Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA DE TERUEL

Departamento de Informática e Ingeniería de Sistemas

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Trabajo Fin de Carrera

Implementación de un clúster HPC en el Centro de Estudios de Física del Cosmos de Aragón

Autor:
Luis Guillén Civera

Director:
Félix Serna Fortea

26 de junio de 2015

Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA DE TERUEL

Departamento de Informática e Ingeniería de Sistemas

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Trabajo Fin de Carrera

Implementación de un clúster HPC en el Centro de Estudios de Física del Cosmos de Aragón

TRIBUNAL

Presidente:

Secretario:

Vocal:

Calificación: _____

Fecha: _____

"La pereza es el escondite preferido del talento."

@Riki_Lopez

UNIVERSIDAD DE ZARAGOZA

Resumen

Escuela Universitaria Politécnica de Teruel
Informática e Ingeniería de Sistemas

Ingeniería Técnica en Informática de Gestión

Implementación de un clúster HPC en el Centro de Estudios de Física del Cosmos de Aragón

por Luis Guillén Civera

El Centro de Estudios de Física del Cosmos de Aragón es una institución cuya actividad se centra en el desarrollo tecnológico y la operación del Observatorio Astrofísico de Javalambre (Teruel), y en la explotación científica de los datos que aporte. Para poder llevar a cabo dicho proceso de explotación científica es indispensable disponer de las herramientas computacionales necesarias. Por este motivo, la institución se ha propuesto adquirir un pequeño clúster HPC que resuelva una gran parte de las necesidades de computación del personal científico y permitir así un incremento en su productividad.

El objetivo de este proyecto será adquirir, diseñar y desplegar una pequeña solución HPC que cubra el mayor número posible de necesidades de cómputo del personal científico del CEFCA con los recursos disponibles.

Agradecimientos

Mi agradecimiento a todas las personas del CEFCA que han contribuido de un modo u otro a la consecución del proyecto. También me gustaría dar las gracias al director de mi proyecto por su predisposición mostrada a lo largo del mismo.

Finalmente me gustaría dar las gracias a todas las personas de mi entorno más cercano que han aguantado y sufrido mi mal humor.

Índice general

Índice de figuras

Abreviaturas

CEFCA	**C**entro de **E**studios de **F**ísica del **C**osmos de **A**ragón
EDAM	**E**xternal **D**ata **A**ccess **M**achine
OAJ	**O**bservatorio **A**strofísico de **J**avalambre
RF	**R**equisito **F**uncional
RNF	**R**equisito **N**o **F**uncional
TFC	**T**rabajo **F**in de **C**arrera
UPAD	**U**nidad **P**rocesado y **A**lmacenamiento de **D**atos

A mis padres...
por ayudar a ese niño a encontrar su pasión

Capítulo 1

Introducción

1.1. Contexto

La Fundación CEFCA es una institución del Gobierno de Aragón que tiene por objeto la implementación en la ciudad de Teruel de un centro de investigación denominado Centro de Estudios de Física del Cosmos de Aragón, cuya actividad se centra en el desarrollo tecnológico y la operación del Observatorio Astrofísico de Javalambre (Teruel), y en la explotación científica de los datos que aporte. Las principales líneas de investigación del CEFCA se concentran en la Evolución de Galaxias y la Cosmología.

El principal proyecto de explotación científica del observatorio y al que debe su diseño es el proyecto J-PAS. Dicho proyecto es un proyecto astronómico cuyo objetivo es cartografiar el Universo de una forma que no se ha hecho hasta ahora y que permitirá nuevos avances científicos, especialmente en el estudio de la energía oscura. Dicho cartografiado se realizará con un telescopio dedicado de 2.5m de diámetro y el campo de visión más grande del mundo, combinado con una cámara de 1.2 Gigapíxeles formada por 14 CCDs. Como es de suponer, una cámara con esa resolución y 14 CCDs tomando imágenes cada 60 segundos durante 4 o 5 años arroja una cantidad de datos en imágenes brutal que es necesario procesar y archivar. Está estimado que el cartografiado con las imágenes y la información procesada sea superior a los 2 PetaBytes.

Durante la generación del cartografiado y el año siguiente a su finalización, el acceso será exclusivo para el personal investigador del CEFCA y para los investigadores pertenecientes a las instituciones colaboradoras. Trascurrido este periodo, todo el material se liberará y estará a disposición de toda la comunidad científica del mundo.

FIGURA 1.1: Observatorio Astrofísico de Javalambre

El desarrollo y mantenimiento de una instalacion científico-técnica singular como es el Observatorio Astrofsico de Javalambre, requiere de un equipo de ingenieros multidisciplinar. Dicho equipo esta agrupado en el departamento de ingeniera cuya misión es dotar de los medios técnicos necesarios para el funcionamiento tanto del OAJ como de la Unidad de Procesado de Datos, y para que el personal científico pueda realizar su labor de investigación.

La Unidad de Procesado y Archivado de Datos (UPAD) es el departamento de CEFCA responsable de los sistemas de almacenamiento y análisis masivo de datos obtenidos por los telescopios del OAJ. El equipamiento principal de UPAD comprende los sistemas encargados de recibir, archivar, tratar y analizar los datos producidos por los telescopios. No obstante este departamento también dirige otras instalaciones de análisis másivo de datos como los sistemas de cómputo científico de uso general instalados en CEFCA.

Finalmente, para poder explotar la información, el CEFCA cuenta con personal científico de primer nivel en los campos de Cosmología y Evolución de Galaxias. Dicho personal pertenece a lo que se denomina el departamento de ciencia.

Este TFC es un proyecto vinculado al área de sistemas y telecomunicaciones, perteneciente al departamento de ingeniera y dentro del marco del departamento UPAD. Dicho área tiene la responsabilidad de diseñar, desplegar y mantener todos los sistemas informáticos y las telecomunicaciones existentes en el Observatorio Astrofísico de Javalambre y en el centro de investigacion ubicado en Teruel.

1.2. Motivación

El departamento de ciencia está formado por investigadores y estudiantes de doctorado que se encuentran realizando tesis relacionadas con alguna de las líneas de investigación. Para que todos ellos puedan realizar su trabajo, necesitan de recursos de cómputo sobre los que poder realizar cálculos y simulaciones. Actualmente todas las personas disponen

de una potente estación de trabajo para satisfacer dichas necesidades. Sin embargo, el tiempo ha mostrado que esta solución es ineficiente e insuficiente.

Aunque las estaciones de trabajo son equipos potentes si las comparamos con PCs normales, no cuentan con los recursos de procesador, memoria y espacio de almacenamiento suficientes para abordar muchos de los problemas que son necesarios resolver. Algunos problemas que se tratan en CEFCA, como simulaciones cosmológicas, son tan grandes que requieren supercomputadores para poder ser abordados. Para este tipo de problemas la opción más económica y eficiente para la institución es recurrir a tiempo en supercomputadores. Sin embargo, existen muchos otros tipos de problemas, como puede ser la realización de análisis de datos para el estudio de evolución de galaxias, que aunque requieren bastante más capacidad de cómputo de lo que una estación de trabajo puede ofrecer, pueden ser abordados mediante técnicas de programación paralela y máquinas más potentes.

Debido a esta creciente necesidad, la institución se ha propuesto adquirir y desplegar un pequeño sistema HPC que sea capaz de resolver una gran parte de las necesidades de computación del personal científico, permitiendo así incrementar su productividad.

El desarrollo de la memoria de este TFC servirá también como documentación interna del departamento de ingeniería del CEFCA.

1.2.1. Motivación personal

Llevo más de 10 años dedicandome profesionalemente a la informática en las más diversas áreas: técnico de soporte, formador, desarrollador de software y administrador de sistemas. Pese a lo negativo que pueda parecer, creo que en general la experiencia me ha hecho ganar perspectiva y me ha proporcionado un amplio arsenal de herramientas a la hora de solucionar problemas. Por ello, y aunque en la actualidad mayormente me dedique al diseño, integración y administración de redes y sistemas, mi principal ventaja competitiva reside en mi habilidad de combinar el mundo de la administración de sistemas con el mundo del desarrollo de software. Por ello trato de enfocar mi carrera profesional hacia una especialización en el desarrollo de software de automatización e integración de sistemas GNU/Linux.

En el mundo HPC la automatización de sistemas es una pieza clave, ya que el número de sistemas es tal que realizar manualmente la administración de dichos sistemas requeriría de una buena plantilla de personal. Es aquí donde reside mi principal motivación, sumada al reto que supone este proyecto por ser el mundo de la computación paralela y de los clusters HPC un misterio para mi. Uno de los grandes principios que aprendí durante

mi etapa de formador es que cuando te ves en la obligación de transmitir conocimientos, el hecho en sí mismo se convierte en un proceso de autoaprendizaje que proporciona una mayor asimilación de los conceptos y un exclarecimiento de los principios existentes detrás de ellos. A este interés por "obligarme a aprender más" se unen las circunstancias de que además se trata de un proyecto delimitado y que puede ser abordable para un TFC.

1.3. Objetivos

El objetivo de este proyecto será adquirir, diseñar y desplegar un pequeño cluster HPC que cubra la mayor parte de las necesidades de cómputo del personal científico del CEFCA con los recursos disponibles.

El clúster HPC estará compuesto de sistemas de cómputo, sistemas de almacenamiento y equipamiento de red propios, por lo que en el proceso de adquisición será necesaria una correcta asignación de los recursos disponibles sobre estos tres factores, teniendo en cuenta una posible ampliación del clúster a corto y medio plazo.

Tras el proceso de adquisición del clúster, habrá que realizar un análisis y modelar el sistema HPC bajo una especificación de requisitos en la que se definirán los diferentes tipos de requerimientos de la solución. Como se verá en esta fase, el clúster HPC no se tratará de un sistema aislado y por lo tanto deberá integrarse con una infraestructura existente. Esto requerirá integración del clúster HPC con:

- Infraestructura de red
- Servicios de red
- Active Directory
- Servicio de logging centralizado
- Sistemas de monitorización
- Sistemas de backup
- Sistemas de distribución de software
- Software de automatización

Tras el análisis y la especificación de requisitos, se realizará un diseño que posteriormente deberá ser implementado, siendo todo el proceso recogido en la memoria. En la etapa

de desarrollo, para la implementación de los sistemas se adoptará el paradigma de *la infraestructura como código*. Siguiendo este paradigma, el despliegue de toda la infraestructura será codificada en software, esto nos permitirá realizar el desarrollo en una infraestructura de pruebas y luego, usando el mismo código, generar la infraestructura de producción. Para ello haré uso de un conjunto de herramientas que personalmente he venido desarrollando a lo largo del tiempo: lgvbox, lgsetup y lgdeploy. El código de todas estas herramientas así como el código completo de despliegue de una infraestructura casi idéntica a la realizada se entregará junto a esta memoria (por motivos de seguridad no proporciono passwords, certificados y datos del desarrollo real).

El sistema final será visto por los usuarios como una caja negra a la cual se conectan, lanzan sus trabajos y obtienen resultados sin preocuparse por la asignación de los recursos del clúster. Dispondrán además de herramientas para conocer el estado de sus trabajos, el estado del clúster y de la documentación necesaria para poder empezar a usarlo.

1.4. Estructura de la memoria

He optado por estructura la memoria en seis capítulos.

- El primer capítulo es esta introducción a la memoria.
- En el segundo capítulo se hace un repaso sobre el estado de la cuestión o del arte en el ámbito de los clústers HPC.
- En el tercer capítulo se realiza la etapa de análisis y especificación de requisitos.
- En el cuarto capítulo se presenta la solución hardware adoptada y se hace una planificación del desarrollo.
- En el quinto capítulo se realiza el desarrollo de la solución.
- En el sexto y último capítulo se comentan las conclusiones y las próximas acciones.
- En los anexos he incluido material adicional desarrollado durante el proyecto.

Capítulo 2

Estado del Arte

Antes de proseguir con el análisis y la especificación de los requisitos, es necesario adquirir una perspectiva global y conocer el estado actual de la tecnología en el ámbito de la computación de alto rendimiento.

2.1. Introducción

2.1.1. Computación de alto rendimiento

Existen problemas en los que el tiempo requerido para resolverlos computacionalmente en un único computador es grande y es imposible esperar todo ese tiempo para obtener los resultados. Un ejemplo claro sería un modelo de predicción meteorológica que se toma dos días en obtener una predicción de temperatura del día siguiente. Tener una predicción del tiempo que hizo ayer podrá servirnos para comprobar que nuestro modelo es correcto pero ¡jamás nos sería práctico!. Otros problemas, sin embargo, pueden ser demasiado grandes para los recursos de un único computador. Por ejemplo, podríamos querer simular el caudal de un río y que, con la memoria disponible en el computador, nuestro modelo pudiese simular únicamente unos cuantos metros cúbicos. En la práctica se trataría de un modelo estupendo que realizaría simulaciones perfectas del efecto de la caída de una esponja en nuestra bañera pero no sería útil para nada más.

El campo de la Computación de Alto Rendimiento (High Performance Computing) viene a cubrir estas necesidades especiales de cómputo. Como se verá posteriormente, hasta hace poco tiempo los supercomputadores eran las únicas herramientas capaces de abordar este tipo de problemas. En la actualidad los clústers y los avances en la computación paralela y distribuida han logrado abaratar enormemente los costes de las soluciones y con ello la popularización de este campo.

2.1.2. Definición y tipos de clústers

Acabo de introducir el concepto de clúster, pero ¿qué es un clúster?. Un clúster es un sistema de procesamiento de tipo paralelo o distribuido, que consiste en una colección de computadores (nodos) interconectados trabajando juntos como si se tratasen de un único e integrado recurso de cómputo. Un clúster podrá ofrecernos las siguientes características (usaré su forma inglesa para evitar confusiones):

- **High Performance**: alto rendimiento, obtendremos un rendimiento mucho mayor
- **Expansible y Escalable**: será fácil de extender agregando nuevos nodos sin entrar en rendimientos decrecientes
- **Hight Throughput**: alta capacidad, el número de trabajos que podrá concluir será mucho mayor
- **High Availability**: alta disponibilidad, la disponibilidad global del sistema será mayor

Además existen diferentes clasificaciones en función de diversos factores:

- Según su *ámbito o aplicación*
  - High Performance: obtener un mejor rendimiento
  - High Availability: dotar al sistema de una mayor disponibilidad y tolerancia a fallos
- Si la *tarea que realiza* es específica
  - clúster dedicado
  - clúster no dedicado
- Según el *hardware* del nodo
  - clúster de PCs
  - clúster de Workstations
  - clúster se servidores multiprocesador
- Según el *sistema operativo*
- Según la *configuración* hardware/software del nodo
  - clústers homogeneos: todos los nodos son exactamente iguales
  - clústers heterogeneos: los nodos son distintos en mayor o menor grado (pueden tener hasta distinta arquitectura)

2.1.3. Soluciones HPC

Hemos introducido anteriormente el campo de la Computación de Alto Rendimiento (HPC) y el concepto de clúster. La revolución tecnológica en la que estamos inmersos y sus avances en materia de telecomunicaciones, han propiciado la aparición de nuevas soluciones y de nuevos conceptos. Por ello vamos a realizar una clasificación de las soluciones HPC disponibles en función de dónde se obtienen los ciclos de CPU necesarios para la ejecución de nuestro software:

- **El clúster HPC commodity**: con el término *commodity* hacemos referencia a que el tipo de hardware empleado es un hardware común, ampliamente utilizado y proporcionado por multitud de fabricantes en el mercado. Los grandes avances tecnológicos junto al progreso en el paradigma de la computación paralela que se han llevado a cabo en las últimas décadas, han hecho posible el uso de este tipo de hardware para la resolución de problemas complejos.

- **Supercomputadores dedicados**: son máquinas que usan componentes específicamente diseñados para el problema que tratan de resolver. Aunque este hecho los convierta en la mejor opción para resolver cierto tipo de problemas, también hace que el coste sea muy elevado. En el pasado este tipo de supercomputadores era la única forma que había para solucionar problemas complejos, sin embargo el elevado coste unido a los avances tecnológicos llevados a cabo han hecho que muchos supercomputadores hayan sido reemplazados por clústers HPC de tipo *commodity*.

- **HPC cloud computing**: los avances en multitud de disciplinas han convertido a la capacidad de resolver problemas computacionalmente en una ventaja competitiva. Dicha ventaja competitiva ha impulsado la demanda de servicios de tipo HPC propiciando la aparición de un nuevo mercado con proveedores de este tipo de servicios. Los clientes demandantes son generalmente empresas de tipo pequeño y mediano que sus necesidades de cómputo no son tan grandes como para poder permitirse los elevados costes que tiene la adquisición y el mantenimiento de una infraestructura HPC propia. El lado de la oferta lo constituyen ciclos de cómputo o sistemas de almacenamiento bajo demanda que permiten al cliente pagar únicamente por lo que usa. Esta solución, sin embargo, no está exenta de problemas, como pueden ser los derivados por la delegación en un tercero (pérdida de control de los procesos y la información, etc) o meramente tecnológicos (velocidades de transmisión, etc).

- **Grid computing**: si el *cloud* surge de la necesidad de obtención de recursos proporcionados por un tercero, el *grid* surge de la necesidad de compartir los recursos

con terceros con el fin de optimizar al máximo el conjunto de los recursos. Sus principales usuarios son instituciones académicas que tienen clústers HPC propios y desean interconectarlos y compartirlos con otras instituciones para optimizar al máximo el agregado de los recursos. Pero el concepto de Grid va mucho más allá y la tecnología ha permitido que ordenadores personales distribuidos a lo largo de todo el mundo puedan unirse y ofrecer sus recursos a distintos proyectos científicos. Ejemplos destacados de este tipo de computación son el proyecto SETI o los proyectos de la fundación Ibercivis. Finalmente, comentar que presentan los mismos problemas que enumeramos en el *cloud computing*.

2.1.4. El clúster HPC

2.1.4.1. Causas de su éxito

Varias son las circunstancias que han permitido al clúster HPC liderar el mundo de la computación de alto rendimiento:

- **Hardware**: durante su crecimiento, el mercado de los equipos de consumo ha ido incluyendo nuevas características que han hecho posible que dichos equipos sean capaces de resolver un número cada vez mayor de problemas. Las avances en arquitectura y organización así como el desarrollo de nuevas tecnologías de integración e interconexión, han permitido que un equipo *commodity* dispusiera progresivamente componentes que eran exclusivos de los supercomputadores. Disponer de coprocesadores matemáticos, múltiples unidades de proceso, procesadores vectoriales (GPUs), etc, son hoy en día características habituales en un equipo *commodity*.
- **Software**: en el ámbito del software, el surgimiento y maduración del sistema operativo libre GNU/Linux ha permitido disponer de un sistema que es un reemplazo natural de UNIX (sistema empleado en la mayoría de los supercomputadores), que no tiene costes adicionales de licenciamiento y que sus características cubren la mayoría de las necesidades de los HPC. De manera adicional, los avances en las herramientas de desarrollo han ido haciendo cada vez más fácil el desarrollo de aplicaciones paralelas. Dichas herramientas se han ido depurando y estandarizando a lo largo del tiempo, obteniendo una base sólida sobre la que desarrollar las más diversas aplicaciones.
- **Telecomunicaciones**: el auge de Internet ha propiciado que tecnologías asociadas a las telecomunicaciones progresasen tanto en prestaciones como en reducción

de costes. Esto ha permitido que la interconexión de equipos se abaratase notablemente y se hayan obtenido unas prestaciones adecuadas para la resolución de problemas complejos.

- **Transmisión del conocimiento**: para finalizar, ese surgimiento de internet también ha permitido que se crearan muchas comunidades de HPC que han multiplicado la velocidad de transmisión del *know-how*.

2.1.4.2. Perspectivas

En función del tipo de problemas que vayamos a resolver, podemos ver a nuestro clúster como si fuese:

- **Granja de cómputo**: en este caso podríamos enviar al clúster los "pequeños" trabajos que deseemos procesar y dejaremos que sea él quien asigne los recursos necesarios. Para que esto sea posible necesitaremos un software que compruebe el estado de los recursos disponibles y gestione la ejecución de los trabajos de forma autónoma.
- **Supercomputador**: en este caso veríamos al clúster como si fuese un gran recurso de cómputo al que enviaremos nuestro trabajo. Para que esto sea posible se ofrecerán al programador distintas herramientas que harán posible esta abstracción.

2.1.4.3. Cómo son

Del mismo modo que existe una diversidad de problemas, existe una diversidad de soluciones. Aun así hay algunas características comunes que todos los clústers comparten.

- Existe un **nodo maestro** o **cabecera** (*head*) que ofrece un mecanismo de *login* donde los usuarios se logean y utilizan los recursos del clúster. Dicho nodo está conectado a una o más redes privadas a las que se encuentran conectados los nodos de cómputo.
- Los **nodos de cómputo** (*workers*) son los encargados de realizar las tareas de cómputo y no son accesibles directamente. Suelen ser nodos idénticos aunque todo dependerá del tipo de aplicaciones que necesitemos ejecutar.
- Suele existir un **almacenamiento compartido** que los nodos de cómputo pueden necesitar para realizar su trabajo (datos de entrada, salida, temporales, etc). Este almacenamiento será diseñado para soportar el tipo de aplicaciones a ejecutar. En

su versión más simple, dicho almacenamiento podría ser un NFS compartido por el nodo maestro. En su versión más compleja podemos irnos a sistemas de archivos paralelos, sistemas multicapa (formados por jerarquías de almacenamiento) o sistemas de archivos que combinan ambos como es el caso de la UPAD.

- En lo referente al **interconexionado de red**, un clúster suele tener tres tipos de tráfico:
  - Tráfico de cómputo
  - Tráfico de almacenamiento
  - Tráfico de administración

Dependiendo del tipo de aplicación, el tráfico de cómputo o el tráfico de almacenamiento pueden ser dominantes y pueden llegar a causar esperas en los nodos. Por ello se suelen separar en redes independientes y utilizar tecnologías de interconexión de mejores prestaciones como pueden ser ethernet 10G o Infiniband.

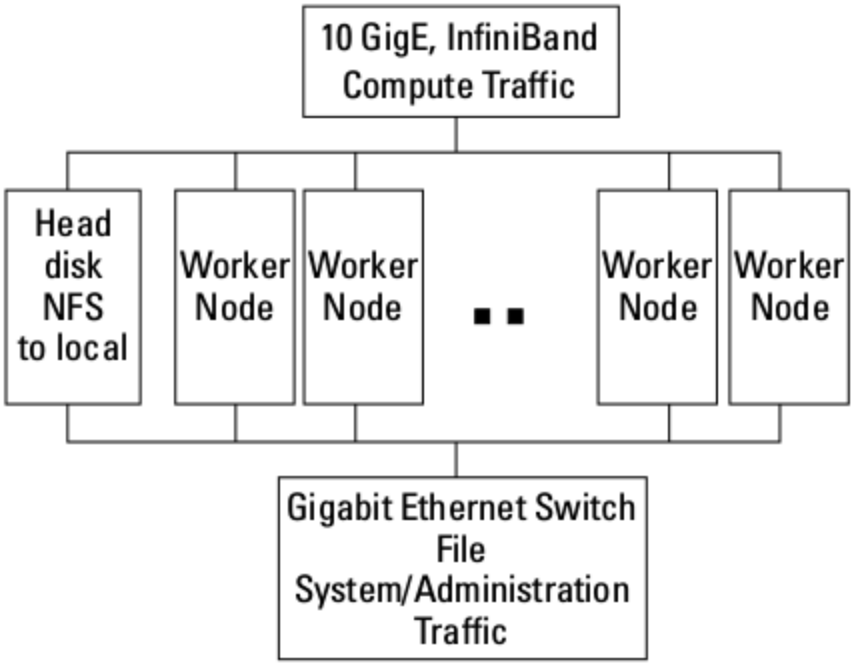


FIGURA 2.1: Configuración típica de un clúster

Como puede verse en la figura 2.1 se muestra cómo es una configuración típica. En esta configuración el nodo *cabecera* tiene acceso directo a una cantidad grande de almacenamiento que comparte vía NFS con el resto de los nodos. Luego tenemos un número variable de nodos de cómputo se comunican con los demás y con el nodo cabecera mediante redes privadas.

2.1.4.4. El rendimiento en un clúster HPC

La medida del rendimiento estándar de un HPC generalmente se mide en Gigaflops. Los FLOPS son las operaciones de coma flotante por segundo que es capaz de realizar, indicando si son de precisión simple o doble. Para poder utilizar dicha medida de rendimiento, es necesario establecer una referencia estándar para las computadores que deseamos comparar, siendo el estándar LINPACK la referencia más más utilizada en este ámbito.

Como ya se comentó, uno de los factores causantes del gran éxito de los HPC han sido los avances en la computación paralela. En este ámbito cabe destacar la existencia de una ley llamada Ley de Amdahl. Esta ley dice: "la mejora obtenida en el rendimiento de un sistema debido a la alteración de uno de sus componentes está limitada por la fracción de tiempo que se utiliza dicho componente". Esto llevado a nuestro ámbito quiere decir que es el algoritmo el que decide la velocidad y no el número de procesadores. Finalmente se llega a un momento en el que no es posible paralelizar más el algoritmo y en el que por lo tanto la parte serial determina el límite en el rendimiento.

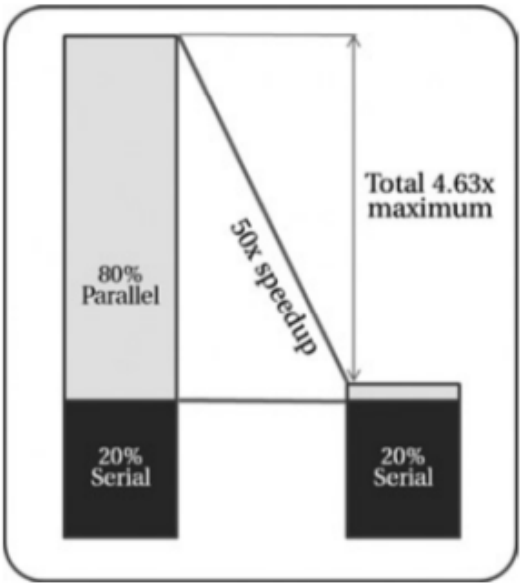


FIGURA 2.2: Ilustración de la Ley de Amdahl

2.2. Elementos de un clúster HPC

2.2.1. Nodos

Como ya se ha comentado, los nodos de cómputo son los elementos que realizan el procesamiento en un clúster HPC. Un **nodo de cómputo** está formado por un conjunto

de placa base, procesador/es, memorias, interfaces de red y opcionalmente discos duros. Aunque un nodo puede ser una workstation de gama alta, generalmente se emplearán servidores que puedan ser enrackados.

A continuación mostraré algunas cuestiones acerca del tipo de nodo a escoger para clúster HPC de servidores y trataré los componentes que lo forman.

2.2.1.1. Tipo de nodo

A la hora de seleccionar los nodos, bien porque deseamos una configuración muy específica o bien por motivos económicos, podemos optar por construirlo nosotros mismos a través de proveedores como Supermicro seleccionando chasis, placa base, etc. La otra opción y la más recomendada consiste en escoger configuraciones prediseñadas de servidores de alguno de los múltiples proveedores existentes: Dell, HP, Fujitsu, IBM, etc.

Una de las primeras cosas que hay que comprobar es que la ubicación física reúne los requisitos de suministro eléctrico y acondicionamiento de temperatura y humedad necesarios. Por ello los primeros datos importantes a comprobar a la hora de seleccionar un nodo son sus datos de eficiencia energética: consumo y energía que disipa en forma de calor.

Una de las grandes decisiones que habrá que tomar es si deseamos un sistema de tipo rack o de tipo blade. Cada uno tiene sus ventajas e inconvenientes.

- En los sistemas tipo rack, cada servidor tiene sus propios componentes, por lo que el coste crece de forma lineal conforme agregamos nodos. Dependiendo del servidor, lo normal es que ocupe una o dos Us en el rack.
- Los sistemas tipo blade están compuestos por un chasis de cuatro o más Us, al que se van insertando servidores blade. El número de servidores que caben en el chasis dependerá del fabricante, modelo, etc, pero un chassis de cuatro Us es normal que pueda contener hasta ocho servidores. Esto hace que ofrezca una mayor densidad de cómputo (más cómputo en menos espacio) que una solución de sistemas en rack. Sin embargo, esto exige mayores necesidades de suministro eléctrico y enfriamiento. Otra de las características es que el chasis contiene algunos elementos que compartirán todos los servidores. Como mínimo se suele compartir las fuentes de alimentación redundantes que dispone, pero algunos chasis pueden proporcionar conexiones de consola de administración común, lector óptico, usbs e incluso switches de red. Uno de los grandes problemas de los sistemas blade es su coste, ya que hay que comprar el chasis independientemente del número de servidores que vayamos a insertar.

FIGURA 2.3: Servidores tipo blade

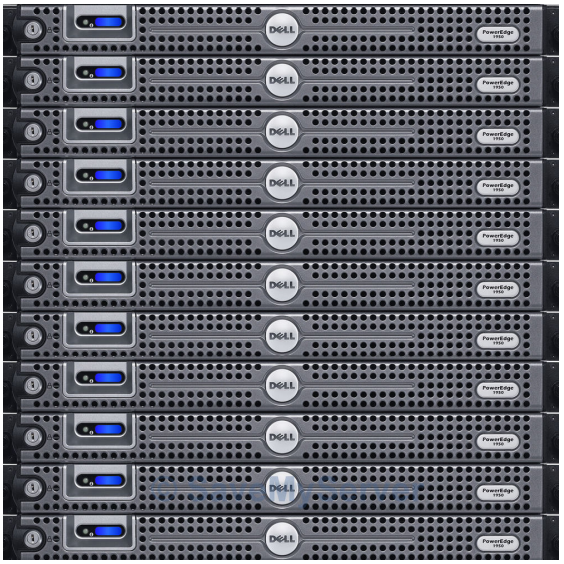


FIGURA 2.4: Servidores tipo rack

Otro de los elementos a valorar son las posibilidades de administración y monitorización que ofrece el hardware del servidor, aunque esto lo veremos con más detalle cuando tratemos la placa base.

2.2.1.2. Procesador

El procesador es el caballo de batalla de la computación: nuestro éxito dependerá de tener los procesadores ocupados la mayor parte del tiempo.

En la actualidad, aunque existen otras arquitecturas, la arquitectura predominante es la x86_64. Los **procesadores** que implementan dicha arquitectura pueden llevar integrados

4, 6, 8, 10 o incluso 12 **cores** (unidades de proceso). Además los cores actuales suelen disponer de características avanzadas como:

- la tecnología Hyperthreading o EMT, que duplica algunos de los componentes internos del core, facilitando la ejecución concurrente de dos tareas en el mismo core
- las tecnologías VTx o AMDv, que agregan soporte para tecnologías de virtualización
- Turbo Boost, que permite subir manera automática la frecuencia del procesador sobre su límite si se dan las condiciones necesarias (no confundir con *speedsteep* que lo regula para minimizar el consumo)
- extensiones SSE, SSE2, SSE3, AVX y AVX2 que dotan de registros e instrucciones optimizadas para el procesamiento vectorial

En la actualidad, en lo que respecta a Intel, la última generación es la llamada "Haswell". Esta última generación supone un rediseño en la arquitectura con respecto a la generación anterior (lo que ellos llaman un tock[1]) y que aporta numerosas mejoras en cuanto a lo que a eficiencia energética se refiere. Aun así, lo más normal todavía es adquirir un procesador la generación anterior "Ivy Bridge" que es tick de "Sandy Bridge" y que viene heredando de varias generaciones una comunicación directa con la memoria y una arquitectura tipo NUMA en la que se interconectan las CPUs mediante la tecnología QPI.

Por lo general, a la hora de seleccionar un procesador nos iremos a alguno de la familia Xeon de Intel (o de la correspondiente en el caso de AMD) y la generación que nos interese. Después existen múltiples modelos en los que los parámetros en los que diferirán serán: número de cores, frecuencia de procesador, si tienen hyperthreading o no y el tamaño de las cachés.

2.2.1.3. Placa base

La placa base es el elemento de interconexión de los componentes de un nodo, por lo que su elección estará condicionada por el tipo de procesador y memorias que queramos emplear. Aunque es un elemento muy importante, ya que las interconexiones que realiza van a ser un cuello de botella en las comunicaciones entre los componentes, los proveedores de servidores generalmente sólo suelen indicar algunos de los elementos de

[1] http://computadoras.about.com/od/Tecnologias/a/Intel-Tick-Tock.htm

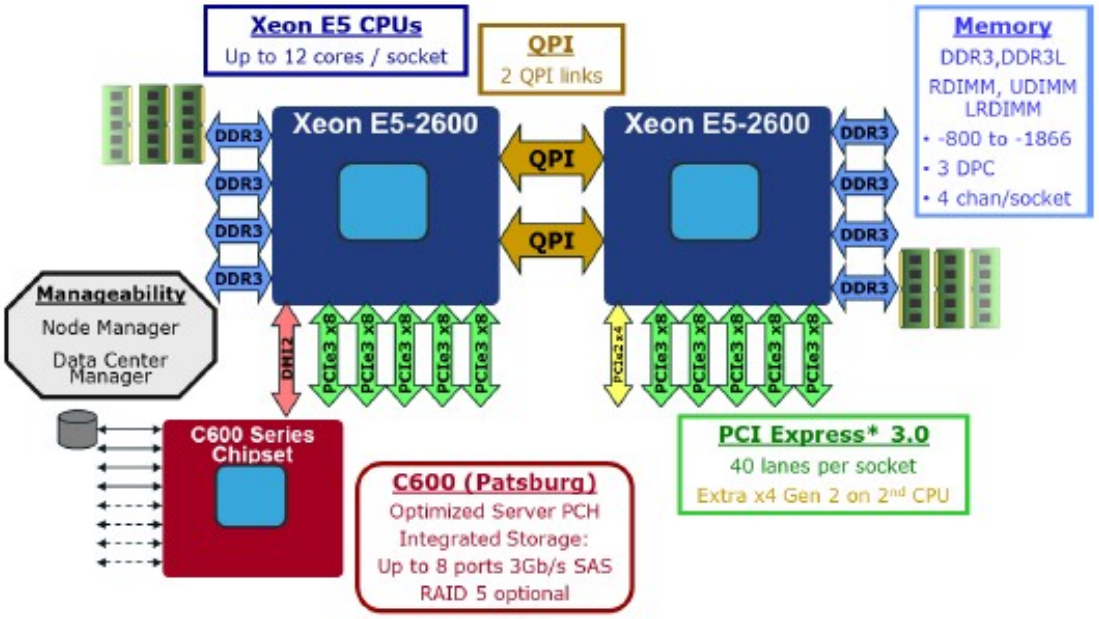


FIGURA 2.5: Organización procesador Xeon E5

FIGURA 2.6: Placa base para cuatro procesadores

la placa y rara vez especifican el modelo. Los elementos a destacar de una placa son: el chipset, el número de sockets para procesadores que ofrece, los canales y slots de memoria disponibles para cada socket, tipo y ranuras PCI express disponibles, conexiones SAS, interfaces de red integradas y elementos para la administración y monitorización.

Las placas base de la actualidad (estamos hablando de hardware "commodity") pueden llevar sockets para 1, 2 o hasta 4 procesadores. Dado que los procesadores pueden llevar integrados 10 o 12 cores, podemos llegar a alcanzar hasta 48 cores en un sólo nodo. Este valor de multiplicar el número de procesadores por el número de cores por procesador se le llama **cores por nodo** y se suele utilizar a la hora de describir un clúster HPC.

El tipo de bus de alta velocidad que se usa en la actualidad para conectar dispositivos es el PCI express. Este bus se encuentra en su versión 3.0 y una ranura puede alcanzar 1GB/s de tráfico direccional y 2GB/s bidireccional. Si se usa una ranura x16 puede alcanzar un máximo de 16GB/s direccionales y 32GB/s bidireccionales. Esto es importante de entender, porque existen distintos tipos de ranura: x1, x4, x8 y x16. Algunas tarjetas,

como las HBAs, tarjetas de red 10Gb, etc, pueden requerir conexiones x16 y el número de ranuras de este tipo en una placa suele escasear. También es importante tener en cuenta la disposición de las mismas y el espacio, ya que puede ser necesario recurrir a tarjetas tipo Riser perpendiculares para poder conectar las tarjetas.

A nivel de interfaces de red, las placas base de tipo servidor suelen llevar integrados un par de puertos de red, por lo que, dependiendo de nuestra arquitectura de HPC, puede ser necesario adquirir tarjetas de red adicionales y para esto será necesario disponer de slots PCI express libres.

Para finalizar, casi todos las fabricantes suelen implementar el estándar IPMI. Dicho estándar es una especificación que provee de funcionalidades de administración y monitorización. Para ello, se integra un chip llamado BMC al que se le conectan diversos sensores y determinados elementos de la placa. Generalmente dispone de una interfaz interna a la que puede accederse mediante un driver en el sistema operativo y de interfaces externas. Dichas interfaces externas pueden estar en forma de puertos serie o pueden incluso utilizar los chip de las tarjetas de red integradas para su uso antes de que el procesador tome su control. Esto permite realizar tareas administrativas y de monitorización cuando el sistema operativo no esté todavía cargado.

De manera adicional, los fabricantes suelen ofrecer en sus gamas Enterpise su propio sistema IPMI que ofrece características avanzadas. Este tipo de sistemas suelen estar formados por un hardware casi "independiente" que se inserta y que ofrece una interfaz de red adicional dedicada. Esto nos permitirá realizar operaciones sobre el servidor incluso cuando el sistema operativo está cargado. Suele ser habitual y es una funcionalidad a tener muy en cuenta, que ofrezcan un KVM virtual que nos permita visualizar lo que sale por la VGA y enviar pulsaciones de teclado y ratón remotamente al servidor. Un buen ejemplo de estos sistemas son las ILO de HP y las iDRAC de Dell.

2.2.1.4. Memoria RAM

La memoria es un recurso crítico para el cómputo. Muchos programas por su naturaleza pueden necesitar grandes cantidades de RAM. Pero no sólo eso, como el objetivo es maximizar la eficiencia, se tratará siempre de minimizar el número de accesos a recursos varios órdenes de magnitud más lentos como pueden ser la red y el disco. Una estrategia habitual suele ser cargar los datos de entrada en memoria para evitar lecturas adicionales. A estos programas que sólo leen del disco los datos de entrada al iniciarse y escriben únicamente una vez finalizados se les denomina **In-Core**.

Un valor que se suele utilizar es la cantidad de **memoria por core**, que es el total de la memoria presente en el nodo divido por el número de cores presentes. Es importante

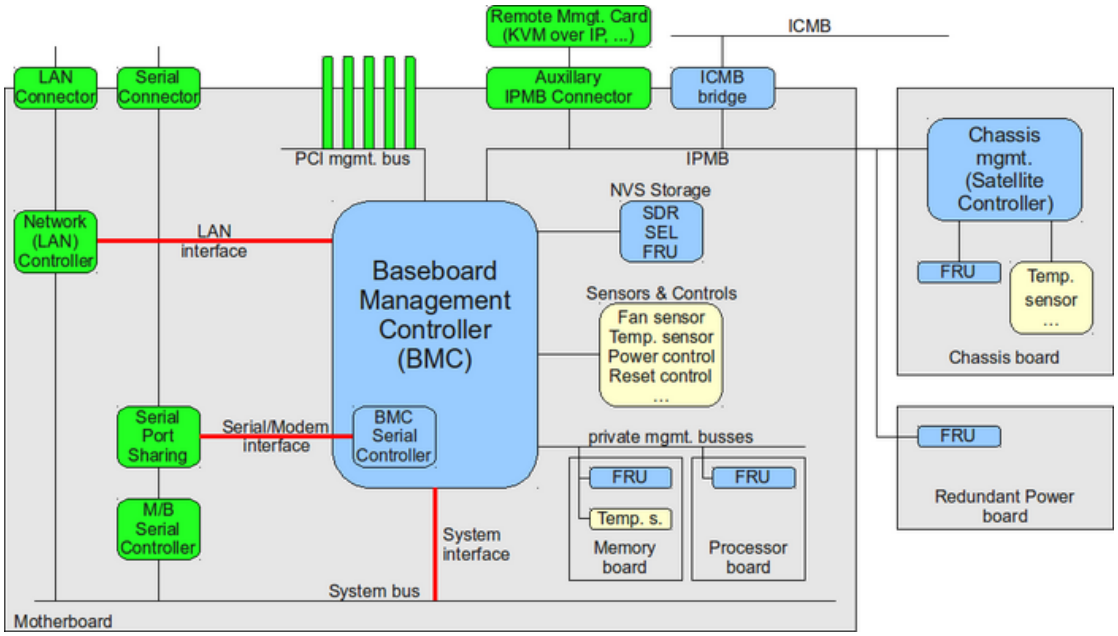


FIGURA 2.7: Diagrama BMC

resaltar que el aumento del número de cores por nodo debe llevar también un incremento en la memoria disponible, de lo contrario la memoria disponible por core disminuirá.

A un nivel de organización, en las últimas generaciones se ha realizado conexión directa con las CPUs a través de un bus dedicado, esto evita contenciones en el acceso al bus y permite una mayor velocidad. Esto también implica que el propio procesador tendrá una velocidad máxima de acceso a memoria y que a su vez condicionará la elección del tipo de memoria. En cuanto a tecnología, la memoria RAM se proporciona en DIMMs, que son un conjunto de chips DRAM agrupados en una única pastilla (también llamado módulo). En la actualidad, las memorias más modernas existentes en el mercado son las DDR4 que traen una mejora sustancial en el rendimiento. Aun así, lo común y lo más asequible a día de hoy son las DDR3. Actualmente podemos encontrarnos DIMMs con tamaños comprendidos entre 2GB y 32GB y unas tasas de transferencia entre 800 MT/s (6.4GB/s) y 1866 MT/s (14.9GB/s). Los tipos de DIMM que podemos encontrarnos son UDIMMs, RDIMMs, LRDIMMs y HDIMMs. Añadir también que podremos seleccionar entre memorias ECC o non ECC, esto dotará al módulo de memoria una mejor corrección de errores y detección de datos corruptos pero a cambio de un coste mayor. Es muy recomendable que optemos por módulos ECC.

Como hemos comentado anteriormente, los sistemas actuales siguen una arquitectura NUMA en la que cada CPU tiene su propia memoria local. Esto hace especialmente importante una correcta distribución de los DIMM en los canales disponibles.

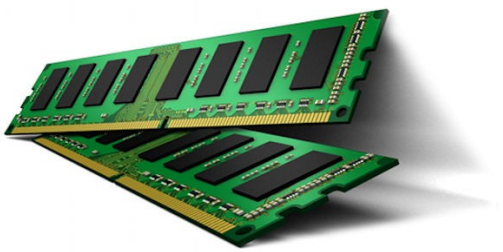


FIGURA 2.8: Módulos de memoria RAM

2.2.1.5. GPUs, coprocesadores y FPGAs

Las tarjetas gráficas se caracterizan por tener su propio sistema CPU (llamada **GPU**) y memoria. Por la propia naturaleza del problema a resolver, proporcionan un modelo de computación **SIMD** (Single Instruction Multiple Data). Esto quiere decir que una única instrucción se aplica sobre un conjunto de datos al mismo tiempo y es especialmente eficiente para resolver determinados tipos de problemas. Este modelo de computación es el que han venido proporcionando los procesadores vectoriales de algunas supercomputadoras y que ahora gracias a las GPUs está disponible en los clústers de tipo "commodity".

Sin embargo, para poder incorporar las GPUs a la resolución de problemas, es necesario realizar desarrollos específicos para su arquitectura. Por ello los fabricantes proporcionan su propio conjunto de herramientas de desarrollo (cabe destacar al popular CUDA de NVIDIA). Debido a la creciente diversidad de GPUs, sistemas operativos y fabricantes, se está haciendo un esfuerzo por estandarizar su uso para la computación mediante el estándar **OpenCL**. Como GPU actual de referencia en el mercado podemos mencionar a la NVIDIA Tesla.

Este emergente mercado de HPC ha impulsado la aparición de elementos de cómputo adicionales como son los **coprocesadores**. Cabe destacar la reciente aportación de Intel con su **Xeon Phi**. Dicho coprocesador dispone de una arquitectura multicore que incluye elementos que nos permiten la computación SIMD. Para poder usarlo en nuestros programas, Intel provee junto a este coprocesador herramientas de desarrollo que incluyen compiladores y librerías. La gran ventaja sobre las GPUs es que las librerías que se ofrecen explotan los recursos del coprocesador y son implementaciones de librerías ya existentes y comúnmente empleadas en el desarrollo de aplicaciones de tipo científico. Esto permite que los programas ya desarrollados puedan hacer uso del coprocesador con poco más esfuerzo que recompilarlos.

FIGURA 2.9: Publicidad Xeon Phi

Pero el elemento que puede disparar la capacidad de cómputo de nuestro HPC son las FPGAs. Tradicionalmente el desarrollo de electrónica específica para el cómputo ha sido exclusivo de los supercomputadores. Sin embargo, los grandes avances surgidos en esta tecnología y la aparición de múltiples herramientas de desarrollo han propiciado que sea posible desarrollar hardware específico para resolver problemas. La herramienta más empleada en este ámbito son las FPGAs. Sin embargo, su mayor ventaja es también su mayor inconveniente: ser hardware dedicado permite una optimización total en los cálculos pero requiere de un desarrollo específico y personal especializado. Este inconveniente lo aleja de su posible uso generalizado.

2.2.1.6. Almacenamiento local

Los nodos pueden llevar o no almacenamiento local. Los motivos para su existencia o ausencia responderán básicamente a tres: administrativos, de aplicación o económicos. Los motivos "administrativos" tienen que ver fundamentalmente con si el nodo va a tener instalado el sistema operativo en un disco local. Los de tipo de "aplicación", tendrán que ver con el uso que pudiesen hacer de este almacenamiento las aplicaciones al ejecutarse en los nodos. Por supuesto, los motivos económicos tendrán que ver con que el almacenamiento adicional en cada nodo incrementará el precio total de nuestra solución HPC.

Usar almacenamiento local para el sistema operativo tiene sus ventajas y sus inconvenientes. La principal ventaja es que el nodo no tiene que emplear la red para cargar el

sistema operativo: esto reduce el tráfico de red y evitará la dependencia de la disponibilidad de la infraestructura necesaria para el arranque por red. Otra ventaja es que nos ofrece un lugar de almacenamiento para ficheros temporales y de logs que de otro modo bien podrían generar tráfico de red y necesitar espacio en un almacenamiento compartido o bien ocupar espacio en RAM mediante el uso de un ramdisk. Para finalizar, el disponer de almacenamiento individualizado por nodo facilitará la "personalización" de la instalación del nodo. Por otro lado, realizar una instalación del sistema operativo independiente en cada uno de los nodos llevará a la necesidad de desplegar el sistema en todos los nodos así como distribuir su configuración.

Según el tipo de aplicación, trataremos de evitar el uso de recursos compartidos de almacenamiento, evitando de este modo tráfico de red y retrasos por la competencia con otros nodos. La forma más habitual es utilizar dicho almacenamiento a modo de "scratch" o temporal: para almacenar resultados intermedios de la ejecución de los programas.

Posteriormente ampliaré información acerca del estado actual de la tecnología, en el apartado dedicado al almacenamiento.

2.2.2. Interconexión

Para minimizar el tiempo ocioso del procesador es necesario una buena interconexión que permita que el tiempo de obtención de la información requerida para que el procesador pueda continuar operando sea el menor posible. La información que puede necesitar un proceso puede ser desde ficheros en un almacenamiento compartido hasta mensajes de señalización con otro proceso. También siempre será necesario disponer de conectividad entre los nodos para la administración, monitorización, despliegue de software y otras tareas.

La definición de la interconexión al final dependerá del tipo de problema, ya que en ocasiones el problema requerirá un alto ancho de banda si se realizan transferencias de memoria o ficheros, en otras ocasiones una baja latencia si se realiza mucha señalización y en otras no será necesario disponer de grandes prestaciones. En cualquier caso, la interconexión tiene un coste significativo en el coste de adquisición de HPC.

2.2.2.1. Tipos de tráfico

Cuando se describió cómo es un HPC se distinguían básicamente tres tipos de tráfico: de cómputo, de almacenamiento y de administración.

- **El tráfico de cómputo** es el tráfico que generan los nodos al comunicarse entre sí para resolver los problemas de forma paralela. Este tráfico suelen ser mensajes MPI que se caracterizan por ser mensajes que esperan ser entregados lo más rápido posible. La prestación más importante para este tipo de tráfico es una baja latencia. Además si se incluyen muchas transferencias de memoria entre nodos, un buen ancho de banda será fundamental.
- **El tráfico de almacenamiento** es el tráfico necesario para obtener la información desde un recurso de almacenamiento compartido. Este tipo de tráfico mueve grandes cantidades de información. Aunque siempre es deseable tener una baja latencia, la prestación más deseable es disponer de un buen ancho de banda.
- Para finalizar, **el tráfico de administración** es un tipo de tráfico que está a medio camino entre los dos tipos de tráfico descritos. En general, podríamos decir que este tipo de tráfico no requiere grandes prestaciones.

2.2.2.2. Tecnologías existentes

En términos de tecnología, actualmente las opciones más extendidas son: Infiniband, Myrinet, Fibre Channel, 10GigE y 1GigE.

- **1GigE**: es el estándar de comunicación de cualquier red local actual. Si no requerimos una capacidad de conexión grande, es una buena solución especialmente en lo que al precio se refiere. La mayoría de los servidores tienen en placa instaladas dos interfaces y existen switches de alta densidad (con muchos puertos) a precios económicos.
- **Myrinet 10G**: es una tecnología basada en fibra óptica muy usada en redes de cómputo ya que proporciona una muy baja latencia pero con un coste bastante elevado. Estaba muy extendida en los grandes supercomputadores hace unos años, sin embargo ha venido siendo reemplazada por Infiniband.
- **Fibre Channel**: se hablará de ella en la parte de almacenamiento. Suele utilizarse como tecnología para confeccionar redes internas de almacenamiento y es muy común su uso en soluciones dedicadas de almacenamiento.
- **Infiniband**: es un estándar de la industria y es considerado por mucha gente como la mejor interconexión debido a su baja latencia y alto throughput. Actualmente en el mercado es posible acceder a equipos a un precio con la tecnología FDR que proporcionan anchos de banda de hasta 56G.

- **10/40GigE**: es una evolución del Ethernet, pero proporcionando un mayor ancho de banda. Está empezando a competir directamente con Infiniband ya que presenta un coste más bajo y, al ser derivado de ethernet, no requiere de formación adicional por parte de los administradores de sistemas. La latencia no llega a ser la de Infiniband, pero los switches están empezando a mejorar su velocidad de conmutación y existen tecnologías implementadas por hardware en las tarjetas como TCP Off Load que permiten un mejor desempeño.

Como ya comenté, hay que tener muy claro que lo más importante antes de decantarse por una solución es entender muy bien los requerimientos de comunicación de nuestras aplicaciones. Si nos decantamos por una tecnología de alto rendimiento como puede ser Infiniband, el coste de la solución global aumenta y podemos dedicar menos dinero a nodos de cómputo. Esto es, cuanto más rápida queramos que sea la red, menos dinero tendremos para el cómputo. Otro elemento importante es el tipo de conexionado, cuya elección estará condicionada la tecnología y las distancias entre los nodos.

2.2.3. Almacenamiento

En prácticamente la totalidad de las soluciones, es necesario un almacenamiento compartido. En los escenarios más sencillos, los procesos leerán sus datos de entrada de manera secuencial, estarán la mayor parte del tiempo procesando y escribirán los resultados. En escenarios más complejos, múltiples procesos pueden requerir de acceso al disco cada poco tiempo y de manera concurrente. En otros escenarios, la información con la que se trabajará podrá tratarse de bases de datos enormes del orden de petabytes de información. Por si todo esto fuera poco, casi siempre será necesario realizar un respaldo de la información existente en determinados espacios del almacenamiento compartido para no perder los resultados de la investigación así como definir unos controles de acceso a la misma.

Podemos deducir que el tipo de problema condicionará enormemente la solución de almacenamiento a escoger. Dicha solución deberá satisfacer en mayor o menor grado las prestaciones de interconexión, rendimiento, capacidad, disponibilidad e integridad de los datos.

2.2.3.1. Clasificación de los sistemas de almacenamiento

Antes de ver la clasificación más empleada de los sistemas de almacenamiento conviene entender cómo dichos sistemas pueden ofrecer la información.

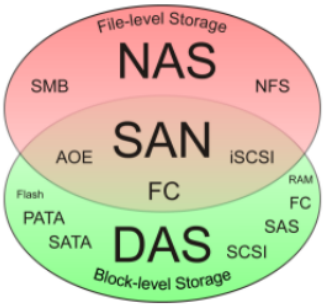


FIGURA 2.10: Relación entre NAS, SAN y DAS

- Nivel de bloque: el almacenamiento se nos monstrará como si fuese un conjunto indexado de bloques. No será responsabilidad del sistema de almacenamiento la ordenación y organización de la información a lo largo de dichos bloques ni tampoco gestionar el acceso concurrente a los mismos.
- Nivel de fichero: el almacenamiento se nos mostrará como un sistema de archivos. En este caso todas las responsabilidades de ordenación y organización recaerán en el sistema de almacenamiento y, dependiendo del sistema, también será responsable de gestionar el acceso concurrente.

Dicho esto, cuando se habla de almacenamiento se clasifica básicamente en tres tipos: DAS, SAN y NAS.

Direct attached storage (DAS) Se trata del sistema más básico de almacenamiento y provee acceso a nivel de bloque al almacenamiento. Un sistema DAS se encuentra conectado directamente a un host (servidor o workstation) sin existir una red de almacenamiento intermedia.

Las tecnologías más usadas en este nivel son:

- **Serial ATA** (SATA): es una evolución del P-ATA (Parallel ATA). Este tipo de tecnología sigue una arquitectura punto a punto del dispositivo con el controlador y permite alcanzar una velocidad de 600MB/s. Existe gran cantidad de dispositivos SATA en el mercado que ofrecen grandes capacidades de almacenamiento a un bajo coste.
- **Serial Attached SCSI** (SAS). El SAS es una evolución de SCSI y SATA. La tecnología está concebida para tener conectados múltiples dispositivos y permite hasta 12Gbs, aunque lo normal a día de hoy es encontrar todavía 6Gbs. Es compatible con SATA, por lo que a una controladora SAS podemos conectar dispositivos tanto SAS como SATA. El coste de los dispositivos es más elevado que los SATA,

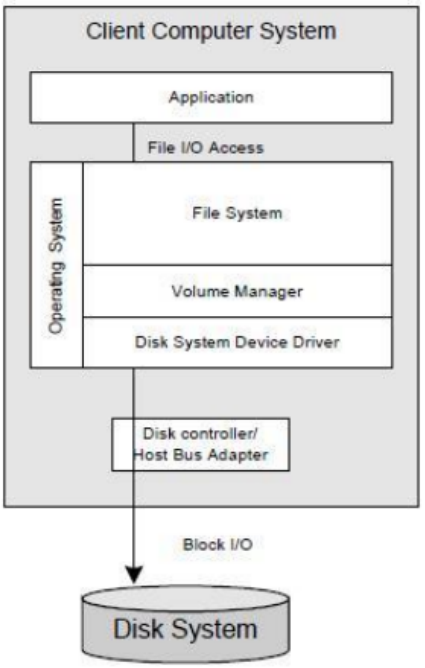


FIGURA 2.11: Arquitectura DAS

pero esto es debido principalmente a una mejor la calidad de los componentes empleados.

Storage area network (SAN) El acceso, del mismo modo que se hacía en DAS, es a nivel de bloque. Sin embargo ofrece un nivel de funcionalidad mucho más alto, ya que permite a múltiples hosts vincularse a múltiples dispositivos de almacenamiento a través de una red. Esto permite una flexibilidad enorme, ya que desvinculamos físicamente el almacenamiento y nos permite asignar los dispositivos a los hosts a nivel de configuración y software.

Las tecnologías más usadas en este nivel son:

- **Fibre Channel** (FC): es una tecnología de alta velocidad usada principalmente para almacenamiento y permite conexiones de hasta 8 y 16G. Está definida por un protocolo multicapa en el cual la última capa ofrece servicio de transporte sobre el que pueden transportarse mensajes SCSI. Para su implementación es necesario disponer de hardware FC dedicado, necesitando en el lado de conexión del servidor una HBA (Host Bus Adapter) dedicada. El mecanismo más general de interconexión de redes de almacenamiento es el uso de switches, por ello será necesario switches especiales que soporten FC. Esta tecnología es la predominante en redes SAN de altas prestaciones pero tiene un coste muy elevado.

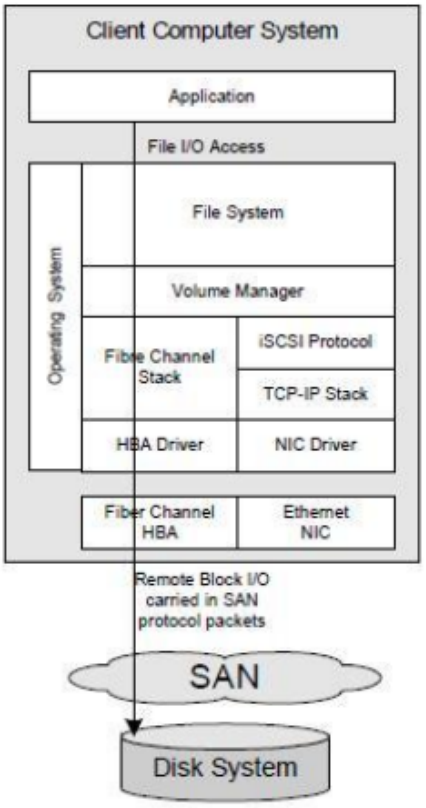


FIGURA 2.12: Arquitectura SAN

- **Internet SCSI** (iSCSI): es una tecnología que permite encapsular el protocolo SCSI sobre una red TCP/IP. Esto hace posible utilizar la tecnología de red existente y por lo tanto, utilizar el mismo equipamiento de red y cableado, incluyendo por supuesto a redes de alta velocidad como ethernet 10G. Por último destacar que el proceso de encapsulación de los comandos SCSI puede suponer una sobrecarga que puede ser mitigada con dispositivos que implementen por hardware esta tarea.

Network attached storage (NAS) Ofrece el nivel más alto y se construye sobre SAN o DAS. Ofrece un acceso a nivel de fichero a través de la red mediante un protocolo específico de compartición de archivos. Este tipo de almacenamiento permite el acceso concurrente de múltiples procesos ubicados en diferentes hosts.

Los protocolos tradicionalmente más comunes son NFS y CIFS, el primero es un protocolo usado generalmente por sistemas UNIX y el segundo por sistemas Microsoft. Sin embargo y como veremos más adelante, en el ámbito HPC y en otros entornos con altos requerimientos han ido apareciendo nuevas soluciones en este nivel.

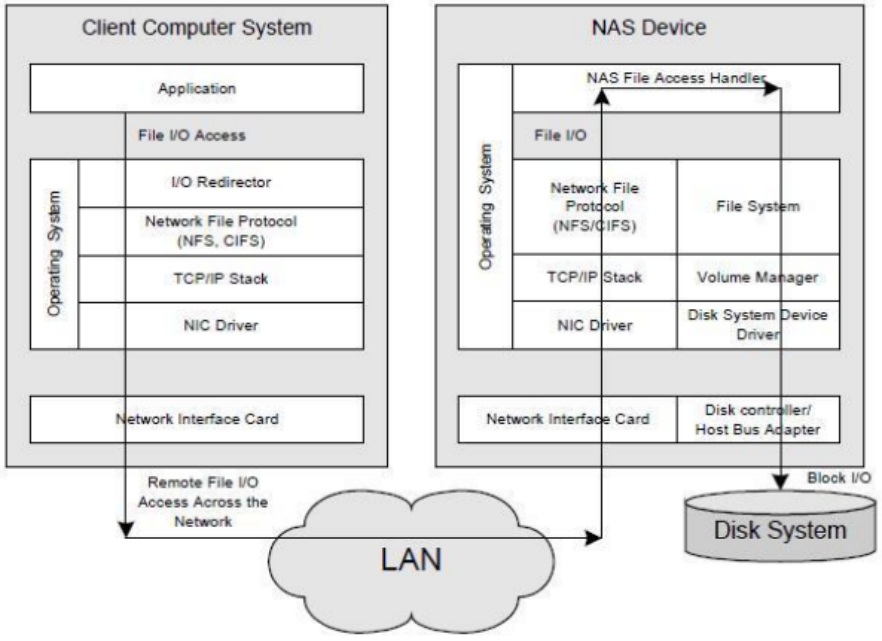


FIGURA 2.13: Arquitectura NAS

2.2.3.2. Tipos de dispositivos de almacenamiento

El último eslabón en un sistema de almacenamiento son los dispositivos en los cuales se almacena la información. Estos dispositivos, además de clasificarlos por el tipo de interfaz de interconexión directa que emplean (SAS o SATA), podremos clasificarlos en función de su tecnología. Bajo esta clasificación encontramos a los tradicionales discos duros magnéticos, las unidades de estado sólido y a los discos híbridos.

- **Disco duro** (HDD, Hard Disk Drive): son los discos duros tradicionales. Se componen de una parte mecánica y una magnética en la que almacenan la información. En función de la parte mecánica y concretamente de las velocidades a la que es capaz de que girar el disco, podemos encontrar versiones de 7.2K, 10K y 15K. Esto representan las revoluciones por minuto que tiene el disco, es decir 7.2K son 7200 revoluciones por minuto, etc. Esto es importante, ya que aunque a mayores revoluciones obtendremos mayores velocidades de acceso, más difícil será aumentar la capacidad de almacenamiento. Es por ello que podemos adquirir discos de hasta 3TB de 7.2K a un bajo coste, mientras que los discos de 10K y 15K a precios razonables los encontraremos por debajo del TB. Desde el punto de vista de prestaciones, cabe destacar que los discos duros responden razonablemente bien ante accesos secuenciales por una cuestión meramente mecánica ya que el brazo del lector no debe desplazarse. Sin embargo, por el mismo motivo mecánico, no se comportan así de bien ante accesos aleatorios.

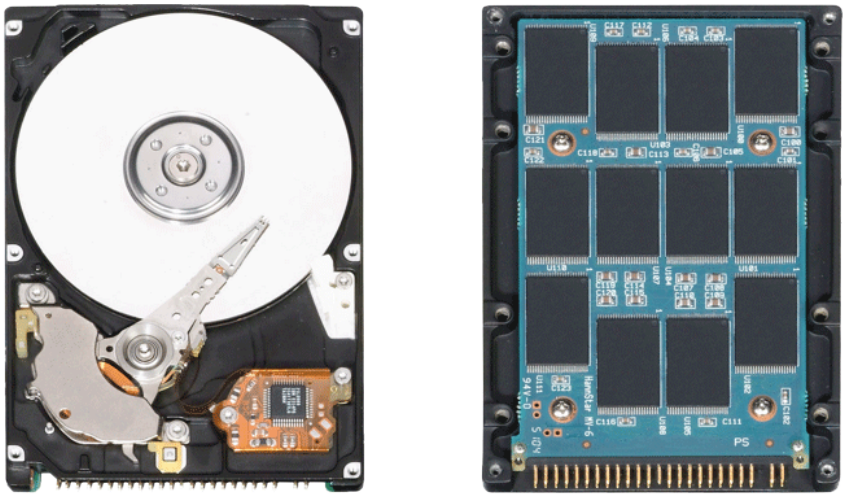


FIGURA 2.14: Disco duro tradicional y ssd

- **Unidad de estado sólido** (SSD, solid-state drive): son dispositivos que usan memorias no volátiles, como memorias flash, para almacenar la información. Al no disponer de elementos mecánicos, ofrecen unos tiempos de acceso muy rápidos tanto a operaciones de tipo secuencial como de tipo aleatorio. La capacidad de almacenamiento es muy inferior en comparación con los discos duros así como su precio es bastante más elevado. El otro gran inconveniente es que, dependiendo del tipo de tecnología empleado, tienen limitado el número de veces que puede escribirse en un mismo sector, causando la degradación del dispositivo al cabo del tiempo.
- **Unidades de tipo híbrido**: vienen a ser una combinación de ambas tecnologías. Su diseño aprovecha los mismos principios de localidad espacial y temporal de los que se benefician las memorias caché. Básicamente disponen de una parte reducida de flash que hace de primer nivel y luego un disco duro magnético tradicional como segundo nivel de almacenamiento.

2.2.3.3. Arrays de discos

Desde el punto de vista de las prestaciones, hasta la llegada de las unidades de estado sólido, el acceso a disco ha sido siempre el cuello de botella de los sistemas. Por ello una estrategia comunmente utilizada ha sido disponer de varios dispositivos en paralelo, de forma que las lecturas y escrituras se distribuyesen a lo largo de varios discos. Esta estrategia proporciona además de unas mejores prestaciones, redundancia en algunos casos de los datos en varios dispositivos, mejorando de este modo el grado de disponibilidad. A esta estrategia se le denomina RAID (Redundant Array of Independent Disks)

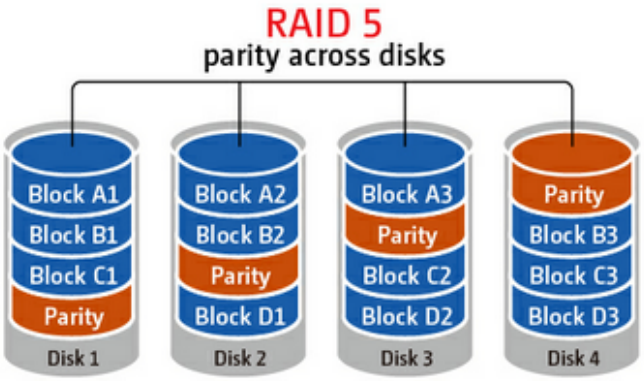


FIGURA 2.15: Distribución de la paridad en un RAID5

y existen múltiples configuraciones a los que se les denomina niveles de RAID. Voy a enumerar los niveles de RAID más empleados:

- **RAID 0**: consiste en distribuir los datos equitativamente a lo largo de varios dispositivos. La capacidad del conjunto así como el rendimiento tanto de lectura y de escritura se multiplica junto con el número de unidades del RAID. El gran problema es que si uno de los discos falla, se pierde toda la información.
- **RAID 1**: consiste en replicar la información de un dispositivo en otro. Esto limita la capacidad del conjunto a la de un único dispositivo pero mejora la disponibilidad de la información. Mientras que las prestaciones de las escrituras no se ven mejoradas, las prestaciones de las lecturas se duplican, ya que pueden realizarse en paralelo.
- **RAID 5 y 6**: son niveles de RAID que distribuyen la información en los discos y almacenan la paridad de la misma para podr ser capaces de recomponer la información ante el fallo de un disco. En el caso del RAID 5 se guarda la paridad una vez en un disco, mientras que en 6 se guardan dos de paridad. La capacidad de información del conjunto son N-1 y N-2, siendo N el número de discos existentes. En ambos casos se mejoran los rendimientos de lectura y escritura, pero tienen penalizaciones en las escrituras si los comparamos con un RAID 0 ya que tiene que escribir la información de paridad.
- **RAID 10**: consiste en un emplear RAID 0 y RAID 1. Es decir, se mantiene un conjunto en paralelo mientras que se hace una réplica exactamente igual en otro conjunto. La capacidad del conjunto es N/2.

En la práctica es común emplear RAID 1 para almacenar el sistema operativo y garantizar así la disponibilidad del servidor. En cuanto a la información, RAID 10 es el tipo de

RAID que ofrece el mejor rendimiento y disponibilidad. Sin embargo, su gran problema es que no podremos usar la capacidad de la mitad de los discos. Por ello suele emplearse RAID 5 o 6. RAID 5 ofrece un mejor rendimiento y un mejor aprovechamiento de la capacidad, sin embargo, con los discos duros actuales tan grandes aumenta la probabilidad de que, habiendo fallado un disco, en el tiempo que transcurren entre que sustituimos el disco y recomponemos el RAID pueda fallar otro disco (son generalmente discos de la misma serie). Por ello, además de disponer de un disco "hot-spare" (un disco de reserva que automáticamente se utiliza en el momento de detección de fallo de un disco), se ha empezado a generalizar la utilización del RAID 6 como sustituto al RAID 5.

2.2.3.4. Sistemas de archivos paralelos

El mismo principio de distribución de la información en distintos medios para poder realizar un acceso en paralelo que vimos para los arrays de discos, es perfectamente aplicable a los sistemas de archivos de red. En un sistema de archivos tradicional de red como pueden ser NFS y CIFS, el acceso a la información se realiza en última instancia a través de un único servidor que presentará el sistema de archivos a los clientes. Esto supondrá que, en determinados escenarios, el servidor pasará a ser el cuello de botella en el acceso al almacenamiento. Para solucionar este problema surgieron los sistemas de archivos paralelos que básicamente distribuyen la información a lo largo de distintos servidores, permitiendo que el acceso a la información se realice en paralelo. Los sistemas de archivos paralelos en el ámbito del mundo HPC más usados son:

- **Lustre**: es un sistema de archivos paralelo que permite crear un volumen unificado a partir de diferentes espacios de disco heterogéneo accesible por conexiones TCP/IP. Estos espacios de disco pueden ser SAN, NAS o DAS. Se compone de un servidor (o clúster de servidores) llamado *metadata server* (MDS) que contendrán un *metadata target* (MDT) por filesystem y en el que almacenarán los metadatos (nombres de ficheros, permisos y file layout de los ficheros). Después estarán los servidores llamados *object storage servers* (OSS) que almacenarán la información. Para finalizar, tendremos los clientes que serán los que harán uso del almacenamiento y que deberán tener el software cliente instalado. Este sistema está muy extendido, aunque es sobradamente conocida la dificultad de su puesta en marcha.
- **pNFS**: se trata de un protocolo y viene a ser el nuevo estándar que reemplace al vetusto NFS. Este sistema proporciona funcionalidades de sistema de archivos paralelo y está inspirado en Lustre. Sin embargo, pretende ser familiar a los

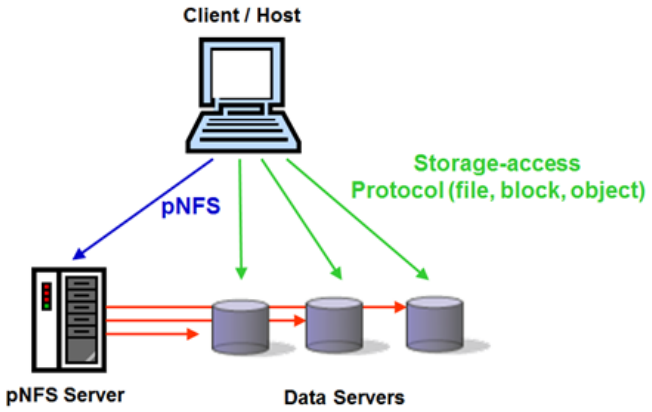


FIGURA 2.16: Diagrama pNFS

administradores de NFS, disponiendo de la misma sintaxis y herramientas de administración. Numerosos fabricantes ya dan soporte en sus soluciones de almacenamiento propietarias a pNFS y actualmente ya está disponible la versión 4.1. Sin embargo, esta solución lleva poco tiempo en el mercado y existe todavía muchas reservas por parte de los administradores.

2.2.3.5. Sistemas de archivado

En ocasiones, los volúmenes de información llegan a ser enormes y es necesario recurrir a los sistemas de archivado. Dichos sistemas hacen uso de medios más baratos de almacenamiento como las librerías de cinta y nos permitirán archivar la información, es decir, moverán la información desde nuestro sistema de almacenamiento a los dispositivos de cinta. Sin embargo, la información estará siempre disponible y podrá desarchivarse (a una velocidad muchos órdenes de información más lenta) para su uso. Estos sistemas de archivado suelen ofrecer políticas de archivado automática (según tiempo de acceso, tamaño, etc) y ofrecer mecanismos de desarchivado transparente al usuario (como la creación de ficheros stubs en el filesystem). Estos sistemas obviamente requieren del hardware apropiado y suelen proporcionarse como soluciones integradas. Algunos de los proveedores de este tipo de soluciones son Oracle o Netapp.

2.2.4. Software

2.2.4.1. Sistema operativo

El sistema operativo predominante en el mundo HPC es GNU/Linux. Se encuentra instalado en el 97 % de los 500 supercomputadores más potentes del mundo. Pero su éxito

FIGURA 2.17: Librería de cintas

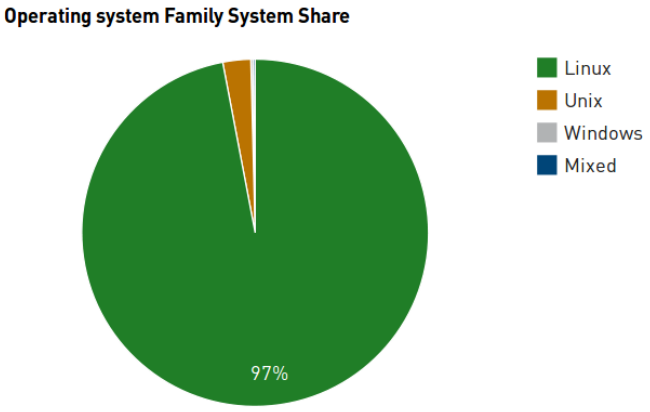


FIGURA 2.18: Linux en el Top500

no sólo se da entre los grandes, como ya he comentado anteriormente, ha contribuido a ser una de las causas del éxito y la popularización de los HPC.

Pero a pesar de este predominio absoluto, existe diversidad en lo que a distribución se refiere y es que el ser una plataforma libre ha contribuido a que existan multitud de distribuciones disponibles. No sólo eso, sino que permite además que sea posible construir una distribución a medida para un supercomputador determinado.

No voy a entrar en el análisis de distribuciones porque bien podría llevar un libro entero. Simplemente comentaré que en ocasiones puede ser necesario disponer de soporte comercial y entre las distribuciones más populares con dicho soporte tenemos a RedHat, SuSE y Ubuntu. Es importante conocer que aunque no necesitemos de dicho soporte, es muy común que el hardware sólo esté soportado y certificado por el fabricante para estas distribuciones y ofrezca únicamente drivers binarios para determinadas versiones de las distribuciones. Este hecho hace que en ocasiones la elección del sistema operativo esté condicionada por el hardware que escojamos.

2.2.4.2. Herramientas de desarrollo

Desde la perspectiva del usuario, disponer de las herramientas necesarias para poder compilar y ejecutar sus programas es el aspecto más importante del clúster.

Shell y entorno La puerta de acceso a los servicios del clúster para los usuarios son los nodos de login. En estos nodos los usuarios dispondrán de una shell predeterminada que será ejecutada tras el proceso de login. Generalmente en los HPC suelen estar disponibles varias shell y es posible que los usuarios soliciten al administrador el cambio de la shell que emplean por defecto. Suele ser habitual disponer de:

- Bourne (sh)
- Korn (ksh)
- C shell (csh)
- Extended C-shell (tcsh)
- GNU Bourne-again shell (bash)
- Z shell (zsh)

Con respecto a esto, simplemente comentar que con los años bash se ha consolidado como la shell por defecto de todas las distribuciones, pero la shell tcsh sigue siendo muy usada en el mundo científico.

Durante su inicialización la shell cargará una serie de variables de entorno, algunas de estas variables estarán relacionadas con cómo se encontrarán los binarios o cómo el enlazador dinámico encontrará las librerías. Será posible tener múltiples versiones en el sistema de un binario o librería y, mediante la gestión de dichas variables, indicar qué versión deseamos utilizar. Es posible que los usuarios gestionen esto a mano, pero suele ser habitual dotar de un software específico para realizar este tipo de tarea. A este software se le suele denominar "módulos de entorno" y es muy popular en los HPC. Este tipo de software permite definir módulos que contienen la información necesaria para configurar el entorno de la shell para determinadas aplicaciones, librerías y versiones. Existen varias implementaciones:

- environment-modules-c
- environment-modules-tcl

- lmod

Aunque cada una tiene su propio lenguaje de definición de módulos, todas ofrecen la misma interfaz de uso desde el punto de vista del usuario.

Editores, utilidades, herramientas de construcción, control de versiones Suele proveerse de un conjunto de editores para que los usuarios puedan realizar modificaciones en sus programas. Entre los editores más empleados: emacs, vi y vim, nano, nedit, pico, joe

No pueden faltar las típicas utilidades coreutils, bc, par, grep, gawk, findutils, more, less, screen, zlib, wget...

Ni por supuesto las herramientas que ayudan a la construcción del software como: make, cmake, qmake, autoconf, automake, autotools...

O herramientas de control de versiones: rcs, cvs, subversion, git...

Compiladores e intérpretes Aunque los programas pueden ser compilados fuera del clúster y copiados posteriormente para su ejecución, suele ser habitual compilarlos en el mismo clúster. Esto se hace tanto por comodidad a la hora de disponer de compilador y librerías como porque el compilador pueda realizar optimizaciones en tiempo de compilación para la arquitectura del clúster. Los lenguajes compilados en el mundo científico más extendidos son C, C++ y FORTRAN. Por ello se suelen incluirse los compiladores de estos lenguajes:

- Compiladores de GNU
- Compiladores de Intel

Los compiladores de GNU son libres y ofrecen un muy buen rendimiento en general pero los de Intel (de pago) generan código más optimizado para su hardware.

Cabe mencionar también a Java que poco a poco está empezando a usarse en este contexto, por lo que también se hace necesario al menos la disponibilidad de una máquina virtual Java.

Además de emplear lenguajes compilados suele ser muy habitual recurrir a lenguajes de scripting que, aunque ofrecen un peor desempeño, permiten aumentar la productividad. Tradicionalmente el lenguaje más usado en el mundo científico para el análisis de datos

ha sido Perl, pero en los últimos años ha venido siendo reemplazado por Python. De manera adicional, existen múltiples herramientas que también ofrecen lenguajes de script y proveen de una alta productividad al encontrarse enfocados en determinados casos de uso. Cabe mencionar Matlab o IDL en el aspecto matemático o R en el aspecto estadístico. Podría resumirse que un HPC en este aspecto debería contener:

- Perl
- Python
- Ruby
- Tcl, tk y expect
- R
- Matlab y/o Octave
- IDL

Debuggers y profilers De forma complementaria a los compiladores suele ser muy útil disponer de herramientas que ayuden a depurar el código, dejándolo libre de bugs (al menos intentarlo). También es útil disponer de herramientas que ayuden a perfilarlo, permitiendo conocer aspectos sobre el consumo de memoria, accesos a discos, etc que nos ayuden a identificar el mal uso de recursos y con ello mejorar el desempeño de los programas.

Del mismo modo que teníamos los compiladores de GNU e Intel tenemos:

- GDB: el debugger de GNU
- IDB: el debugger de Intel

Además existen algunos propietarios como el conocido TotalView.

Con respecto a perfiladores, existe un amplio abanico:

- gprof: GNU profiler
- Valgrind: análisis de memoria
- Intel Vtune Analyzer: herramienta de optimización del rendimiento
- UNITE: debugger y análisis de rendimiento de MPI y OpenMP

Librerías de paralelización En el ámbito de la paralelización tenemos dos tipos de arquitecturas paralelas. La primera, llamada de memoria distribuida, considera que cada procesador tiene su propio espacio de memoria RAM y no puede ser accedido por otros procesadores. En el segundo tipo, llamada de memoria compartida, habilita a los procesadores a acceder al mismo área de memoria RAM de manera simultanea y transparente.

La implementación de la arquitectura de memoria distribuida basa su concepto en el paso de mensajes. Existe un intento de estandarización del paso de mensajes en unas librerías que es llamado MPI (Message Passing Interface). Existen varias implementaciones, las más extendidas en la actualidad son MPICH, OpenMPI e Intel MPI.

En cuanto a la implementación de la arquitectura de memoria distribuida, aunque es posible el uso de los mecanismos dados por el sistema operativo como los PosixThreads, es muy común el uso de OpenMP (Open Muli Processing). Esta librería proporciona una abstracción que oculta muchos detalles de la complejidad del desarrollo de aplicaciones paralelas. Tango GCC como Intel proveen una versión de OpenMP junto al compilador.

Por último, comentar que ambas librerías suelen ser usadas al mismo tiempo, permitiendo la combinación de ambas arquitecturas de paralelismo.

Librerías matemáticas Existen multitud de librerías en el ámbito matemático que por supuesto los usuarios demandarán en un HPC. Entre las más destacadas encontramos:

- FFTW: librería para realizar trasformadas de Fourier.
- PETSc: es una suite de librerías científicas
- LAPACK: librería con rutinas de álgebra lineal.
- ScaLAPACK: un subconjunto de rutinas paralelizadas de LAPACK.
- BLAS: son librerías de álgebra lineal. Existen varias implementaciones:
  - ATLAS
  - MKL: implementación de Intel
  - OpenBLAS
- GSL: librería de GNU con una amplia variedad de rutinas matemáticas
- SPRNG: generador de números aleatorios.

2.2.4.3. Software de gestión de recursos

Finalmente encontramos el software de gestión de recursos, que será el software que se encargará de monitorizar el estado de los recursos computacionales y planificar y asignar recursos necesarios a los trabajos de acuerdo a nuestras políticas de uso.

Existen diversos gestores de recursos y todos tienen características muy similares e incluso parecidos desde el punto de vista de utilización para los usuarios. Generalmente todos admiten trabajos en forma de shell script en los que se suministran atributos especificando los recursos que se necesitan, la prioridad, etc y la invocación del propio trabajo.

Los más populares son:

- **PBS** (Portable Batch System): es un sistema desarrollado por la NASA a principios de los 90. Actualmente existe version comercial y version libre. La version libre se llama Torque.
- **HTCondor**: es un sistema libre que se desarrolló con la capacidad de ser desplegado en estaciones de trabajo para detectar el tiempo que estuviesen en idle y aprovecharlo para realizar tareas de cómputo.
- **GridEngine**: su origen es una compañía llamada Gridware que fue adquirida por Sun y liberó el código llamándolo Sun Grid Engine. Posteriormente con la adquisición de Oracle pasó a ser Oracle Grid Engine y posteriormente vendido a Univa. Existen dos versiones libres de gridengine, la primera es Gridscheduler, derivada de la última versión que liberó Univa. La otra es Son Of Grid Engine, que fue un fork posterior.

2.3. Gestión de un clúster HPC

2.3.1. Gestión de los usuarios

La mayoría de los gestores de tareas, tras seleccionar el/los nodos objetivos, proceden a lanzar los procesos necesarios. La ejecución de dichos procesos a nivel de sistema operativo se llevan a cabo como si fuesen el usuario que mandó la tarea a ejecutar, esto es, ejecutará los procesos con el uid del usuario que solicitó ejecutar la tarea. Esto implica que deberemos de tener la información de los uid de usuario consistente en todos los nodos, es decir, que debemos disponer de una base de datos común de usuarios. GNU/Linux provee de un sistema muy flexible llamado nsswitch que permite que la

base de datos de usuarios que utiliza el sistema pueda ser cambiada y extendida más allá del clásico /etc/passwd.

Existen múltiples plugins nsswitch que permiten los más diversos orígenes. Tradicionalmente en el mundo UNIX la base de datos de usuarios centralizada que se ha venido usando ha sido el NIS y NIS+ de Sun. Este sistema ha demostrado su efectividad a lo largo del tiempo pero tiene serias carencias de flexibilidad y de seguridad. Por ello ha venido siendo reemplazado por directorios LDAP, mucho más flexibles, seguros y escalables, aunque algo más difíciles de desplegar. Las soluciones en este terreno son múltiples: OpenLDAP, Fedora Directory Server, Active Directory, etc..

2.3.2. Gestión del software

Todas las distribuciones GNU/Linux disponen de su propio sistema de empaquetado de software. Un paquete de software no sólo contiene ficheros ejecutables, librerías, documentación, etc, también posee información acerca de versiones, dependencias, incompatibilidades y hasta opciones de configuración. Además de las herramientas que gestionan e instalan los paquetes, existen herramientas de más alto nivel que gestionan la resolución automática de las dependencias y conflictos. Esto es posible gracias a que se crean unos repositorios en los que se almacenan los paquetes de software y que estarán ubicados en algún medio de distribución de los soportados como dvds, servidores http, etc.

Para la gestión de los repositorios en un clúster puede optarse por tres estrategias:

- No tener repositorios locales y usar los de internet: los equipos descargarán el software directamente de algún repositorio de internet. Esto hará necesario que todos los nodos descarguen a través de internet el mismo paquete, siendo muy ineficiente desde el punto de vista del tráfico. También será dependiente del nivel de congestión del repositorio y a las caídas en el servicio que pudieran ocurrir. Esta práctica suele estar desaconsejada.
- Disponer de un proxy especializado de repositorios: se usará un servidor local que hará de proxy, de este modo el paquete sólo se descargará una vez de internet y todos los nodos bajarán de manera local el paquete. Además de la eficiencia que supone, esta solución es muy fácil de poner en marcha. El gran problema de esta solución es que no tenemos un control completo del repositorio, ya que está en manos de quien controla el repositorio origen. Esto puede hacer que se introduzcan nuevas versiones que puedan "romper" nuestra infraestructura. En el caso de las distribuciones basadas en Debian (como Ubuntu) es muy usado apt-cacher-ng.

- Disponer de un mirror de los repositorios: en esta solución se hará una copia en espejo de la información de un servidor de internet. Ofrece un mayor control que la versión de proxy ya que podremos dejar "congelado" nuestro repositorio y realizar las sincronizaciones cuando hayamos probado que todo funciona correctamente. El gran problema es que requiere de un mayor mantenimiento y necesitamos la capacidad para albergar todos los paquetes del repositorio. En el caso de Debian está muy extendida la solución apt-mirror.

Además de los repositorios ofrecidos por las distribuciones, existen repositorios de terceros que podremos emplear. También es posible crear nuestro propio repositorio en el que publiquemos software que no se encuentre en la distribución. Una solución sencilla para crear nuestros propios repositorios en Debian es Reprepro.

El sistema de paquetes y repositorios es una solución estupenda de propósito general. Sin embargo, lo que es bueno como propósito general puede no serlo en determinados ámbitos. Así como el sistema de paquetes y repositorios va a solucionar la instalación básica de utilidades del sistema, no va a ser suficiente para resolver en el ámbito del desarrollo:

- los paquetes que realizan las distribuciones no están específicamente optimizados
- no está exento de dificultad realizar paquetes y pueden llevar un gran esfuerzo, especialmente cuando no se llevan buenas prácticas de desarrollo (algo característico en el ámbito científico)
- los paquetes son un sistema estupendo para actualizar las versiones y ofrecer alternativas a programas, pero en el ámbito científico lo que se desea es mantener versiones antiguas y diversas de forma paralela, de modo que siempre sea posible reproducir los resultados obtenidos en alguna simulación

Estos motivos, hacen necesario un sistema adicional que complemente al sistema de paquetes para la gestión del software de desarrollo y librerías. En este aspecto ha tomado mucha relevancia en el mundo HPC la solución Easybuild. Dicha solución dispone además de integración con los módulos de entorno y existen builds para gran parte del software necesario en los HPC.

2.3.3. Gestión de la configuración

Cuando disponemos de muchos nodos, se hace necesario tener algún sistema que nos permita gestionar la configuración de forma eficiente. Un buen sistema que permita la gestión de la configuración debe al menos:

- poder realizarse de forma centralizada en algún lugar
- disponer de un control de versiones que permita tener un seguimiento y poder revertir configuraciones
- escalar fácilmente el número de nodos

Dicha gestión puede llevarse a cabo de forma "artesanal" empleando algún software de gestión de versiones, scripts personalizados y algún servidor NFS o RSYNC para gestionar los archivos de configuración. Sin embargo, el emergente mercado del cloud computing junto a tecnologías de virtualización han impulsado el surgimiento de herramientas específicas de gestión de la configuración. Actualmente hay que destacar herramientas como Puppet, Cfengine o Cheff.

2.3.4. Despliegue del sistema operativo

Antes de poder gestionar el software o la configuración de los nodos es necesario que estos dispongan de un sistema operativo. La instalación del sistema es uno de los aspectos que también se debe automatizar, ya que ir uno a uno en cientos de nodos es inviable. El proceso suele iniciarse mediante un arranque por red vía PXE y se requiere por tanto montar la infraestructura necesaria. El despliegue automatizado de un sistema operativo puede realizarse básicamente de dos maneras:

- Clonación de una imagen del sistema: una opción ampliamente utilizada, es realizar una instalación básica del sistema operativo y el software necesario sobre uno de los nodos y posteriormente tomar una imagen del sistema del nodo. La imagen puede ser tomada a nivel de bit o a nivel de sistema de archivos, en cualquier caso se dispondrá de una imagen que podrá ser instalada posteriormente en cualquier otro nodo. La distribución de las imágenes puede realizarse de múltiples formas, desde un simple servidor NFS hasta sistemas de distribución multicast o p2p. El inconveniente de las imágenes suele ser que en algunos sistemas está vinculado al hardware del que se ha tomado la imagen, también se hace necesario realizar algunos cambios posteriores a la instalación de la imagen y además se requiere el mantenimiento de una imagen "master". Existe multitud de software que permite esto, como clonezilla o fog.
- Instalación desatendida del sistema: los instaladores de los sistemas operativos ofrecer un mecanismo de instalación desatendida de los sistemas. Esto básicamente consiste en proporcionar un fichero de respuestas en la fase de instalación que permite la automatización del despliegue del sistema operativo independientemente

del tipo de hardware. En caso de Debian, el debian installer puede utilizar un fichero de respuesta al que llama preseed que puede localizar tanto en un cd como por red. Se han construido frontends que permiten crear una infraestructura. Uno de los más conocidos es fai.

Dependiendo de la infraestructura a desplegar, el hardware, etc se optará por uno u otro mecanismo. En máquinas virtuales suele emplearse algún mecanismo basado en imágenes ya que el hardware es idéntico y los propios hipervisores disponen de mecanismos de clonación de discos duros. En caso de máquinas físicas, es muy cómodo realizar instalaciones desatendidas ya que no requiere el mantenimiento de ninguna imagen "master".

2.3.5. Monitorización

La monitorización es uno de los aspectos que a menudo se suele pasar por alto y se deja en segundo plano, sin embargo es uno de los aspectos más importantes ya que nos permite medir y garantizar la disponibilidad.

Hay un dicho que viene a decir que "si un servicio no está monitorizado no es un servicio", ya que si no lo monitorizas ni si quiera puedes garantizar que lo estás proporcionando. Por ello deberemos estar continuamente controlando de forma automática que nuestros sistemas están en funcionamiento y ser capaces de reaccionar lo más rápido posible ante un fallo en los mismos. Pero la monitorización no sólo es capaz de mejorar la disponibilidad del servicio vía reactiva, también puede hacerlo de forma proactiva analizando tendencias y anticipando una denegación del servicio por falta de capacidad. Además de mejorar la disponibilidad, la monitorización nos proporcionará información tanto a nivel operativo, permitiendo realizar estimaciones de tiempo y recursos, como a nivel gerencial, al permitirnos calcular el retorno de la inversión o prever inversiones futuras.

Por estos motivos será necesario monitorizar todos los aspectos del clúster: desde el entorno físico en el que se encuentra hasta los servicios de los que depende, nodos de cómputo, red, almacenamiento, etc. No voy a entrar en detalles acerca de la monitorización, porque bien se podrían llenar libros enteros. Simplemente voy a comentar lo que creo imprescindible que debe disponer un sistema de monitorización:

- Datos en "tiempo real" y archivado de histórico: es muy conveniente que podamos visualizar de forma gráfica la información que consideremos relevante (consumo de cpu, memoria, tráfico de red, etc) y que dispongamos de un histórico con el que la podamos contrastarla o realizar análisis de tendencias.

- Gestión de eventos y alarmas: debemos ser capaces de definir los más diversos eventos, que pueden ser desde que una máquina ha superado un determinado umbral de uso continuado de CPU hasta que un cable ha sido desconectado de un switch. Y responder a dichos eventos tratándolos de forma automatizada, como podría ser establecer una conexión de backup o bien disparando una alarma que informe del evento.

Todo esto es posible debido a que se realiza una recolección de información que se realiza o bien vía chequeos con los más diversos agentes o vía algún protocolo orientado a esta función como puede ser SNMP. El éxito en la elección del software dependerá de la facilidad con que podamos integrar nuestros sistemas, por lo que será un factor muy a tener en cuenta.

Entre el software más enfocado en la recolección de información podemos encontrar a Cacti, Munin o al líder en HPC: Ganglia. En cuanto a la gestión de eventos y alarmas, Monit es un sistema que funciona muy bien en respuesta proactiva a eventos en un host, sin embargo el líder indiscutible es Nagios. Una herramienta de monitorización integral que también hay que tener en cuenta es Zabbix.

2.3.6. Distribuciones específicas HPC

Como se ha podido ver, las necesidades de un HPC son un tanto particulares. La mayoría de las distribuciones incluso carecen de algunos paquetes clave para el funcionamiento de un clúster. Por ello surgieron distribuciones específicas "todo en uno" que incorporan además del software necesario, instaladores que permiten tener todo el conjunto del software integrado y funcionando con muy poco esfuerzo. Algunas de las distribuciones son:

- Redhat Enterprise for HPC
- Rocks Clusters
- Oscar

Este tipo de distribuciones específicas son un buen mecanismo para implementar un clúster para personas con un perfil científico que disponen de un bajo conocimiento técnico e incluso puede serlo para personas que, teniendo un buen conocimiento técnico, simplemente quieren tener una solución con el mínimo esfuerzo. Sin embargo este tipo de soluciones son bastante rígidas y no permiten realizar fácilmente integraciones con otro

software. Se produce además una pérdida por parte del administrador de sistemas sobre cómo funciona y cómo están integrados los componentes que lo forman, redundando en su capacidad de resolver problemas.

Capítulo 3

Análisis y especificación

3.1. Análisis preliminar

3.1.1. Entrevistas y reuniones

Una vez estudiado el Estado del arte ya tenía mucho más claro el estado de la tecnología. El siguiente paso sería obtener más información acerca del tipo de problemas que se trataban de resolver en CEFCA y cómo podrían resolverse mediante el uso de un HPC. Para ello en primer lugar identifiqué a las personas clave que podrían proporcionarme más información, clasificándolas en varios tipos:

- Usuarios potenciales del sistema: personas que demandaban las mayores necesidades de cómputo y que iban a ser los usuarios finales.
- Usuarios con experiencia en HPC: personas que habían trabajado y/o tenían acceso a otros sistemas HPC.
- Jefes

Para el primer tipo de usuario, busqué personal de las dos líneas de investigación que se llevan a cabo en el CEFCA: cosmología y evolución de galaxias. El objetivo principal era obtener información principalmente del tipo de herramientas que usaban, caracterizar a muy alto nivel el tipo de problemas que iban a tratar de solucionar, conocer la demanda de recursos de dichos problemas, el grado de paralelización y conocer cual era la expectativa de "experiencia de uso" que tenían. De estas entrevistas pretendía obtener un listado de la herramientas software necesarias, realizar una estimación de los recursos computacionales, anticiparme a posibles problemas de capacidad y obtener las expectativas de uso.

Del segundo tipo de usuario, buscaba principalmente conocer cómo eran las soluciones en otros HPC, cuál era su forma de trabajar, si era posible iniciar sesión en un HPC para ver su entorno, cómo se encontraba estructurada la información, las colas, etc. De estas entrevistas pretendía obtener modelos en los que "inspirarme" y obtener los típicos casos de uso desde el punto de vista de un usuario de un clúster HPC.

De mis entrevistas con los jefes esperaba obtener, además de responder a la cuestión fundamental ¿cuáles eran sus expectativas con el HPC?, la respuesta a cuestiones que marcarían las principales especificaciones y restricciones. En primer lugar, cuál iba a ser el ámbito del HPC y qué personas tendrían acceso(si privado, privado con colaboraciones, de acceso público, integrado con otros grids, etc). La siguiente cuestión sería relativa a la seguridad en cuanto a qué información se podría tener acceso desde el HPC y si deberían fijarse algunas políticas de seguridad especiales. Para finalizar, quería saber si debería existir algún tipo de política en cuanto a tiempos de uso, reservas a proyectos, etc.

Para la realización de las entrevistas realicé un pequeño guión de entrevista que adjunto en el anexo.

Además de las entrevistas tuvieron lugar una serie de reuniones con el objetivo de definir el equipamiento necesario. A dichas reuniones asistieron dirección, los principales responsables de investigación, personal científico con experiencia en HPC, personal de la UPAD y la persona encargada de poner en marcha el HPC (es decir, yo).

3.1.2. Tipos de problemas a resolver y necesidades de cómputo

Aunque el uso del HPC mayormente va a estar enfocado en la resolución de unos tipos de problemas que comentaré a continuación, se hacía hincapié en que también había más tipos de problemas que se querrían resolver y en la necesidad en un futuro de procesar también imágenes. En general, lo que se quería es que el HPC fuese lo suficientemente genérico, tuviese las herramientas con las que trabajaban, pudiesen obtener resultados en un tiempo mucho más reducido y ejecutar programas con tamaños de entrada más grandes y mayor de complejidad de lo que eran capaces de ejecutar sus estaciones de trabajo. Lo cierto es que me hubiese gustado obtener métricas caracterizando de manera empírica los problemas, pero cada científico tenía sus propios programas, de diferente tipo, con diferente lenguaje, etc. Pronto abandoné la idea, lo más que pude obtener al respecto fue una "percepción" de lo que costaban ejecutarse y lo que consumían algunos programas en determinadas estaciones de trabajo, con lo que en cierto modo tenía una comparativa relativa. Otra medida que me hubiese gustado obtener era una estimación cuantitativa del uso del HPC pero tampoco pude obtenerla, en general cuando consultaba sobre esta cuestión, los futuros usuarios no podían realizar una estimación

pero afirmaban que tenían muchos programas y datos que procesar. Para tener un cálculo sobre el que trabajar, establecía una relación entre las horas semanales y el número de usuarios del HPC.

Desde el punto de vista del área de cosmología, recogí que se van a realizar principalmente dos tipos de problemas:

- Análisis estadístico de datos del OAJ: tendrán por objetivo medir las propiedades fundamentales del universo. En este tipo de problemas se llevará a cabo una carga en memoria de los datos al principio y una escritura al final y serán mayormente intensivos en cómputo. La entrada de datos estará entre 100GB y 200GB con una salida del orden de MB y requerirá de manera adicional de catálogos que estarán alrededor de los 100GB.
- Simulaciones numéricas de procesos físicos: las más grandes simulaciones se llevarán a cabo en supercomputadores (como simulaciones de big-bang), pero algunas más pequeñas podrán ser realizadas en CEFCA. El requerimiento principal de este tipo de problemas es la RAM. La entrada de datos serán centenares de archivos de 5GB que serán leídos y escritos cada media hora. Para este tipo de problemas se hace uso intensivo de MPI, siendo necesarias transferencias de memoria.

Desde el punto de vista de del área de evolución de galaxias los siguientes:

- Spectral Fitting: el objetivo principal es estimar los tipos de galaxia y las distancias existentes entre ellas (redshift). Este tipo de problemas se llevarán a cabo sobre una entrada de datos de 100 objetos (del orden de KB) y se realizará un contraste iterativo contra un conjunto de modelos que serán cargados de un fichero de 100MB. Tras realizar el ajuste, se guarda el mejor ajuste (del orden de Kb). Sin embargo, se está trabajando en almacenar también el cálculo de probabilidades de ajuste, por lo que la salida para 100 modelos es de aproximadamente 30MB. Para paralelizar este tipo de problemas tan sólo es necesario dividir la entrada de datos para alimentar múltiples procesos. El principal requerimiento de este tipo de problemas es su consumo de RAM.
- Cálculo de photo Z: se trata de un subconjunto del spectral fitting y consistirá en la estimación del parámetro Photo Z.

Tras este análisis saqué mis propias conclusiones. En general, el factor predominante en este HPC iba a estar en el cómputo. Estaba claro dado el tipo de problemas y por algunas de las comparaciones relativas que teníamos de las ejecuciones, que la mayor

necesidad eran cores y memoria RAM. También pensé que podría ser interesante el uso de GPUs y/o coprocesadores, especialmente si en un futuro se trabaja sobre imágenes, pero los programas de los usuarios no estaban optimizados para el aprovechamiento de estas tecnologías y, aunque había una predisposición a aprender por parte de los usuarios, no tenían el nivel técnico necesario para su aprovechamiento. Pude comprobar que la mayoría de los programas realizaban sus cálculos en memoria y que únicamente se iban a disco durante el proceso de lectura y de escritura, siguiendo en ambos casos un patrón de acceso secuencial. Dado que la mayoría de los programas demoraban muchísimo más tiempo de procesado que leyendo o escribiendo de disco, la probabilidad de concurrencia por el almacenamiento iba a ser baja. La excepción a esto eran programas que hacen uso intensivo de catálogos, que por su tamaño no pueden ser cargados en RAM y su acceso es aleatorio. Por lo tanto, en general no existen grandes requerimientos pero será necesario disponer de una parte rápida para el acceso a catálogos. En lo que respecta a las necesidades de interconexionado entre nodos, pude comprobar que había una mayoría de programas que usaban el paradigma de sistema de memoria compartida contra una minoría que usaba el paradigma de memoria distribuida. Dicho de otra manera, la mayoría de los programas utilizaban paralelización o bien a nivel de entrada de datos, dividiendo el problema y ejecutando múltiples procesos con diferentes entradas de datos o bien utilizaban paralelización multihilo. Sin embargo, en el área de cosmología sí se hacen programas distribuidos que emplean MPI y lo usan tanto para señalización como para transferencias de memoria. Por lo tanto, habrá que buscar una solución de compromiso. En resumen:

- el factor predominante es el cómputo
- será necesaria una buena cantidad de cores y de memoria RAM por core
- puede ser interesante la adquisición de GPUs pero no debería suponer una gran inversión
- para el almacenamiento hay necesidades de capacidad ya que se van a almacenar imágenes y catálogos. Desde el punto de vista del rendimiento, en general no hay grandes exigencias pero sí hará falta una parte rápida para el acceso a catálogos
- en general no hay un gran requerimiento de red, sin embargo para un rendimiento óptimo de los programas de cosmología será necesario recurrir a Infiniband o 10G.

3.2. Proceso de definición de la solución

Como comenté con anterioridad, la toma de decisiones con respecto a la selección del equipamiento tuvieron lugar durante las reuniones citadas. El objetivo final de dichas reuniones era definir unas especificaciones generales del equipamiento necesario para poder sacarlo a licitación pública. Con un presupuesto muy aproximado de partida, había que tomar una serie de decisiones:

- Lugar en el que se iba a instalar y acondicionamiento necesario
- Diseño básico del clúster
- Especificación de los nodos
- Especificación del almacenamiento
- Especificación de la red
- Especificación del software

Para la selección del lugar no hubo ningún tipo de dudas: el laboratorio de informática. El laboratorio tiene su ubicación en la planta sótano y se diseñó para que pudiese albergar físicamente hasta cuatro racks con sus limitaciones de capacidad por rack (debido a limitaciones del lugar, burocráticas y de presupuesto). El objetivo principal de este lugar es alojar todos los equipos de comunicaciones, cortafuegos, servidores de servicios internos de oficina, servidores en DMZ, EDAM (el sistema que servirá en la DMZ a la comunidad científica los catálogos) y otros sistemas informáticos adicionales, como el clúster HPC. En el momento en que se produjeron las reuniones se estaban realizando tareas de acondicionamiento del lugar, la instalación del sistema de climatización y los cuadros eléctricos. En cuanto al rack en el que podría ser instalado, sería un rack Dell quipado con PDUs que en aquellos momentos había quedado libre. Por lo tanto, no sería necesaria la adquisición de un rack, PDUs, ni realizar labores adicionales de acondicionamiento. Esto era una buena noticia ya que dejaba más dinero disponible para cómputo.

Como se vio durante el estado del arte, existen decisiones de compromiso que hay que tomar a la hora de elegir una solución HPC. La decisión que tomemos deberá tener principalmente en cuenta el tipo de problemas que el HPC tratará de resolver. La definición de dichos problemas determina los requerimientos de los principales factores: cómputo, almacenamiento y red. Esto es necesario, ya que si se usa mucho dinero en un factor, se tiene poco dinero para los otros dos.

Como se ha visto con anterioridad, el clúster HPC no va a tratarse de un sistema que requiera grandes prestaciones desde el punto de vista del almacenamiento: no se trata de una nueva UPAD. Por lo tanto no iba a ser necesaria una solución dedicada de almacenamiento, usar sistemas de ficheros distribuidos, redes de alta velocidad para el acceso al almacenamiento, etc. Por este motivo y por las limitaciones presupuestarias, iba a tratarse de un clúster básico que tuviese:

- un servidor de nodo de cabecera, que proporcionaría los servicios básicos del clúster, servicio de login y que sirviese el almacenamiento compartido por NFS.
- varios servidores de nodos de cómputo

El nodo de cabecera debería tener, además del equipamiento necesario para la conexión al almacenamiento, una buena cantidad de cores y de memoria RAM. Esto sería debido a que el nodo, además de llevar algunos servicios necesarios para el funcionamiento del clúster y servir el almacenamiento por NFS, debería de proporcionar las funciones de nodo de login y desarrollo.

Como se vio anteriormente, la mayor necesidad es de cores y de memoria. Por ello los nodos de cómputo deberían ser nodos con multiprocesador de la última generación del momento de Intel, y deberían proporcionar un número de cores agregado igual o superior a 90 cores. En cuanto a las necesidades de memoria se estimó que deberían tener un mínimo de 5GB por core. Con respecto a llevar o no GPU o coprocesador, como no había software preparado para su aprovechamiento, su presencia disparaba el presupuesto y limitaba los modelos de servidores existentes en el mercado, se tomó la decisión de que algunos nodos deberían de ofrecer la opción de ampliación para un futuro. De este modo, el nodo de desarrollo (cabecera) y uno de los nodos de cómputo deberían ser capaces de ampliarse.

Con respecto al almacenamiento, en primer lugar se determina que una buena solución a la necesidad de acceso rápido a los catálogos, dado que dicho acceso es únicamente de lecturas, consistiría en dotar con un disco SSD a los nodos de cómputo. Para ello, habría que distribuir los catálogos más utilizados al SSD de los nodos de cómputo y los nodos atacarían directamente a su almacenamiento local. Esto evitaría la necesidad de adquirir un sistema compartido de almacenamiento y red de altas prestaciones. Se determinó un disco SSD mínimo de 120GB por nodo. Solucionado este problema, había que abordar el sistema de almacenamiento compartido, menos exigente en prestaciones. Para este sistema se determina la necesidad de emplear una matriz de discos. Para el conexionado a la matriz y dado que el nodo de cabecera va a ser el único que se conecte de forma directa, se usará SAS y deberá admitir multipath. En la elección del tipo de

discos a usar se determina que no se emplearán discos rápidos SSD, 10K o 15K. En su lugar se buscará el rendimiento de disco poniendo muchos discos de 7.2K. Por supuesto la matriz de discos deberá poder ser fácilmente ampliable sin la necesidad de adquirir nuevas controladoras. Finalmente se determinan 24 discos de 2.5" de 1TB.

En lo que respecta a la red, aquí había un problema. Si se atendían a los requisitos de cosmología, sería necesario recurrir a una red Infiniband o 10G. Esto incrementaría en mucho el presupuesto y supondría menos dinero para cómputo. Finalmente se decidió que dicho tipo de red se ampliaría en un futuro. Por lo tanto, se decidió que para la conectividad finalmente se usaría Gigabit Ethernet para todas las redes (general, cómputo y almacenamiento) y se emplearía agregaciones de enlace para suministrar mayor ancho de banda. Por ello, se especificó una tarjeta de 4 puertos Gigabit en todos los nodos. En cuanto a la especificación del equipamiento, por motivos de compatibilidad e integración de las redes general y de administración con la infraestructura existente, se determinó que el equipo no saldría en la misma licitación al tener que ser obligatoriamente un equipo de comunicaciones Cisco.

El último de los puntos era la selección del software que debería de llevar el clúster HPC. Los dos puntos básicos que había que conocer eran: el sistema operativo que debería llevar y el gestor de recursos del clúster. Determinar el sistema operativo era crítico ya que el hardware que debería adquirirse debería estar certificado para garantizar su plena compatibilidad con dicho sistema operativo. Finalmente se decidió que el hardware debería estar certificado para Ubuntu, ya que era el sistema que se estaba desplegando en todas las estaciones de trabajo del personal científico, garantizando de este modo una fácil adopción del sistema por parte de los usuarios y la compatibilidad con los desarrollos ya realizados. Por otro lado, el gestor de recursos del clúster debería ser el empleado en los sistemas de la UPAD: gridengine.

3.3. Situación actual de la infraestructura CEFCA

El clúster HPC deberá ser capaz de integrarse con la infraestructura existente. A continuación paso a describir brevemente los sistemas implicados en la integración.

3.3.1. Infraestructura de red

Para poder realizar el diseño de red del HPC, será necesario analizar la situación actual en lo que se refiere al diseño de red de la organización. Este diseño de red no será alterado y supondrá por lo tanto una restricción a tener en cuenta a la hora de diseñar los sistemas de red del HPC.

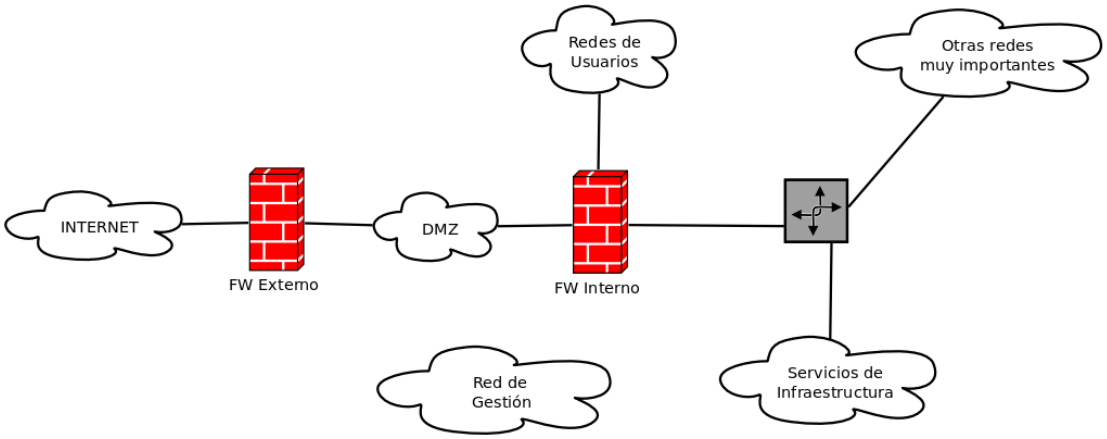


FIGURA 3.1: Diagrama lógico de red CEFCA

Como todo diseño, la red actual sigue unos principios de diseño. El primero de los principios es el de la seguridad, cualquiera de las soluciones tiene que tener en cuenta la seguridad en las comunicaciones, especialmente en los sistemas críticos. La seguridad deberá suponer un buen equilibrio entre el riesgo de materialización de las amenazas, los recursos disponibles y la satisfacción de los requerimientos de uso. El segundo principio de diseño son la satisfacción de los requerimientos desde el punto de vista de la capacidad y la escalabilidad. Esto quiere decir que la solución, además de satisfacer los anchos de banda requeridos, deberá poder ampliarse con nuevo equipamiento sin variar el diseño. El tercero de los principios será la disponibilidad. Habrá diferentes áreas de la red que necesitarán de alta disponibilidad, esto es, existirá redundancia en los equipos de comunicaciones que permitirán que los sistemas críticos puedan seguir funcionando.

Siguiendo estos principios, en primer lugar se llevó a cabo una segmentación física y lógica en subredes y en segundo lugar la interconexión de las mismas, siendo esta última quizás la parte más compleja. Para esto había que tener en cuenta especialmente el equipamiento disponible y sus ventajas e inconvenientes. Por un lado teníamos switches capaces de trabajar en capa 3, que ofrecían un alto throughput y posibilidades de stackado (permitiendo de este modo alta disponibilidad) a cambio de permitir un filtrado muy básico (no permitiendo hacer tracking de conexiones, etc). Por otro lado teníamos firewalls, que ofrecían grandes capacidades de filtrado a costa de un menor throughput (y en nuestro caso no pudiendo ser redundados). Finalmente se optó por el diseño representado por el siguiente diagrama lógico la figura 3.1.

En primer lugar vemos la existencia de la red DMZ. En esta red están todos los servidores que ofrecen servicios en Internet (web, correo, etc). El principio básico es que desde Internet sólo se permitirá el tráfico hacia DMZ y desde DMZ no se permitirá nunca el tráfico hacia redes internas salvo el tráfico cuyo origen haya sido la red interna. Para su diseño, y dado el equipamiento disponible, se desarrolló una arquitectura de doble

firewall. Esta arquitectura reduce significativamente uno de los riesgos más fácilmente materializables, que son los fallos de configuración por parte del administrador de sistemas. Además desde el punto de vista del rendimiento permite dividir el tráfico e incluso en un determinado momento implementar un IPS en uno de los firewalls (generalmente en FS Externo) sin afectar al rendimiento del otro firewall. El principal inconveniente que tenemos es la disponibilidad de estos sistemas, que no se encuentran redundados. (Además, dadas las diferentes tecnologías de los firewalls Cisco ASA y Linux, era muy difícil implementar una solución de este tipo). Aunque la arquitectura sí permite sustituir los componentes por dispositivos en alta disponibilidad en un futuro (actualmente uno de los proyectos en marcha es sustituir FW Interno por un clúster HA GNU/Linux).

El otro elemento son las "Redes de usuario". Este elemento engloba las redes desde las cuales el personal de CEFCA conecta sus dispositivos a la red Interna, esto es: red cableada, red wifi y red privada virtual. Como puede observarse, se pone a los usuarios siempre detrás de un firewall. Esto nos permitirá filtrar las conexiones, permitiendo solamente acceder a las redes y servicios que nosotros definamos independientemente de su ubicación: Internet, DMZ, servicios de infraestructura, etc.

Otra red significativa de mostrar para el desarrollo del proyecto es la red de "Servicios de Infraestructura". En esta red se ubican todos los servicios comunes y necesarios para el funcionamiento general de los sistemas. Dicha red, como puede observarse, se encuentra detrás de un encaminador. Dicho encaminador se encontrará implementado mediante las funciones de capa 3 de un stack de switches Cisco. Esto garantizará un alto throughput, menor latencia y una alta disponibilidad de los servicios ofrecidos en esta red (al encontrarse el equipamiento en stack). Además estos equipos nos permitirán un filtrado básico de nivel 3 que nos permitirá aislar redes enteras que cuelguen del mismo encaminador.

Las redes que se encuentran englobadas bajo "Otras redes muy importantes", son redes entre las que encontraremos las de telefonía, videovigilancia, observatorio, etc. Estas redes tienen a su vez su propia topología tanto física como lógica, pero como carecen de relevancia para el desarrollo de este proyecto se han representado como una nube.

Para finalizar, encontramos flotando la "Red de Gestión". En esta red se ponen las interfaces de administración de todos los equipos de comunicaciones, cortafuegos, hipervisores de los sistemas de virtualización, PDUs administrables, sistemas de gestión del hardware de servidores (iDRACs e iLOs), etc, y únicamente pueden tener acceso los administradores de sistemas. Para garantizar la seguridad, en esta red no existe pasarela al resto de redes. Por lo tanto, la única forma posible de acceder a los equipos de esta red es mediante la conexión directa. Aunque teóricamente esto es muy bonito, siempre es necesaria cierta conectividad hacia el exterior. Algunos ejemplos de ello son la necesidad de

sincronización horaria, envío de emails de alerta, monitorización, etc. Para implementar esto se recurre a sistemas con doble interfaz que proporcionen los servicios necesarios, es decir, sistemas con una interfaz en una red con acceso al exterior (generalmente en servicios de infraestructura) y la otra interfaz en la red de gestión. Para minimizar el riesgo de intrusiones, los equipos que disponen de doble interfaz, únicamente ofrecen servicios en la red de gestión.

3.3.2. Servicio DNS

No voy a comentar cómo es el diseño global del servicio DNS ya que se encuentra fuera del ámbito del PFC. Por lo tanto únicamente comentaré cómo será el servicio DNS para el dominio office.cefca.es, que es uno de los dominios internos existentes en la infraestructura CEFCA y en el que deberán integrarse los nodos del clúster. La solución existente para este dominio consiste en dos servidores con el software BIND, integrados con Samba4 mediante el módulo DLZ que realiza la lectura de la información de las zonas desde el directorio almacenado por Samba. Esto hace que no sean necesarias las transferencias de zona (la replicación de la información la realiza el software de directorio LDAP). Estos servidores, además de servir las consultas de sus zonas, cachean y reenvían peticiones externas a otros servidores. Las consultas a servidores DNS externos se encuentran bloqueadas en toda la organización mediante cortafuegos (salvo para los propios DNS) para que todos los equipos usen los DNS internos.

3.3.3. Servicio DHCP

El servicio DHCP existente se encuentra implementado con dos servidores autoritativos únicos para todas las redes del sitio, que comparten y balancean los pool en las redes en que son necesarios. La configuración de las reservas estáticas se encuentra sincronizada mediante el software Unison. Para finalizar, es necesario el uso de agentes DHCP relay para poder realizar las asignaciones en los dominios de broadcast a los que los servidores no tienen conexión.

3.3.4. Servicio NTP

El servicio NTP se compone de dos servidores NTP que utilizan como fuentes de tiempo servidores de estrato 1 existentes en Internet. Se ha definido a nivel de cortafuegos que sólo los servidores NTP pueden conectarse a Internet, de este modo obligaremos a toda la infraestructura a que usen los servidores internos. El primero de los servidores tiene como fuente de estrato superior a su reloj interno y el otro servidor tiene como fuente

al primer servidor, de forma que si se pierde conexión se mantiene una fuente de tiempo consistente en toda la organización.

3.3.5. Servicio Syslog

El servicio no es un servicio crítico para el funcionamiento de la infraestructura, por lo que no existe un despliegue en alta disponibilidad. Se utiliza un único servidor que almacena la información sobre ficheros de texto, haciendo muy fácil realizar la depuración de un problema con herramientas de filtros estándar. Dichos logs se comprimen y eliminan automáticamente usando la utilidad logrotate. Además dichos registros se agregan a una base de datos MySQL que reside en el mismo servidor y que crea un histórico que puede consultarse.

3.3.6. Active Directory

La autenticación y autorización deberá integrarse con la solución de Active Directory existente en la organización. La solución consiste en un sistema de alta disponibilidad integrado por dos servidores con Samba4 y BIND9 integrados. Dispone con un directorio creado con un nivel funcional de 2008R2 y el esquema de RFC2307 agregado. Dicho esquema permite el almacenamiento en el directorio de la información requerida por cuentas de usuario y grupo del estándar Posix. El primero de los servidores (llamados controladores de dominio) lleva todos los roles Active Directory del directorio. La replicación del directorio la realiza el mismo Samba4 mientras que la replicación del recurso sysvol se realiza mediante la utilidad rsync y siempre desde el primer servidor hacia el segundo.

3.3.7. Distribución de software

Con motivo de homogeneizar las estaciones de trabajo y portátiles GNU/Linux del personal de CEFCA, se trabajó en realizar una estandarización general. Para ello, se definió que la distribución base sobre la que se trabajaría sería la última versión LTS de Ubuntu. Además se determinó que la estandarización llamada *ubuntucefca14* debería:

- Tener diferente selección de software en función del tipo de usuario: científico o ingeniero.
- Disponer de una versión sin entorno gráfico que pudiera instalarse en servidores.

- Desplegar los certificados de la CA interna además de instalar un "artwork" y una apariencia "corporativa" en las versiones de escritorio.
- Incluir el software necesario que no se encontrase en los repositorios oficiales.

Para el despliegue del software, los paquetes se publican en un repositorio APT interno gestionado por Reprepro y publicado usando el servidor web Apache. Además, en el mismo servidor de software está disponible un repositorio APT caché que cachea los paquetes de la distribución oficial de Ubuntu, permitiendo acelerar mucho el proceso de instalación de software.

3.3.8. Monitorización

Existen dos herramientas de monitorización que se emplean en la infraestructura:

- **Cacti**: es la herramienta que usamos para obtener la información de uso de los equipos de comunicaciones. Esta solución hace uso de las conocidas RRDtools para almacenar la información en bases de datos de tipo round robin y para generar gráficos a partir de dicha información. Para ello implementa un componente software que mediante un polling va obteniendo la información de los equipos vía SNMP y almacenando la información en bases de datos. El componente principal de Cacti es una aplicación web que nos ofrece una manera sencilla de gestionar todo este proceso, dotándonos de plantillas y de un entorno mediante el cual es sencillo monitorizar multitud de dispositivos.
- **Nagios**: es la herramienta con la que principalmente gestionamos las alarmas de nuestros sistemas. También hacemos uso de PNP4nagios para almacenar información de valores obtenidos de los plugins en bases de datos round robin con las que se pueden generar posteriormente gráficas adicionales de rendimiento. Esta solución es prácticamente un estándar en la industria y existen multitud de plugins para realizar chequeos de todo tipo de dispositivos y servicios. Los chequeos que puede realizar son tanto pasivos como activos, es decir, por un lado podrá recibir eventos desde los dispositivos como puede ser un trap SNMP (chequeos pasivos) y por otro lado podrá realizar consultas a los dispositivos usando plugins especiales para ello (chequeos activos). Es una solución muy completa que almacena históricos de incidentes, estadísticas de disponibilidad, permite avisos y monitorización según intervalos horarios, múltiples tipos de alarmas (email, sms, jabber, etc), múltiples equipos de soporte y un larguísimo etc. Una de las grandes ventajas es que existen multitud de extensiones, una de ellas es pnp4nagios, que se aprovechará de la

información proporcionada por los plugins en cada chequeo para ir almacenando la información en bases de datos RRD, proporcionándonos también un histórico.

3.4. Especificación de requisitos

3.4.1. Acceso

- **RNF_01**: El clúster HPC será un clúster de uso privado al personal investigador de CEFCA y no existe ningún plan para la integración en un Grid. El acceso al mismo, por lo tanto, no deberá ser público y su conexión estará restringida a las redes de usuario de oficinas, del OAJ y a través de internet usando la VPN.
- **RNF_02**: Todos los usuarios deberán ser personal investigador de CEFCA, por lo que se usará la misma base de datos de autenticación y autorización que se usa para el resto de servicios. Esto evitará que los usuarios tengan que recordar un nuevo usuario y contraseña. También se valorará por parte de los usuarios un mecanismo Single Sign On (como Kerberos) que no les obligue a introducir la contraseña una y otra vez. Dado que la contraseña será la misma que para el resto de servicios, seguirá la misma política y los usuarios podrán cambiar la contraseña desde el mismo lugar.
- **RNF_03**: Desde el clúster será posible descargarse información de imágenes y bases de datos de la UPAD y OAJ. También deberán poder realizarse conexiones desde y hacia las workstations del personal investigador y a otros sistemas HPC empleando SSH.
- **RNF_04**: El acceso a los servicios del clúster se realizará exclusivamente a través del "nodo de login". Dicha conexión se realizará usando el protocolo SSH. No podrán realizarse conexiones al resto de nodos ni de forma directa ni indirecta a través del nodo de login. En el nodo de login el usuario podrá trabajar en su código, compilarlo y realizar ejecuciones de prueba, pero nunca ejecuciones reales y "completas" de sus programas.La shell que tendrán por defecto será bash y podrá ser cambiada mediante solicitud helpdesk. Sin embargo, sólo podrán obtener soporte si usan bash.
- **RNF_05**: El acceso al almacenamiento del clúster desde las redes de usuario se realizará también a través del "nodo de login" a través del protocolo SSH. Permitiendo de este modo el uso de herramientas tipo sftp, scp, rsync, git, etc. No se pondrán a disposición de las redes de usuario protocolos más "cómodos" como CIFS o NFS para evitar que el clúster acabe siendo un lugar de almacenamiento privado.

3.4.2. Hardware

- **RNF_06**:Los nodos de cómputo deben cumplir los requerimientos marcados en el documento "Fabricación y suministro de un clúster de computación intensiva para el Centro de Estudios de Física del Cosmos de Aragón", disponible en el perfil del contratante del Gobierno de Aragón [1].
- **RNF_07**: El nodo cabecera (llamado también de "desarrollo y servicios") y su sistema de almacenamiento deben cumplir con los requerimientos marcados en el documento "Fabricación y suministro de un clúster de computación intensiva para el Centro de Estudios de Física del Cosmos de Aragón", disponible en el perfil del contratante del Gobierno de Aragón.
- **RNF_08**: El switch de comunicaciones, por motivos de compatibilidad, deberá ser un switch Cisco que proporcione 48 puertos Gigabyte Ethernet de acceso y ofrezca funcionalidad de capa 3.

3.4.3. Software

- **RNF_09**: El sistema operativo se instalará en los nodos de manera local.
- **RNF_10**: Con el fin de garantizar la mayor compatibilidad con las estaciones de trabajo del personal científico, todos los nodos del clúster deberán llevar instalado el sistema operativo Ubuntu Server 14.04LTS. Además se instalarán todos los paquetes de desarrollo y científicos que se desarrollarán en la normalización de la distribución ubuntucefca14.
- **RNF_11**: En un recurso NFS que montarán todos los nodos se instalará todo el software especialmente compilado para el hardware del clúster. En este recurso podrá almacenar diferentes compiladores, librerías, toolchains, etc y además tener diferentes versiones de los mismos.
- **RF_01**: Los usuarios del clúster podrán escoger la versión de compiladores, librerías y toolchains utilizando para ello el comando "module".
- **RNF_12**: Serán necesaria la instalación optimizada de las siguientes librerías
  - ATLAS
  - BLACS
  - OpenBLAS

[1] https://servicios.aragon.es/pcon/pcon-public/controlAdjudicacionPublico?accion=ACCION_SELECCIONAR_ADJUDICACION_PUBLICO&iddatoadjudicacion=316478

- FFTW versiones 2 y 3
- ScaLaPACK
- GSL
- Numpy
- Scipy
- HDF5

3.4.4. Ejecución y supervisión

- **RF_02**: Toda ejecución de programas en el clúster deberá realizarse mediante el software de gestor de recursos del clúster. Por motivos de homogeneización y hacer más fácil las cosas a los usuarios, se utilizará el mismo software que se usa en la UPAD. El software sera gridscheduler un derivado libre de Sun Grid Engine. Para poder usarlo los usuarios tendrán a su disposición en el nodo de login todas las utilidades cliente.
- **RF_03**: Para poder ejecutar programas, los usuarios deberán definir trabajos o "jobs". En dichos trabajos los usuarios deberán poder definir:
  - Programa o programas a ejecutar
  - Ficheros de entrada y de salida de datos
  - Tipo de trabajo que es (se describirá a continuación)
  - Prioridad relativa que le da al trabajo
  - Email al que se le notificará cuando se inicie o finalice el trabajo
  - Recursos que necesitarán para completar el trabajo (memoria, tiempo, licencias de software, etc)
- **RF_04**: El proceso general para la ejecución de trabajos que deberán seguir los usuarios en el clúster será el siguiente:
  - El usuario inicia sesión en el nodo de login.
  - El usuario escribe y/o compila el software que necesita ejecutar.
  - El usuario puede lanzar alguna ejecución breve de prueba en el nodo de login para comprobar que el programa funciona bien.
  - El usuario escribe un fichero de trabajo en el que especificará los datos indicados anteriormente.
  - El usuario ejecuta el software gestor de recursos pasando como parámetro el fichero de trabajo generado.

  - El usuario comprobará comprobar mediante el software gestor de recursos el estado en la cola y de ejecución de su trabajo.
  - Una vez finalizado el trabajo, el usuario obtendrá la salida de su trabajo en el lugar especificado.

- **RF_05**: Podrán existir trabajos cuya entrada de datos deba proporcionarse de manera interactiva. Para ello el clúster proporcionará una cola que permita el envío de trabajos interactivos.
- **RF_06**: Además de los trabajos que requieran interacción con el usuario, el clúster admitirá:
  - Trabajos simples
  - Trabajos de tipo array: este tipo de trabajos permitirá la ejecución de un programa de forma paralela con distintos conjuntos de entrada de datos
  - Trabajos de tipo paralelo: este tipo de trabajos requerirán de un entorno de ejecución paralelo configurados en el clúster.
- **RF_07**: Los tipos entornos de ejecución paralelo que se deberán configurar en primera instancia serán:
  - openmp: la paralelización se realizará sobre múltiples cores de un mismo nodo
  - mpi: la paralelización se realizará a nivel de nodo y también a lo largo de varios nodos
- **RNF_13**: Al no tener de experiencia previa, las políticas de limitaciones de recursos, de ejecución paralela, asignaciones por defecto, etc, se dejarán por defecto y se irán modulando conforme se vaya obteniendo un feedback real.
- **RF_08**: Los usuarios podrán comprobar el estado de ejecución de sus programas en cualquier momento desde el nodo de login mediante las utilidades cliente del software de gestión de recursos.
- **RF_09**: Los usuarios podrán cancelar sus trabajos cuando así lo deseen. Únicamente los usuarios administradores del clúster podrán cancelar trabajos de otros usuarios.
- **RF_10**: Los usuarios podrán ver el estado de ejecución de trabajos de los nodos mediante las utilidades cliente.
- **RF_11**: Los usuarios tendrán a su disposición una web de información del clúster donde podrán ver:
  - si todos los servicios del clúster están operativos

  - consumo actual de recursos
  - estadísticas de uso general

- **RNF_14**: Existirán tres tipos de usuario.
  - superusuarios: tendrán un control total sobre todo el sistema. No serán usuarios con los que se envíen trabajos al clúster.
  - administradores del clúster: serán usuarios que podrán cancelar trabajos de otros usuarios, sincronizar catálogos, instalar software y tendrán permisos administrativos en los nodos de cómputo.
  - usuarios del clúster: serán los usuarios normales del clúster.

3.4.5. Almacenamiento

- **RNF_15**: Cada usuario tendrá un HOME personal que será visible desde todos los nodos del clúster. Con el motivo de facilitar la colaboración, por defecto al HOME de los usuarios tendrán acceso de lectura para los demás usuarios del clúster.
- **RNF_16**: La velocidad de transferencia de datos desde y hacia el clúster tendrá un máximo agregado de 1Gbps, limitado por la interfaz del nodo de login en clúster, las características del firewall y en última instancia del puerto de comunicaciones al que se encuentre conectado la workstation (100Mbps).
- **RNF_17**: El espacio de nombres del sistema de archivos seguirá el estándar FHS salvo por las excepciones comentadas a continuación.
- **RNF_18**: Todo recurso de almacenamiento disponible en todos los nodos del HPC (salvo los HOME) deberá colgar del directorio `/global`. El contenido será el siguiente:

```
/global
        /shared
        /catalog
                /distrib
        /scratch
                /$user
```

- **RNF_19**: Los usuarios deberán poder usar en sus trabajos parte del almacenamiento local de cada nodo, implementado mediante un disco SSD.
- **RNF_20**: Todo recurso de almacenamiento local del nodo de cómputo colgará del directorio `/local`. El contenido será el siguiente:

```
/local
        /catalog
        /scratch
                /$user
```

- **RNF_21**: El recurso `/home/CEFCA` será un recurso de lectura/escritura que contendrá el home de cada usuario. Los homes estarán protegidos, teniendo cada usuario sólo acceso a su home. Dicho recurso tendrá una cuota por usuario por determinar.
- **RNF_22**: El recurso `/global/shared` será un recurso de lectura/escritura para todos los usuarios. Cada usuario tendrá un directorio con su username. Todos los usuarios tendrán permisos sobre los ficheros que allí se creen salvo el permiso de borrado (sticky bit).
- **RNF_23**: El recurso `/global/catalog` será un recurso de sólo lectura salvo para los usuarios Administradores del Clúster.
- **RNF_24**: El recurso `/global/scratch` será un recurso de lectura y escritura para todos los usuarios. Cada usuario tendrá un directorio con su username. Existirá una tarea programada que elimine el contenido de este recurso cada cierto tiempo.
- **RNF_25**: El recurso `/local/catalog` será un recurso de sólo lectura. El contenido de este recurso se sincronizará mediante una tarea programada con el contenido de `/global/catalog/distrib`.
- **RNF_26**: El recurso `/local/scratch` será un recurso de lectura y escritura para todos los usuarios. Cada usuario podrá crearse un directorio con su username. Existirá una tarea programada que elimine el contenido de este recurso cada cierto tiempo.

3.4.6. Integración

- **RNF_24**: Al no disponer de hardware dedicado para el nodo de login, se creará una máquina virtual o un contenedor en el nodo cabecera para implementar dicho rol.
- **RNF_25**: Todos los nodos deberán ser agregados al dominio `office.cefca.es` y usarán como servidores de resolución a los servidores principales del dominio.
- **RNF_26**: Todos los nodos del clúster deberán disponer de reservas estáticas en el servicio y poder obtener su dirección de manera dinámica.
- **RNF_27**: Todos los nodos del clúster deberán estar correctamente sincronizados contra los servidores de tiempo de la infraestructura.
- **RNF_28**: Todos los nodos del clúster deberán enviar la información de logging al servidor syslog.

- **RNF_29**: Todos los nodos del clúster se agregarán al dominio para mantener la consistencia de UIDs y GIDs. Además el nodo de login podrá ofrecer autenticación Kerberos en el servicio SSH.
- **RNF_30**: Se agregará la monitorización del switch a Cacti.
- **RNF_31**: Todo el hardware y los servicios importantes del clúster estarán monitorizados mediante la herramienta Nagios.

Capítulo 4

Descripción de la solución adoptada y planificación

4.1. Proceso de selección del equipamiento

Todo lo anteriormente expuesto relacionado con el hardware de la solución, se redactó y trató de dejar bien atado en un documento de licitación pública que salió bajo el nombre de "Fabricación y suministro de un clúster de computación intensiva para el Centro de Estudios de Física del Cosmos de Aragón" expediente 2014/01. Toda la información referente a este proceso se encuentra en el Perfil del Contratante del Gobierno de Aragón [1].

De acuerdo a las condiciones expuestas en la licitación y conforme a como marca la Ley, se procedió a valorar las ofertas recibidas. Tras el proceso de valoración la oferta ganadora y a la que finalmente se adjudicó fue la oferta de HP[2].

Para la selección del switch finalmente se optó por un modelo de Cisco 3750X. El principal motivo para seleccionar este modelo es que ya existen múltiples switches de este modelo en CEFCA. Para la adquisición, se pidió presupuesto a varios proveedores y se eligió la oferta más económica.

[1] https://servicios.aragon.es/pcon/pcon-public/controlAdjudicacionPublico?accion=ACCION_SELECCIONAR_ADJUDICACION_PUBLICO&iddatoadjudicacion=316478

[2] https://servicios.aragon.es/pcon/pcon-public/controlPrincipalPublico?iddocumento=299034

FIGURA 4.1: Servidor HP Proliant DL380 Gen8

4.2. Especificaciones finales del hardware

4.2.1. Nodo cabecera

El equipo será un servidor HP Proliant DL380Gen8p de 2U de tamaño con:

- 2 procesadores E5-2630V2. Las características de este procesador son: 2.6GHz, 6 cores, 15MB, 7.2GT-s QPI, 80W, soporte DDR3-1600, HT, Turbo Boost
- 128GB de RAM en módulos HP 16GB 2RX4 PC3-14900R-13
- 2 discos de 300GB a 10K
- Tarjeta ethernet flexlom de 4 puertos 1gbps y tarjeta adicional 4 puertos 1gbps
- 2 fuentes de 1200 W

El nodo tendrá conectado vía SAS la cabina de discos, que será una HP MSA 2040 de 2U's con:

- Doble controladora
- Doble fuente de alimentación y ventiladores redundantes
- 8GB de caché (4 en cada controladora)
- 8 puertos SAS a 12Gbps con soporte multipath
- 24 discos MDL-SAS SFF de 1 TB de capacidad
- Ampliable hasta 199 discos SFF con 7 bandejas adicionales de 2U's
- Tamaño de LUN de hasta 64TB

FIGURA 4.2: Matriz de discos HP MSA 2040 SAS

- Soporte de RAID 0, 1, 3, 5, 6, 10 y 50
- Licencia para realizar hasta 64 snapshots y/o clones
- Licencia opcional de replicación

4.2.2. Nodos de cómputo

Todos los nodos irán ubicados en un Chasis SL6500 de 4Us con:

- Fuentes redundantes de 1500 W
- 8 ventiladores redundantes

Habrá 4 nodos de tipo HP Proliant SL230 gen8 con:

- 2 procesadores E5-2670V2
- 128GB de RAM en dimms de HP 16GB HP 16GB 2RX4 PC3-14900R-13
- Tarjeta de 4 puertos 1gbps y 2 puertos integrados en placa
- Disco SSD HP 200GB 6G SATA 2.5

Y 1 nodo de tipo HP Proliant SL250 gen8 con:

- 2 procesadores E5-2670V2
- 128GB de RAM en dimms de HP 16GB HP 16GB 2RX4 PC3-14900R-13
- Tarjeta de 4 puertos 1gbps y 2 puertos integrados en placa
- Disco SSD HP 200GB 6G SATA 2.5
- Capacidad de llevar hasta 3 aceleradoras o bien Xeon Phi o bien GPGPU

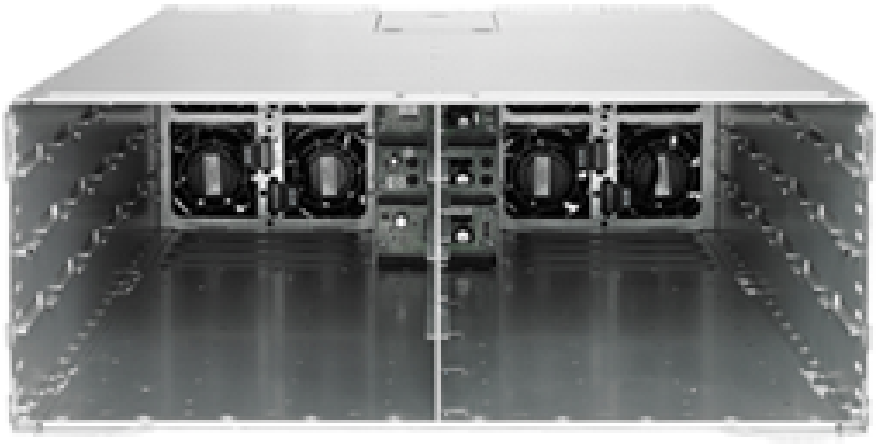


FIGURA 4.3: Chasis HP SL6500

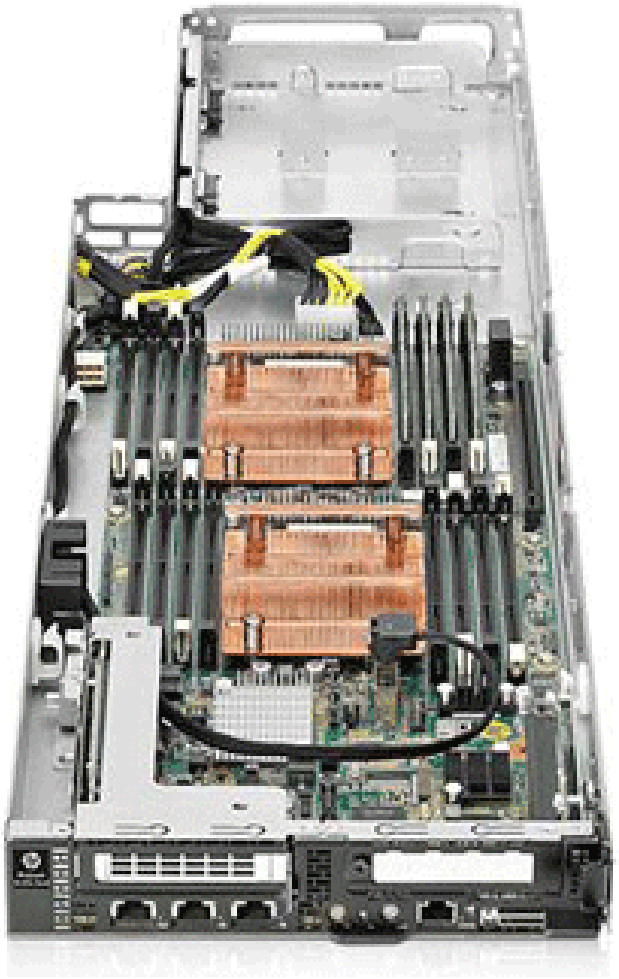


FIGURA 4.4: Servidor HP Proliant SL250

FIGURA 4.5: Switch Cisco 3750X

4.2.3. Switch de comunicaciones

El switch es un Cisco WS-C3750X-48

- 48 puertos GE
- Módulo C3KX-NM-1G para uplinks 1GB
- Fuente redundante

4.2.4. Consumo eléctrico

Consumo total de los servidores:

- 900.25 watts
- 3069 BTUs

Consumo total del almacenamiento:

- 374 watts
- 1622 BTUs

Consumo de switch (100 % throughput)

- 152 watts
- 516 BTUs

Cómputo y almacenamiento		
Concepto	**Uds**	**Precio**
Servidor HP Proliant DL380Gen8	1	
MSA 2040 SAS	1	
Chasis SL6500	1	
Servidor HP Proliant SL230 gen 8	4	
Servidor HP Proliant SL250 gen 8	1	
	Subtotal	52.000,00 €

Networking y cableado		
Concepto	**Uds**	**Precio**
Switch CISCO 3750X	1	4.356,73 €
Cableado y accesorios		243,84 €
	Subtotal	4.600,57 €

Total	56.600,57 €

FIGURA 4.6: Coste del hardware del HPC

4.2.5. Coste final

El presupuesto al que salió la licitación fue de 60.000€y fue adjudicada a HP por un precio de 52.000€. Si a esto añadimos los costes del switch y cableado obtenemos el precio total del coste del hardware de la solución que podemos ver en la figura 4.6.

4.3. Planificación

El desarrollo de la solución seguirá un proceso iterativo en el cual se proporcionará una versión funcional del producto a cada iteración. Cada versión cumplirá un hito en el proyecto que a su vez estará compuesto de una serie de objetivos que deberán ser cumplidos antes de poder pasar al siguiente hito. Entre la finalización de una versión y el inicio de la siguiente, se dejará un espacio de tiempo suficiente para que salgan a la luz problemas y para obtener retroalimentación por parte de los usuarios y del propio sistema, siendo esto último especialmente importante debido al desconocimiento de partida en la implementación de soluciones de este tipo. Tras el análisis de la retroalimentación obtenida, aparecerán nuevos objetivos y la importancia relativa de los objetivos marcados para versiones posteriores variará. Con este análisis, la dependencia entre objetivos

y una estimación orientativa acerca del tiempo necesario para cumplirlos, se definirán los objetivos necesarios para el siguiente hito y una fecha límite para su realización.

En el inicio de cada hito se analizarán los objetivos, y se realizará una enumeración detallada de las tareas necesarias para cumplir dichos objetivos. Se realizará una estimación en tiempo de las tareas y se emplazarán en el tiempo. Aunque he recibido ayuda en momentos puntuales como en el montaje físico del clúster, soy yo el único recurso humano disponible por lo que en este proyecto no hay un reparto de tareas.

A continuación expongo la foto final, sin entrar en más detalles, de los hitos y los objetivos que se han definido a lo largo del proyecto. Cabe destacar que el último de los hitos no estuvo presente en la planificación original del proyecto y se añadió al final.

4.3.1. Hito 1: Versión inicial del HPC

En este hito el objetivo principal es obtener una versión funcional básica del HPC con la que los científicos puedan empezar a trabajar. Para ayudar a obtener la mayor retroalimentación del sistema para la definición de los objetivos del siguiente hito, deberá implementarse un sistema de monitorización básico de los recursos del clúster.

Los objetivos en esta fase serán:

- Disponer de una infraestructura de pruebas virtual del clúster HPC
- Integrar en las redes de CEFCA al clúster y tener el networking interno funcionando
- Disponer de un sistema de almacenamiento compartido funcional con los recursos especificados
- Realizar la integración de los nodos a nivel de red, autenticación, etc con la infraestructura CEFCA.
- Disponer de una versión funcional de nodo de login
- Desplegar el software gestor de recursos con una configuración funcional
- Desplegar del software estándar científico de ubuntucefca14 en los nodos de cómputo y login
- Despliegue inicial del sistema Ganglia para monitorizar la utilización de los nodos
- Desarrollar la automatización del despliegue de la configuración de los nodos de cómputo

4.3.2. Hito 2: Optimización del software científico

En este hito el objetivo principal es instalar el software científico requerido optimizado. Además se completará el proceso de automatización del despliegue de los nodos de cómputo del clúster.

Los objetivos de esta fase serán:

- Implementar mediante el sofware EasyBuild el repositorio de software científico optimizado
- Migrar el nodo de login de VirtualBox a un contenedor LXC
- Implementar la automatización completa del sistema operativo usando LGDeploy

4.3.3. Hito 3: Integración del sistema de monitorización

En este hito el objetivo es la integración de los sistemas que conforman el clúster con los sistemas de monitorización de CEFCA.

Los objetivos de esta fase serán:

- Integración del switch en el sistema de monitorización Cacti
- Integración del hardware de nodos en sistema de monitorización Nagios
- Integración de los servicios en el sistema de monitorización Nagios

4.3.4. Hito 4: Ajustes en el gestor de recursos y documentación

Para la consecución de este hito se realizarán los ajustes en el gestor de recursos del clúster y se elaborará una documentación de usuario.

Los objetivos en esta fase serán:

- Realización de mediciones y ajustes en los sistemas
- Tuneado del gestor de recursos
- Agregar monitorización de trabajos y slots a Ganglia
- Creación del portal del usuario

4.3.5. Hito 5: Actualización del clúster a 10G

En este hito el objetivo principal es la actualización de los sistemas de red a Ethernet 10G.

Los objetivos en esta fase son:

- Migrar clúster a rack HP
- Actualización networking a Ethernet 10G
- Realizar mediciones y ajustes. Realizar una comparativa con red 1G.
- Presentación de la memoria de este PFC

Capítulo 5

Desarrollo del proyecto

Acabamos de ver en la fase de planificación cómo el proyecto se ha definido en hitos y objetivos y es así como se llevará a cabo de manera real. Sin embargo este desarrollo iterativo es muy complejo de documentar, especialmente si dicha documentación debe ser entendida por personas ajenas al proyecto. Esto es debido principalmente a que existen muchos elementos interdependientes y saltos entre conceptos que habría que dar necesarios para detallar el proceso. Por ello se ha ordenado todo el proceso de desarrollo de una manera secuencial de forma que sea más fácil de entender.

La secuencia realizada es una secuencia lógica en la cual en cada paso existe una dependencia clara del anterior y es la que brevemente describo a continuación:

1 Analizamos y diseñamos las redes internas del clúster y la conexión con CEFCA

2 Analizamos y diseñamos cómo se implementará el sistema de almacenamiento compartido

3 A partir de los puntos anteriores ya conocemos cómo ha de realizarse el cableado y tenemos más información del sistema, por lo que procedemos a realizar la instalación física de los equipos, el cableado y la configuración inicial de los mismos

4 Con los equipos de almacenamiento y red funcionando, ya podemos configurar los nodos del clúster. En este paso se trabajará en la configuración a nivel general de sistema, es decir: integración de los nodos en la infraestructura de CEFCA, configuración del almacenamiento y configuración de la red.

5 Con los nodos plenamente operativos con una configuración general, ya podemos centrarnos en el despliegue del software particular. En este paso nos dedicaremos a desplegar las herramientas software de tipo científico que los usuarios finalmente ejecutarán en el clúster.

6 En el paso anterior aprendimos y conocimos los programas que los usuarios necesitarán ejecutar: qué son y cómo funcionan. Identificamos el software que requiere de comunicación entre los nodos y por lo tanto tiene un tratamiento especial. Con toda esta información ya podemos trabajar en el diseño e implementación del software desde el cual los usuarios lanzarán dichos programas: el gestor de recursos.

7 Ya tenemos todo funcionando y deberemos garantizar que siga así, para ello integraremos el clúster en los sistemas de monitorización de CEFCA.

8 Como ya conocemos toda la configuración final, incluyendo la de los agentes de monitorización necesarios, procederemos a elaborar el proceso de automatización del despliegue de los nodos de cómputo.

9 Ya tenemos prácticamente la solución completa, vamos a ver sus prestaciones y comentaremos algunos de los ajustes realizados durante el proceso de configuración.

10 Se realiza la actualización del networking a ethernet 10G, por lo que se deberán realizar los cambios necesarios en el clúster para integrar dicha tecnología. Además realizaremos una comparativa con los datos que obtuvimos en el apartado anterior para evaluar la mejora.

11 Con el producto ya completado, dotaremos a los usuarios de una documentación y unas herramientas de soporte.

5.1. Diseño e integración de la red

5.1.1. Análisis de red y conectividad del clúster

Ya vimos en la fase de especificación los requerimientos y las restricciones del sistema en cuanto a la interconexión con la infraestructura existente que se muestra en la figura 3.1. A continuación trataremos de recapitular los requerimientos de necesidades de conectividad del HPC y hacer un pequeño análisis adicional. Para ello usaremos distintos puntos de vista: desde el punto de vista del usuario del sistema, desde el punto de vista de funcionamiento propio del sistema y desde el punto de vista del administrador del sistema.

Desde el punto de vista del usuario del sistema, toda la conectividad será realizada a través del nodo de login, no pudiendo tener acceso a ningún otro servicio interno del HPC. A dicho nodo solamente podrá acceder vía SSH ,permitiéndose también el uso del redireccionamiento de X11 hacia su máquina tunelado en ese protocolo. Para subir y descargar archivos del HPC también lo hará usando dicho protocolo, pudiendo emplear

utilidades como sftp, scp, rsync o git. El motivo por el que no se ofrecerá ningún servicio de almacenamiento "más cómodo" como pudiera ser NFS o CIFS, se hace con el fin de evitar que los usuarios conviertan el HPC en su almacén personal de backups. Por lo tanto, desde las redes de usuario hacia el HPC únicamente se admitirá el tráfico SSH hacia el nodo de login. Una vez dentro del nodo de login, se permitirá al usuario conectar a su estación de trabajo y a otros clústers HPC (dados de alta previamente por el administrador de sistemas) usando SSH y se permitirá el tráfico HTTP y HTTPS para que pueda descargar software desde la red. Una restricción adicional del sistema es que los usuarios desde el nodo de login no podrán obtener sesiones interactivas en el resto de nodos del clúster a no ser que la obtengan a través del gestor de colas.

Ya se comentó en el "Estado del arte" las necesidades y el diseño típico de red de un sistema HPC desde el punto de vista del sistema. Resumiendo, en primer lugar necesitaremos de una red "general" por la que circularán, entre otros: el tráfico SSH, tráfico del gestor de colas y el tráfico de monitorización. Esta red será poco exigente en cuanto a prestaciones. En segundo lugar tendremos una red dedicada al acceso al sistema de almacenamiento. Esta red será especialmente exigente en lo referente al throughput. Para finalizar, hará falta una red de comunicación entre nodos dedicada al tráfico de cómputo (generalmente MPI), que será muy exigente en prestaciones: baja latencia si se usan protocolos de sincronización y alto throughput si se realizan grandes transferencias de memoria entre nodos. La tecnología con la que contamos no es óptima para solucionar este tipo de problemas (Gigabit Ethernet), por este motivo será necesaria una segunda fase de ampliación del HPC utilizar otra tecnología de networking dedicada a esta red. Sin embargo, habrá que dar una solución que pueda ser fácilmente reemplazable en el futuro.

Además de todo esto, el clúster necesitará comunicación con el exterior para los servicios de infraestructura de los que depende (DNS, Directorio, etc.) y con los sistemas de monitorización.

Desde el punto de vista del administrador del sistema, además de poder conectar a través de la red "general" de forma interactiva a todos los nodos usando SSH, deberá poder conectarse a los diferentes sistemas de gestión que ofrece el hardware del clúster. Para ello estos sistemas deberán integrarse dentro de la red que hemos llamado anteriormente "Red de Gestión". En esta red incluiremos: gestión de switch del HPC, gestión de la matriz de discos y gestión de las iLOs de los servidores.

Una vez identificadas las necesidades, listaremos los elementos con los que contamos para implementar la interconexión del clúster:

- Switch Cisco 3750:

- 48 puertos Ge servicio
- 4 puertos Ge de uplink

- Matriz de discos:
  - Controladora A: 1 puerto de administración.
  - Controladora B: 1 puerto de administración.
- Servidor Cabecera:
  - 1 puerto iLO
  - Placa: 4 puertos Ge
  - Tarjeta adicional: 4 puertos Ge
- 5 nodos cómputo:
  - 1 puerto iLO
  - Placa: 2 puertos Ge
  - Tarjeta adicional: 4 puertos Ge

5.1.2. Diseño de las redes del clúster

Una vez realizado el análisis vamos a proceder con el diseño de la redes del clúster.

5.1.2.1. Red general

Esta red será la que lleve el tráfico "general" del clúster, esto es: tráfico SSH, gestor de colas, sistemas de monitorización, tráfico DNS, autorización y autenticación de usuarios, etc. Dicho de otro modo, llevará todo el tráfico de red salvo el tráfico de almacenamiento y de cómputo. A esta red conectaremos todos los nodos (cabecera, login y cómputo) además del cortafuegos externo, que usaremos de pasarela hacia "el mundo exterior". Puede verse en el diagrama un diagrama lógico de la red. En el diagrama se puede ver una topología de bus a la que están conectados todos los nodos mas el firewall.

En nuestro diseño, no realizaremos ningún tipo de NAT y la red podrá ser direccionada desde fuera del HPC (será la única en serlo). Por lo tanto se elegirá una red dentro del direccionamiento privado de CEFCA. La dirección de red asginada será 10.50.85.0/24.

```
Address:   10.50.85.0           00001010.00110010.01010101. 00000000
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000. 11111111
=>
```

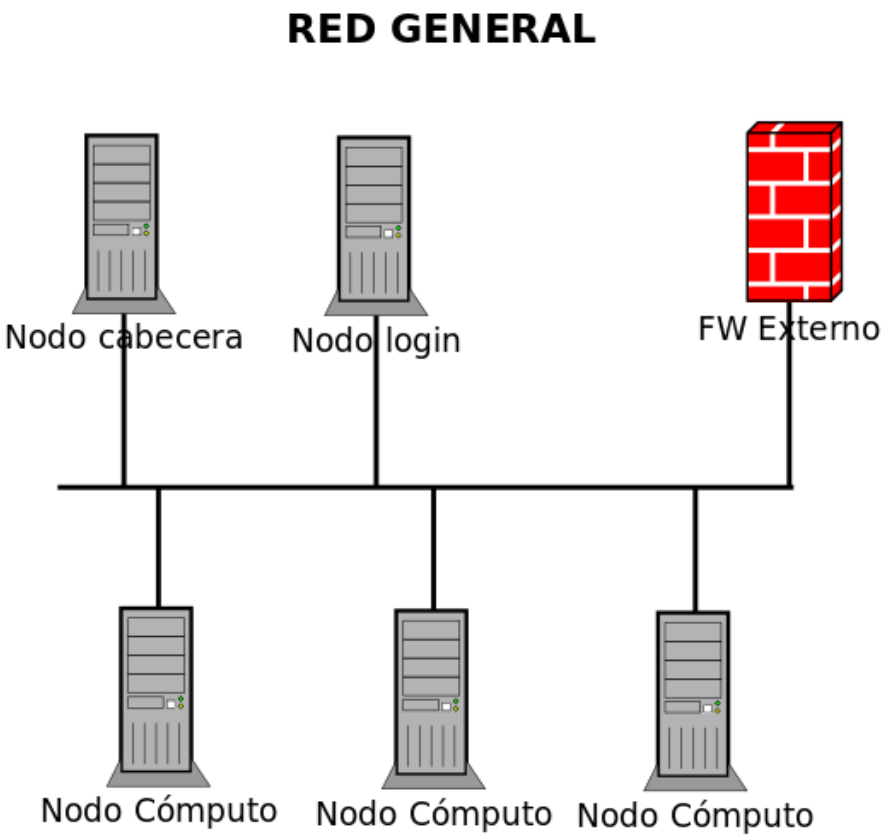


FIGURA 5.1: Diagrama lógico red general HPC

```
Network:    10.50.85.0/24          00001010.00110010.01010101. 00000000
HostMin:    10.50.85.1             00001010.00110010.01010101. 00000001
HostMax:    10.50.85.254           00001010.00110010.01010101. 11111110
Broadcast:  10.50.85.255           00001010.00110010.01010101. 11111111
Hosts/Net: 254                      Class A, Private Internet
```

Las direcciones quedarán de la siguiente manera:

- hpc-master: 10.50.85.1
- hpc-login: 10.50.85.2
- hpc-node1: 10.50.85.10
- hpc-node2: 10.50.85.11
- hpc-node3: 10.50.85.12
- hpc-node4: 10.50.85.13
- hpc-node5: 10.50.85.14
- firewall: 10.50.85.254

De este modo dejaremos las direcciones (3-9) para posibles nodos de login y servicios y las direcciones (15-253) para nodos de cómputo.

Todas las conexiones, salvo la del cortafuegos, se realizarán en el switch 3750 dedicado del clúster (posteriormente se hablará de cómo se conectará con los otros equipos de comunicaciones). Como ya se comentó, el nodo hpc-login será un "container" hubicado en el nodo hpc-master y no existirá físicamente. En este caso optaremos por asignar una interfaz exclusiva para contenedor en esta red en lugar de compartirla. Para los nodos de cómputo asignaremos un puerto exclusivo en esta red. Quedará del siguiente modo:

- (nodo cabecera) hpc-master: Puertos 0/1 y puerto 0/2.
- hpc-node1: Puerto 0/1
- hpc-node2: Puerto 0/1
- hpc-node3: Puerto 0/1
- hpc-node4: Puerto 0/1
- hpc-node5: Puerto 0/1

5.1.2.2. Red de almacenamiento

La red de almacenamiento llevará todo el tráfico NFS del clúster HPC y será de uso interno del HPC, no pudiendo ser accesible desde fuera (los usuarios conectarán al nodo de login y verán el almacenamiento, pero nunca la red). A esta red conectaremos todos los nodos: el nodo cabecera (hpc-master) para servir el almacenamiento vía NFS y los nodos de cómputo. En el caso del nodo hpc-login no hará falta, ya que al tratarse de un contenedor tendrá acceso local al almacenamiento.

Como se trata de una red no accesible, usaremos en nuestro diseño una red privada no direccionable en CEFCA. La dirección de red asignada será 192.168.14.0/24.

```
Address:   192.168.14.0          11000000.10101000.00001110. 00000000
Netmask:   255.255.255.0 = 24    11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255             00000000.00000000.00000000. 11111111
=>
Network:   192.168.14.0/24       11000000.10101000.00001110. 00000000
HostMin:   192.168.14.1          11000000.10101000.00001110. 00000001
HostMax:   192.168.14.254        11000000.10101000.00001110. 11111110
Broadcast: 192.168.14.255        11000000.10101000.00001110. 11111111
Hosts/Net: 254                    Class C, Private Internet
```

Las direcciones quedarán de la siguiente manera:

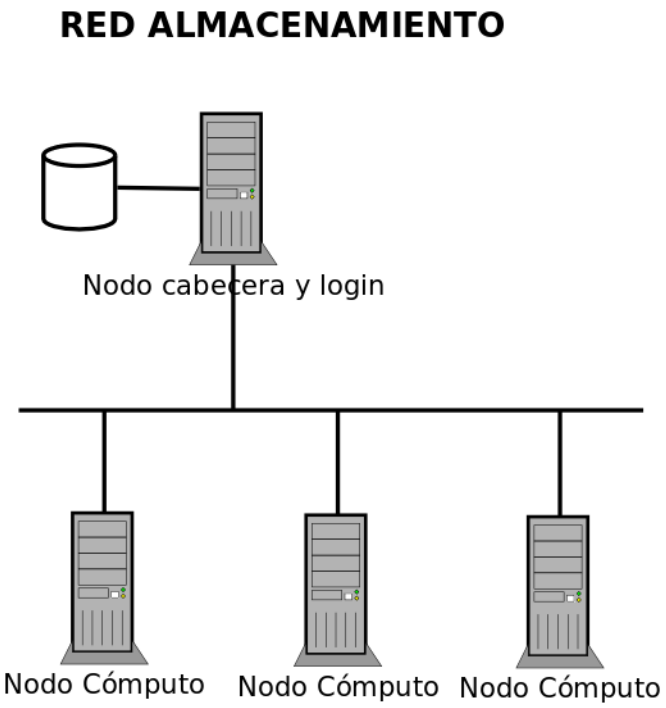


FIGURA 5.2: Diagrama lógico red de almacenamiento HPC

- hpc-master: 192.168.14.1
- hpc-node1: 192.168.14.10
- hpc-node2: 192.168.14.11
- hpc-node3: 192.168.14.12
- hpc-node4: 192.168.14.13
- hpc-node5: 192.168.14.14

De este modo dejaremos las direcciones (2-9) para posibles nuevos orígenes de almacenamiento y (15-253) para nodos de cómputo. Nótese además que el último octeto en las direcciones coincide exactamente con la red general. Esto nos será de suma utilidad a la hora de realizar scripts de automatización.

Todas las conexiones se realizarán en el switch 3750 dedicado al clúster. Como ya se ha comentado, el nodo de login es un "container" y no necesita conexión al almacenamiento. Al tratarse de una red con mucha exigencia en throughput, se hará uso de LACP, un protocolo de agregación de enlace (lo que se conoce comúnmente como bonding) y se fijará la MTU a 9000 (por defecto la longitud máxima de la trama ethernet es de 1500, aumentando el tamaño de la misma se consigue una mayor carga por paquete, permitiendo de este modo un menor overhead y aumentando de este modo el throughput).

Usaremos 4 puertos en el nodo cabecera (hpc-master) que es el que servirá el almacenamiento. En los nodos de cómputo se previeron originalmente 2 puertos sin embargo, debido a las características del switch, se comprobó que no iba a ser posible el aprovechamiento del agregado de las interfaces (esto lo explicaré en el apartado de Mediciones y ajustes) por lo que se dejará una única interfaz por nodo. Quedará del siguiente modo:

- (nodo cabecera) hpc-master: Puertos 1/1, puerto 1/2, puerto 1/3 y puerto 1/4 en Bonding y MTU 9000
- hpc-node1: Puerto 0/2 y MTU 9000
- hpc-node2: Puerto 0/2 y MTU 9000
- hpc-node3: Puerto 0/2 y MTU 9000
- hpc-node4: Puerto 0/2 y MTU 9000
- hpc-node5: Puerto 0/2 y MTU 9000

5.1.2.3. Red de cómputo

La red de cómputo será la red dedicada por la que los nodos de cómputo podrán comunicarse para el intercambio de mensajes en programas paralelos distribuidos en distintos nodos. Como se comentó anteriormente, en una segunda fase de ampliación del HPC se usará un networking dedicado con una tecnología superior al Gigabit Ethernet del que disponemos. En nuestro diseño deberemos dar una solución con la que funcionen los problemas y que sea fácilmente reemplazable por tecnologías superiores. A diferencia de las otras redes, y como puede verse en el diagrama, a esta red sólo conectaremos los nodos de cómputo.

Como se trata de una red no accesible, usaremos en nuestro diseño una red privada no direccionable en CEFCA. La dirección de red asignada será 192.168.13.0/24.

```
Address:   192.168.13.0          11000000.10101000.00001101. 00000000
Netmask:   255.255.255.0 = 24    11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255             00000000.00000000.00000000. 11111111
=>
Network:   192.168.13.0/24       11000000.10101000.00001101. 00000000
HostMin:   192.168.13.1          11000000.10101000.00001101. 00000001
HostMax:   192.168.13.254        11000000.10101000.00001101. 11111110
Broadcast: 192.168.13.255        11000000.10101000.00001101. 11111111
Hosts/Net: 254                    Class C, Private Internet
```

Las direcciones quedarán de la siguiente manera:

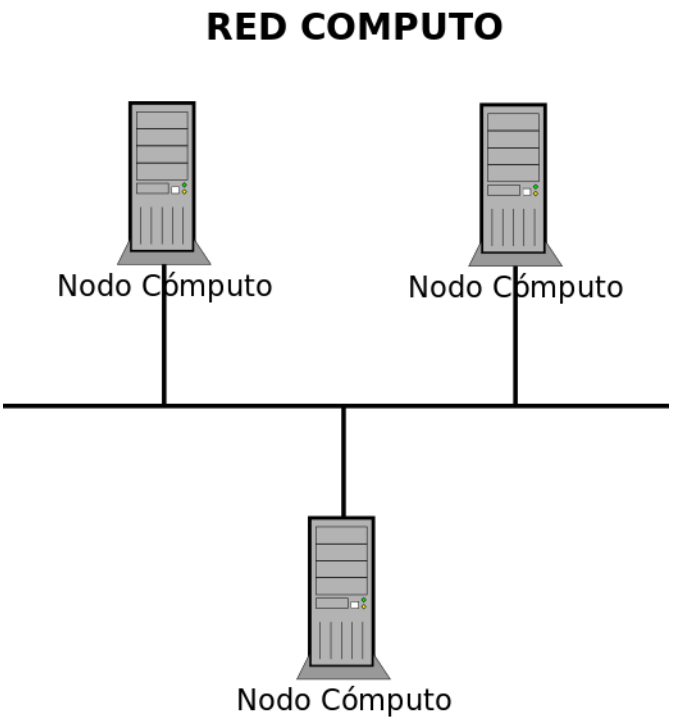


FIGURA 5.3: Diagrama lógico red de cómputo HPC

- hpc-node1: 192.168.13.10
- hpc-node2: 192.168.13.11
- hpc-node3: 192.168.13.12
- hpc-node4: 192.168.13.13
- hpc-node5: 192.168.13.14

De este modo dejaremos las direcciones (15-253) para nodos de cómputo. Nótese además que el último octeto en las direcciones coincide exactamente con la red general. Esto nos será de suma utilidad a la hora de realizar scripts de automatización.

Todas las conexiones, hasta que puedan sustituirse por un networking dedicado 10G o Infiniband, se realizarán en el switch 3750X dedicado del HPC. En este caso sí podrá aprovecharse el uso de bondings, por lo que realizaremos dos conexiones por nodo de cómputo con una MTU de 9000. Además se quedando del siguiente modo:

- hpc-node1: Puerto 1/1 y puerto 1/2 en bonding y MTU 9000
- hpc-node2: Puerto 1/1 y puerto 1/2 en bonding y MTU 9000
- hpc-node3: Puerto 1/1 y puerto 1/2 en bonding y MTU 9000

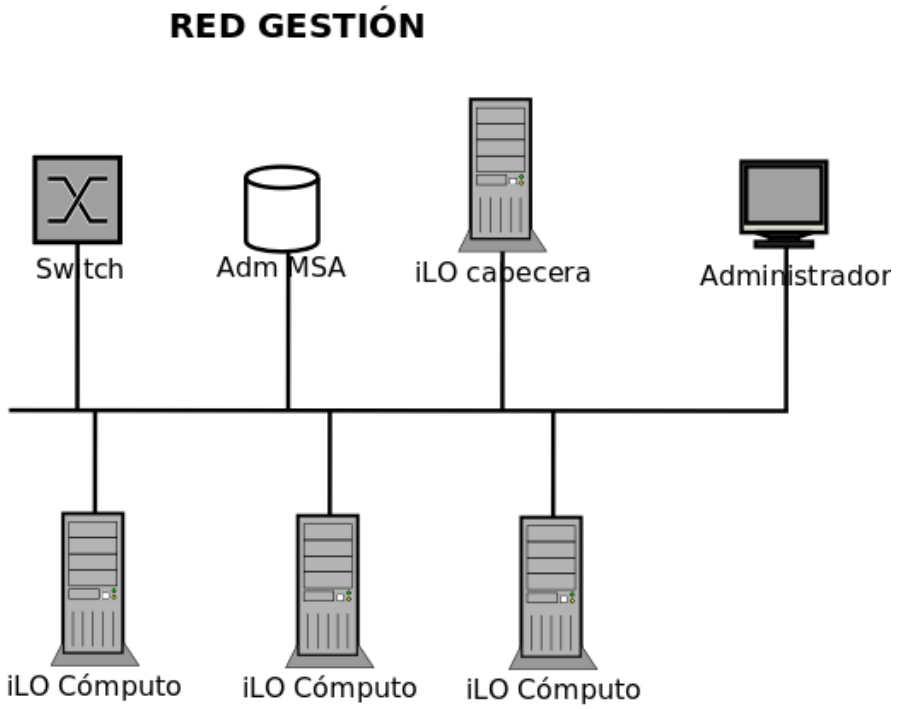


FIGURA 5.4: Diagrama lógico red de gestión

- hpc-node4: Puerto 1/1 y puerto 1/2 en bonding y MTU 9000
- hpc-node5: Puerto 1/1 y puerto 1/2 en bonding y MTU 9000

5.1.2.4. Red de gestión

La red de gestión será la misma red empleada en el resto de CEFCA para la gestión de dispositivos. En este caso será necesario conectar la gestión del switch, la gestión de las controladoras de la matriz de discos y las iLO de todos los servidores. Como se vio anteriormente, esta red únicamente es accesible por los administradores de sistemas mediante una conexión física a dicha red.

En este caso, no necesitaremos definir una nueva red, sino que asignaremos unas direcciones en la red de gestión ya existente. Quedará del siguiente modo:

- Switch: 10.10.10.26
- iLO Nodo cabecera: 10.10.10.80
- Matriz de discos, controladora A: 10.10.10.81
- Matriz de discos, controladora B: 10.10.10.82
- iLO Nodo cómputo 1: 10.10.10.83

- iLO Nodo cómputo 2: 10.10.10.84
- iLO Nodo cómputo 3: 10.10.10.85
- iLO Nodo cómputo 4: 10.10.10.86
- iLO Nodo cómputo 5: 10.10.10.87

Las conexiones se realizarán al switch 3750X dedicado del HPC. Las conexiones quedarán del siguiente modo:

- iLO Nodo cabecera: Puerto iLO
- Matriz de discos, controladora A: Puerto Mgmt
- Matriz de discos, controladora B: Puerto Mgmt
- iLO Nodo cómputo 1: Puerto iLO
- iLO Nodo cómputo 2: Puerto iLO
- iLO Nodo cómputo 3: Puerto iLO
- iLO Nodo cómputo 4: Puerto iLO
- iLO Nodo cómputo 5: Puerto iLO

5.1.2.5. Segmentación del switch

Para separar los dominios de colisión de las redes, procederemos a segmentar el switch 3750X en vlanes. Las vlanes necesarias quedarán del siguiente modo:

- vlan 85: hpc_general
- vlan 14: hpc_almacenamiento
- vlan 15: hpc_mpi
- vlan 10: gestion

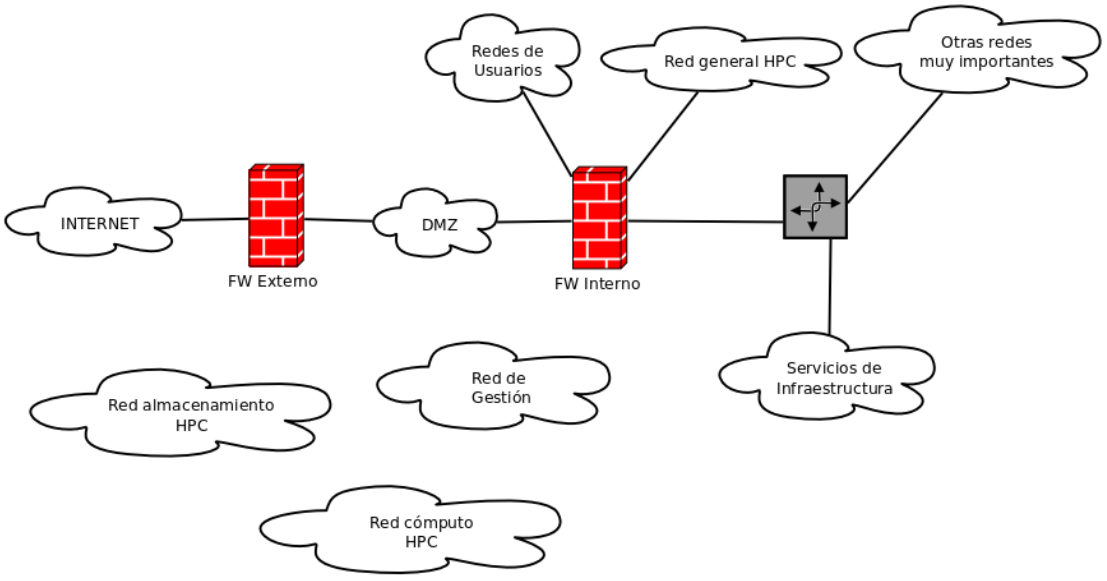


FIGURA 5.5: Diagrama lógico red CEFCA con HPC

5.1.3. Conexión del clúster con infraestructura CEFCA

Ya hemos visto que la red general será la única red que será direccionable del HPC y hemos visto que poníamos una interfaz del cortafuegos FW Interno en dicha red. Pues bien, el firewall será la pasarela por defecto de todos los nodos del clúster y, para la ruta de vuelta, el enrutamiento en todos los equipos de comunicaciones de CEFCA hacia la red 10.50.85.0/24 se realizará por el cortafuegos mencionado. Podemos ver un diagrama completo de la red. En este caso vemos que se ha agregado una nueva "pata" al cortafuegos FW Interno con una conexión en la red general HPC. Podemos ver también "flotando" sin conexión las redes de almacenamiento y cómputo hpc. Esto se debe a que no tienen conexión con ninguna red.

Actualmente el rack en el que se instalará el clúster HPC y el rack de comunicaciones general de infraestructura de CEFCA se encuentran en lugares separados y el único medio físico para unir ambas ubicaciones es un tirado de fibra óptica monomodo. Para poder llevar a cabo el diseño lógico:

- Daremos de alta la vlan general_hpc con id 85 en el switch de infraestructura de cefca.
- Configuraremos la interfaz de red de firewall y cablearemos a una boca de la vlan general_hpc
- Conectaremos un SFP de fibra monomodo en infraestructura cefca y en switch clúster hpc

- Configuraremos ambos puertos en modo trunk, permitiendo el paso de las vlanes cluster_hpc y gestion.

En los próximos meses se trasladará el equipamiento de infraestructura CEFCA al mismo cuarto en el que se encuentra el clúster HPC. En ese momento se podrá sustituir el SFP de fibra por dos SFP de cobre, realizar una conexión a cada switch del stack de infraestructura y realizar un LACP.

5.1.4. Escenario completo

En la figura 5.6 podemos ver una pequeña simplificación de cómo están distribuidos los sistemas implicados en nuestro proyecto.

Algunos comentarios al respecto:

- las máquinas *dc1* y *dc2* son los controladores de dominio Active Directory y además llevarán los servicios DNS y NTP.
- las máquinas *dhcp1* y *dhcp2* son el clúster que tiene el servicio dhcp.
- la máquina *syslog* es la máquina que recopila los logs de los sistemas.
- la máquina *software* contiene los repositorios apt de *ubuntucefca14*. Además hace de repositorio apt-cacher.
- la máquina *lgdeploy* contiene el servidor con el software con el que se realizará el despliegue automatizado del clúster.
- la máquina *it-monitor* contiene el software de monitorización cacti y nagios con el que se realizará la monitorización de los sistemas.
- la máquina *cluster-info* contendrá la wiki de usuarios con la información, además de las gráficas de utilización del clúster.

5.2. Diseño del sistema de almacenamiento

5.2.1. Análisis del almacenamiento

Ya vimos en la fase de especificación los requerimientos y las restricciones del sistema en cuanto a las necesidades de almacenamiento. A continuación trataremos de recapitular

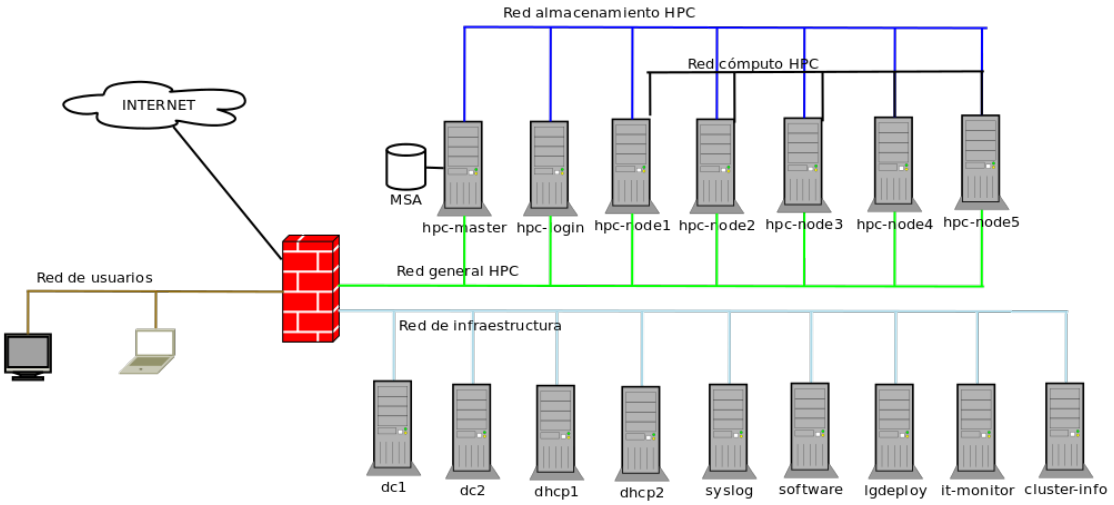


FIGURA 5.6: Escenario completo de red

de los requerimientos las necesidades de conectividad del HPC y hacer un pequeño análisis adicional. Para ello usaremos distintos puntos de vista: desde el punto de vista del usuario del sistema, desde el punto de vista de funcionamiento propio del sistema y desde el punto de vista del administrador del sistema.

Desde el punto de vista del usuario del sistema, el usuario sabe que el espacio de nombres usado para el sistema de archivos es común en todos los nodos del clúster y existen dos tipos de recursos: locales al nodo y compartidos. Los recursos locales podrán usarlos para cálculos intermedios y si desean un acceso rápido de lecturas y escrituras de tipo aleatorio. Además allí podrán encontrar algunos catálogos para tener un acceso más rápido. Los recursos globales a los que podrán acceder serán para el almacenamiento persistente de información y para el almacenamiento de cálculos intermedios que deban estar a disposición de todos los nodos. Además se deberá proveer de un espacio para un almacenamiento global de catálogos que podrá ser accedido desde todos los nodos. De estos recursos globales, para el almacenamiento persistente de información, los usuarios demandarán espacio de almacenamiento y buenas prestaciones de lectura y escritura secuencial. En lo que respecta a cálculos intermedios y catálogos querrán disponer de un acceso más rápido especialmente en lo que a lecturas y escrituras se refiere. El espacio de nombres definido y aprobado por los usuarios que deberá desplegarse será el siguiente:

```
/home/CEFCA/$user           # home de los usuarios (global)

/global                     # espacio de nombres global
        /shared             # almacenamiento persistente compartido
        /catalog            # catálogos globales
                /distrib    # directorio distribución de catálogos
        /scratch            # directorio de scratch

/local                      # espacio de nombres local
        /catalog            # catálogos locales
```

```
/scratch          # directorio de scratch
```

Desde el punto de vista del sistema, el almacenamiento local estará en los discos duros SSD existentes en los nodos en una partición reservada a esta tarea. En lo que respecta al almacenamiento global, deberá emplearse la matriz de discos adquirida conectada al servidor de cabecera, se crearán los volúmenes necesarios y se servirá vía NFS a todos los nodos de cómputo. La conexión desde el servidor cabecera deberá realizarse empleando la doble controladora disponible en la matriz de discos.

Desde el punto de vista del administrador, el diseño de los volúmenes de almacenamiento deberá ser lo más sencillo posible y ser fácilmente escalable si se agregan posteriormente cajones de discos. Además se instalará algún sistema de cuotas al almacenamiento persistente para que los usuarios no abusen de los recursos. Por último, se deberán reservar discos de global spare para que formen parte de los volúmenes de manera automática en el momento en que falle algún disco.

Una vez identificadas las necesidades, listaremos los elementos con los que contamos para implementar el almacenamiento del clúster:

- Servidor Cabecera:
  - 1 HBA SAS con 2 puertos SAS
- Matriz de discos:
  - Controladora A: 4 puertos SAS a 12Gbps
  - Controladora B: 4 puertos SAS a 12Gbps
  - 24 discos MDL-SAS SFF de 1 TB de capacidad
- Nodos de cómputo
  - Disco SSD HP 200GB 6G SATA 2.5

5.2.2. Diseño del almacenamiento compartido

5.2.2.1. Diseño de los volúmenes

Tras realizar diversas simulaciones de posibilidades que admite la cabina para crear los array de discos y ver diversos documentos y benchmarkings disponibles sobre la MSA, finalmente se opta por crear dos volúmenes independientes a nivel de discos. Esto nos permitirá proveer de las características demandadas por los usuarios y poder ampliarse

posteriormente fácilmente (moviendo uno de los volúmenes a una nueva bandeja de discos). La asignación de volúmenes y discos será:

- **Volumen persistente o de usuarios**: este volumen será el que contenga la información persistente de los usuarios y que por lo tanto deberá tener cierta fiabilidad. Además deberá disponer de bastante tamaño útil y responder bien a lecturas y escrituras secuenciales. Para este volumen se opta por RAID 6 (ver Estado del Arte) ya que es el modelo de RAID que mejor encaja en estos requisitos. Para realizarlo usaremos 14 discos de la cabina.
- **Volumen scratch**: en este volumen se almacenará tanto el scratch como los catálogos. No importa la fiabilidad de la información en este volumen, lo que se buscará es una buena respuesta a lecturas y escrituras aleatorias. Para este volumen se opta por RAID 10 (ver Estado del Arte) ya que es el modelo que mejor encaja en estos requisitos. Para realizarlo se usarán 8 discos de la cabina.
- **Discos spare**: se dejarán los dos discos restantes como global spare.

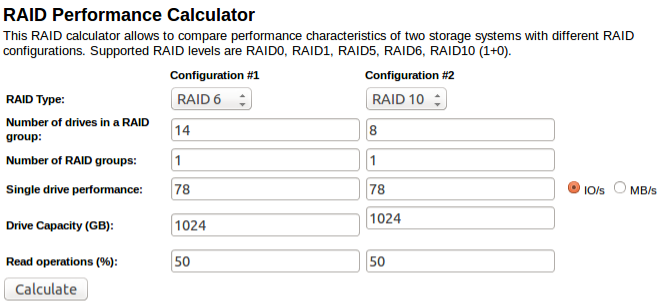
Para facilitar la posterior ampliación, los discos del volumen persistente serán continuos y anterior al volumen de scratch. Esto permitirá que eliminemos fácilmente el scratch (no nos importa la información que hay en él), lo creemos en la nueva bandeja de discos y que ampliemos el volumen persistente.

Además de esta separación a nivel de matriz, a nivel de sistema operativo crearemos volúmenes lógicos para estos volúmenes, esto nos permitirá una mayor flexibilidad de cara a variar el diseño establecido.

Para el diseño, además de consultar benchmarks del fabricante, se han realizado cálculos orientativos de la configuración seleccionada con una calculadora [1] y así tener una referencia del tamaño útil y del rendimiento:

- El primero es un cálculo comparativo de IOPS de tipo aleatorio que pueden dar los arrays tomando un valor de 78 IOPS por disco y un 50 % de lecturas. Como puede observarse un en la figura 5.7, el array de discos de RAID 10 aun con menos discos ofrece un mejor desempeño para este tipo de operaciones.
- El segundo es un cálculo comparativo de ancho de banda tomando 100MBs por disco y un 80 % de lecturas. Puede verse en la figura 5.8 cómo el array de discos en RAID 6 ofrece un tamaño mayor y un rendimiento de acceso secuencial también mayor (esto es debido al empleo de más discos).

[1] http://wintelguy.com/raidperf.pl



RAID Performance Calculator

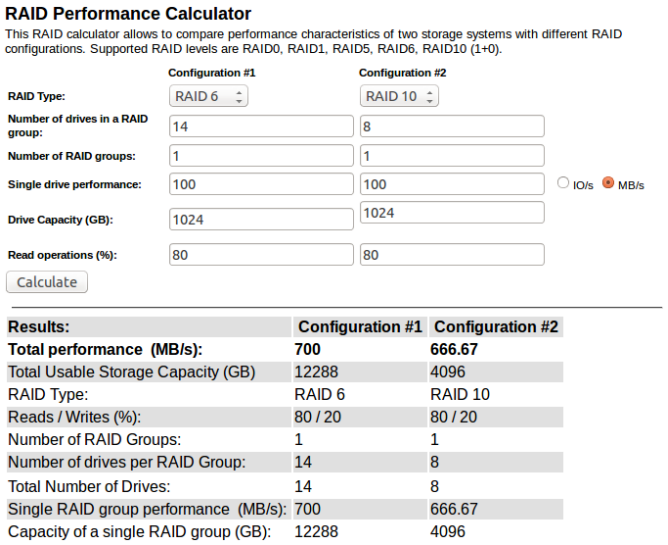
This RAID calculator allows to compare performance characteristics of two storage systems with different RAID configurations. Supported RAID levels are RAID0, RAID1, RAID5, RAID6, RAID10 (1+0).

	Configuration #1	Configuration #2	
RAID Type:	RAID 6	RAID 10	
Number of drives in a RAID group:	14	8	
Number of RAID groups:	1	1	
Single drive performance:	78	78	IO/s MB/s
Drive Capacity (GB):	1024	1024	
Read operations (%):	50	50	

Calculate

Results:	Configuration #1	Configuration #2
Total performance (IO/s):	**312**	**416**
Total Usable Storage Capacity (GB)	12288	4096
RAID Type:	RAID 6	RAID 10
Reads / Writes (%):	50 / 50	50 / 50
Number of RAID Groups:	1	1
Number of drives per RAID Group:	14	8
Total Number of Drives:	14	8
Single RAID group performance (IO/s):	312	416
Capacity of a single RAID group (GB):	12288	4096

FIGURA 5.7: Cálculo de iops acceso aleatorio



RAID Performance Calculator

This RAID calculator allows to compare performance characteristics of two storage systems with different RAID configurations. Supported RAID levels are RAID0, RAID1, RAID5, RAID6, RAID10 (1+0).

	Configuration #1	Configuration #2	
RAID Type:	RAID 6	RAID 10	
Number of drives in a RAID group:	14	8	
Number of RAID groups:	1	1	
Single drive performance:	100	100	IO/s MB/s
Drive Capacity (GB):	1024	1024	
Read operations (%):	80	80	

Calculate

Results:	Configuration #1	Configuration #2
Total performance (MB/s):	**700**	**666.67**
Total Usable Storage Capacity (GB)	12288	4096
RAID Type:	RAID 6	RAID 10
Reads / Writes (%):	80 / 20	80 / 20
Number of RAID Groups:	1	1
Number of drives per RAID Group:	14	8
Total Number of Drives:	14	8
Single RAID group performance (MB/s):	700	666.67
Capacity of a single RAID group (GB):	12288	4096

FIGURA 5.8: Cálculo de ancho de banda acceso secuencia

5.2.2.2. Conexionado con la cabina

Para el conexionado se usarán ambos puertos de la HBA SAS de la controladora pinchada en el servidor de cabecera. Conectaremos físicamente uno a cada controladora según la configuración de la figura 5.9.

Cuando se realizan dos conexiones en un servidor que apuntan al mismo almacenamiento, el sistema operativo las verá como volúmenes distintos. Por ello es necesario usar un software de *multipath* que se integre con el sistema operativo. Este software ofrecerá una abstracción sobre las rutas que apuntan al mismo volumen de almacenamiento y se encargará de detectar si una de las rutas no está disponible. En este caso usará la que

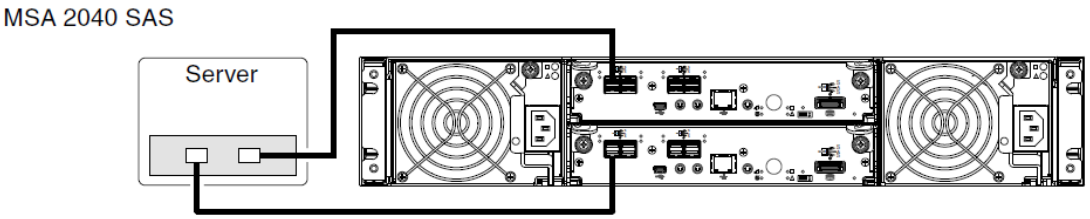


FIGURA 5.9: Conexión HBA del servidor con MSA 240 SAS

lo está. Además permitirá distribuir la carga entre distintas rutas para optimizar el rendimiento.

En la documentación sobre la MSA se recomienda una configuración multipath que use una política round robin. Esto hará que las peticiones se vayan despachando por una o por otra controladora en cada petición, realizando un uso activo de ambas controladoras y distribuyendo la carga. Esta configuración multipath con round robin es la que realizaremos en el sistema operativo.

5.2.2.3. Sistemas de archivos, estructura y compartición NFS

Para el uso de ambos volúmenes se formateará con el sistema de archivos EXT4. Se estuvo valorando el uso de otro sistema de archivos como XFS pero finalmente por simplicidad, por el relativamente pequeño tamaño de los volúmenes y la falta de unos requerimientos más definidos, se optará por el oficialmente soportado por la distribución. Al sistema de archivos persistente le agregaremos soporte de cuotas y realizaremos algunas pequeñas optimizaciones como que no se actualicen los timestamp con los accesos.

Una vez creados los filesystem en el servidor de cabecera, se montarán los volúmenes en `/mnt/vg_users` y `/mnt/vg_scratch` y se creará la siguiente estructura de directorios

```
/mnt/vg_users/homes
/mnt/vg_users/shared
/mnt/vg_users/software
/mnt/vg_scratch/scratch
/mnt/vg_scratch/catalog
```

Posteriormente, se realizarán los montajes necesarios internamente en el servidor y se realizará la compartición vía NFS, quedando los recursos estarán disponibles de la siguiente manera:

```
/export/homes
/export/software
/export/shared
/export/catalog
/export/scratch
```

Para no ocupar recursos del servidor NFS cuando los nodos estén ociosos, para mejorar la resilencia ante posibles caídas de los recursos y para simplificar la administración, se hará uso del software automount en los nodos del clúster para que realice el montaje automático cuando un proceso demande un fichero o directorio que se encuentre en el espacio de direcciones compartido. Para ello configuraremos en este software los recursos exportados por NFS y las trayectorias especificadas.

5.2.2.4. Vista global del almacenamiento compartido

Como recapitulación, en la figura 5.10 puede verse la estructura de capas en las que se define el almacenamiento compartido. En el nivel más bajo encontramos los propios arrays de discos que son vistos por cada una de las controladoras de la matriz. A continuación tenemos la conexión realizada de la HBA contra cada una de las controladoras. Aquí el driver del sistema operativo ofrecerá 4 dispositivos de bloque en total. Esos dispositivos serán tratados por el multipath y se encargará de mostrar tan sólo dos dispositivos de bloque. Crearemos dos grupos de volumen, uno de usuarios (almacenamiento persistente) y otro de scratch y añadiremos los dispositivos a cada uno de los volúmenes y crearemos los volúmenes lógicos pertinentes. Esos volúmenes lógicos serán tratados por el sistema operativo como nuevos dispositivos de bloque, en dichos dispositivos crearemos el sistema de archivos y realizaremos los puntos de montaje. Finalmente el servidor NFS exportará los recursos. Estos recursos serán lo que finalmente verán los nodos de cómputo del clúster.

5.2.2.5. Limitaciones encontradas

Como vimos según los cálculos orientativos de velocidad del sistema, el sistema puede llegar a dar en disco hasta 700MB en secuencial (asumiendo combinación de 80/20 lecturas/escrituras). Esto es perfectamente soportado por las conexiones SAS 12Gbps existentes pero los nodos jamás van a poder obtener ese rendimiento debido a la red.

El máximo teórico de transferencia de una interfaz de 1G con una MTU de 9000 es de 123 MB/s, por lo que en el servidor de cabecera con cuatro interfaces podrá servir a un máximo teórico de 492MB/s. Este máximo teórico es inferior al throughput que calculamos para arrays, indicándonos por tanto que la red es el factor limitante del sistema.

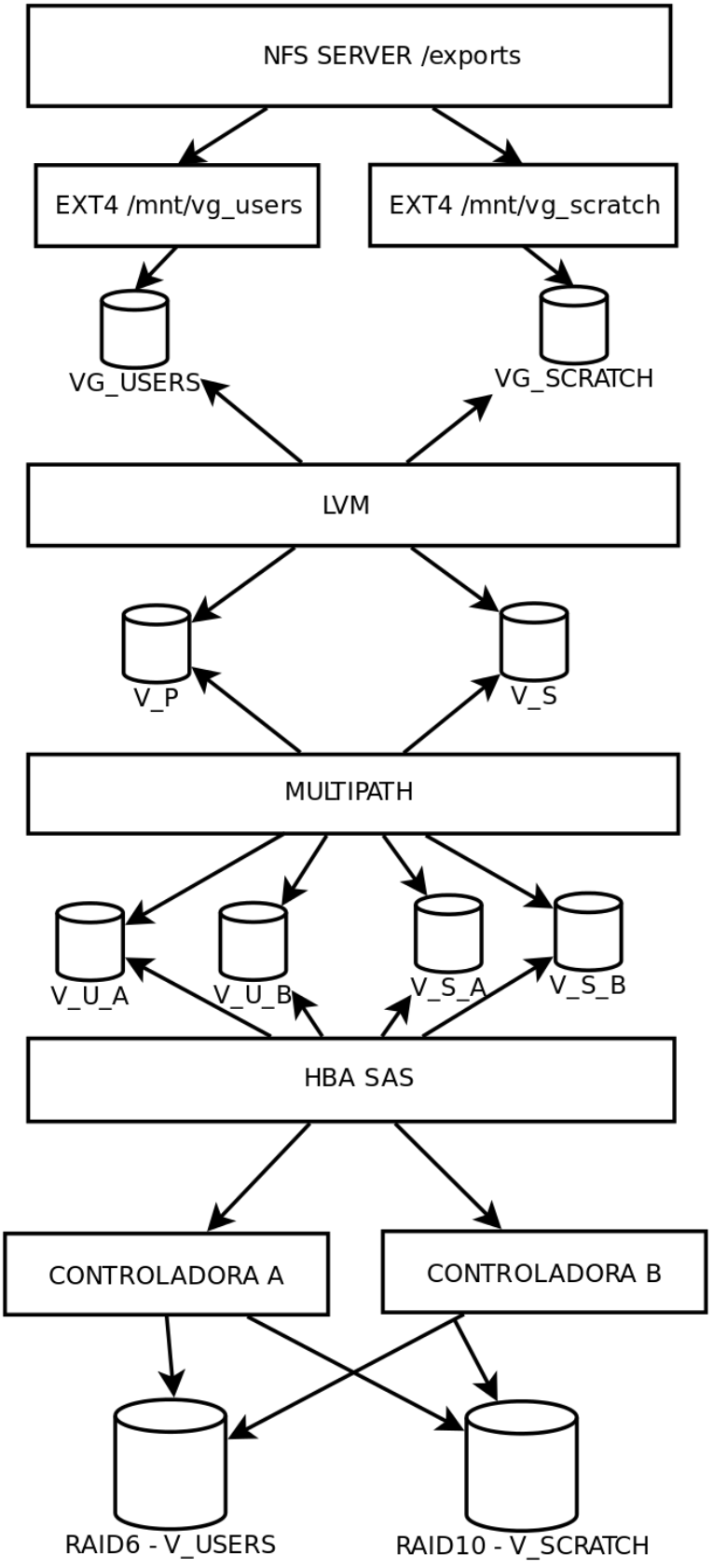


Figura 5.10: Vista global del sistema de almacenamiento

5.2.3. Diseño del almacenamiento local

5.2.3.1. Creación y montaje

Para el almacenamiento local simplemente se realizará una partición en los discos duros SSD locales y se realizará un montaje vía fstab en `/local`. Posteriormente se crearán los directorios de la estructura requerida.

```
/local/scratch
/local/catalog
```

5.2.3.2. Espacio y rendimiento

Los discos son de 200GB y se ha definido una partición de 12GB para el sistema operativo, luego el punto de montaje local tendrá 188GB.

El rendimiento del disco SSD según el fabricante es de:

- Random Reads: 64000 IOP/s
- Random Writes: 4700 IOP/s
- Sequential Reads: 410 MiB/s
- Sequential Writes: 130 MiB/s

5.3. Instalación física y configuración inicial

Una vez que tenemos definido el diseño general del clúster, ya podemos realizar la instalación física.

5.3.1. Enrackado y cableado

Tras finalizar las labores de acondicionamiento en el laboratorio de informática se lleva el rack Dell físicamente, se realiza una pequeña planificación del rack y posteriormente se llama al personal de HP para que realice el enrackado de los servidores. Sin embargo hay algunos problemas, en un primer momento no se logra enrackar el chasis ya que las guías que venían eran para racks HP. Posteriormente se piden unas guías de rack estándar y al cabo de unos días se procede a la instalación y, como el chassis tiene mucho fondo, deja varios conectores schuko de las PDU inutilizables. Después se lleva a cabo el

enrackado del nodo de cabecera y la matriz de discos y se deja un espacio inferior por si se agregan más bandejas de discos o se agrega alguna SAI en el futuro.

Tras el enrackado físico se lleva a cabo la instalación del cableado. Para ello se realiza una planificación que incluye la instalación de organizadores de cables, el etiquetado y la confección de un pequeño código de colores.

En la figura 5.11 puede verse una serie de fotos del proceso completo de enrackado y cableado del clúster HPC.

5.3.2. Configuración del switch

Vamos a omitir el proceso de configuración inicial del switch de:

- conectar vía serie mediante la utilidad `minicom`
- fijar hostname y dominio
- habilitar usuarios
- habilitar acceso SSH
- banner, etc.

Y nos centraremos en la parte más relevante. El proceso de configuración consiste en:

- entrar en modo configuración
- realizar la configuración
- salir del modo configuración
- comprobar que todo funciona correctamente
- guardar en la memoria persistente la configuración actual

Valga el siguiente ejemplo:

```
cefca-swt-013#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
cefca-swt-013(config)#vlan 10
cefca-swt-013(config-vlan)#name Gestion
cefca-swt-013(config-vlan)#exit
cefca-swt-013(config)#exit
cefca-swt-013#wr
Building configuration...
[OK]
```

FIGURA 5.11: Instalación física del clúster HPC

5.3.2.1. Configuración nivel 2

En primer lugar definiremos las vlan que tendrá este switch de acuerdo a lo visto en el diseño.

```
vlan 10
 name Gestion
vlan 13
 name Cluster_Mpi
vlan 14
 name Cluster_Storage
vlan 85
 name Cluster_General
```

5.3.2.2. Configuración nivel 3

En este switch no vamos a realizar labores de enrutamiento en capa 3, sin embargo será necesario darle una ip en la vlan de gestión para que podamos gestionarlo por ssh, para ello definiremos la interfaz.

```
interface Vlan10
 ip address 10.10.10.26 255.255.255.0
 no ip route-cache
 no ip mroute-cache
```

5.3.2.3. Configuración del enlace

Configuraremos el enlace con el otro switch, para ello simplemente pondremos el puerto de uplink usado en modo trunk.

```
interface GigabitEthernet1/1/2
 description RPS2_FA ; CEFCA-SWT-006 ; Gi1/1/4 ; Enlace con stack
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
```

5.3.2.4. Configuración de los puertos

Ahora procederemos a configurar los puertos del switch. Usando el conexionado que hemos utilizado en el proceso de instalación procederemos a realizar la configuración oportuna de cada una de las redes.

Puertos red de gestión Ejemplo de configuración de los puertos de gestión.

```
interface GigabitEthernet1/0/30
 description ; cefca-sv-013 ; ILO ; Conexion a red gestion
 switchport access vlan 10
 switchport mode access
 spanning-tree portfast
!
        ....
```

Puertos red general Ejemplo de configuración de los puertos de red general.

```
interface GigabitEthernet1/0/4
 description ; cefca-sv-015 ; gi0/1 ; Conexion a red general
 switchport access vlan 85
 switchport mode access
 spanning-tree portfast
!
        ....
```

Puertos red de cómputo Ejemplo de configuración de los puertos de red de cómputo.

```
interface GigabitEthernet1/0/18
 description ; cefca-sv-013 ; gi0/2 ; Conexion a red mpi
 switchport access vlan 13
 switchport mode access
 spanning-tree portfast
!
        ....
```

Puertos red de almacenamiento Ejemplo de configuración de los puertos de red de almacenamiento.

```
interface GigabitEthernet1/0/29
 description ; cefca-sv-013 ; gi1/1 ; Conexion a almacenamiento
 switchport access vlan 14
 switchport mode access
 spanning-tree portfast
!
        ....
```

Bonding de puertos En primer lugar será necesario definir las interfaces portchannel que serán necesarias para realizar la agregación de enlaces. Los portchannel se asignarán de forma consecutiva.

```
interface Port-channel2
 description ; cefca-sv-013; ; Port-channel interfaz computo
 switchport access vlan 13
 switchport mode access
 spanning-tree portfast
!
         ....
```

Luego iremos a los puertos asociados y los vincularemos al portchannel indicando el channel-group.

```
interface GigabitEthernet1/0/18
 description ; cefca-sv-013 ; gi0/2 ; Conexion a red mpi
 switchport access vlan 13
 switchport mode access
 spanning-tree portfast
 channel-group 2 mode active
!
interface GigabitEthernet1/0/31
 description ; cefca-sv-013 ; gi0/2 ; Conexion a red mpi
 switchport access vlan 13
 switchport mode access
 spanning-tree portfast
 channel-group 2 mode active
!
         ....
```

Definiremos la política usada para el balanceo de los bondings (se explicará en el apartado de mediciones y ajustes).

```
port-channel load-balance src-ip
```

5.3.2.5. Modificación del tamaño de la mtu

Modificaremos el tamaño de la MTU Jumbo para admitir jumbo frames.

```
# show system mtu

System MTU size is 1500 bytes
System Jumbo MTU size is 1500 bytes
System Alternate MTU size is 1500 bytes
Routing MTU size is 1500 bytes

# system mtu jumbo 9000
Changes to the system jumbo MTU will not take effect until the next reload is
    done
# reload
```

5.3.3. Configuración hardware los nodos

5.3.3.1. Actualización firmware del servidor

En primer lugar, deberemos de descargar y crear un USB autoarrancable desde nuestra máquina de gestión. Para ello bajaremos el último Service Pack disponible, que en el momento de la instalación fue el HP Service Pack for ProLiant 2014.09.0.

Antes de proceder a la actualización del firmware, comprobaremos que el servidor funciona correctamente. Para ello simplemente conectaremos un monitor externo y comprobaremos que el servidor arranca y no da ningún error en el proceso POST de verificación tal y como puede verse en la figura 5.12 .

FIGURA 5.12: Proceso POST de arranque de servidor Proliant

Tras esta verificación, procedemos a introducir el USB y pulsar la tecla F11 que nos mostrará el menú de arranque. Nos preguntará si deseamos realizar la actualización automática o la interactiva. Responderemos que la interactiva. Tras aceptar el acuerdo de licencia, tendremos el menú principal en el que seleccionaremos *Firmware Update*. Tras esto, el sistema empezará un proceso de tres pasos, en primer lugar escaneará el firmware disponible en el Service Pack y posteriormente, comprobará qué es lo que se encuentra instalado en el nodo (como puede verse en la figura 5.13). Después nos permitirá seleccionar los componentes que queremos actualizar, seleccionaremos todos y pasaremos al siguiente paso en el que el sistema automáticamente realizará el despliegue de las actualizaciones. Tras realizarse este proceso y ver que no ha dado ningún error, se reiniciará el servidor y se verificará que el sistema arranca correctamente.

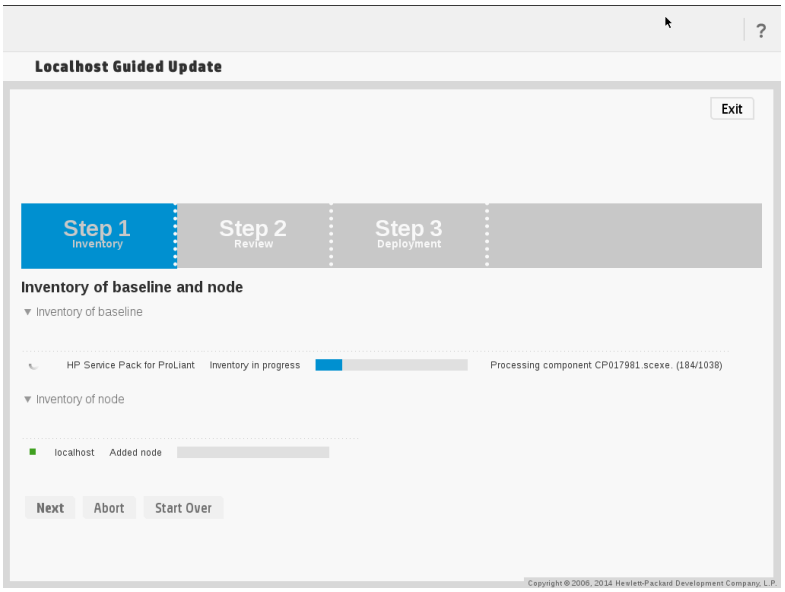


FIGURA 5.13: Proceso de actualización de firmware de servidor

NOTA: en el despliegue, se realizó el modo automático. Aparentemente fue todo bien, pero tras instalar los sistemas operativos, el módulo de kernel del driver de las tarjetas de red en algunos de los servidores daba un error de checksum al cargar. Se solucionó el problema volviendo iniciar el proceso de actualización de firmware, pero esta vez de forma Interactiva y seleccionando a que forzase la reinstalación del firmware de las tarjetas de red.

5.3.3.2. Configuración inicial iLO

Para proseguir con la configuración del servidor, lo mejor es configurar la interfaz de red de la iLO y seguir realizando el resto de tareas de forma cómoda desde nuestra workstation en red de gestión. Para ello durante el proceso de arranque pulsaremos la tecla F8 que nos mostrará una pequeña interfaz tipo ncurses desde la que podremos configurar la interfaz de red de la iLO. Seleccionaremos que no usamos DHCP y le pondremos la ip estática del servidor sin indicar una pasarela como puede verse en la figura 5.14.

5.3.3.3. Configuración iLO

Una vez realizada la configuración inicial, procederemos a conectarnos desde un navegador usando https://direccion_ip_ilo. Iniciaremos sesión con el usuario Administrator y la clave que por defecto escribe HP en los servidores y procederemos a realizar la configuración. En la figura 5.15 puede verse el aspecto de la pantalla de login de una ILO.

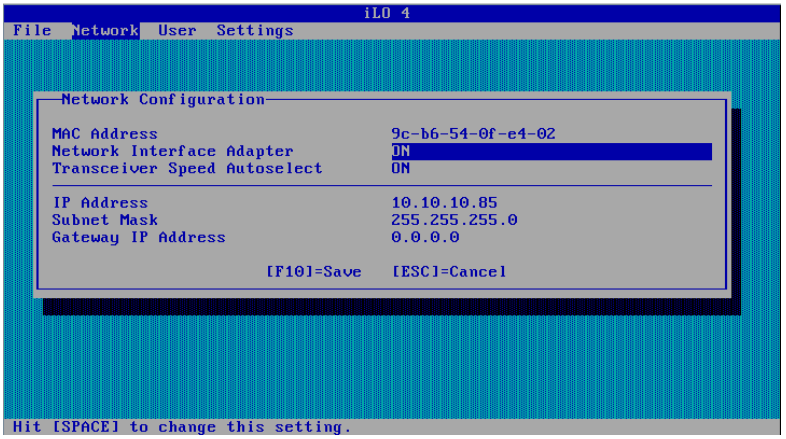


FIGURA 5.14: Configuración inicial de la iLO

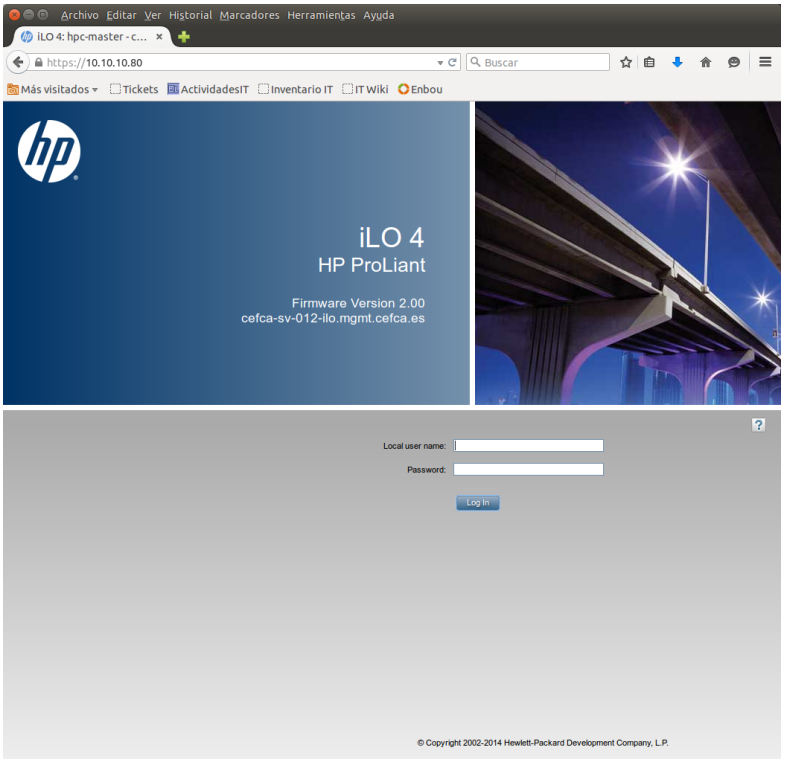


FIGURA 5.15: Login web de la iLO

Introducción de la clave de licencia Para ello nos iremos a `Administration -> Licensing`. Introduciremos la clave de licencia que nos proporcionó HP y pulsaremos sobre el botón `Install`.

Configuración de acceso Iremos a `Administration -> Access Settings`. Aquí configuraremos las opciones de acceso, habilitando IMPI, SNMP y la gestión de la iLO desde el arranque. Será importante indicar el nombre del servidor y su FQDN completo, que como veremos nada tiene que ver con el hostname de la iLO, que configuraremos a continuación. Puede verse en la figura ILO - Access Settings.

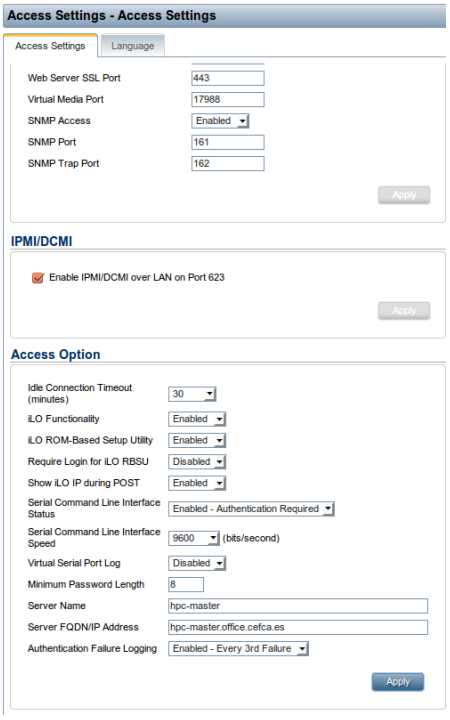


FIGURA 5.16: Opciones de acceso de la iLO

Configuración de red de la iLO Iremos a `Network -> iLO Dedicated Network Port`. Nos iremos a la pestaña `General` y configuraremos el hostname de la iLO y su dominio, asegurándonos de que se encuentra chequeada la opción de usar el puerto dedicado. Puede verse en la figura 5.17. Después en la pestaña `IPv4` comprobaremos la dirección IP y deshabilitaremos todas las opciones de registro y ping a la pasarela. En la pestaña `IPv6` desactivaremos todos los parámetros ya que no usaremos el protocolo ipv6. Finalmente en la pestaña `SNTP` indicaremos el servidor NTP disponible en la red de gestión, esto es muy importante especialmente por la fiabilidad de los logs y al depurado de problemas. Puede observarse en la figura 5.18. En cada pestaña hay que aplicar pulsando en `Submit`. Una vez realizados estos cambios, será necesario reiniciar la iLO, para ello pulsaremos el botón `Reset`. Al cabo de unos minutos podremos volver a contectarnos.

Creación de usuarios administradores Iremos a `Administration -> User Administration` . Sobre `Local Users` pulsaremos el botón `New`. Puede observarse en la figura 5.19 la creación de un usuario con todos los privilegios marcados, tras pulsar el botón `Create User`, el usuario administrador estará creado. En nuestro despliegue, hemos optado por usar usuarios locales de cada iLO en lugar de unir al directorio.

Creación de certificado SSL y firma con XCA Iremos a `Administration -> Security` . Sobre la pestaña `SSL Certificate` rellenaremos los campos de nuestro certificado como

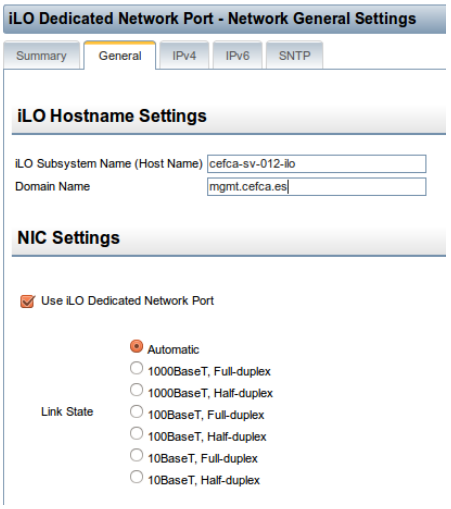


FIGURA 5.17: Configuración general de red de la iLO

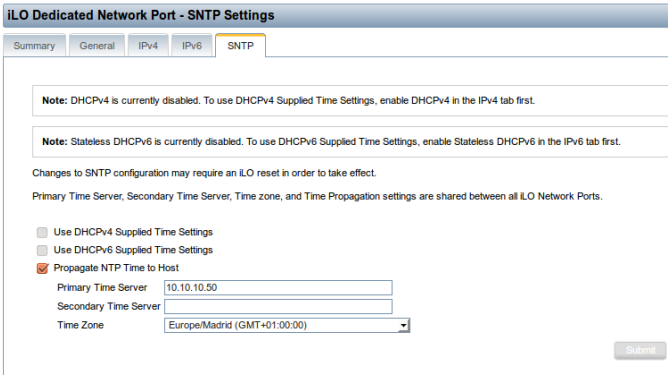


FIGURA 5.18: Configuración de tiempo de la iLO

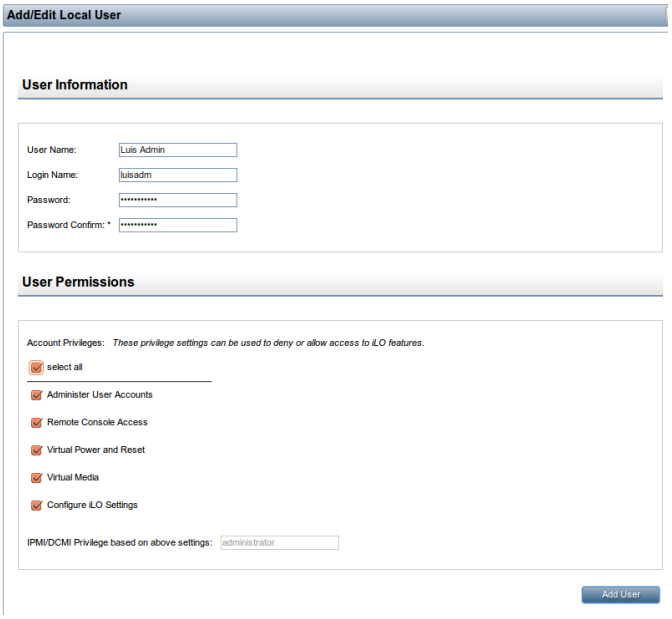


FIGURA 5.19: Creación de usuarios en la iLO

puede observarse en la figura 5.20 y pulsaremos el botón `Generate CSR`. Esperaremos unos minutos y volveremos a pulsar sobre `Generate CSR`. Nos aparecerá la solicitud de firma que copiaremos y pegaremos sobre un fichero de texto. Posteriormente utilizaremos la utilidad XCA para firmar la solicitud del certificado con nuestra CA y aplicando la plantilla de servidor SSL como puede observarse en la figura 5.21. Una vez firmado el certificado, lo exportaremos seleccionando la opción de formato `PEM with Certificate chain` para así incluir el certificado de la CA. Ahora podremos pulsar el botón `Import Certificate` de la interfaz de gestión de la iLO y copiaremos y pegaremos el contenido del fichero exportado en el cuadro de diálogo que nos presenta. Una vez realizada la importación, la iLO se reiniciará con el nuevo certificado cargado.

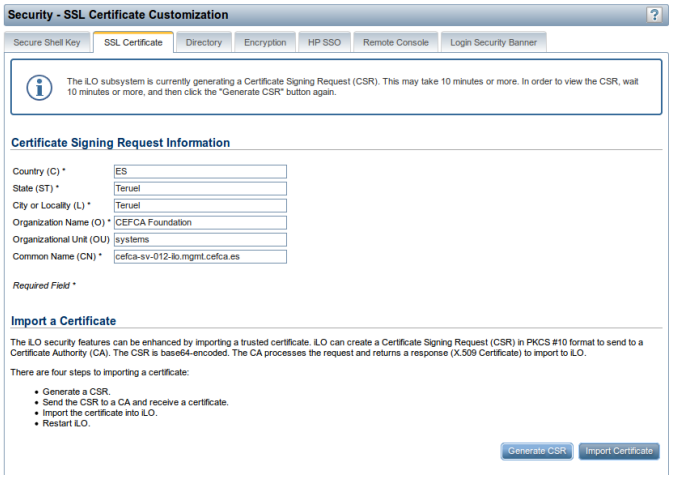


FIGURA 5.20: Configura certificado SSL en la iLO

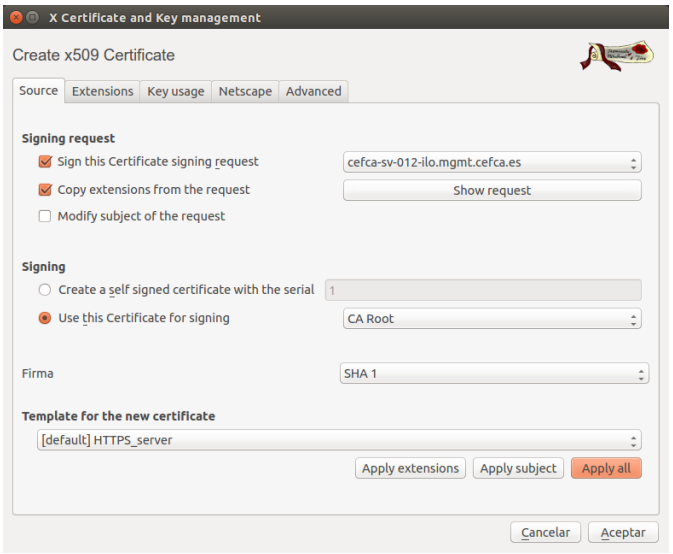


FIGURA 5.21: Firma de CSR con XCA

Configuración de alertas por email Para ello iremos al menú `Administration -> Management`. En la pestaña `AlertMail` configuraremos las alertas por email enviándoselas al

servidor smtp smarthost que existente en la red de gestión. Puede verse en la figura 5.22.

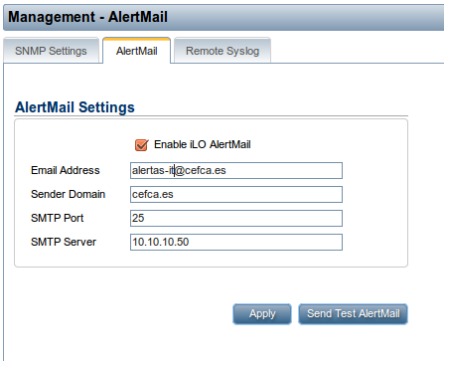


FIGURA 5.22: Configura alertas por email en la iLO

Configuración de grupo multicast Para ello iremos al menú `Administration -> iLO Federation`. Seleccionaremos el grupo `DEFAULT` existente y pulsaremos `Delete`. Una vez confirmada la acción, pulsaremos sobre el botón `Join Group`, que nos llevará a la pantalla de creación de un grupo. Allí daremos el nombre a nuestro grupo, que será `ILO-HPC` y una contraseña para el grupo. Además indicaremos los permisos que concederemos al grupo en nuestra iLO, en nuestro caso daremos permiso a todo menos a administrar usuarios y modificar opciones. Puede verse en la figura 5.23.

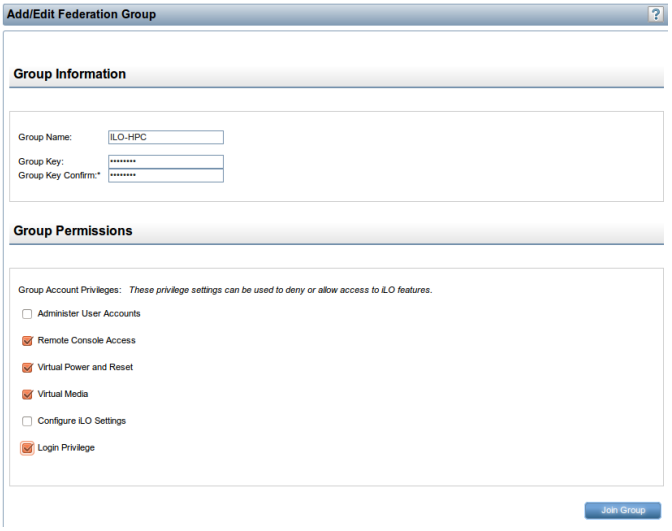


FIGURA 5.23: Agrega a grupo multicast en la iLO

Transcurridos unos minutos de la configuración del grupo multicast en todos los nodos, podremos ver el estado y realizar diferentes tareas administrativas sobre todos los nodos desde el apartado `iLO Federation`. Podemos ver en la figura 5.24 una vista general del estado de todos los nodos del clúster HPC.

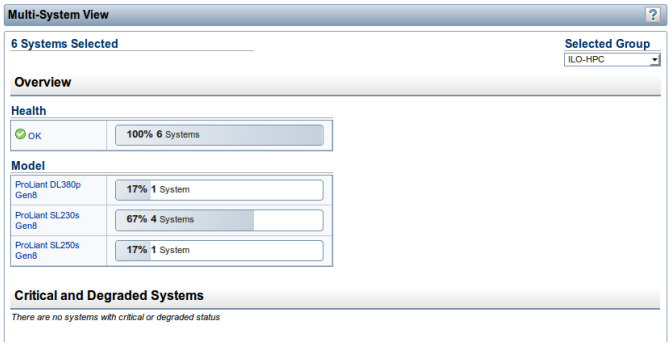


FIGURA 5.24: Muestra a grupo multicast en la iLO

5.3.3.4. Opciones BIOS

Una vez que tenemos ya correctamente configurada la iLO, podemos prescindir completamente del uso de teclado, monitor y ratón. Para ello podremos conectarnos remotamente mediante el menú `Remote Console -> Remote Console` y pulsar el botón `Launch` de la consola que nos interese. En el caso de Ubuntu, es necesario emplear la consola Java. Las consolas disponibles pueden verse en la figura 5.25. Desde esta consola podremos interactuar con el servidor como si estuviesemos delante de él físicamente.

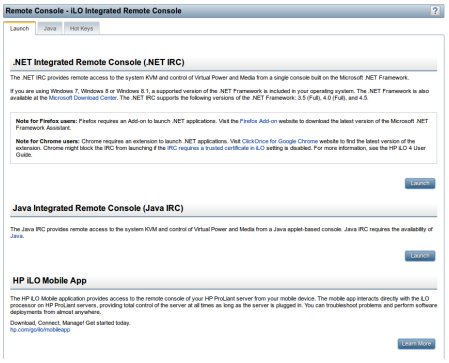


FIGURA 5.25: Acceso a la consola del servidor desde la iLO

Cuando arranque el servidor pulsaremos la tecla F9 para entrar en el Setup de la ROM y realizar los siguientes ajustes. Ya se tratará en el apartado de mediciones y ajustes el porqué de estas opciones. Puede verse en la figura 5.26 algunas de las opciones.

- entraremos en el menú `System Options` para dejar a `Enabled` el `Intel Hyperthreading Options` .
- entraremos en el menú `Power Management Options -> HP Power Profile` y seleccionaremos Maximum Performance
- entraremos en el menú `Power Management Options -> HP Power Regulator` y seleccionaremos `HP Static Performance Mode`

- entraremos en el menú `Advanced Options -> Advanced Performance Tuning Options` y seleccionaremos `Intel Performance Counter Monitor (PCM)`

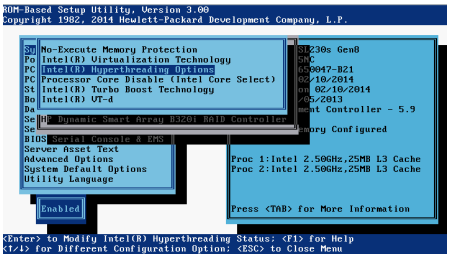


FIGURA 5.26: BIOS de servidor Proliant

5.3.3.5. Configuración RAID

En el caso del servidor de cabecera, el equipo lleva dos discos SAS de 15k que deberemos poner en RAID 1. Para ello pulsaremos la tecla FX para entrar en las opciones del Smart Array P420i integrado que lleva el servidor y al que están conectados ambos discos y procederemos a la creación del volumen RAID 1 con los dos discos. Quedando el estado tal y como se muestra en la figura 5.27.

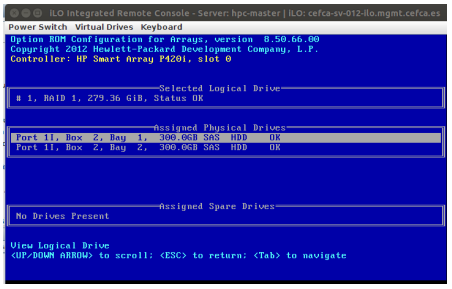


FIGURA 5.27: Creación de RAID en servidor cabecera

5.3.4. Configuración de la matriz de discos

5.3.4.1. Configuración de la controladora

Información del sistema En primer lugar rellenaremos la información del sistema, para ello nos iremos al menú `Configuration -> System Settings -> System Information` y pulsaremos `Apply`. Puede verse en la figura 5.28.

Actualización del firmware Después se instalará el último firmware disponible, para ello desde el menú `Tools -> Update Firmware`, se sube el fichero y se reinician las controladoras.

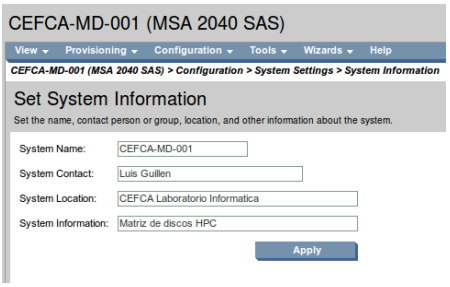


FIGURA 5.28: Información del sistema de la MSA

Configuración de acceso Tras el reinicio, comprobamos el correcto funcionamiento y pasamos a configurar los servicios que proveerán las controladoras, para ello iremos al menú `Configuration -> Services -> Management` y quitaremos los servicios no seguros HTTP y Telnet. Puede verse en la figura 5.29.

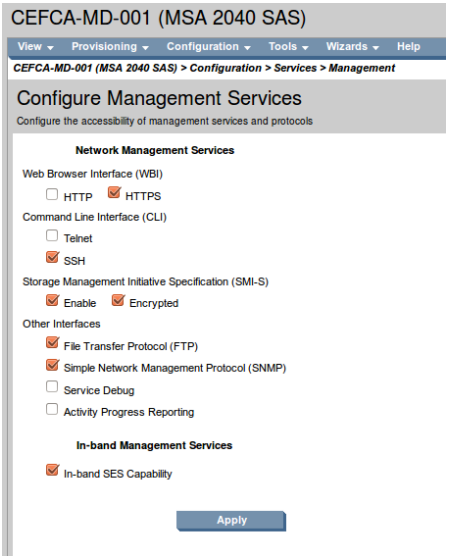


FIGURA 5.29: Gestión de servicios de la MSA

Configuración de red Después de configurar los servicios, pasamos a la configuración de red. En primer lugar configuramos el direccionamiento mediante el menú `Configuration -> System Settings -> Network Interfaces`. Para ello seleccionamos una asignación manual y rellenamos dicha información, pulsando Apply y volviendo a introducir la nueva dirección en el navegador. Puede verse en la figura 5.30.

Tras configurar las direcciones e las controladoras, fijaremos el servidor NTP de la red de gestión. En el menú `Configuration -> System Settings -> Date, Time`.

Creación de administradores Desde el menú menú `Configuration -> Users -> Add New User` crearemos un nuevo usuario con perfil Manage tal y como se ve en la figura 5.31

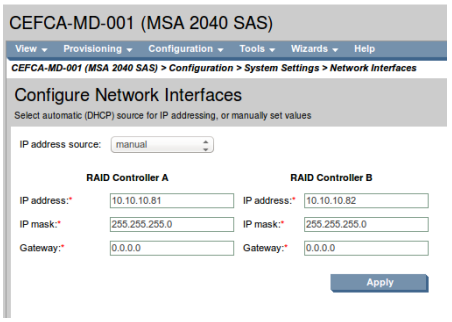


FIGURA 5.30: Configuración de red de la MSA

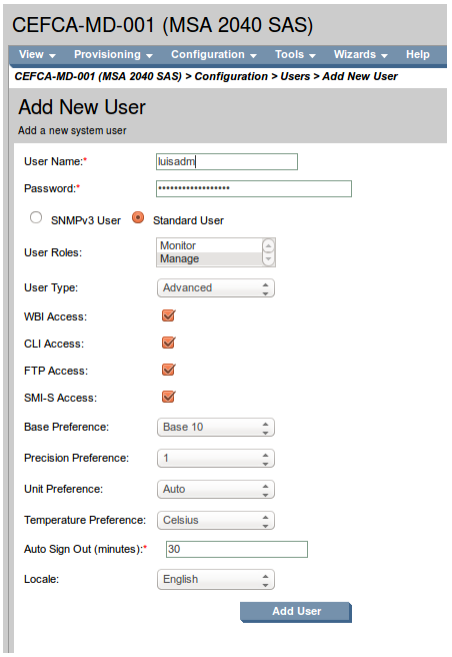


FIGURA 5.31: Creación de usuario administrador en la MSA

Configuración de alarmas Ya se tratará en el apartado de monitorización cómo monitorizaremos la cabina tanto de manera activa como pasiva. De momento dejaremos activo el envío de mensajes vía email como puede verse en la figura 5.32.

5.3.4.2. Configuración del almacenamiento

Configuración de los hosts El proceso de configuración del almacenamiento es muy sencillo. Por un lado tenemos los hosts que, como su nombre indica, son los los hosts que tienen conectividad con las controladoras y que por estar conectado directamente por SAS, las controladoras detectan de manera automática. En nuestro caso tras seleccionar el nodo Hosts del árbol de configuración, veremos dos hosts. Esto es debido a que como vimos, cada controladora está conectado a un puerto de nuestro Host. Puede verse en la imagen 5.33.

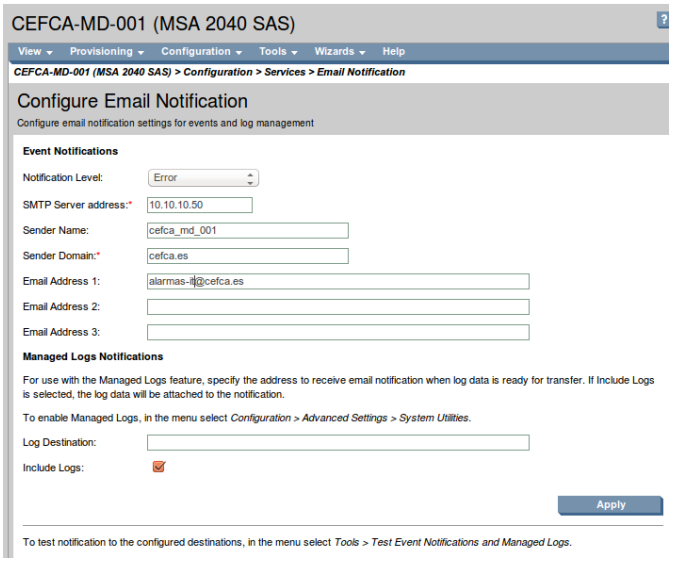


FIGURA 5.32: Configurar notificaciones por email de la MSA

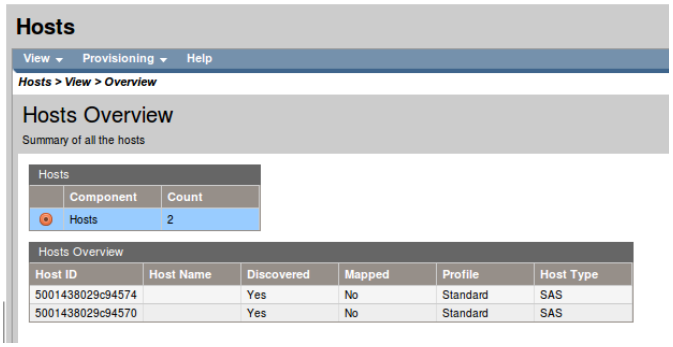


FIGURA 5.33: Configurar hosts en la MSA

Procederemos a darle un nombre a cada host identificándolo con cada uno de los puertos físicos del host, para ello renombrarmos a un nombre sv12-1 o sv12-2 dependiendo del puerto.

Configuración de vdisks y volúmenes A continuación pasaremos a crear los Vdisk y los volúmenes lógicos. Por un lado, los Vdisk serán arrays de discos que se crearán siguiendo alguno de los tipos de RAID disponibles y por otro lado los volúmenes lógicos, que serán volúmenes realizados sobre los Vdisks, pudiendo crear volúmenes con todo el Vdisk. Un Vdisk no podrá ser presentado a un Host, únicamente podrán presentarse los volúmenes. Como veremos, un volumen podrá presentarse a múltiples hosts y a un host se le podrá presentarse múltiples volúmenes.

Procederemos en primer lugar a crear los Vdisk, para ello usaremos el menú `Provisioning -> Create Vdisk` y seleccionaremos el nombre, el tipo de RAID los Sub-vdisk para RAIDs paralelos como RAID 10, el tamaño del chunk y si queremos que realice una inicialización

en línea. Después crearemos los volúmenes en cada Vdisk, ocupando todo el espacio disponible si es necesario y habilitando o deshabilitando el uso de snapshots. En nuestra configuración finalmente se realiza siguiente modo:

- Almacenamiento usuario:
  - Nombre vdisk: users
  - Número de discos: 14
  - Tipo de RAID: RAID6
  - Tamaño chunk: 512KB
  - Volúmenes: users_v0001 100 %
- Almacenamiento scratch
  - Nombre vdisk: scratch
  - Número de discos: 8
  - Tipo de RAID: RAID 10
  - Tamaño chunk: 512KB
  - Volúmenes: scratch_v0001 100 %
- Global spare
  - Número de discos: 2

Puede verse la configuración realizada en la figura 5.34.

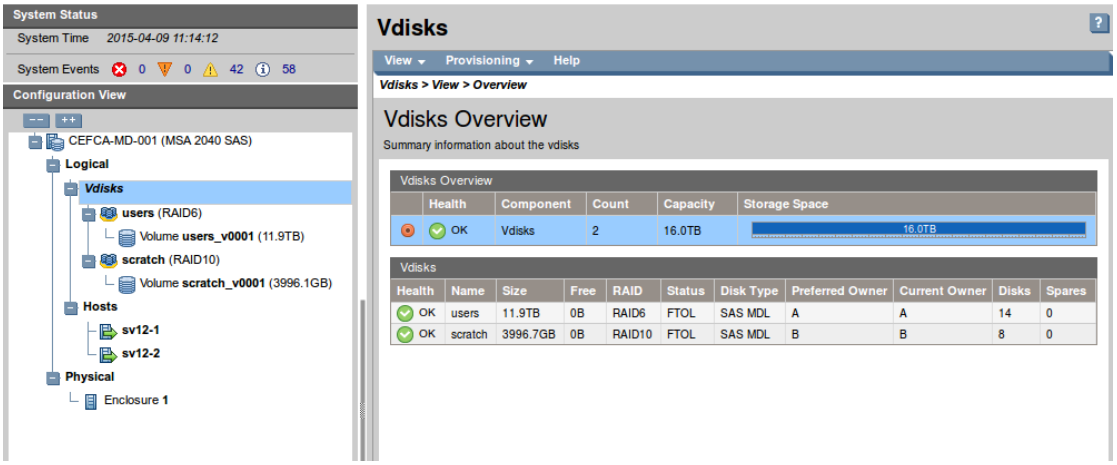


FIGURA 5.34: Configuración final de volúmenes de la MSA

Mapeo de volúmenes a hosts Tras la creación de los volúmenes, podremos presentárselos al host, esto lo haremos desde `Provisioning -> Map Volumes`. En nuestra configuración, mapearemos los dos volúmenes como lectura y escritura a cada uno de los puertos a los que está conectado el servidor. Puede verse en la figura 5.35.

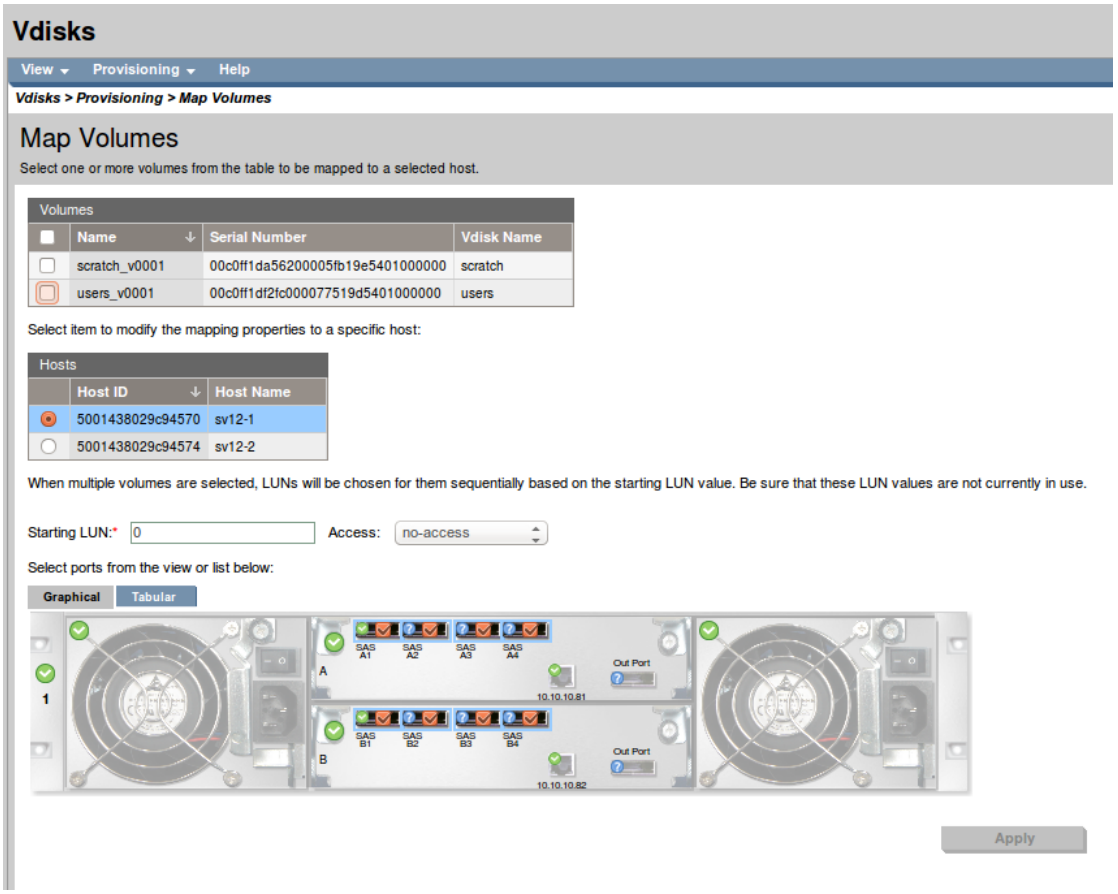


FIGURA 5.35: Mapeo de volúmenes a host en la MSA

Configuración de la caché Configuraremos la caché desde `Configuration -> Advanced Settings -> Cache`. Dejaremos el modo de cache inmediato, deshabilitaremos que el host tenga control de la caché Write-Back y marcaremos las opciones indicadas para el Write Through. Estas opciones ya se tratarán en el apartado dedicado a mediciones y ajustes.

5.3.5. Instalación del sistema operativo en los nodos

Como veremos en una sección posterior, se desarrollará una automatización que desplegará el sistema operativo empleando la red y de forma automática. Pero antes de que esto sea posible es necesario realizar una instalación en un par de los nodos para realizar las configuraciones y pruebas necesarias, por ello procederemos a instalar el sistema operativo de manera manual. Para hacer esto de una manera cómoda y de manera remota,

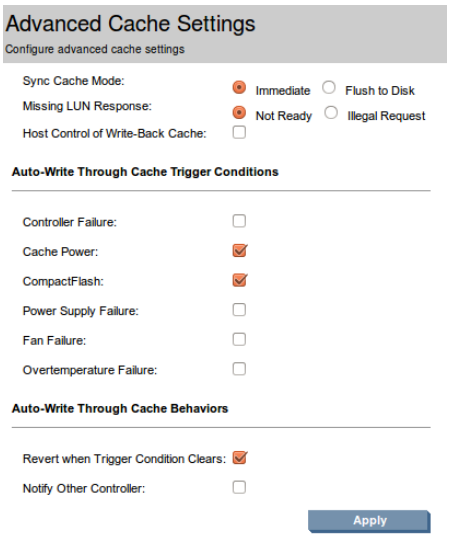


FIGURA 5.36: Configuración de la caché en la MSA

dejaremos en un servidor web (que puede estar en nuestra propia máquina) una imagen iso del instalador de Ubuntu Server y nos iremos a la pestaña `Virtual Media -> Virtual Media`. Allí en el cuadro de texto de `Connect CD/DVD-ROM` introduciremos la URL completa en la red de gestión de la imagen iso y pulsaremos el botón `Boot on Next Reset` para que no sea necesario alterar opciones de arranque. Una vez realizado esto, pulsaremos el botón `Insert Media` e iniciaremos o resetearemos el servidor. Puede verse en la figura 5.37.

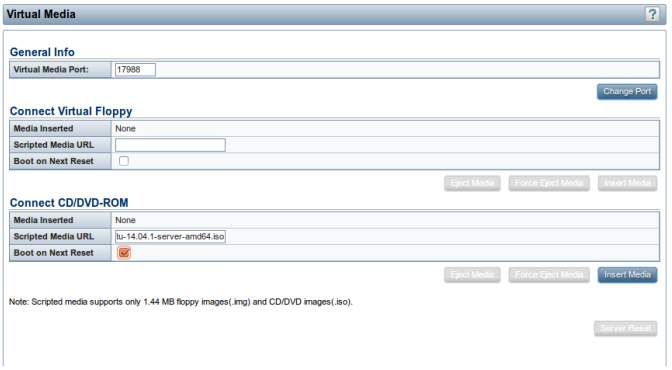


FIGURA 5.37: Inserción de un DVD virtual a través de la iLO

Tras el inicio del servidor en la consola virtual veremos el instalador de Ubuntu Server y procederemos con la instalación del sistema operativo, como puede verse en la figura Instalación SO.

Para la instalación realizaremos el siguiente proceso:

- **Configurar las opciones de ubicación y teclado**: como Español.
- **Configuración de red**: usaremos em1 y DHCP, confirmando el nombre de host visto.

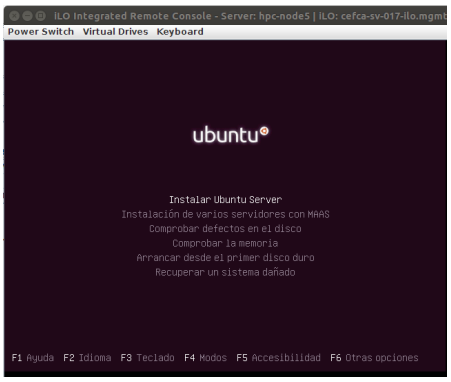


FIGURA 5.38: Instalación del sistema operativo

- **Configuración del usuario administrador**: indicaremos el nombre completo, usuario y contraseña. No cifraremos la carpeta personal.
- **Configuración de la zona horaria**: seleccionaremos Península.
- **Realización del particionado de discos**:
  - en el caso del nodo cabecera, seleccionaremos la opción de `Guiado - Utilizar todo el disco`.
  - en el caso de los nodos de cómputo, seleccionaremos `Manual`. A continuación crearemos dos particiones primarias, la primera de 12GB de tipo EXT4 con etiqueta de arranque y montando el sistema de archivos / y la segunda con el resto del espacio disponible indicando simplemente que se usará para EXT4, sin formatear ni crear un punto de montaje.
- **Configuración del gestor e paquetes**: indicaremos la dirección de nuestro servidor proxy `http://apt-cacher.office.cefca.es:3142`.
- **Configuración de tasksel**: indicaremos que no deseamos actualizaciones automáticas y utilizaremos el perfil de instalación `OpenSSH server`.
- **Instalación cargador de arranque**: procederemos a instalar en el registro principal de arranque.

5.4. Configuración de los nodos

Con el clúster ya instalado y cableado, la cabina y el switch configurados y el sistema operativo desplegado en los nodos podemos realizar la configuración de los mismos.

5.4.1. Configuración de la red

Lo primero que configuraremos en los nodos será la red. Aunque, como veremos posteriormente cuando tratemos la automatización, la interfaz de la red general podrá configurarse de manera dinámica vía DHCP, realizaremos una configuración estática en todas las interfaces.

El primer parámetro de configuración importante es el nombre de host o hostname. Existe una política clara con respecto al nombrado de las máquinas en CEFCA, y es que se usa su código de inventario bien sea físico o bien virtual como hostname. Posteriormente se hace uso de los nombres canónicos o alias DNS para indicar el rol o funciones que realiza cada host. Sin embargo, debido a la dependencia del gestor de recursos del clúster de los nombres dns (aunque existen formas de atajar en parte este problema) y por facilitar la vida a los usuarios, se ha optado por hacer la excepción y romper con el criterio definido, empleando en este caso el rol como hostname tal y como se ha visto en la parte de diseño de red. El dominio será el dominio de la infraestructura, que será `office.cefca.es`. Por ello será necesario crear los nombres de host en dicho dominio y crear la zona de resolución inversa en los controladores de dominio Samba4, para ello en uno de los controladores de dominio ejecutaremos:

```
#  creacion zona inversa
samba-tool dns zonecreate localhost 85.50.10.in-addr.arpa -Uadministrator
#  para cada host creamos registro A y registro puntero en zona inversa
samba-tool dns add localhost office.cefca.es hpc-master A 10.50.85.1 -
    Uadministrator
samba-tool dns add localhost 85.50.10.in-addr.arpa 1 PTR hpc-master.office.cefca.
    es -Uadministrator
#  repetir para cada hpc-node
```

Debido a la dependencia tanto del gestor de recursos como del sistema de autenticación kerberos del servicio de resolución y dado que la primera resolución es local, será muy importante configurar bien el hostname. Para ello deberá estar correctamente asociado el hostname con la dirección de la red general, funcionando correctamente la resolución inversa del host. Para ello editaremos el fichero `/etc/hostname` y en el fichero `/etc/hosts` sustituiremos la dirección que viene por defecto `127.0.1.1` por la dirección que le corresponda al nodo en la red general para que funcione correctamente la resolución inversa de la ip de manera local. Además ejecutaremos el comando `hostname` pasándole el nombre para que tenga efecto en el sistema en ejecución y no sea necesario reiniciar. Los ficheros implicados quedarán del siguiente modo:

```
root@hpc-master:~#  cat /etc/hostname
hpc-master
root@hpc-master:~#  cat /etc/hosts
127.0.0.1 localhost
```

```
10.50.85.1      hpc-master.office.cefca.es      hpc-master


#  The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Después verificaremos con los comandos `hostname` y `hostname --fqdn` que devuelve el nombre correctamente.

A continuación configuraremos las interfaces de red de red de los nodos. Como se ha visto durante el diseño, se hará uso de LACP para hacer agregación de interfaces y obtener un ancho de banda agregado mayor. Ya vimos en la configuración del Cisco 3750X su lado de configuración, ahora es necesario configurarlo en los nodos. Para ello es necesario instalar el software de espacio de usuario necesario y cargar los módulos de kernel. El software que necesitaremos lo obtendremos en el paquete `ifenslave-2.6` y los módulos de kernel necesarios serán `bonding` y `mii` que serán agregados al fichero `/etc/modules` para que se carguen al inicio. Para ello:

```
root@hpc-master:~#  apt-get install ifenslave-2.6
root@hpc-master:~#  modprobe bonding
root@hpc-master:~#  modprobe mii
root@hpc-master:~#  cat /etc/modules
#  /etc/modules: kernel modules to load at boot time.
#
#  This file contains the names of kernel modules that should be loaded
#  at boot time, one per line. Lines beginning with "\# " are ignored.
#  Parameters can be specified after the module name.

lp
rtc
bonding
mii
```

La configuración de las interfaces de red se hace mediante el fichero `/etc/network/interfaces` en este fichero. Dado Ubuntu Server hace uso de `resolvconf` como mecanismo de configuración del cliente DNS también contendrá dicha información para generar automáticamente el fichero `/etc/resolv.conf`. Como podremos ver se configura únicamente en la interfaz de la red general la configuración del cliente y la pasarela o gateway. En la configuración dns indicaremos los servidores dns y los prefijos dns a buscar (esto hará que no será necesario especificar el fqdn para la resolución). En cuanto a la configuración de la agregación de enlaces, se define una interfaz bond0 con la información de red, una mtu de 9000, los parámetros de la agregación necesarios (nótese que el modo 802.3ad

es el estándar LACP) y las interfaces que lo componen. Además será necesario definir las interfaces que componen el bonding para que se agreguen al mismo durante la inicialización.

Para el caso del hpc-master, como se vio en el diseño de red, tendrá:

- interfaz em1 en la red general
- interfaz em2 definida pero sin configurar, ya que será la que usará el contenedor del hpc-login
- interfaz bond0 en la red de almacenamiento usando LACP formado por las interfaces p1p1, p1p2, p1p3 y p1p4 y una mtu de 9000

```
root@hpc-master:~#  cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto em1
iface em1 inet static
        address 10.50.85.1
        netmask 255.255.255.0
        gateway 10.50.85.254
        dns-nameservers 10.50.84.1 10.50.84.2
        dns-search office.cefca.es cefca.es

auto em2
iface em2 inet manual

auto bond0
iface bond0 inet static
        address 192.168.14.1
        netmask 255.255.255.0
        bond-mode 802.3ad
        bond-miimon 100
        bond-downdelay 200
        bond-updelay 200
        bond-slaves p1p1 p1p2 p1p3 p1p4
        mtu 9000

auto p1p1
iface p1p1 inet manual
        bond-master bond0

auto p1p2
iface p1p2 inet manual
        bond-master bond0

auto p1p3
iface p1p3 inet manual
        bond-master bond0
```

```
auto p1p4
iface p1p4 inet manual
        bond-master bond0
```

En el caso de los nodos de cómputo, la configuración es similar:

- interfaz em1 en la red general
- interfaz em2 en la red de cómputo
- interfaz bond0 en la red de almacenamiento usando LACP formado por las interfaces em5 y em6 y una mtu de 9000

```
root@hpc-node1:~#  cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto em1
iface em1 inet static
        address 10.50.85.10
        netmask 255.255.255.0
        gateway 10.50.85.254
        dns-nameservers 10.50.84.1 10.50.84.2
        dns-search office.cefca.es cefca.es

auto em2
iface em2 inet static
        address 192.168.14.10
        netmask 255.255.255.0

auto bond0
iface bond0 inet static
        address 192.168.13.10
        netmask 255.255.255.0
        bond-mode 802.3ad
        bond-miimon 100
        bond-downdelay 200
        bond-updelay 200
        bond-slaves em5 em6
        mtu 9000

auto em5
iface em5 inet manual
        bond-master bond0

auto em6
iface em6 inet manual
        bond-master bond0
```

Tras configurar las interfaces, se realizará un `ifdown` y un `ifup` de cada una de las interfaces. Podrá verse el la correcta configuración mediante el comando `ifconfig` y el funcionamiento del bonding con `cat /proc/net/bonding/bond0`.

5.4.2. Configuración de repositorios de software

5.4.2.1. Configuración de proxy APT

Para la instalación del software, se usará un servidor de caché disponible en la infraestructura CEFCA, para que los nodos usen el servidor caché bastará con crear un fichero de configuración `/etc/apt/apt.conf.d/02proxy` con la información del proxy. Aprovecharnos dicho fichero de configuración también para indicar las fuentes desde las cuales no deseamos que realice caché, que en nuestro caso será el repositorio interno de la oficina y la web de descargas de Oracle, que da problemas al instalar determinados paquetes relacionados con Java.

```
root@hpc-master:~#  cat /etc/apt/apt.conf.d/02proxy
Acquire::http::Proxy "http://apt-cacher.office.cefca.es:3142";
Acquire::http::Proxy {
        download.oracle.com DIRECT;
        software.office.cefca.es DIRECT;
}
```

5.4.2.2. Configuración de repositorio de software local

De manera adicional, incluiremos el repositorio de software de CEFCA y con el que podremos realizar la instalación posterior de la distribución *ubuntucefca14* que contendrá algunos de los paquetes científicos necesarios. Para ello, habrá que importar la key con la que se firma el contenido del repositorio y agregar un fichero a `/etc/apt/sources.list.d`:

```
root@hpc-master:~#  wget http://software.office.cefca.es/packages/apt-key.asc
root@hpc-master:~#  apt-key add apt-key.asc
root@hpc-master:~#  rm apt-key.asc
root@hpc-master:~#  echo "deb http://software.office.cefca.es/packages/
    ubuntucefca ubuntucefca14 main" > /etc/apt/sources.list.d/ubuntucefca.list
root@hpc-master:~#  apt-get update
```

Para finalizar con la configuración del repositorio local, usaremos el mecanismo de "pinning" de APT para dar prioridad a los paquetes que se encuentran en el repositorio local sobre los existentes en la distribución.

```
root@hpc-master:~#  cat /etc/apt/preferences.d/pinning-releases
Package: *
Pin: release n=ubuntucefca14
Pin-Priority: 900
```

5.4.3. Configuración NTP

De cara al buen funcionamiento del clúster, será necesario que todos los nodos se encuentren correctamente sincronizados, para ello usaremos un cliente ntp que será lanzado tras levantarse las interfaces de red al arranque y cada hora con el fin de mantener sincronizado el sistema. Para ello simplemente instalaremos el paquete `ntpdate`, configuraremos en el fichero `/etc/default/ntpdate` los servidores NTP de nuestra infraestructura, usaremos el comando `ntpdate-debian` para sincronizar y crearemos un script en `/etc/cron.hourly` para que se sincronice a cada hora:

```
root@hpc-master:~#  apt-get install ntpdate
root@hpc-master:~#  cat /etc/default/ntpdate
#  The settings in this file are used by the program ntpdate-debian, but not
#  by the upstream program ntpdate.

#  Set to "yes" to take the server list from /etc/ntp.conf, from package ntp,
#  so you only have to keep it in one place.
NTPDATE_USE_NTP_CONF=no

#  List of NTP servers to use  (Separate multiple servers with spaces.)
#  Not used if NTPDATE_USE_NTP_CONF is yes.
NTPSERVERS="ntp1.office.cefca.es ntp2.office.cefca.es"
#  Additional options to pass to ntpdate
NTPOPTIONS=""
root@hpc-master:~#  ntpdate-debian
 4 Apr 20:00:25 ntpdate[21026]: adjust time server 10.50.84.1 offset 0.011568 sec
root@hpc-master:~#  cat /etc/cron.hourly/ntpdate
# !/bin/bash
/usr/sbin/ntpdate-debian &>/dev/null
```

5.4.4. Configuración syslog

A continuación configuraremos todos los nodos para que usen el syslog de la infraestructura de CEFCA y de este modo tener centralizado el archivado de logs. La configuración propuesta seguirá almacenando los logs de manera local en `/var/log` pero además los enviará al servidor central de archivado syslog. Para ello se configurará el software `rsyslogd` de forma que use el protocolo TCP en la comunicación con el servidor para poder detectar los fallos de comunicaciones y en ese caso usará un caché local para reenviar los mensajes que no hayan podido ser entregados por una eventual caída del mismo. La configuración es sencilla, para ello crearemos el directorio en el cual se almacenará la caché local nombrada y configuraremos el software añadiendo un fichero al directorio `/etc/rsyslog.d`.

```
root@hpc-master:~# mkdir -p /var/spool/rsyslog
root@hpc-master:~# chown root:adm /var/spool/rsyslog
root@hpc-master:~# chmod 750 /var/spool/rsyslog
```

```
root@hpc-master:~# cat /etc/rsyslog.d/60-client_syslog.office.cefca.es.conf
$WorkDirectory /var/spool/rsyslog  # default location for work (spool) files

$ActionQueueType LinkedList   # use asynchronous processing
$ActionQueueFileName srvrfwd  # set file name, also enables disk mode
$ActionResumeRetryCount -1    # infinite retries on insert failure
$ActionQueueSaveOnShutdown on # save in-memory data if rsyslog shuts down
*.*       @@syslog.office.cefca.es
```

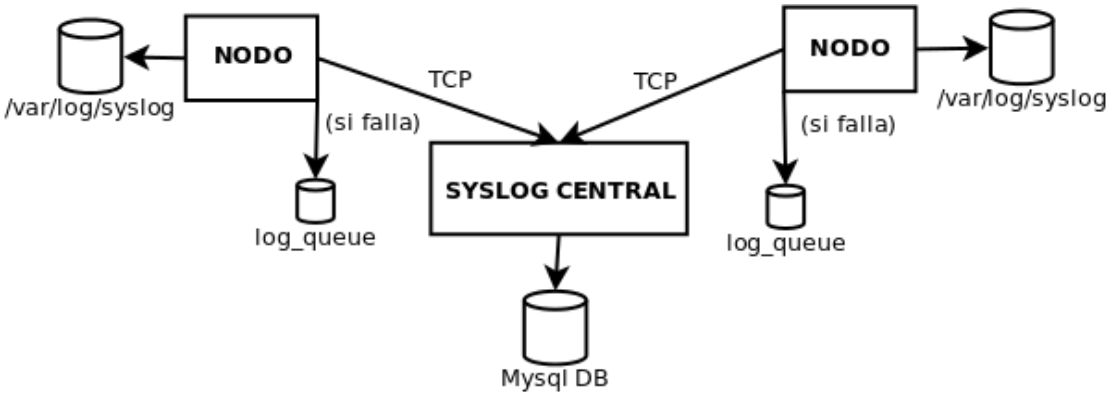


FIGURA 5.39: Diagrama syslog

5.4.5. Configuración autenticación y autorización

Un requerimiento tanto del gestor de recursos del clúster como del almacenamiento compartido, es la consistencia de UIDs y GIDs de los usuarios en todos los sistemas del clúster. Esto se consigue teniendo una base de datos desde la cual obtendrán dicha información todos los nodos. Dicha base de datos se encontrará en un servicio Active Directory implementado mediante dos servidores Samba4. Para ello se han hecho uso de las extensiones RFC2307 al esquema del directorio que permite el almacenamiento la información requerida por cuentas de usuario y grupo del estándar Posix. Active Directory hace uso intensivo del servicio DNS para el descubrimiento de servidores y servicios, del estandar LDAP para la consulta al directorio y de Kerberos como mecanismo preferido de autenticación.

5.4.5.1. Creación de usuarios y grupos

Ya se definió que habría tres tipos de usuarios del clúster: superusuarios, administradores y usuarios normales. Los superusuarios se implementan mediante cuentas locales en los nodos y serán los miembros del grupo sudo. Los administradores y los usuarios normales sí se encuentran implementados en el directorio en los grupos llamados `hpc_users` y `hpc_admins`. La creación de estos grupos se realiza como cualquier otro grupo de seguridad en Active Diretory. Pero en este caso además habrá que mapear los grupos a

cuentas de grupo Posix. Para ello editaremos las propiedades del grupo y en la pestaña `UNIX Attributes` desplegaremos el `NIS Domain` para seleccionar nuestro dominio y el sistema asignará automáticamente el siguiente `GID` como puede verse en la figura 5.40.

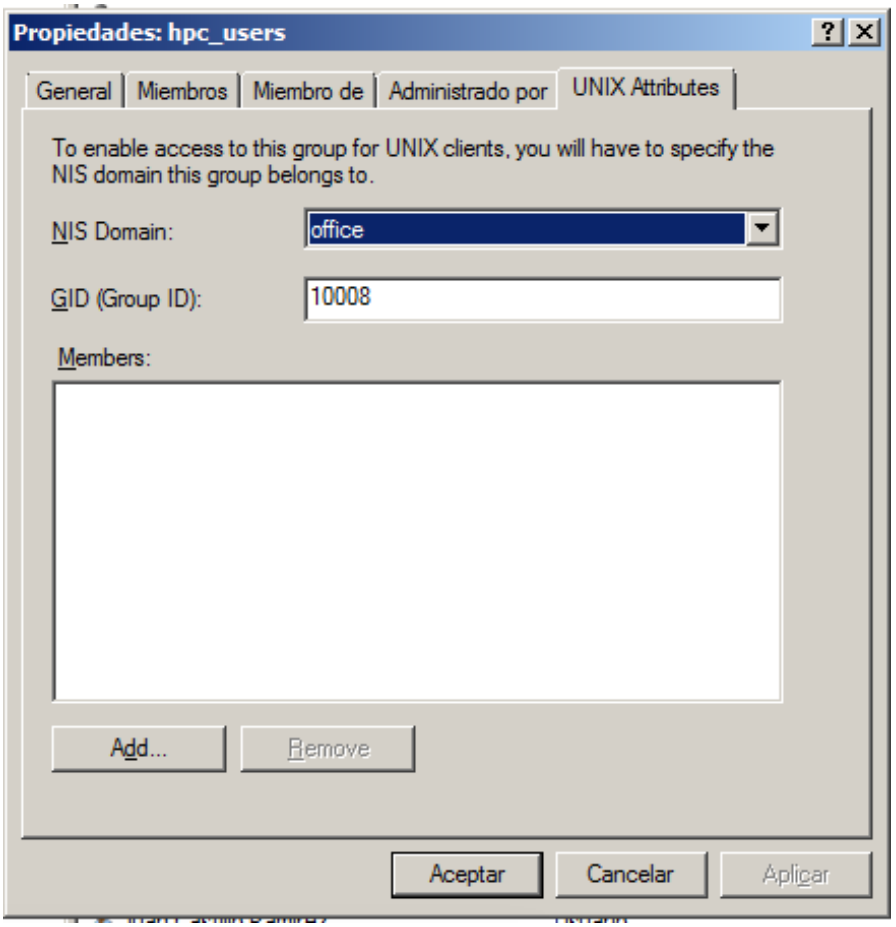


FIGURA 5.40: Atributos UNIX de grupo en Active Directory

Con los usuarios se procederá prácticamente de la misma forma. En este caso agregaremos al usuario a los grupos hpc que le corresponda y nos iremos a la pestaña `UNIX Attributes`. Seleccionaremos como hicimos el `NIS Domain` y modificaremos los atributos de `Login Shell` y `Home Directory` tal y como se ve en la figura 5.41. Asignaremos que el grupo primario sea el grupo `Domain Users` (este grupo también fue mapeado a Posix previamente).

5.4.5.2. Integración de los nodos

Para la integración de los nodos con la infraestructura Active Directory, utilizaremos el software `sssd`. La figura 5.42 es muy ilustrativa de cómo funciona dicha integración. Por un lado tendremos los componentes cliente de NTP y DNS que serán necesarios para que la infraestructura funcione correctamente y de los que ya hemos descrito su configuración anteriormente. Por otro lado tenemos el componente software `sssd` que se compone de

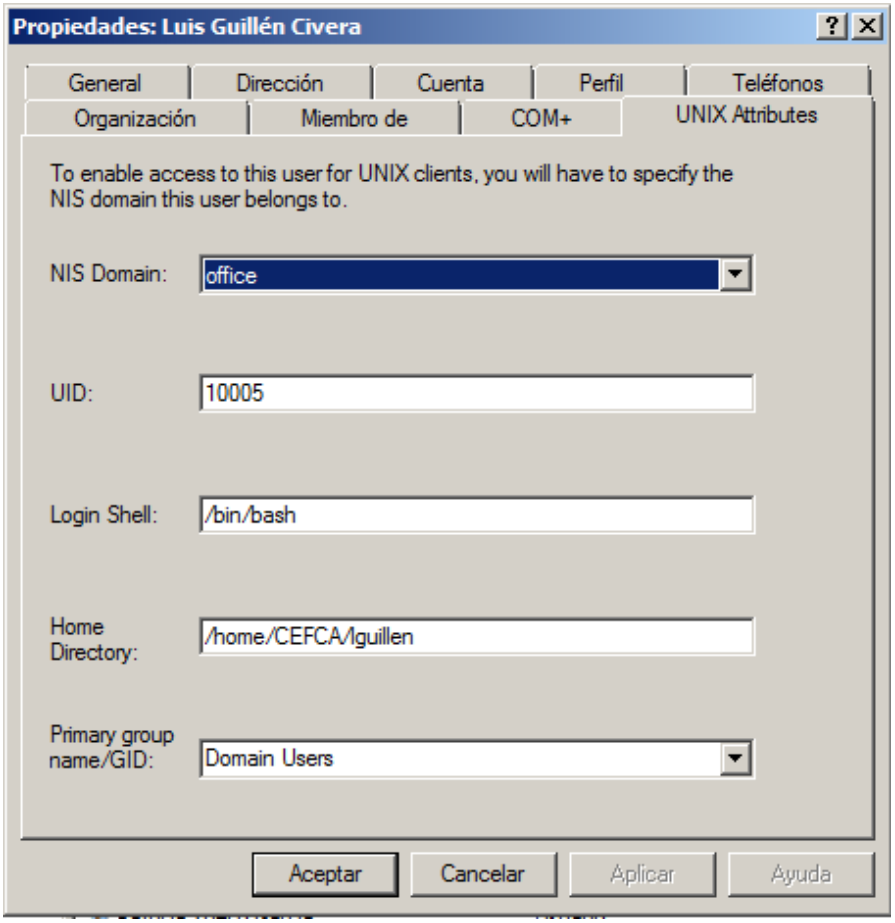


FIGURA 5.41: Atributos UNIX de usuario en Active Directory

un demonio que permite diversas extensiones y de los componentes necesarios para la integración con el sistema: `nss_sss` y `pam_sss`. Podemos ver en la figura que el demonio implementará una parte de cliente de LDAP para consultar al Active Directory sobre los datos de identificación, esto es: UIDs, GIDs, nombres de usuarios y grupos, etc. La otra parte implementará un cliente Kerberos, que se usará para autenticar a los usuarios contra el Active Directory y obtener un ticket de sesión. El componente que hace visible al sistema la información de usuarios y grupos se trata de un módulo nsswitch cuyo nombre es `nss_sss`. El componente que permitirá ofrecer servicios de autenticación de usuarios a los programas ofreciendo la funcionalidad de módulo pam es `pam_sss`.

En primer lugar instalaremos y configuraremos la parte cliente de kerberos, para ello instalaremos los paquetes indicados a continuación y a la pregunta del realm kerberos contestaremos OFFICE.CEFCA.ES (realmente da igual, vamos a sustituir el fichero de configuración). Después reemplazaremos el contenido del fichero de configuración `/etc/krb5.conf` como se muestra a continuación para que use la resolución DNS para encontrar los servidores kerberos. En este momento podremos usar el comando `kinit usuario` para obtener un ticket de un usuario existente y comprobar que funciona correctamente.

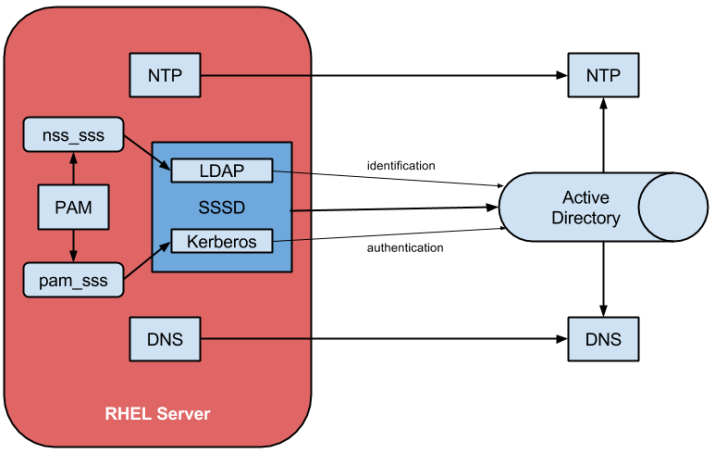


FIGURA 5.42: Integración de nodo con Active Directory

```
root@hpc-master:~# apt-get install krb5-user libsasl2-modules-gssapi-mit
root@hpc-master:~# cat /etc/krb5.conf
[libdefaults]
        default_realm = OFFICE.CEFCA.ES
        dns_lookup_realm = true
        dns_lookup_kdc = true
root@hpc-master:~# kinit lguillen
Password for lguillen@OFFICE.CEFCA.ES:
root@hpc-master:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: lguillen@OFFICE.CEFCA.ES
Valid starting     Expires            Service principal
04/04/15 22:28:58  05/04/15 08:28:58  krbtgt/OFFICE.CEFCA.ES@OFFICE.CEFCA.ES
        renew until 05/04/15 22:28:56
```

El siguiente paso será agregar la máquina al dominio, para ello instalaremos la parte cliente de samba y sobreescribiremos la configuración generada de `/etc/samba/smb.conf` por el contenido que se muestra a continuación. Después procederemos a agregar la máquina al dominio mediante el comando `net join` empleando un usuario con privilegios de agregar máquinas al dominio y posteriormente probaremos que se ha unido correctamente con el comando `net ads testjoin`.

```
root@hpc-master:~# apt-get install samba-client cifs-utils
root@hpc-master:~# cat /etc/samba/smb.conf

[global]
   workgroup = OFFICE
   server string = %h server
```

```
   client signing = yes
   client use spnego = yes
   kerberos method = secrets and keytab
   realm = OFFICE.CEFCA.ES
   security = ads

   log file = /var/log/samba/log.%m
   max log size = 1000

   wins support     = no

root@hpc-master:~# net join -Uaddsrv
Enter addsrv's password:
Using short domain name -- OFFICE
Joined 'HPC-MASTER' to dns domain 'office.cefca.es'
root@hpc-master:~# net ads testjoin
Join is OK
```

Al agregar la máquina al dominio, se han agregado al repositorio de claves kerberos `/etc/krb5.keytab` varios SPN del servicio de tipo `host`, esto permitirá al demonio sssd autenticarse contra el Active Directory sin necesidad de utilizar un usuario dedicado a ello, en su lugar se autenticará con las keys del SPN contenidas en el keytab. Además nos servirá para utilizar kerberos como mecanismo de autenticación para servicios que hagan uso del SPN de tipo `host` como es el SSH. Podremos ver el contenido del keytab mediante el comando `ktutil`, y una vez con los comandos `rkt /etc/krb5.keytab` y `list`.

Aunque no vamos a hacer uso del NFS con kerberos como mecanismo de autenticación (de hecho lo desactivaremos explícitamente), puede ser interesante agregar un SPN que permita el uso de dicho protocolo. Para ello podremos usar el comando `net ads keytab` especificando el SPN.

```
root@hpc-master:~#  net ads keytab add nfs -Uusuarioagregadominio
```

Ya tenemos todos los prerequisitos listos, ahora podremos instalar el software `sssd`. Para ello instalaremos el paquete `sssd-ad` y configuraremos el fichero `/etc/sssd/sssd.conf` como se describe a continuación. Una vez realizado, podremos reiniciar el servicio y con el comando `getent passwd` podremos ver en el listado de usuarios a los usuarios del directorio.

```
root@hpc-master:~# apt-get install sssd-ad
root@hpc-master:~# cat /etc/sssd/sssd.conf
[sssd]
config_file_version = 2
domains = office.cefca.es
services = nss, pam
debug_level = 0

[nss]

[pam]
```

```
[domain/office.cefca.es]
# Uncomment if you need offline logins
# cache_credentials = true
enumerate = true

id_provider = ad
auth_provider = ad
access_provider = ad

# Uncomment if you want to use POSIX UIDs and GIDs set on the AD side
ldap_id_mapping = False

root@hpc-master:~# service sssd restart
```

5.4.6. Configurando el almacenamiento local

5.4.6.1. Configuración del nodo de cabecera

En primer lugar, realizaremos la configuración del almacenamiento local del nodo cabecera. Como vimos en la anterior sección, ya se particionó el disco de sistema dejando que el instalador hiciera el particionado por nosotros, asignando todo a la partición raíz. El esquema resultante es el que vemos a continuación:

```
root@hpc-master:~# parted /dev/sda print
Model: HP LOGICAL VOLUME (scsi)
Disk /dev/sda: 300GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system     Flags
 1      262kB   163GB   163GB   primary   ext4            boot
 2      163GB   300GB   137GB   extended
 5      163GB   300GB   137GB   logical   linux-swap(v1)
```

Ya vimos en la configuración de la MSA que creamos dos volúmenes, uno de scratch y otro de usuarios, que presentábamos al host. En nuestro caso vimos que desde el punto de vista de la cabina veíamos dos hosts, uno conectado a cada controladora y que se correspondían a los dos puertos de la controla HP P431 que hay instalada en el servidor de cabecera. Por lo tanto desde el punto de vista del host deberíamos ver cuatro dispositivos de almacenamiento, dos por cada conexión. En el módulo de kernel hpsa que viene con la versión de Ubuntu Server 14.04 esto no ocurría y únicamente se veían dos dispositivos, esto se debe a un bug que se trata en el anexo. En dicho apéndice se acaba generando un paquete hpsa-dkms que se publica en el repositorio visto antes y que deberemos instalar y reiniciar el servidor posteriormente. Para ello simplemente

instalaremos el paquete, reiniciaremos y nos aseguraremos de que el módulo hpsa es el actualizado:

```
[2:0:0:0]    disk    HP       LOGICAL VOLUME   6.00  /dev/sda
[2:3:0:0]    storage HP       P420i            6.00  -
[3:0:0:1]    disk    HP       MSA 2040 SAS     G105  /dev/sdb
[3:0:0:2]    disk    HP       MSA 2040 SAS     G105  /dev/sdc
[3:3:0:0]    storage HP       P431             2.04  -
root@hpc-master:~# cat /sys/module/hpsa/version
3.4.0-1
root@hpc-master:~# apt-get install hpsa-dkms
root@hpc-master:~# reboot
root@hpc-master:~# cat /sys/module/hpsa/version
3.4.8-140
root@hpc-master:~# lsscsi
[2:0:0:0]    disk    HP       LOGICAL VOLUME   6.00  /dev/sda
[2:3:0:0]    storage HP       P420i            6.00  -
[3:1:1:1]    disk    HP       MSA 2040 SAS     G105  /dev/sdb
[3:1:1:2]    disk    HP       MSA 2040 SAS     G105  /dev/sdc
[3:1:2:1]    disk    HP       MSA 2040 SAS     G105  /dev/sdd
[3:1:2:2]    disk    HP       MSA 2040 SAS     G105  /dev/sde
[3:3:0:0]    storage HP       P431             2.04  -
```

Ahora que vemos los cuatro dispositivos, deberemos configurar el multipath en un modo activo/activo. De este modo cada dispositivo tendrá dos posibles caminos, que irá alternando distribuyendo la carga usando una política round-robin y que deshabilitará un camino si detecta que falla, permitiendo que el acceso siga funcionando. Para ello instalaremos el paquete `multipath-tools`, generaremos un fichero vacío `/etc/multipath.conf`, para posteriormente generar una configuración basada en la configuración que tiene por defecto:

```
root@hpc-master:~# apt-get install multipath-tools
root@hpc-master:~# touch /etc/multipath.conf
root@hpc-master:~# service multipath-tools restart
root@hpc-master:~# echo 'show config' | multipathd -k > multipath.conf-live
```

Editaremos el fichero generado eliminando los prompt existentes y en la configuración por defecto pondremos que no use nombres "user friendly" y agregaremos a la blacklist el dispositivo sda. Una vez realizado esto, sustituiremos al `/etc/multipath.conf` y reiniciaremos el servicio. A continuación se ve una pequeña parte de la configuración:

```
root@hpc-master:~# cat /etc/multipath.conf | more
defaults {
        verbosity 2
        user_friendly_names no
}
blacklist {
        devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
        devnode "^hd[a-z]"
        devnode "^dcssblk[0-9]*"
        devnode "^cciss!c[0-9]d[0-9]*"
```

```
        devnode "^sda"
                ....
root@hpc-master:~# service multipath-tools restart
```

Una vez hecho esto, crearemos en primer lugar los caminos de manera automática basada en la configuración vista y posteriormente los visualizaremos:

```
root@hpc-master:~# multipath -v2
create: 3600c0ff0001df2fc77519d5401000000 undef HP,MSA 2040 SAS
size=11T features='0' hwhandler='0' wp=undef
|-+- policy='round-robin 0' prio=1 status=undef
| `- 3:1:1:1 sdb 8:16 undef ready running
`-+- policy='round-robin 0' prio=1 status=undef
  `- 3:1:2:1 sdd 8:48 undef ready running
create: 3600c0ff0001da5625fb19e5401000000 undef HP,MSA 2040 SAS
size=3.6T features='0' hwhandler='0' wp=undef
|-+- policy='round-robin 0' prio=1 status=undef
| `- 3:1:1:2 sdc 8:32 undef ready running
`-+- policy='round-robin 0' prio=1 status=undef
  `- 3:1:2:2 sde 8:64 undef ready running


root@hpc-master:/# multipath -ll
3600c0ff0001df2fc77519d5401000000 dm-1 HP,MSA 2040 SAS
size=11T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=1 status=active
| `- 3:1:1:1 sdb 8:16 active ready running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 3:1:2:1 sdd 8:48 active ready running
3600c0ff0001da5625fb19e5401000000 dm-0 HP,MSA 2040 SAS
size=3.6T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=1 status=active
| `- 3:1:1:2 sdc 8:32 active ready running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 3:1:2:2 sde 8:64 active ready running
```

También podremos obtener más información mediante el intérprete interactivo que ofrece multipath usando el comando `multipathd -k`. Además veremos que se han asignado nuevos dispositivos en el mapper con el comando dmsetup.

```
root@hpc-master:~# dmsetup ls
3600c0ff0001df2fc77519d5401000000       (252:1)
3600c0ff0001da5625fb19e5401000000       (252:0)
```

A continuación pasaremos a particionar los nuevos dispositivos para poder asignarlos posteriormente a volúmenes lógicos mediante la utilidad parted. Para ello y debido al tamaño de los discos, crearemos una tabla de particionado tipo GPT y crearemos una partición primaria que ocupe todo el disco y sea de tipo LVM. Después crearemos un volumen físico de la partición resultante para poder agregarlo a LVM. Para ello para cada dispositivo:

```
root@hpc-master:~# DISPOSITIVO=/dev/mapper/3600c0ff0001df2fc77519d5401000000
root@hpc-master:~# parted -s $DISPOSITIVO mklabel gpt
root@hpc-master:~# parted -s -a optimal $DISPOSITIVO mkpart primary 0% 100%
root@hpc-master:~# parted -s $DISPOSITIVO set 1 lvm on
root@hpc-master:~# partprobe
root@hpc-master:~# pvcreate $DISPOSITIVO-part1
```

Después crearemos el grupo de volúmenes agregando los dispostivos y los volúmenes lógicos, para ello:

```
root@hpc-master:~# vgcreate vg_scratch /dev/mapper/3600
    c0ff0001da5625fb19e5401000000-part1
root@hpc-master:~# vgcreate vg_users /dev/mapper/3600
    c0ff0001df2fc77519d5401000000-part1
root@hpc-master:~# lvcreate -l 100%VG -n lv0 vg_scratch
root@hpc-master:~# lvcreate -l 100%VG -n lv0 vg_users
```

Una vez realizado esto, procederemos a crear los sistemas de ficheros de ambos volúmenes y etiquetarlos, para ello:

```
root@hpc-master:/# mkfs.ext4 /dev/vg_scratch/lv0
root@hpc-master:/# mkfs.ext4 /dev/vg_users/lv0
root@hpc-master:/# e2label /dev/vg_scratch/lv0 vg_scratch
root@hpc-master:/# e2label /dev/vg_users/lv0 vg_users
```

Será MUY importante, que modifiquemos las preferencias de LVM para que en el siguiente reinicio ignore el escaneado de los dispositivos de disco y evite de esa forma la correcta carga del multipath, para ello será necesario editar el fichero `/etc/lvm/lvm.conf` y generar un nuevo `initrd` que incluya esta nueva configuración , para ello:

```
root@hpc-master:~# cat /etc/lvm/lvm.conf
                ....
        filter = [ "r/block/","r/disk/","r/sd.*/", "a/.*/" ]
                ....
root@hpc-master:~# update-initramfs -u -k all
```

Por supuesto, crearemos los puntos de montaje, entradas en el `/etc/fstab` y realizaremos el montaje del sistema de archivos:

```
root@hpc-master:~# mkdir -p /mnt/vg_users
root@hpc-master:~# mkdir -p /mnt/vg_scratch
root@hpc-master:~# cat /etc/fstab
                ....
LABEL=vg_users /mnt/vg_users ext4 defaults,usrquota 0 2
LABEL=vg_scratch /mnt/vg_scratch ext4 defaults,noatime 0 2
        ....
root@hpc-master:~# mount /mnt/vg_users
root@hpc-master:~# mount /mnt/vg_scratch
```

Para finalizar, configuraremos las cuotas de usuario (aunque todavía no editaremos lo disponible para cada usuario) en el sistema de archivos de vg_users. Para ello instalaremos el paquete `quota`, crearemos e inicializaremos el fichero `aquota.user` mediante el comando `quotacheck` y habilitaremos el uso de cuotas:

```
root@hpc-master:~# apt-get install quota
root@hpc-master:~# quotacheck -cum /mnt/vg_users
root@hpc-master:~# quotaon /mnt/vg_users
```

5.4.6.2. Configuración de los nodos de cómputo

Durante la instalación del nodo, ya se realizó el particionado del disco y se formateó la partición raíz para su instalación. Como vimos, no se creó partición swap de intercambio ya que se tratan de discos SSD y se dejó una partición primaria de tipo linux sin formatear.

```
root@hpc-node5:~# parted /dev/sda print
Model: ATA MK0200GCTYV (scsi)
Disk /dev/sda: 200GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  12.0GB  12.0GB  primary  ext4         boot
 2      12.0GB  200GB   188GB   primary  ext4
```

Procederemos pues a realizar la configuración del almacenamiento local del nodo, para ello crearemos el sistema de archivos, etiquetaremos y realizaremos el montaje previa configuración de fstab:

```
root@hpc-node5:/# mkfs.ext4 /dev/sda2
root@hpc-node5:/# e2label /dev/sda2 sd_local
root@hpc-node5:/# mkdir -p /mnt/sd_local
root@hpc-node5:/# cat /etc/fstab
        ....
LABEL=sd_local /mnt/sd_local ext4 defaults,noatime 0 2
        ....
root@hpc-node5:/# mount /mnt/sd_local
```

Una vez que hemos creado y montado la partición local, configuraremos el almacenamiento de acuerdo al espacio de nombres visto en la especificación. Para ello crearemos los directorios catalog y scratch y realizaremos los puntos de montaje.

```
root@hpc-node5:/# mkdir -p /mnt/sd_local/catalog
root@hpc-node5:/# mkdir -p /mnt/sd_local/scratch
root@hpc-node5:/# mkdir -p /local/catalog
root@hpc-node5:/# mkdir -p /local/scratch
root@hpc-node5:/# cat /etc/fstab
```

```
        ....
        /mnt/sd_local/catalog /local/catalog none bind 0 0
        /mnt/sd_local/scratch /local/scratch none bind 0 0
        ....
root@hpc-node5:/# mount /local/catalog
root@hpc-node5:/# mount /local/scratch
```

5.4.7. Configuración del almacenamiento compartido

5.4.7.1. Configuración del nodo de cabecera

Crearemos en primer lugar los directorios vistos en la especificación en su volumen correspondiente y les asignaremos los permisos.

```
root@hpc-master:/# chmod 755 /mnt/vg_scratch
root@hpc-master:/# mkdir -p /mnt/vg_scratch/catalog
root@hpc-master:/# mkdir -p /mnt/vg_scratch/scratch
root@hpc-master:/# chgrp hpc_admins  /mnt/vg_scratch/catalog
root@hpc-master:/# chmod g+s  /mnt/vg_scratch/catalog
root@hpc-master:/# chmod g+w  /mnt/vg_scratch/catalog
root@hpc-master:/# chgrp hpc_users  /mnt/vg_scratch/scratch
root@hpc-master:/# chmod g+s  /mnt/vg_scratch/scratch
root@hpc-master:/# chmod g+w  /mnt/vg_scratch/scratch
root@hpc-master:/# chmod +t /mnt/vg_scratch/scratch

root@hpc-master:/# chmod 755 /mnt/vg_users
root@hpc-master:/# mkdir -p /mnt/vg_users/homes
root@hpc-master:/# mkdir -p /mnt/vg_users/software
root@hpc-master:/# mkdir -p /mnt/vg_users/shared
root@hpc-master:/# chgrp hpc_admins  /mnt/vg_users/software
root@hpc-master:/# chmod g+s  /mnt/vg_users/software
root@hpc-master:/# chmod g+w  /mnt/vg_users/software
root@hpc-master:/# chgrp hpc_users /mnt/vg_users/shared
root@hpc-master:/# chmod g+s  /mnt/vg_users/shared
root@hpc-master:/# chmod g+w  /mnt/vg_users/shared
root@hpc-master:/# chmod +t /mnt/vg_users/shared
```

A continuación crearemos los puntos de montaje que se exportarán por NFS y se configurará el montaje a los mismos en el fstab.

```
root@hpc-master:/# mkdir -p /export
root@hpc-master:/# mkdir -p /export/homes
root@hpc-master:/# mkdir -p /export/software
root@hpc-master:/# mkdir -p /export/shared
root@hpc-master:/# mkdir -p /export/catalog
root@hpc-master:/# mkdir -p /export/scratch
root@hpc-master:/# cat /etc/fstab
          ....
/mnt/vg_users/homes /export/homes none bind 0 0
/mnt/vg_users/software /export/software none bind 0 0
/mnt/vg_users/shared /export/shared none bind 0 0
```

```
/mnt/vg_scratch/catalog /export/catalog none bind 0 0
/mnt/vg_scratch/scratch /export/scratch none bind 0 0
          ....
root@hpc-master:/# mount /export/homes
root@hpc-master:/# mount /export/software
root@hpc-master:/# mount /export/shared
root@hpc-master:/# mount /export/catalog
root@hpc-master:/# mount /export/scratch
```

Procederemos a instalar el software mediante el paquete `nfs-kernel-server` y a configurar el fichero `/etc/exports`.

```
root@hpc-master:~# apt-get install nfs-kernel-server
root@hpc-master:~# cat /etc/exports
/export 192.168.14.0/24(rw,sync,fsid=0,crossmnt,no_subtree_check)
/export/homes 192.168.14.0/24(rw,sync,fsid=1,no_subtree_check)
/export/software 192.168.14.0/24(rw,sync,fsid=2,no_subtree_check)
/export/shared 192.168.14.0/24(rw,sync,fsid=3,no_subtree_check)
/export/catalog 192.168.14.0/24(rw,async,fsid=4,no_subtree_check)
/export/scratch 192.168.14.0/24(rw,async,fsid=5,no_subtree_check)
root@hpc-master:~# service nfs-kernel-server restart
```

5.4.7.2. Configuración de los nodos de cómputo

Para el montaje en los nodos se utilizará el automontador, pero en primer lugar será necesario instalar la parte cliente de nfs. Para ello instalaremos el paquete nfs-common. Dado que no vamos a realizar montajes usando kerberos como autenticación contra el servidor nfs y existe un pequeño bug en nuestra versión que hace que el proceso de montaje sea más lento si el punto de montaje no usa kerberos, pondremos en blacklist el módulo que trata de realizar la operación.

```
root@hpc-node5:~# apt-get install nfs-common
root@hpc-node5:~#  echo "blacklist rpcsec_gss_krb5" >> /etc/modprobe.d/blacklist.
    conf
```

Procederemos a crear la estructura sobre la cual se realizarán los montajes.

```
root@hpc-node5:~# mkdir -p /home/CEFCA
root@hpc-node5:~# mkdir -p /usr/local/Cluster-software
root@hpc-node5:~# mkdir -p /global
root@hpc-node5:~# mkdir -p /global/shared
root@hpc-node5:~# mkdir -p /global/catalog
root@hpc-node5:~# mkdir -p /global/scratch
```

A continuación instalaremos el software mediante el paquete `autofs` y realizaremos la configuración.

```
root@hpc-node5:~# apt-get install autofs
root@hpc-node5:~# mkdir -p /etc/auto.master.d
root@hpc-node5:~# cat /etc/auto.master.d/
global.autofs    home.autofs      software.autofs
root@hpc-node5:~# cat /etc/auto.master.d/home.autofs
/home/CEFCA     /etc/auto.home
root@hpc-node5:~# cat /etc/auto.master.d/software.autofs
/-      /etc/auto.software
root@hpc-node5:~# cat /etc/auto.master.d/global.autofs
/-      /etc/auto.global
root@hpc-node5:~# cat /etc/auto.home
* -fstype=nfs4 192.168.14.1:/homes/&
root@hpc-node5:~# cat /etc/auto.software
/usr/local/Cluster-software -fstype=nfs4 192.168.14.1:/software
root@hpc-node5:~# cat /etc/auto.global
/global/shared -fstype=nfs4 192.168.14.1:/shared
/global/catalog -fstype=nfs4 192.168.14.1:/catalog
/global/scratch -fstype=nfs4 192.168.14.1:/scratch
root@hpc-node5:~# service autofs restart
```

Para finalizar, crearemos en los nodos el script con el que se realizará la sincronización de los catálogos al recurso local. No lo meteremos en el cron ya que lo lanzaremos de manera centralizada.

```
root@hpc-node5:~# cat /usr/local/bin/update-local-catalog.sh
#!/bin/bash

## fuerzo al automontador a montar el catálogo
ls /global/catalog > /dev/null
if [ $? -ne 0 ]; then
        echo "Error montando punto de distribución" 1>&2
        exit 2
fi

if [ ! -d /global/catalog/distrib ]; then
        echo "No existe directorio de distribución" 1>&2
        exit 2
fi

if [ ! -d /local/catalog ]; then
        echo "No existe directorio local de catálogos" 1>&2
        exit 2
fi

rsync -a --delete /global/catalog/distrib/ /local/catalog
```

5.4.8. Virtualización del nodo de login

Aunque en una primera versión, principalmente por motivos de aprovechar la automatización existente se implementó usando VirtualBox, finalmente se sustituyó la tecnología

de contenedor para implementar el nodo de login. Como puede apreciarse en la figura 5.43, la tecnología de contenedor utiliza el mismo kernel a la vez que mantiene al sistema del contenedor aislado y controlando su uso de recursos (en Linux principalmente gracias las tecnologías cgroups y namespaces). Esto evita todo el overhead que agrega una solución de virtualización tradicional (tanto hipervisores tipo 1 como 2).

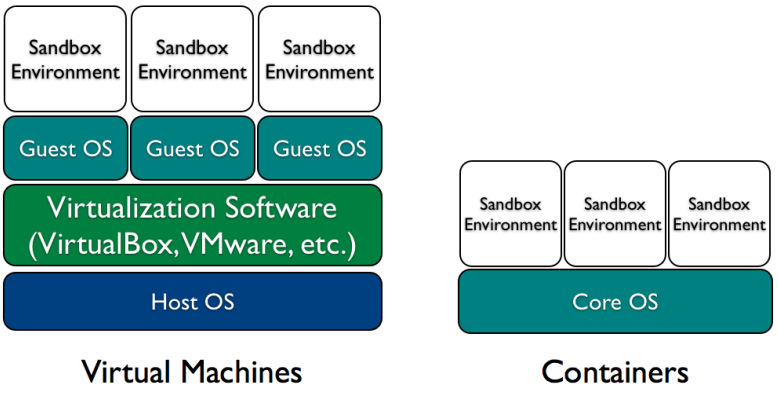


FIGURA 5.43: Virtualización vs contenedor

Para ello se instalará el software mediante el paquete `lxc`, desactivaremos las funciones de puente de red y crearemos el contenedor con el comando `lxc-create`.

```
root@hpc-master:~# apt-get install lxc
root@hpc-master:~# cat /etc/default/lxc-net
          ....
USE_LXC_BRIDGE="false"
          ....
root@hpc-master:~# lxc-create -n hpc-login -t ubuntu
Checking cache download in /var/cache/lxc/trusty/rootfs-amd64 ...
Installing packages in template: ssh,vim,language-pack-en
Downloading ubuntu trusty minimal ...
          ....
##
# The default user is 'ubuntu' with password 'ubuntu'!
# Use the 'sudo' command to run tasks as root in the container.
##
```

Tras este comando, tenemos una tty corriendo en el contenedor, pasaremos a realizar el cambio de nombre de usuario ubuntu al usuario que empleamos como administrador y le fijaremos una password. Posteriormente realizaremos un apagado del contenedor mediante `poweroff` y procederemos a configurar la interfaz de red, los recursos que podrá utilizar el contenedor y haremos que se inicie con el arranque del equipo. Para todo esto bastará editar el fichero `/var/lib/lxc/hpc-login`.

```
root@hpc-master:~# cat /var/lib/lxc/hpc-login/config
         ....
# Network configuration
lxc.network.type = phys
lxc.network.flags = up
```

```
lxc.network.link = em2
lxc.network.name = eth0
#
lxc.cgroup.memory.limit_in_bytes = 96G

lxc.start.auto = 1
lxc.start.delay = 5
```

Haremos que se creen los directorios necesarios del host local y se monten en el próximo arranque del contenedor.

```
root@hpc-master:~# cat /var/lib/lxc/hpc-login/fstab
/mnt/vg_users/homes     /var/lib/lxc/hpc-login/rootfs/home/CEFCA        none
    bind,create=dir 0       0
/mnt/vg_users/software  /var/lib/lxc/hpc-login/rootfs/usr/local/Cluster-software
            none    bind,create=dir 0       0
/mnt/vg_users/shared    /var/lib/lxc/hpc-login/rootfs/global/shared     none
    bind,create=dir 0       0
/mnt/vg_scratch/catalog /var/lib/lxc/hpc-login/rootfs/global/catalog    none
    bind,create=dir 0       0
/mnt/vg_scratch/scratch /var/lib/lxc/hpc-login/rootfs/global/scratch    none
    bind,create=dir 0       0
```

Con todo ya creado, procederemos a iniciar el contendor con `lxc-start` y configurar como lo hemos hecho con el resto de nodos salvo:

- únicamente configuraremos la interfaz de red de que dispone en red general con la ip asignada y cuyo identificador de dispositivo será `eth0`.
- no configuraremos la parte de cliente ntp (ya lo hace el host).
- no configuraremos nada en lo que respecta al almacenamiento ya que lo hemos hecho en el `fstab` del contenedor visto anteriormente
- si desplegamos la automatización de la configuración, deberemos instalar los paquetes `curl`, `wget` y `lsb-release`.

5.4.9. Configuración del servicio SSH y delegación de permisos

Para finalizar configuraremos el servicio ssh en los nodos. Recordemos que:

- en nodo de login pueden tener acceso los usuarios del grupo hpc_users
- en los nodos de cómputo tendrán acceso únicamente los usuarios del grupo hpc_admins
- admitiremos uso de kerberos como mecanismo de login

Para implementar esta configuración simplemente configuraremos en el fichero de preferencias del servicio.

```
root@hpc-login:~# cat /etc/ssh/sshd_config
        ....
# GSSAPI options
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes

# Para permitir X con ssh disabled en los nodos (se accede usando SGE)
# Realizar solo en el nodo hpc-login
X11UseLocalhost no

# Grupos de usuarios de acceso. En nodos de computo sustituir hpc_users por
    hpc_admins
AllowGroups root sudo hpc_users
```

Delegaremos permisos a los usuarios hpc_admins en los nodos de cómputo para que puedan realizar cualquier operación. Editaremos `/etc/sudoers` con la utilidad `visudo`.

```
root@hpc-node1:~# cat /etc/sudoers
    ....
# Damos a los usuarios de hpc_admins permisos de administración del nodo
%hpc_admins     ALL=(ALL:ALL) ALL
    ....
```

5.5. Software de desarrollo científico

Ya disponemos de todos los nodos configurados, compartiendo almacenamiento e integrados con la infraestructura CEFCA. A continuación pasaremos a desplegar el software científico que será lo que en última instancia utilicen nuestros usuarios.

5.5.1. Instalación del software base

5.5.1.1. Nodos de cómputo

En los nodos procederemos a instalar en primer lugar el paquete `ubuntucefca-scientific` que multitud de software que emplea el personal científico y que no requiere entorno gráfico. Dicho paquete se encuentra en la distribución Ubuntucefca14.

```
root@hpc-node5:~#  apt-get install ubuntucefca-scientific
```

Aunque posteriormente instalaremos un entorno OpenMPI optimizado procederemos a instalar el entorno que viene en la distribución.

```
root@hpc-node5:~#  apt-get install openmpi-bin libopenmpi-dev
```

Además instalaremos una serie de librerías que serán necesarias posteriormente para realizar las compilaciones de algunas librerías y utilidades.

```
root@hpc-node5:~#  apt-get install libssl-dev libxp6 libgl1-mesa-dev libxt-dev
```

Como veremos posteriormente, los nodos son sistemas NUMA por lo que instalaremos software que facilitará a los usuarios la programación de los mismos.

```
root@hpc-node5:~#  apt-get install numactl hwloc
```

Por supuesto será necesario instalar el entorno modules con el que posteriormente los usuarios podrán utilizar el software que estará en el repositorio NFS.

```
root@hpc-node5:~#  apt-get install environment-modules
```

5.5.1.2. Nodo de login

Todos los paquetes enumerados, serán instalados en el nodo de login junto con algunos editores, software de control de versiones y herramientas adicionales de desarrollo.

```
root@hpc-node5:~# apt-get install emacs vim geany
root@hpc-node5:~# apt-get install rsync mercurial svn git subversion
root@hpc-node5:~# apt-get install ipython
root@hpc-node5:~# apt-get install gdb valgrind
```

5.5.2. Instalación de software en el recurso NFS

5.5.2.1. Instalación de EasyBuild

Los usuarios con permisos para instalar software en dicho recurso serán los usuarios del grupo hpc_admins. Toda la instalación del software en este recurso, se realizará siempre desde un nodo de cómputo para que la compilación se optimice para la arquitectura de los nodos.

Empezaremos descargando e instalando el software EasyBuild, que será el software con el que gestionaremos todo el software del repositorio. Realizaremos la instalación tal y como recomiendan en su página web, empleando el método bootstrap.

```
lguillen@hpc-node5:~$ curl -O https://raw.githubusercontent.com/hpcugent/
    easybuild-framework/develop/easybuild/scripts/bootstrap_eb.py
python bootstrap_eb.py /usr/local/Cluster-software
```

```
INFO Done!
INFO
INFO EasyBuild v1.16.1 was installed to /usr/local/Cluster-software, so make sure
     your $MODULEPATH includes /usr/local/Cluster-software/modules/all
INFO
INFO Run 'module load EasyBuild', and run 'eb --help' to get help on using
    EasyBuild.
INFO Set $EASYBUILD_MODULES_TOOL to 'EnvironmentModulesC' to use the same modules
     tool as was used now.
INFO
INFO By default, EasyBuild will install software to $HOME/.local/easybuild.
INFO To install software with EasyBuild to /usr/local/Cluster-software, make sure
     $EASYBUILD_INSTALLPATH is set accordingly.
INFO See http://easybuild.readthedocs.org/en/latest/Configuration.html for
    details on configuring
```

Una vez realizada la instalación, habrá generado una sencilla estructura compuesta en su raíz de dos directorios: `software` y `modules`. En el directorio `software` se creará un nuevo directorio para cada programa que se instale incluyendo un directorio con la versión correspondiente. En el directorio `modules/all` se irán guardando bajo el mismo esquema los ficheros module que cargarán el entorno de cada programa. Además se generarán enlaces simbólicos a carpetas bajo el directorio `modules` con el nombre diferentes categorías.

5.5.2.2. Configuración de EasyBuild y modules

Para proseguir con la instalación del entorno, deberemos configurar la utilidad modules para que cargue todos módulos del directorio allí. Para ello simplemente añadiremos la trayectoria al final del fichero `/etc/environment-modules/modulespath`. Esto deberá realizarse en todos los nodos de cómputo y en el nodo de login.

```
root@hpc-node5:~# cat /etc/environment-modules/modulespath
          ....
/usr/local/Cluster-software/modules/all
```

Procederemos a cargar el módulo EasyBuild que se ha creado con la instalación para terminar de configurar el entorno de instalación. Para ello generaremos un fichero plantilla de configuración, en el que indicaremos el correcto `installpath`.

```
lguillen@hpc-node5:~$ module load EasyBuild
lguillen@hpc-node5:~$ eb --configihelp > $HOME/.easybuild/config.cfg
lguillen@hpc-node5:~$ cat $HOME/.easybuild/config.cfg
          ....
installpath=/usr/local/Cluster-software
          ....
```

En las versiones nuevas de EasyBuild, ha cambiado la ubicación de este fichero, será necesario copiarlo a su nueva ubicación para que funcione.

```
lguillen@hpc-node5:~$ mkdir -p .config/easybuild
lguillen@hpc-node5:~$ cp $HOME/.easybuild/config.cfg $HOME/.config/easybuild
```

5.5.2.3. Instalación de software con EasyBuild

Existe abundante documentación del uso de EasyBuild en su página web. El procedimiento es muy simple, basta con cargar el entorno de EasyBuild mediante el comando module y posteriormente usar el comand `eb` para lanzar búsquedas de software o realizar instalaciones.

Por ejemplo, para instalar el compilador GCC en su versión 4.9.2 lo haremos de la siguiente forma.

```
lguillen@hpc-node4:~$ module load EasyBuild/2.0.0
lguillen@hpc-node4:~$ eb GCC-4.9.2.eb
```

Existen multitud de *easyconfigs* realizadas y además EasyBuild proporciona un sistema de dependencias que pueden ser automáticamente resueltas. Sin embargo, la experiencia me ha demostrado que lo mejor es realizar la instalación de las dependencias a mano, cargando además de forma explícita previamente el entorno requerido mediante el comando `module`. Además, conviene echar un vistazo a los ficheros *easyconfigs* a partir de los cuales vamos a instalar el software para ver versiones, flags al compilador, etc. Por ello al final acaba siendo necesario la creación de un pequeño repositorio propio de easyconfigs y parches en el que iremos dejando los ficheros que hemos teneido que crear y personalizar.

Adaptando easyconfigs existentes En el siguiente ejemplo, se trata de instalar el software de benchmarking HPL para la toolchain goolf. Sin embargo, para nuestra versión de toolchain no existe un easyconfig, por lo tanto lo que haremos será copiar uno existente y adaptarlo a las versiones de las que disponemos.

```
lguillen@hpc-node4:~$ module load EasyBuild/2.0.0
lguillen@hpc-node4:~$ module load goolf/1.5.14-no-OFED
lguillen@hpc-node4:~$ module list
Currently Loaded Modulefiles:
  1) EasyBuild/2.0.0
  2) GCC/4.8.2
  3) hwloc/1.8.1-GCC-4.8.2
  4) OpenMPI/1.6.5-GCC-4.8.2-no-OFED
  5) gompi/1.5.14-no-OFED
  6) OpenBLAS/0.2.9-gompi-1.5.14-no-OFED-LAPACK-3.5.0
  7) FFTW/3.3.4-gompi-1.5.14-no-OFED
  8) ScaLAPACK/2.0.2-gompi-1.5.14-no-OFED-OpenBLAS-0.2.9-LAPACK-3.5.0
  9) goolf/1.5.14-no-OFED
```

```
lguillen@hpc-node4:~$ eb --search HPL | grep -i goolf
 * /usr/local/Cluster-software/software/EasyBuild/2.0.0/lib/python2.7/site-
    packages/easybuild_easyconfigs-2.0.0-py2.7.egg/easybuild/easyconfigs/h/HPL/
    HPL-2.0-cgoolf-1.1.7.eb
 * /usr/local/Cluster-software/software/EasyBuild/2.0.0/lib/python2.7/site-
    packages/easybuild_easyconfigs-2.0.0-py2.7.egg/easybuild/easyconfigs/h/HPL/
    HPL-2.0-goolf-1.4.10.eb
 * /usr/local/Cluster-software/software/EasyBuild/2.0.0/lib/python2.7/site-
    packages/easybuild_easyconfigs-2.0.0-py2.7.egg/easybuild/easyconfigs/h/HPL/
    HPL-2.1-goolfc-1.4.10.eb
 * /usr/local/Cluster-software/software/EasyBuild/2.0.0/lib/python2.7/site-
    packages/easybuild_easyconfigs-2.0.0-py2.7.egg/easybuild/easyconfigs/h/HPL/
    HPL-2.1-goolf-1.7.20.eb
 * /usr/local/Cluster-software/software/EasyBuild/2.0.0/lib/python2.7/site-
    packages/easybuild_easyconfigs-2.0.0-py2.7.egg/easybuild/easyconfigs/h/HPL/
    HPL-2.1-goolfc-2.6.10.eb
lguillen@hpc-node4:~$ cp /usr/local/Cluster-software/software/EasyBuild/2.0.0/lib
    /python2.7/site-packages/easybuild_easyconfigs-2.0.0-py2.7.egg/easybuild/
    easyconfigs/h/HPL/HPL-2.0-goolf-1.4.10.eb .
lguillen@hpc-node4:~$ mv HPL-2.0-goolf-1.4.10.eb easybuildrepo/2.0.0
lguillen@hpc-node4:~$ cd easybuildrepo/2.0.0
lguillen@hpc-node4:~/easybuildrepo/2.0.0$ mv HPL-2.0-goolf-1.4.10.eb HPL-2.0-
    goolf-1.5.14-no-OFED.eb
lguillen@hpc-node4:~/easybuildrepo/2.0.0$ cat HPL-2.0-goolf-1.5.14-no-OFED.eb
name = 'HPL'
version = '2.0'

homepage = 'http://www.netlib.org/benchmark/hpl/'
description = """HPL is a software package that solves a (random) dense linear
    system in double precision (64 bits) arithmetic
on distributed-memory computers. It can thus be regarded as a portable as well as
     freely available implementation of the
High Performance Computing Linpack Benchmark."""

toolchain = {'name': 'goolf', 'version': '1.5.14-no-OFED'}
toolchainopts = {'optarch': True, 'usempi': True}

sources = [SOURCELOWER_TAR_GZ]
source_urls = ['http://www.netlib.org/benchmark/%(namelower)s']

# fix Make dependencies, so parallel build also works
patches = ['HPL_parallel-make.patch']

moduleclass = 'tools'
```

Creando easyconfigs nuevos A pesar del extenso repositorio con el que cuenta EasyBuild, hay numerosos programas que no están incluidos en el mismo. Así como para otras ramas de la ciencia existe un numeroso número de software disponible, en el caso de la astrofísica no hay tal diversidad. Por ello se hace necesario generar los easyconfigs necesarios para su instalación. Como ejemplo pondré un sencillo *easyconfig* que creé para la utilidad wcstools de la NASA.

```
lguillen@hpc-login:~$ cat easybuildrepo/2.0.0/wcstools-3.9.2-GCC-4.9.2.eb
##
# This file is an EasyBuild reciPY as per https://github.com/hpcugent/easybuild
#
# Authors::   Luis Guillen <lguillen@cefca.es>
# License::   MIT/GPL
# $Id$
##
easyblock = 'MakeCp'

name = 'wcstools'
version = '3.9.2'

homepage = 'http://tdc-www.harvard.edu/software/wcstools/'
description = """Astronomers often need to relate positions on an
 image of the sky to positions on the real sky to identify catalogued
 objects in images, tell other people where to look to find an
 identified object, or to compute motions of planets, satellites,
 asteroids, or comets. WCSTools is a package of programs and a
 library of utility subroutines for setting and using the world
 coordinate systems (WCS) in the headers of the most common
 astronomical image formats, FITS and IRAF .imh, to relate image
 pixels to sky coordinates. This software is all written in very
 portable C, so it should compile and run on any computer with
 a C compiler."""

toolchain = {'name': 'GCC', 'version': '4.9.2'}

sources = [SOURCELOWER_TAR_GZ]
source_urls = ['http://tdc-www.harvard.edu/software/wcstools/']

buildopts = " all"

files_to_copy = [
        "bin",
        "man",
        (['wcstools'], 'bin')
]

sanity_check_paths = {
        'files': ["bin/getfits"],
        'dirs': []
}

moduleclass = 'tools'
```

5.5.2.4. Instalación e integración de software no libre mediante instaladores

El mecanismo preferido para la instalación de software en el recurso NFS es EasyBuild, sin embargo existe la necesidad de poner a disposición software no libre y de los que no existe mecanismo de instalación con EasyBuild. En nuestro caso el único software de

este tipo que es necesario instalar es IDL. Sin embargo, en un futuro puede ser necesario adquirir software adicional como pudiese ser Mathlab.

Siguiendo el ejemplo de instalación de IDL, simplemente en el proceso de instalación habrá que indicar que se instale en `/usr/local/Cluster-software/software/itt/8.1`.

```
lguillen@hpc-node2:~/idl81/unix$ ./install.sh
        ....
---------------------
Installation Summary:
---------------------
                Product: IDL 8.1
                Installation location: /usr/local/Cluster-software/software/itt
    /8.1
                Login: lguillen
                Dicom Network Services: Yes
Platforms:
                Linux - X86 (32-bit/64-bit)

Install the above configuration? (y/n): y
        ....
```

Una vez instalado, lo que haremos será crear fichero *module* que carge el entorno y luego los enlaces simbólicos necesarios.

```
lguillen@hpc-login:~$  cat /usr/local/Cluster-software/modules/all/idl/8.1
#%Module

proc ModulesHelp { } {
        puts stderr {   IDL - You can put a description here ;)
        }
}

module-whatis {Description: IDL description
}

set root    /usr/local/Cluster-software/software/itt/8.1


prepend-path    PATH            $root/idl/idl81/bin

setenv  ITT_DIR         "$root"
setenv  IDL_DIR         "$root/idl/idl81"
lguillen@hpc-login:~$ cd /usr/local/Cluster-software/modules/lang/idl
lguillen@hpc-login:/usr/local/Cluster-software/modules/lang/idl$ ln ../../all/idl
    /8.1 .
```

5.5.3. Software instalado

5.5.3.1. Compiladores y toolchains

Todo el software (a excepción del IDL anteriormente citado) que se instalará en el cluster será software libre. Esto por supuesto incluyen a las herramientas de desarrollo GCC. Como se verá más adelante, en un futuro podrá evaluarse la adquisición de software licenciado de desarrollo, como son las Intel Cluster Tools que entre otras cosas incluyen el compilador icc y las librerías matemáticas mkl.

Como compiladores instalaremos mediante *Easybuild* los siguientes compiladores:

- GCC: compiladores de C, C++ y Fortran.
- Clang: compilador de C y C++.

En lo que respecta a toolchains, se instalarán las siguientes toolchains:

- GCC: sólo compilador GCC
- gompi: GCC, OpenMPI
- goalf: ATLAS, BLACS, FFTW, GCC, OpenMPI, ScaLAPACK
- goolf: BLACS, FFTW, GCC, OpenBLAS, OpenMPI, ScaLAPACK

5.5.3.2. Entorno paralelización

- OpenMP: para implementar el modelo de paralelismo de memoria compartida, compilaremos GCC con soporte OpenMP.
- OpenMPI: para la implementación del modelo de paralelismo de memoria distribuida, se instalará la implementación de MPI, OpenMPI. Es muy importante que en la compilación se indique la integración con GridEngine. Esto permite que el entorno de ejecución de OpenMPI (llamado *ORTE*) pueda integrarse bien con el software que utilizaremos como gestor de recursos. Esta opción no viene en el easyconfig de los repositorios de Easybuild y hará necesario la creación de unos easyconfigs personalizados.

5.5.3.3. Librerías

- ATLAS
- BLACS
- OpenBLAS
- FFTW: además de la versión 3 que es la incluída en las toolchains, se instalará la versión 2 ya que algunos usuarios tienen software desarrollado para dicha versión.
- ScaLaPACK
- GSL
- Numpy
- Scipy
- HDF5

5.5.3.4. Intérpretes y VMs

- Python
- R
- Java

5.5.3.5. Repositorio de software final

En la figura 5.44 puede verse la salida del comando `module` mostrando todo el software instalado y disponible vía modules en el clúster. Como ya se comentó, para la compilación de todo este software se usaron los easyconfigs que trae el propio Easybuild. Sin embargo hubo que realizar diversos cambios y crear easyconfigs personalizados e incluso parches para su correcta instalación. Todo esto se almacenó en un repositorio personal y cuyo código se incluye en el PFC.

5.5.4. Pruebas del entorno de ejecución

Ya tenemos todas las herramientas software instaladas, pero deberemos comprobar que todo funciona correctamente, especialmente la parte que corresponde al MPI.

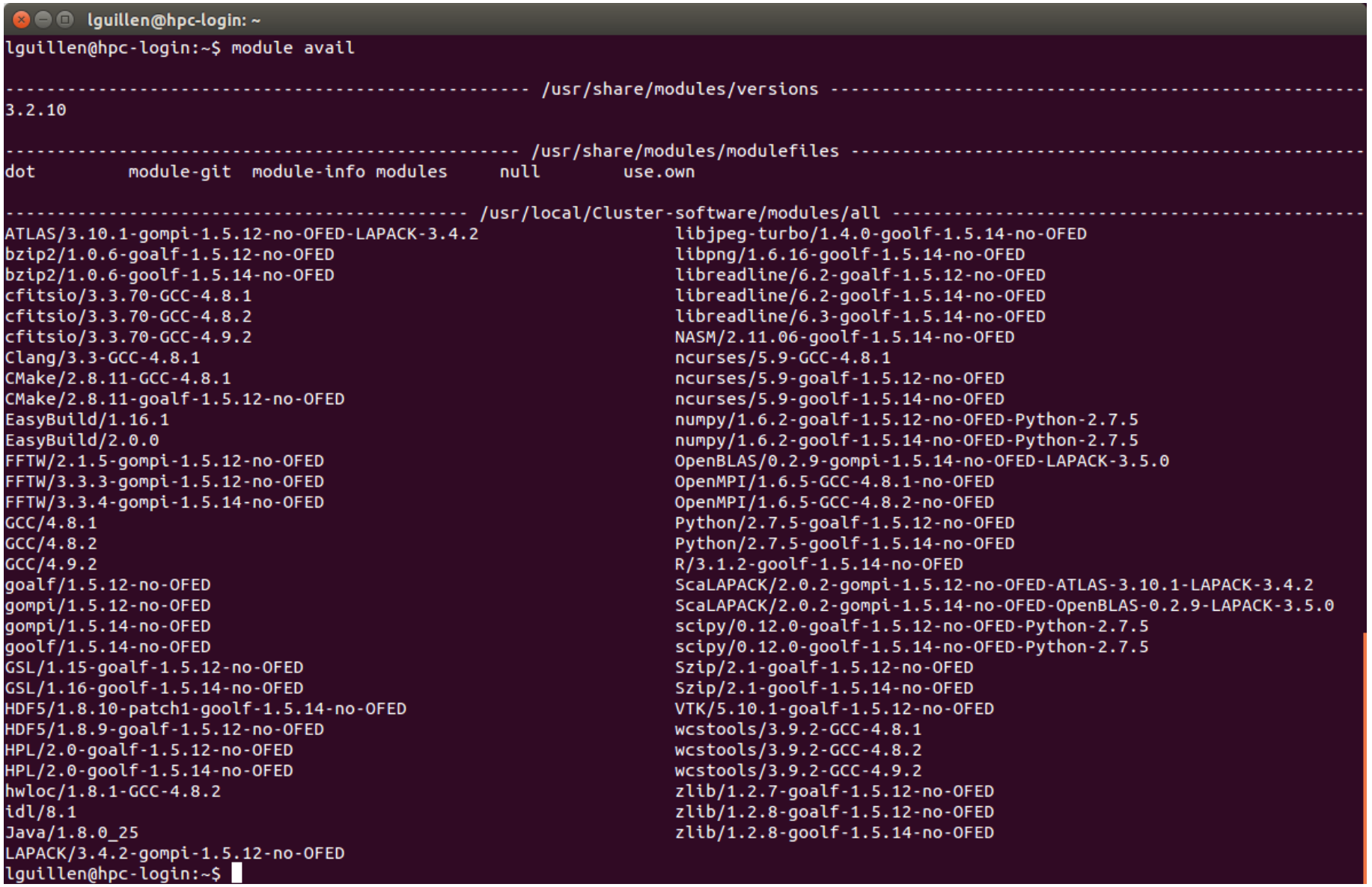


FIGURA 5.44: Software disponible via modules en el HPC

5.5.4.1. Funcionamiento de module

Aunque ya hemos utilizado el sistema de módulos para el uso de *Easybuild* y la instalación de software, probaremos su correcto funcionamiento con módulos generados automáticamente por la utilidad *Easybuild*. Para ello podremos comprobar la versión de gcc que ofrece el sistema, cargar un módulo y luego comprobar de nuevo la versión de gcc.

```
lguillen@hpc-login:~$ gcc --version
gcc (Ubuntu 4.8.2-19ubuntu1) 4.8.2
Copyright (C) 2013 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
lguillen@hpc-login:~$ module load GCC
lguillen@hpc-login:~$ module list
Currently Loaded Modulefiles:
  1) GCC/4.9.2
lguillen@hpc-login:~$ gcc --version
gcc (GCC) 4.9.2
Copyright (C) 2014 Free Software Foundation, Inc.
Esto es software libre; vea el código para las condiciones de copia.  NO hay
garantía; ni siquiera para MERCANTIBILIDAD o IDONEIDAD PARA UN PROPÓSITO EN
PARTICULAR
```

Probaremos también la correcta carga de dependencias de módulos cargando una toolchain. En este ejemplo cargo el toolchain `goolf`, compruebo todos los módulos cargados automáticamente por el sistema y compruebo cómo ha quedado la variable de entorno `LIBRARY_PATH`.

```
lguillen@hpc-login:~$ module load goolf
lguillen@hpc-login:~$ module list
Currently Loaded Modulefiles:
  1) GCC/4.8.2
  2) hwloc/1.8.1-GCC-4.8.2
  3) OpenMPI/1.6.5-GCC-4.8.2-no-OFED
  4) gompi/1.5.14-no-OFED
  5) OpenBLAS/0.2.9-gompi-1.5.14-no-OFED-LAPACK-3.5.0
  6) FFTW/3.3.4-gompi-1.5.14-no-OFED
  7) ScaLAPACK/2.0.2-gompi-1.5.14-no-OFED-OpenBLAS-0.2.9-LAPACK-3.5.0
  8) goolf/1.5.14-no-OFED
lguillen@hpc-login:~$ echo $LIBRARY_PATH
        .... /usr/local/Cluster-software/software/ScaLAPACK/2.0.2-gompi- ....
```

5.5.4.2. Prueba de compilación y ejecución de software con OpenMP

Deberemos comprobar que hemos integrado el soporte OpenMP y que funciona correctamente. Para ello nos valdremos de un sencillo programa que usa OpenMP y al que llamaremos `test_openmp.c`.

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
int nthreads, tid;

/* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid)
  {

  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d\n", tid);

  /* Only master thread does this */
  if (tid == 0)
        {
        nthreads = omp_get_num_threads();
        printf("Number of threads = %d\n", nthreads);
        }

  }  /* All threads join master thread and disband */

}
```

Cargaremos un toolchain, compilaremos y probaremos. En este ejemplo uso la variable de entorno para indicar a openmp el número de hilos que puede usar.

```
lguillen@hpc-login:~/tests$ module load goolf
lguillen@hpc-login:~/tests$ gcc -fopenmp test_openmp.c -o test_openmp
lguillen@hpc-login:~/tests$ OMP_NUM_THREADS=8 ./test_openmp
Hello World from thread = 1
Hello World from thread = 5
Hello World from thread = 2
Hello World from thread = 4
Hello World from thread = 7
Hello World from thread = 6
Hello World from thread = 3
Hello World from thread = 0
Number of threads = 8
```

5.5.4.3. Prueba de la librería FFTW3

Realizaremos una prueba con algún programa que haga uso de la librería. En este ejemplo, me bajo un sencillo programa que hay en la red, lo guardo como `test_fftw3.c` y lo compilo haciendo uso del toolchain `goolf`.

```
lguillen@hpc-login:~/tests$ wget -O test_fttw3.c http://people.sc.fsu.edu/~
    jburkardt/c_src/fftw3/fftw3_prb.c
```

```
lguillen@hpc-login:~/tests$ module load goolf
lguillen@hpc-login:~/tests$ gcc  test_fttw3.c -o test_fftw3 -lfftw3 -lm
lguillen@hpc-login:~/tests$ ./test_fftw3
22 April 2015 07:22:45 AM

FFTW3_PRB
  C version
  Test the FFTW3 library.
        ....
         7     8      0.982060
         7     9      0.791577

FFTW3_PRB
  Normal end of execution.

22 April 2015 07:22:45 AM
```

5.5.4.4. Prueba de librería GSL

Realizamos una prueba con un pequeño ejemplo al que llamamos `test_gsl.c`.

```
#include <stdio.h>
#include <gsl/gsl_sf_bessel.h>

int
main (void)
{
  double x = 5.0;
  double y = gsl_sf_bessel_J0 (x);
  printf ("J0(%g) = %.18e\n", x, y);
  return 0;
}
```

En este caso GSL no forma parte del toolchain, por lo que será necesario cargarlo. Después compilaremos y probaremos.

```
lguillen@hpc-login:~/tests$ module load GSL/1.16-goolf-1.5.14-no-OFED
lguillen@hpc-login:~/tests$ gcc test_gsl.c -o test_gsl -lgsl -lm -lgslcblas
lguillen@hpc-login:~/tests$ ./test_gsl
J0(5) = -1.77596771314338264 2e-01
```

5.5.4.5. Prueba de librería numpy

Probaremos con un sencillo código que hace uso de dicha librería y lo llamaremos `test\_numpy.py`.

```
import time, numpy

def trad_version():
```

```
        t1 = time.time()
        X = range(10000000)
        Y = range(10000000)
        Z = []
        for i in range(len(X)):
                Z.append(X[i] + Y[i])
        return time.time() - t1

def numpy_version():
        t1 = time.time()
        X = numpy.arange(10000000)
        Y = numpy.arange(10000000)
        Z = X + Y
        return time.time() - t1

print trad_version()
print numpy_version()
```

En este caso cargaremos la versión de numpy del toolchain goolf y probaremos.

```
lguillen@hpc-login:~/tests$ module load numpy/1.6.2-goolf-1.5.14-no-OFED-Python
    -2.7.5
lguillen@hpc-login:~/tests$ python test_numpy.py
2.81191897392
0.126993179321
```

5.5.4.6. Prueba y configuración del entorno OpenMPI

Posteriormente integraremos OpenMPI con el gestor de recursos del cluster, pero deberemos comprobar su correcto funcionamiento haciendo uso del protocolo SSH. Para ello crearemos una key y la instalaremos a su vez como clave autorizada. Como todos los nodos hacen uso del mismo home nos permitirá iniciar sesión usando como autenticación la key.

```
lguillen@hpc-login:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/CEFCA/lguillen/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/CEFCA/lguillen/.ssh/id_rsa.
Your public key has been saved in /home/CEFCA/lguillen/.ssh/id_rsa.pub.
The key fingerprint is:
56:bd:28:17:c2:0f:a2:c9:09:1e:84:f7:db:2b:d2:aa lguillen@hpc-login
The key's randomart image is:
+--[ RSA 2048]----+
| ..      XXX     |
|...     .   X.X  |
| .o.   . + o .   |
| . +.+ . = o .   |
|  . =o  S + .    |
|    . .. o       |
```

```
|   .   .         |
|  . o   Eooooo.. |
|E..o .           |
+-----------------+
lguillen@hpc-login:~$ cd .ssh/
lguillen@hpc-login:~/.ssh$ cp id_rsa.pub authorized_keys
```

Probaremos su correcto funcionamiento ejecutando el comando `hostname` en los nodos con los que probaremos OpenMPI y aprovecharemos para guardar la clave pública de los nodos cuando nos lo pregunte.

```
lguillen@hpc-login:~/.ssh$ ssh hpc-node1 hostname
hpc-node1
lguillen@hpc-login:~/.ssh$ ssh hpc-node2 hostname
hpc-node2
```

A continuación usaremos el siguiente código para probar el funcionamiento de MPI y lo llamaremos `test_mpi.c`.

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
  // Initialize the MPI environment. The two arguments to MPI Init are not
  // currently used by MPI implementations, but are there in case future
  // implementations might need the arguments.
  MPI_Init(NULL, NULL);

  // Get the number of processes
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  // Get the rank of the process
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

  // Get the name of the processor
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int name_len;
  MPI_Get_processor_name(processor_name, &name_len);

  // Print off a hello world message
  printf("Hello world from processor %s, rank %d out of %d processors\n",
                 processor_name, world_rank, world_size);

  // Finalize the MPI environment. No more MPI calls can be made after this
  MPI_Finalize();
}
```

Cargaremos el toolchain `gompi` y compilaremos mediante el comando `mpicc`.

```
lguillen@hpc-login:~/tests$ module load gompi
lguillen@hpc-login:~/tests$ module list
Currently Loaded Modulefiles:
  1) GCC/4.8.2                          3) OpenMPI/1.6.5-GCC-4.8.2-no-OFED
  2) hwloc/1.8.1-GCC-4.8.2              4) gompi/1.5.14-no-OFED
lguillen@hpc-login:~/tests$ mpicc test_mpi.c -o test_mpi
```

Probaremos la correcta ejecución en el mismo nodo.

```
lguillen@hpc-login:~/tests$  mpirun ./test_mpi
Hello world from processor hpc-login, rank 0 out of 1 processors
lguillen@hpc-login:~/tests$  mpirun -n 5 ./test_mpi
Hello world from processor hpc-login, rank 0 out of 5 processors
Hello world from processor hpc-login, rank 4 out of 5 processors
Hello world from processor hpc-login, rank 1 out of 5 processors
Hello world from processor hpc-login, rank 2 out of 5 processors
Hello world from processor hpc-login, rank 3 out of 5 processors
```

Ahora probaremos la ejecución en los nodos de cómputo, para ello utilizaremos el parámetro `host`. Nótese cómo por defecto aplica una política round robin. Todo esto puede cambiarse mediante parámetros y un fichero `hostfile`.

```
lguillen@hpc-login:~/tests$ mpirun -n 5 -host hpc-node1,hpc-node2 ./test_mpi
Hello world from processor hpc-node2, rank 1 out of 5 processors
Hello world from processor hpc-node2, rank 3 out of 5 processors
Hello world from processor hpc-node1, rank 2 out of 5 processors
Hello world from processor hpc-node1, rank 4 out of 5 processors
Hello world from processor hpc-node1, rank 0 out of 5 processors
```

Todo funciona mágicamente, pero debemos entender cómo funciona esta magia. Para ello haremos uso de un programa más intensivo que nos permita monitorizar los procesos y las conexiones de red que se establecen. Para ello descargaremos un ejemplo que utiliza mpi [2] para averiguar números primos usando la Criba de Eratóstenes.

Modificaremos el código fuente para tener las máquinas ocupadas durante mucho tiempo, para ello modificaremos la variable `n_hi` que existe en el programa por un mágico `INT_MAX`. Compilaremos y lanzaremos el programa de forma que ejecute una instancia en cada nodo.

```
lguillen@hpc-login:~/tests/prime$ mpicc prime_mpi.c -o prime
lguillen@hpc-login:~/tests/prime$ mpirun -n 2 -host hpc-node1,hpc-node2 ./prime
```

Realizaremos algunas comprobaciones en los nodos. En el nodo de login.

```
root@hpc-login:~# ps aux | grep prime
lguillen 41181  0.1  0.0  51708  3076 pts/4    S+   11:28   0:00 mpirun -n 2 -
    host hpc-node1,hpc-node2 ./prime
```

[2] http://people.sc.fsu.edu/~jburkardt/c_src/prime_mpi/prime_mpi.html

```
root     41185  0.0  0.0  12960   928 pts/5    S+   11:28   0:00 grep --color=
    auto prime
root@hpc-login:~# netstat -natpe | grep 41181
tcp        0      0 0.0.0.0:55065           0.0.0.0:*               LISTEN
    10005      4855445     41181/mpirun
tcp        0      0 10.50.85.2:55065        10.50.85.11:58264       ESTABLISHED
    10005      6256856     41181/mpirun
tcp        0      0 10.50.85.2:55065        10.50.85.10:58315       ESTABLISHED
    10005      6256854     41181/mpirun
```

En el nodo de cómputo 1.

```
root@hpc-node1:~# ps aux | grep lguillen
lguillen 14332  0.0  0.0  47388  2672 ?        Ss   11:28   0:00 /usr/local/
    Cluster-softwar
e/software/OpenMPI/1.6.5-GCC-4.8.2-no-OFED/bin/orted --daemonize -mca ess env -
    mca orte_ess
_jobid 3341942784 -mca orte_ess_vpid 1 -mca orte_ess_num_procs 3 --hnp-uri
    3341942784.0;tcp
://10.50.85.2:55065 -mca plm rsh
lguillen 14333 99.5  0.0  65796  5416 ?        R    11:28   1:32 ./prime
root     14343  0.0  0.0  14276   916 pts/1    S+   11:29   0:00 grep --color=
    auto lguillen
root@hpc-node1:~# netstat -punta | grep orted
tcp        0      0 0.0.0.0:50386           0.0.0.0:*               ESCUCHAR
    14332/orted
tcp        0      0 10.50.85.10:50386       10.50.85.10:37460       ESTABLECIDO
    14332/orted
tcp        0      0 10.50.85.10:58315       10.50.85.2:55065        ESTABLECIDO
    14332/orted
root@hpc-node1:~# netstat -punta | grep prime
tcp        0      0 0.0.0.0:1024            0.0.0.0:*               ESCUCHAR
    14333/prime
tcp        0      0 0.0.0.0:58254           0.0.0.0:*               ESCUCHAR
    14333/prime
tcp        0      0 192.168.13.10:39442     192.168.13.11:1025      ESTABLECIDO
    14333/prime
tcp        0      0 10.50.85.10:37460       10.50.85.10:50386       ESTABLECIDO
    14333/prime
tcp        0      0 192.168.14.10:51726     192.168.14.11:1025      ESTABLECIDO
    14333/prime
tcp        0      0 10.50.85.10:39411       10.50.85.11:1025        ESTABLECIDO
    14333/prime
```

Y en el nodo de cómputo 2.

```
root@hpc-node2:~# ps aux | grep lguillen
lguillen 14716  0.0  0.0  47388  2672 ?        Ss   11:28   0:00 /usr/local/
    Cluster-softwar
e/software/OpenMPI/1.6.5-GCC-4.8.2-no-OFED/bin/orted --daemonize -mca ess env -
    mca orte_ess
_jobid 3341942784 -mca orte_ess_vpid 2 -mca orte_ess_num_procs 3 --hnp-uri
    3341942784.0;tcp
://10.50.85.2:55065 -mca plm rsh
```

```
lguillen 14717 99.8  0.0  65792  5368 ?        R    11:28   2:55 ./prime
root     14727  0.0  0.0  14276   920 pts/0    S+   11:31   0:00 grep --color=
    auto lguillen
root@hpc-node2:~# netstat -punta | grep -i orted
tcp        0      0 0.0.0.0:41705           0.0.0.0:*               ESCUCHAR
    14716/orted
tcp        0      0 10.50.85.11:41705       10.50.85.11:33822       ESTABLECIDO
    14716/orted
tcp        0      0 10.50.85.11:58264       10.50.85.2:55065        ESTABLECIDO
    14716/orted
root@hpc-node2:~# netstat -punta | grep -i prime
tcp        0      0 0.0.0.0:54877           0.0.0.0:*               ESCUCHAR
    14717/prime
tcp        0      0 0.0.0.0:1025            0.0.0.0:*               ESCUCHAR
    14717/prime
tcp        0      0 192.168.13.11:1025      192.168.13.10:39442     ESTABLECIDO
    14717/prime
tcp        0      0 192.168.14.11:1025      192.168.14.10:51726     ESTABLECIDO
    14717/prime
tcp        0      0 10.50.85.11:33822       10.50.85.11:41705       ESTABLECIDO
    14717/prime
tcp        0      0 10.50.85.11:1025        10.50.85.10:39411       ESTABLECIDO
    14717/prime
```

Como podemos ver, en el nodo de login el proceso `mpirun` (que realmente es un enlace a `orterun`) lanza en los nodos instancias de `orted` vía ssh, iniciando de este modo el entorno de ejecución y estableciendo un canal de control del mismo. Posteriormente, el entorno de ejecución comprueba las interfaces de red existentes en los nodos y el direccionamiento que tienen en común para identificar a través de qué redes podrá realizarse el intercambio de mensajes MPI. Como podemos ver, ha establecido conexiones entre los nodos a través de todas las interfaces de red que tienen en común ambos nodos. Esto no es lo deseable, ya que queremos dedicar la interfaz que definimos como dedicada a dicho tráfico. Esto lo haremos indicando con el parámetro `btl_tcp_if_include` indicando la interfaz como en el siguiente ejemplo.

```
lguillen@hpc-login:~/tests/prime$ mpirun -n 2 --mca btl_tcp_if_include
    192.168.13.0/24 -host hpc-node1,hpc-node2 ./prime
```

Para dejar esta opción fijada, deberemos configurar esta opción en los ficheros de configuración de OpeMPI. En el caso de que se use el OpenMPI que trae el sistema, será necesario configurar el fichero de todos los nodos.

```
echo "btl_tcp_if_include = 192.168.13.0/24" >> /etc/openmpi/openmpi-mca-params.
    conf
echo "btl_tcp_if_include = 192.168.13.0/24" >> /usr/local/Cluster-software/
    software/OpenMPI/1.6.5-GCC-4.8.1-no-OFED/etc/openmpi-mca-params.conf
echo "btl_tcp_if_include = 192.168.13.0/24" >> /usr/local/Cluster-software/
    software/OpenMPI/1.6.5-GCC-4.8.2-no-OFED/etc/openmpi-mca-params.conf
```

5.6. Software gestor de recursos

Con todo el software científico desplegado en nuestro clúster únicamente faltará el software con el que los usuarios podrán enviar sus trabajos de cómputo. La interfaz mediante la cual los usuarios enviarán y supervisarán sus trabajos la ofrecerá el software gestor de recursos que vamos a ver a continuación.

5.6.1. Diseño del gestor de recursos

5.6.1.1. Definición de roles de los nodos

Grid Engine es un sistema distribuido formado por diversos componentes tal y como puede verse en la figura 5.45. Por lo tanto, nuestro primer paso será la asignación de los roles en nuestros nodos y cómo será el despliegue de los componentes.

- **hpc-master**: como no podía ser de otra manera, estará implementado en el nodo hpc-master. Llevará el software Qmaster, cerebro de todo el clúster. Sus principales responsabilidades serán las de controlar el estado de los nodos de ejecución y llevar a cabo la planificación de los trabajos.
- **hpc-login**: estará implementado en el nodo de login y será la cara visible del clúster a los usuarios. En dicho nodo los usuarios deberán tener a su disposición el software con el que podrán interactuar con el gestor de recursos. Dicho software se compondrá de todas las utilidades de gestión de sus trabajos, visualización de los estados de colas y nodos del clúster y del API de desarrollo DRMAA.
- **hpc-exec**: este será el rol asignado a los nodos de cómputo. Los nodos llevarán instalado el demonio de ejecución del gestor de recursos.Dicho software será el responsable llevar a cabo la ejecución final de los trabajos y de monitorizar la carga, notificando en todo momento al Qmaster.

Nótese que no hemos asignado los componentes ARCo ni Shadow Master. Estos componentes no se implementarán en nuestro clúster.

5.6.1.2. Definición de las colas

Definiremos dos colas con las siguientes características:

- **main.q**:

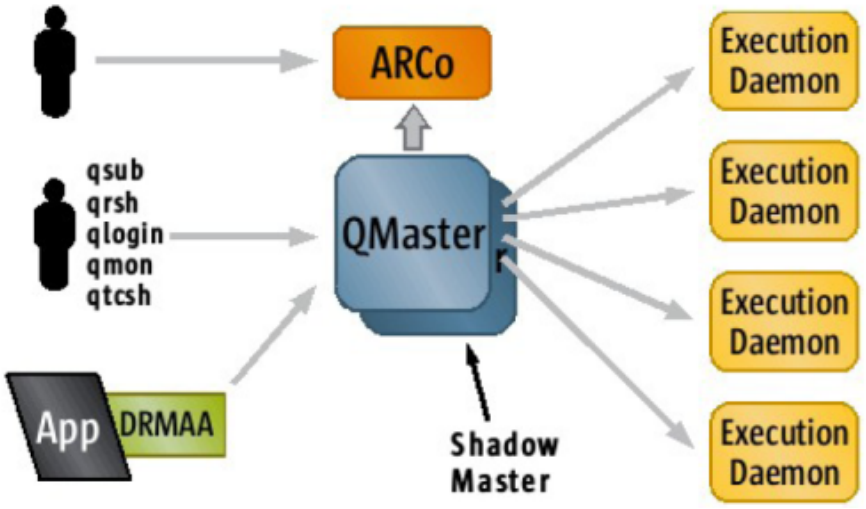


FIGURA 5.45: Componentes de GridEngine

- Irán todos los trabajos excepto los de tipo interactivo.
- Se establecerá una duración máxima de los trabajos de 48 horas.
- En esta cola estarán disponibles todos los nodos de computación del clúster.
- Esta cola tendrá los entornos paralelos definidos que describiremos posteriormente.

▪ **interactive.q**:

- Irán todos los trabajos de tipo interactivo.
- Se establecerá una duración máxima de 6 horas por trabajo.
- Esta cola tendrá un único nodo disponible.
- Esta cola tendrá los entornos paralelos definidos que describiremos posteriormente.

Puede verse en la figura 5.46 la distribución de las mismas, además de la agrupación de hosts que haremos bajo el grupo @allhosts.

5.6.1.3. Ajuste de la capacidad de los nodos

Como ya se vio en la parte de especificación, los nodos de cómputo disponen de 2 procesadores con 10 cores cada uno, formando un total de 20 cores. Si además activamos el hyperthreading el sistema operativo verá un total de 40 unidades de procesamiento. En la figura 5.47 podemos ver cómo es la organización de CPU y memoria de nuestros nodos de cómputo. Como puede verse, estos equipos tienen una arquitectura NUMA, lo que hace que cada procesador tenga una memoria local de 64GB. Sin embargo, ambos

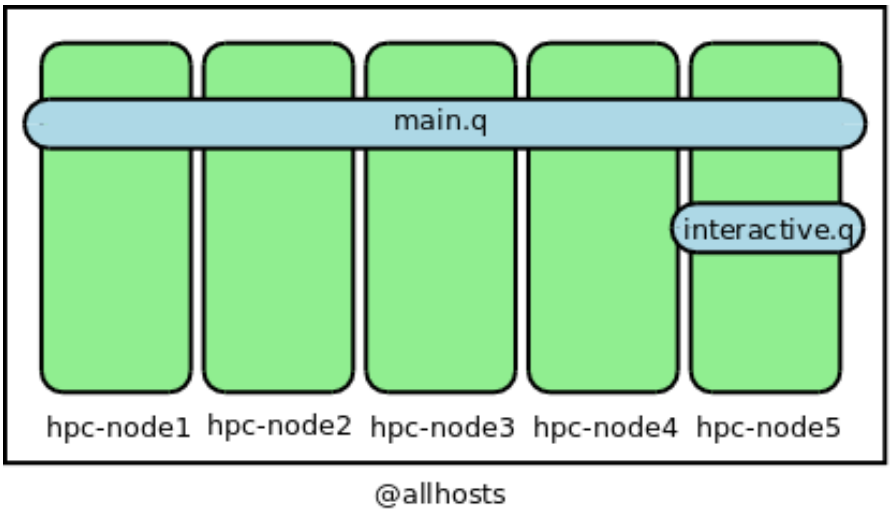


FIGURA 5.46: Distribución de las colas en el clúster HPC

procesadores pueden acceder a la zona no local de su memoria empleando el bus de alta velocidad QPI pero a una velocidad más lenta que el acceso a su memoria local.

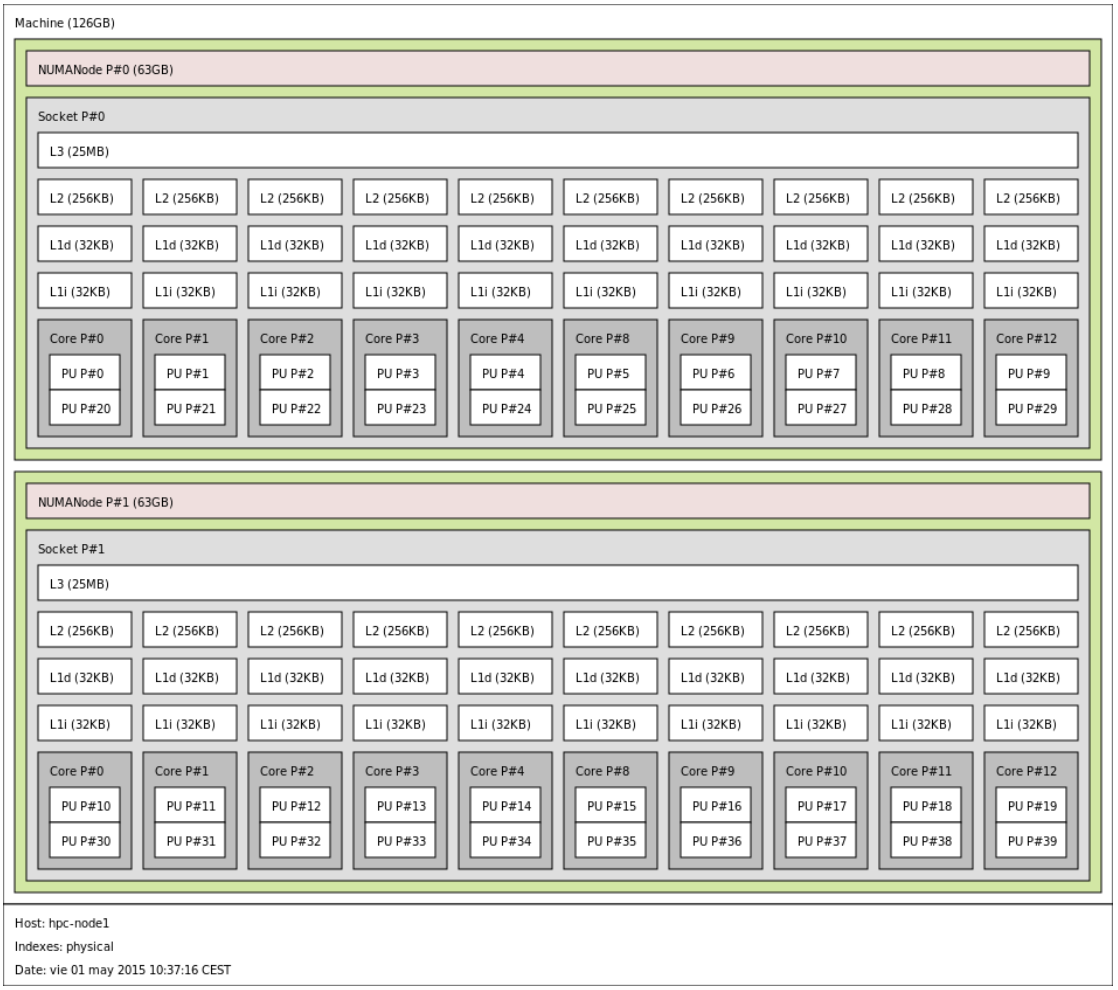


FIGURA 5.47: Vista de CPUs de un nodo con hwloc

Desde el punto de vista de nuestro gestor de recursos y con el fin de evitar la sobresubscripción del nodo, permitiremos tantos *slots* de ejecución como cores tiene el nodo, es decir, a cada nodo le asignaremos 20 slots de ejecución. Esto quiere decir que desde el punto de vista de nuestro gestor de recursos únicamente sabrá que puede asignar un máximo de 20 trabajos no paralelos a cada nodo (obviamente, también podrá asignar trabajos paralelos pero en menor número). Será responsabilidad de los programadores el aprovechamiento del hyperthreading y de la arquitectura NUMA que ofrecen los nodos, utilizando las capacidades de *affinity* que nos ofrece el planificador del kernel de Linux.

Si la sobresubscripción en CPU es mala, ya que introduce más competencia por la CPU y por lo tanto más longitud de cola de espera y cambios de contexto, en memoria es devastadora. Ya vimos que cada nodo tiene 128GB y no tiene Swap. No disponer de Swap se debe a dos motivos: el primero es que nuestro almacenamiento es un disco SSD y usarlo para swap degradaría el disco. El segundo motivo, es que en este tipo de entornos, todas las aplicaciones que se lanzan no tienen apenas tiempo de idle y son muy intensivas en sus accesos a memoria. Esto hace que si el nodo empieza a paginar, en la práctica, el nodo acaba por tumbarse. Acabamos de comentar que vamos a limitar la ejecución a 20 slots por nodo, y como acabamos de ver, la memoria RAM también es un recurso limitado que nuestro gestor de recursos también deberá de tener en cuenta. Si vemos la memoria disponible de uno de los nodos tras la carga del sistema operativo y con un acceso ssh, obtenemos lo siguiente:

```
root@hpc-node1:~#  LANG=C free
                           total        used        free      shared     buffers
    cached
Mem:     131997796     1112764   130885032         988       18344       91580
-/+ buffers/cache:     1002840   130994956
Swap:            0           0           0
```

Con esta información, podemos asumir que tenemos `130885032KB` disponibles para el uso de las aplicaciones. Si dividimos por el número de slots asignados, nos dará un total de `6544251KB` por slot. Esto quiere decir que un nodo a su máxima capacidad de cómputo podrá estar procesando 20 trabajos simultáneamente con un uso máximo de aproximadamente 6GB cada uno. Evidentemente, habrá trabajos que usarán menos memoria y otros que usarán muchísima más memoria, pero será necesario evitar la sobresubscripción del nodo y la entrada en escena del OOM Killer del kernel de linux para tumbar trabajos que lleven muchas horas de cómputo. Por ello seremos un poco más conservadores y dejaremos al sistema operativo ese pico de `1997796KB` (casi 2GB) para su uso y para el alojamiento de buffers y cache y fijaremos el máximo de consumo en trabajos en `130000000KB`. Por defecto fijaremos que cada trabajo tendrá a su disposición `6500000KB` pero permitiremos que el usuario pueda solicitar explícitamente más memoria para su

trabajo, pudiendo solicitar hasta el máximo disponible del nodo (obviamente si no existe ningún otro trabajo en ejecución en ese momento).

Deberemos configurar por lo tanto estos límites en nuestro gestor de recursos por nodo:

- capacidad de slots: 20
- capacidad de memoria: 130000000KB
- asignación por defecto de memoria a cada slot: 6500000KB

Nuestro gestor de recursos llevará la contabilidad de los recursos asignados y solicitados e irá despachando los trabajos según explicaremos a continuación. Sin embargo, aunque llevará una contabilidad de la asignación de la memoria, en el caso del consumo de CPU no será así (en realidad sí lleva la contabilidad pero es en tiempos de uso). Es muy posible que los usuarios lancen aplicaciones multihilo que realicen procesamiento pesado en varios hilos y no hayan especificado un entorno paralelo y por lo tanto no hayan solicitado slots de ejecución adicionales. Para ello estableceremos en el gestor de recursos unos umbrales de carga bajo los cuales no entrarán trabajos adicionales en los nodos. En el caso de la memoria el umbral va a ser el total que hemos definido anteriormente. Para el procesador, gridengine ofrece algunos *complex* que están normalizados al número de procesadores, sin embargo hemos de tener en cuenta que tenemos habilitado el hyperthreading. Por ello, y tras los resultados obtenidos de varias pruebas de carga, fijaremos que a partir de un valor normalizado de 0.9 no admitiremos más trabajos en el nodo.

5.6.1.4. Definición de los entornos paralelos

Como vimos, nuestro clúster deberá responder los dos tipos de paradigmas de paralelización en lo que a memoria se refiere. El primero de ellos es el de la paralelización con memoria compartida y el segundo el de la paralelización con memoria distribuida. Deberemos por lo tanto poner a disposición de los usuarios en el gestor de recursos de los entornos de paralelización necesarios.

Para el primer tipo de paradigma, los usuarios recurrirán generalmente a OpenMP u otras librerías de paralelización multihilo. Sin embargo, también podrán querer lanzar procesos independientes y recurrir a algún mecanismo de memoria compartida que ofrece el sistema operativo. En cualquier caso, el ámbito de ejecución se reducirá a un nodo y el máximo de slots que los usuarios podrán solicitar para su ejecución estará limitado a 20.

Para el segundo tipo de paradigma, en nuestro clúster proporcionamos la librería OpenMPI. Esta librería proporciona un entorno de ejecución llamado ORTE que se encarga de levantar los procesos y establecer los canales de comunicación necesarios entre los mismos. En el caso de que los procesos que levanta ORTE estén en el mismo host, emplea mecanismos de memoria compartida y la comunicación entre estos procesos es muy rápida. En caso de que estén en hosts distintos, los procesos deberán comunicarse a través de la red lo que implica una comunicación mucho más lenta. Por poner un ejemplo, un trabajo que requiere de 15 procesos: si se ponen 5 procesos en un nodo y 10 procesos en otro, rendirá mucho peor que si se ponen los 15 en un único nodo. Por esto será necesaria la maximización del número de procesos alocados en un nodo y se evitarán establecer comunicaciones por red innecesarias.

Por todo lo anteriormente nombrado, especificaremos tres entornos de paralelización:

- **parallel**: este entorno localizará un nodo que pueda satisfacer la demanda de slots del trabajo y lo ejecutará.
- **orte**: este entorno, del mismo modo que el anterior, buscará un nodo con los slots libres solicitados y realizará la ejecución en ese nodo. Sin embargo, a diferencia del anterior, se empleará la integración con ORTE (que compilamos en el apartado anterior) y el proceso que gestiona el runtime no consumirá un slot. Este entorno permitirá la ejecución de programas que usen OpenMPI pero que demanden un número de slots menor que el tamaño disponible en el nodo.
- **orte-20**: este entorno también estará integrado con ORTE, pero en este caso para la solicitud de slots de ejecución deberán suministrarse números múltiplos de 20. Esto obligará a llenar nodos enteros y maximizar el uso y rendimiento de los trabajos de este tipo.

Los entornos de ejecución parallel y orte estarán disponibles tanto en la cola main.q como en interactive.q. El entorno orte-20 sólo estará disponible para la cola main.q.

5.6.1.5. Distribución de los trabajos

Ya vimos con anterioridad la separación que hacíamos entre trabajos interactivos y no interactivos. Todos los interactivos irán a la cola interactive.q y por lo tanto irán todos a hpc-node5 al ser el único miembro de dicha cola. Sin embargo en el caso de los no interactivos podrán ir a cualquiera de los nodos, incluyendo al nodo hpc-node5. Por la naturaleza de los trabajos interactivos, deseamos que los trabajos enviados a main.q vayan al nodo hpc-node5 únicamente cuando no haya recursos en el resto de nodos.

Con respecto a la distribución general de los trabajos en la cola main.q, para facilitar la entrada de trabajos paralelos grandes y con muchas necesidades de cómputo se seguirá una estrategia fillup. Esta estrategia consiste en ir llenando en lo máximo un nodo de trabajos antes de pasar al siguiente, dejando de este modo nodos libres en los que podrán entrar trabajos grandes. Para el éxito en la implementación de esta estrategia es necesario la correcta configuración de la asignación y el control de la capacidad que vimos en el apartado anterior.

5.6.2. Instalación inicial

Importante: hay que destacar que esta instalación se realiza a partir de los paquetes parcheados de gridengine cuyo proceso puede verse en los Anexos. Estos paquetes solucionan un molesto bug en la herramienta gráfica Qmon y agregan algunos binarios necesarios para que funcione correctamente el soporte JSV.

5.6.2.1. Instalación en el nodo master

En el caso del servidor máster instalaremos además del gridengine-master los paquetes de cliente y qmon para poder gestionar la configuración desde el mismo nodo y gestionar trabajos, bien para enviar o bien para eliminar trabajos existentes.

```
#  apt-get install gridengine-master gridengine-client gridengine-qmon
```

5.6.2.2. Instalación en el nodo de login

En este caso instalaremos los paquetes de tipo cliente gridengine-client y gridengine-qmon. Además instalaremos los paquetes que contienen las librerías DRMAA y su binding para python.

```
#  apt-get install gridengine-client gridengine-qmon gridengine-drmaa-dev python-drmaa
```

5.6.2.3. Instalación en los nodos de cómputo

En el caso de los nodos de cómputo instalaremos el paquete gridengine-exec que contiene el demonio de ejecución. Además instalaremos los comandos cliente ya que son necesarios para el correcto funcionamiento de las tareas OpenMPI.

```
#  apt-get install gridengine-client gridengine-exec
```

5.6.2.4. Asistente de instalación debconf

En todos los casos, ya que todos los paquetes dependen de `gridengine-common`, realizaremos la configuración vía *debconf*.

Lo primero que nos preguntará será la configuración del MTA que se instala y que servirá para informar vía email a los usuarios acerca del estado de sus trabajos. El MTA que instala por defecto es Postfix y deberá configurarse:

- En primer lugar seleccionaremos el tipo de sistema, indicando que es de tipo Internet con smarthost, tal y como se puede ver en la figura 5.48.
- Después especificaremos la FQDN del servidor en el nombre del sistema de correo. Esto hará que postfix gestione el correo local del nodo.
- Para finalizar, daremos el nombre del servidor SMTP que usará para entregar el correo no local, es decir el que finalmente se usará para enviar los emails.

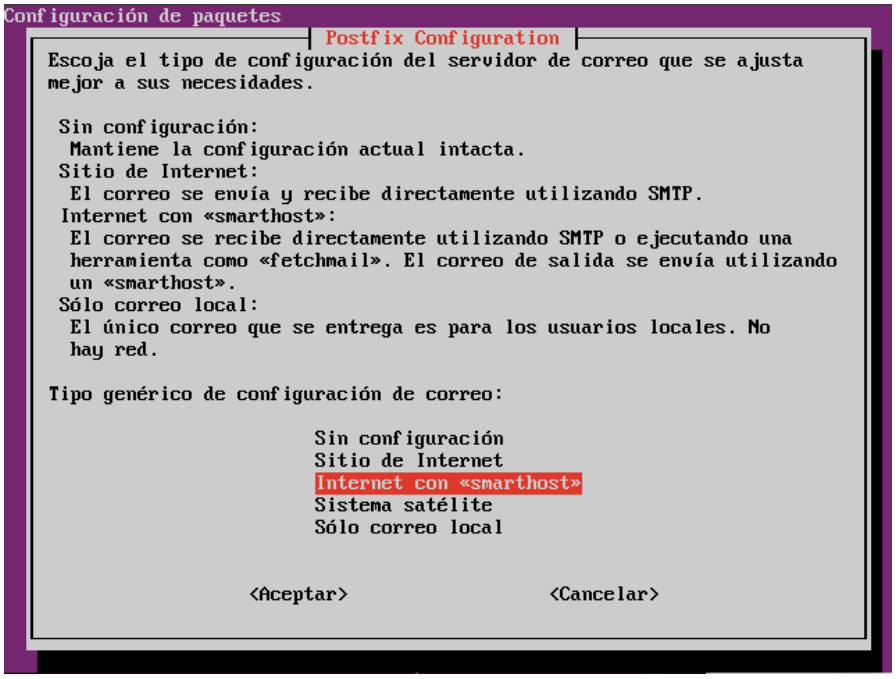


FIGURA 5.48: Configuración vía Debconf de postfix en nodo de cómputo

Tras configurar el MTA, pasará a configurar la configuración de gridengine en si:

- Escogeremos la configuración automática que realiza *debconf*.
- Configuraremos el nombre de célula con el nombre de `default`.
- Finalmente especificaremos el FQDN del servidor que ejecuta el demonio Qmaster, como puede verse en la figura 5.49.

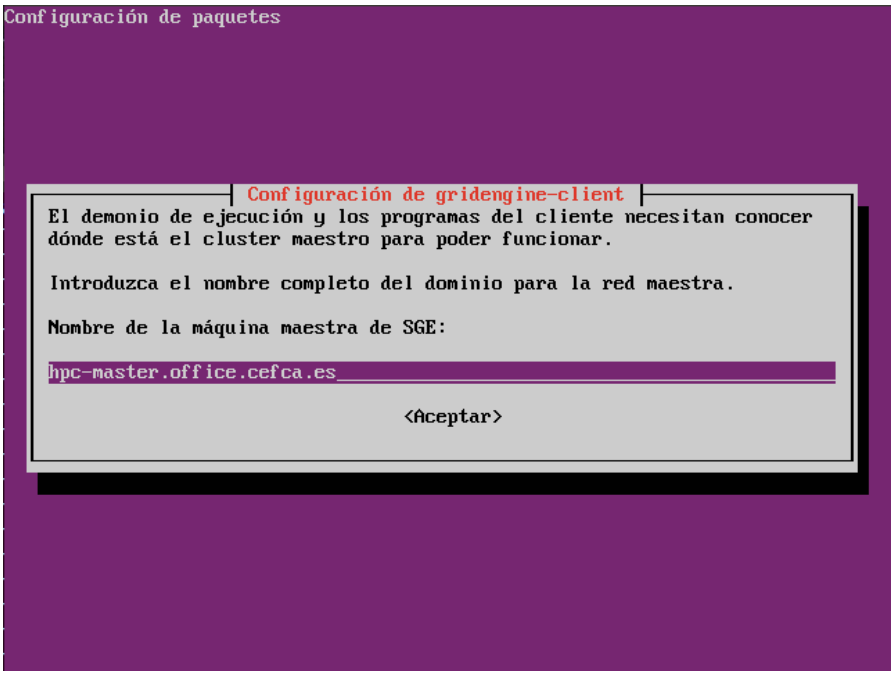


FIGURA 5.49: Configuración vía Debconf de gridengine

5.6.3. Configuración

El proceso de configuración en este proyecto lo vamos a realizar mediante la herramienta gráfica *qmon*, esto nos permitirá ver de un modo gráfico los elementos de la configuración y hacer más fácil el entendimiento de los conceptos. Existe a nuestra disposición otra herramienta de línea de comandos más potente llamada *qconf* que también puede usarse para esta labor y cuyo uso es más recomendable que *qmon*.

Para realizar la configuración, deberemos realizarlo desde el host que tiene Qmaster y desde el usuario root. En nuestro caso realizaremos un ssh a este host realizando un forwarding de las X y lanzaremos la herramienta qmon. Podemos ver en la figura 5.50 el aspecto del mismo.

FIGURA 5.50: Panel de control principal de Qmon

5.6.3.1. Creación de los usuarios administradores

Para no tener que acceder con el usuario root para realizar la configuración, es más que recomendable agregar los usuarios que podrán realizarla. Para ello pincharemos en

`User Configuration` , en la pestaña `Manager` agregaremos el usuario y pulsaremos sobre `Add`. Puede verse en la figura 5.51.

Además, permitiremos que los usuarios del grupo hpc_admins puedan eliminar trabajos en ejecución de otros usuarios. Para ello, en la pestaña `Operator` procederemos del mismo modo, pero agregaremos el grupo hpc_admins poniendo @hpc_admins.

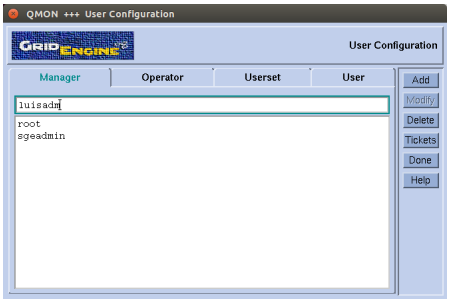


FIGURA 5.51: Agregar administrador en Qmon

5.6.3.2. Configuración de nodos

Será necesario en la configuración especificar los roles de los servidores que vimos en la primera parte. Para ello nos iremos a `Host Configuration`. Una vez ahí, en la pestaña `Submit Host` agregaremos el FQDN de los servidores hpc-master y hpc-login y pulsaremos sobre `Add`. Esto es porque desde el máster también nos será útil eliminar o modificar trabajos. Esto puede verse en la figura 5.52.

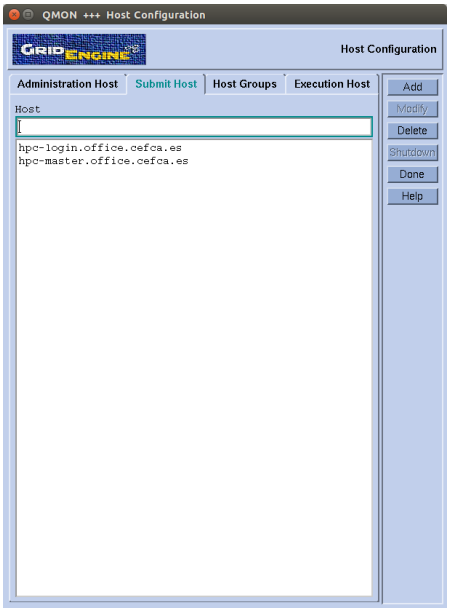


FIGURA 5.52: Agregar submit hosts en Qmon

Nos iremos a la pestaña `Host Groups` y pulsaremos en `Add`. Luego crearemos el hostgroup @allhosts y agregaremos el FQDN de todos los nodos de cómputo, como puede verse en la figura Qmon - Agrega hostgroup.

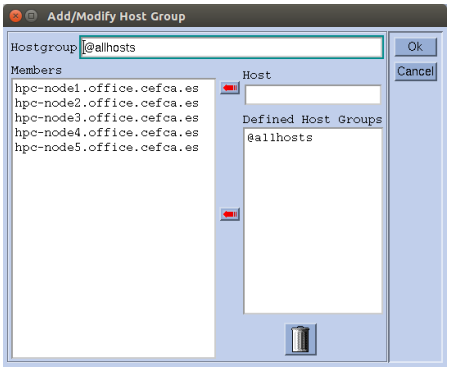


FIGURA 5.53: Agregar hostgroup en Qmon

5.6.3.3. Configuración consumibles

Como ya vimos en la parte de diseño, queremos tener un control de la memoria RAM y evitar la sobresubscripción. Para ello habrá en primer lugar deberemos definir el complex h_vmem como consumible. Para realizar esto, habrá que ir a `Complex Configuration`, modificar el `h\_vmem` para que sea consumible, le daremos el valor por defecto 6500000K y pulsaremos `Modify`, luego `Commit`. Puede verse la figura 5.54.

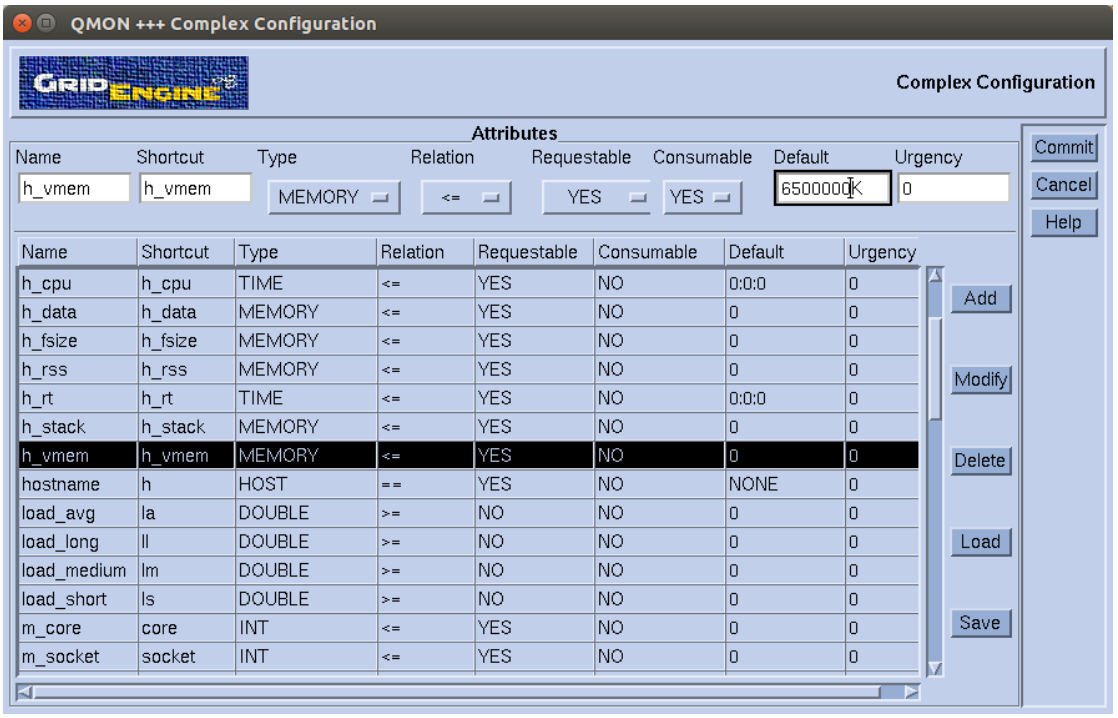


FIGURA 5.54: Configurando la memoria como consumible en Qmon

También modificaremos `h\_fsize` para que sea un consumible y le daremos un valor por defecto de 20G. Esto evitará que un proceso de usuario cree un fichero mayor de 20GB (podrá hacerlo pero deberá especificarlo explícitamente en la definición del trabajo).

Ahora deberemos configurar los valores consumibles a nivel de host para evitar la sobre-subscripción, para ello deberemos irnos de nuevo a `Host Configuration` y luego pinchar en la pestaña `Execution Host`. Pulsaremos en cada uno de los hosts y pulsaremos `Modify`.

Nos iremos a la pestaña `Consumables/Fixed Attributes` y pulsaremos en la cabecera de la rejilla sobre Name (la interfaz es así, no es para ordenar los valores sino para editarlos) y agregaremos slots con un valor de 20 y h_vmem con un valor de 13000000K. Puede verse en la figura 5.55.

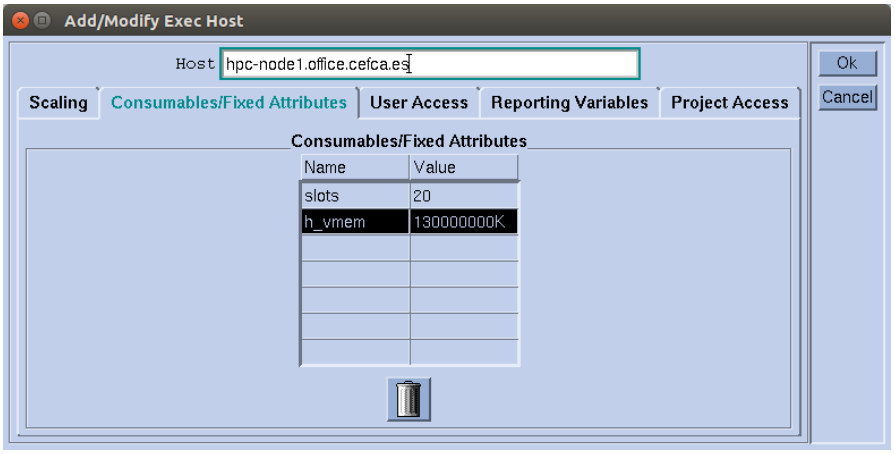


FIGURA 5.55: Configurando consumibles de un host en Qmon

5.6.3.4. Configuración entornos paralelos

Procederemos a configurar los entornos paralelos que definimos en la parte de diseño. Para ello nos iremos a `Parallel Environment Configuration` y pulsaremos en `Add`.

Crearemos el entorno parallel, para ello indicaremos en nombre parallel, en slots especificaremos 100 (que será el máximo de trabajos que se alojen en el clúster) y en `Allocation Rule` especificaremos $ pe_slots. Con esta política obligaremos a que sea todo en el mismo nodo. Podemos verlo en la figura 5.56.

Crearemos el entorno orte. Indicaremos el nombre orte, en slots especificaremos 100 y en la `Allocation Rule` también $ pe_slots. En este caso seleccionaremos la opción `Control slaves` que permitirá la integración de OpenMPI con el gestor de recursos.

Para finalizar crearemos el entorno orte-20. Indicaremos el nombre orte, en slots de nuevo 100, pero en este caso en `Allocation Rule` pondremos 20. Esto hará que aloje 20 trabajos en cada host.

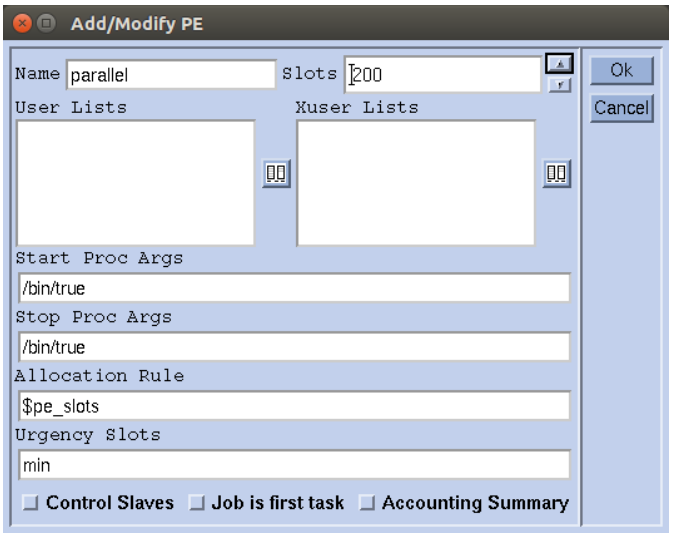


FIGURA 5.56: Creando un entorno paralelo en Qmon

5.6.3.5. Creación de las colas

Procederemos a la creación de las colas, para ello nos iremos a `Queue Control`, luego pestaña `clúster Queues` y pulsaremos `Add`.

Creación de la cola **interactive.q**

- Daremos el nombre de la cola y agregaremos el FQDN del host hpc-node5.
- En la pestaña `General Configuration` pondremos la shell /bin/bash, slots 20 y en el campo Type marcaremos sólo Interactive.
- En la pestaña `Load/Suspends Thresholds` añadiremos los valores de carga `np_load_short` a 0.9 y `mem_used` a 130000000K
- En la pestaña `Limits` en `Hard Limits` indicaremos el wallclock Time de 6 horas.
- En la pestaña `Limits` en `Corefile Size (byte)`indicaremos 0 para no se generen ficheros core.
- En la pestaña `Parallel Environment` moveremos los entornos parallel y orte a Referenced PEs.

Creación cola **main.q**

- Daremos el nombre de la cola y agregaremos el hostgroup @allhosts.
- En la pestaña `General Configuration` pondremos la shell /bin/bash, slots 20 y en el campo Type marcaremos sólo Batch.

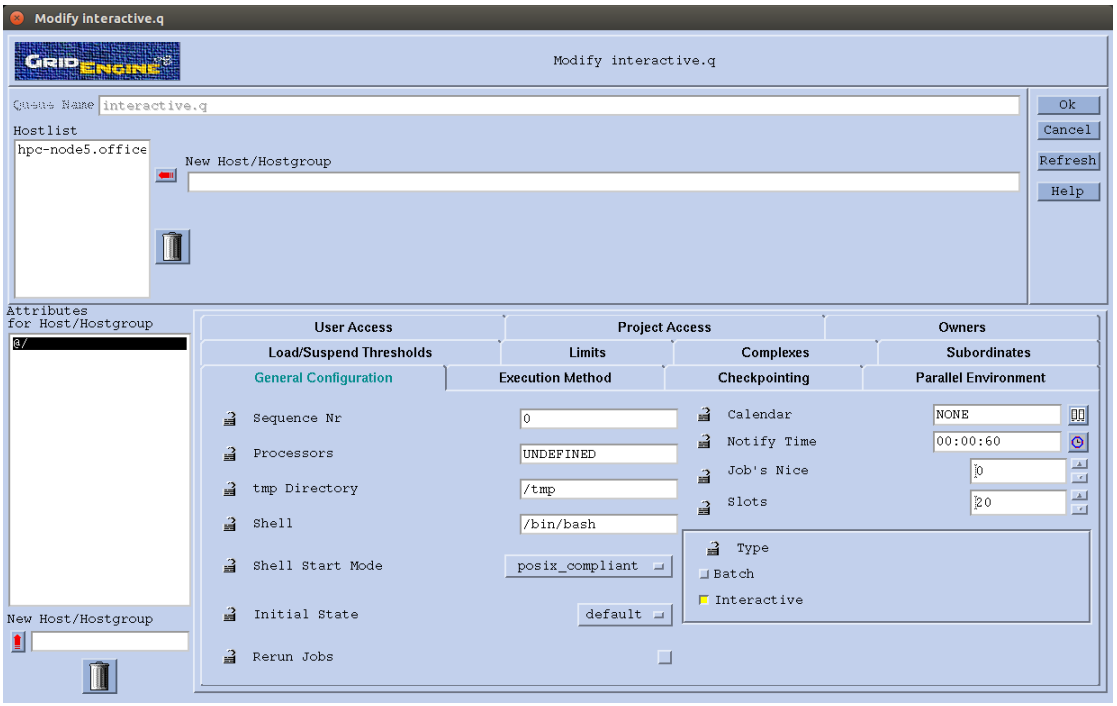


FIGURA 5.57: Creando la cola interactive.q en Qmon

- En la pestaña `Load/Suspends Thresholds` añadiremos los valores de carga `np_load_short` a 0.9 y `mem_used` a 130000000K
- En la pestaña `Limits` en `Hard Limits` indicaremos el wallclock Time de 48 horas.
- En la pestaña `Limits` en `Corefile Size (byte)`indicaremos 0 para no se generen ficheros core.
- En la parte de abajo en `Attributes for Host` agregaremos el FQDN del host hpc-node5, pincharemos sobre él y sobre el candadito que bloquea el valor de Sequence Nr. en este caso pondremos el valor 1.
- En la pestaña `Parallel Environment` moveremos los entornos parallel, orte y orte-20 a Referenced PEs.

5.6.3.6. Configuración general del clúster

Configuraremos la funcionalidad integrada para el entorno interactivo que nos servirá tanto para tener un mayor control de los trabajos lanzados de manera interactiva como para que funcione correctamente la integración con el entorno ORTE de OpenMPI. Para ello nos iremos a `clúster Configuration` en `Host global` pulsaremos en `Modify`. Luego en la pestaña `Advanced Settings` definiremos todos los `Interactive Parameters` al valor de builtin. Puede verse en la figura 5.59.

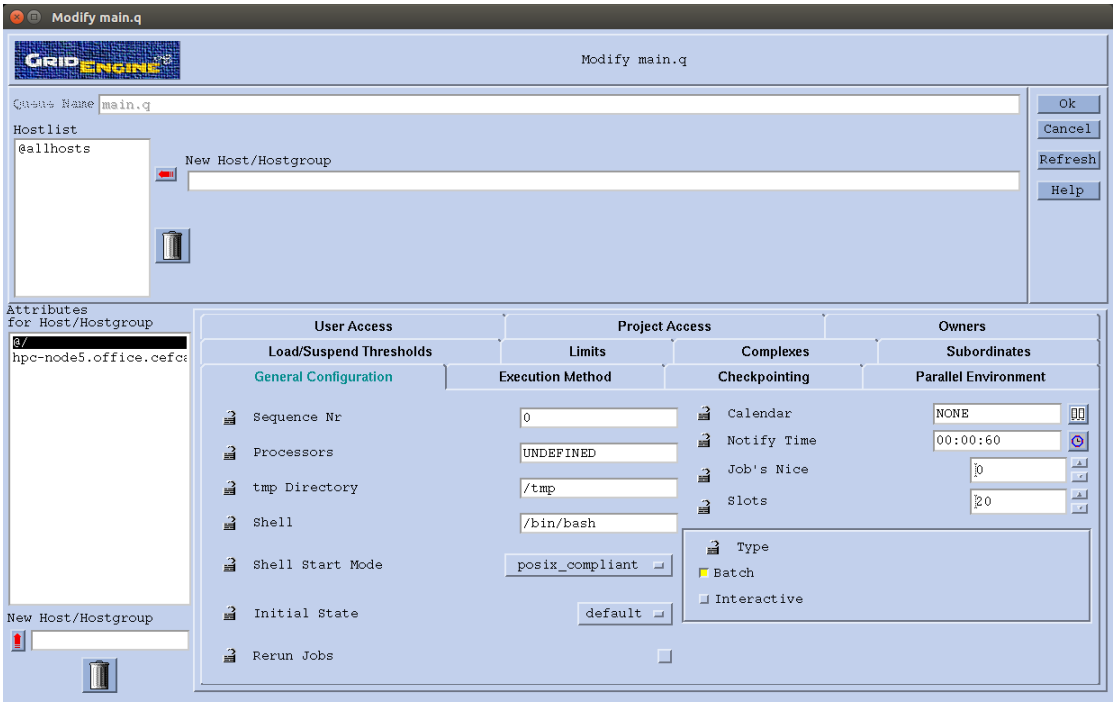


FIGURA 5.58: Creando la cola main.q en Qmon

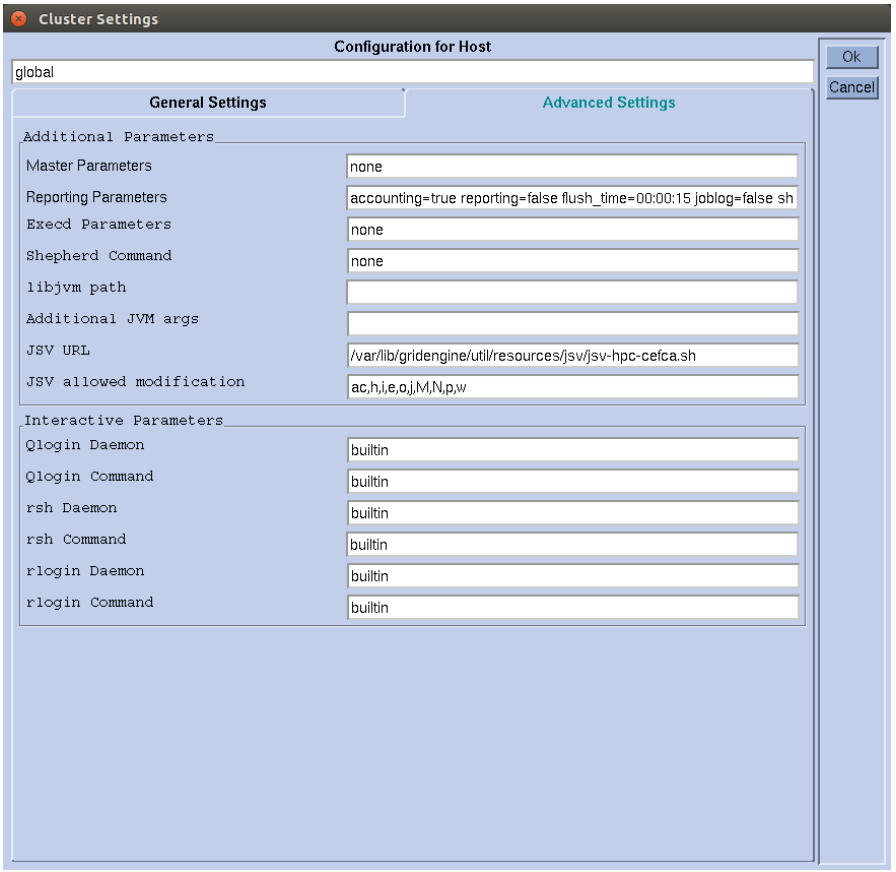


FIGURA 5.59: Configuración avanzada del Clúster en Qmon

Aprovecharemos para crear las configuraciones por defecto que los usuarios pedirán al clúster. En nuestro en el host máster crearemos únicamente le daremos al parámetro prioridad el valor de -100 para que de este modo los usuarios puedan dar valores más altos y de ese modo cambiar la prioridad de sus trabajos (los usuarios normales no pueden especificar prioridades mayores que cero).

```
root@hpc-master:~#  cat /var/lib/gridengine/default/common/sge_request
-p -100
```

Además es muy común cuando se utiliza el comando qstat ver el estado de los trabajos de todos los usuarios. Para ello agregaremos este parámetro por defecto.

```
root@hpc-master:/var/lib/gridengine/default/common#  cat /var/lib/gridengine/
    default/common/sge_qstat
-u *
```

Esto está configurado en el nodo maestro, pero los usuarios emplean el nodo de login. Aprovecharemos que el nodo de login es un contenedor del nodo maestro para crear enlaces duros de estos ficheros y además crearemos un enlace al fichero `accounting` para que los usuarios puedan ver la contabilidad de sus trabajos desde el nodo de login.

```
root@hpc-master:/var/lib/gridengine/default/common# ln accounting /var/lib/lxc/
    hpc-login/rootfs/var/lib/gridengine/default/common/
root@hpc-master:/var/lib/gridengine/default/common# ln sge_request /var/lib/lxc/
    hpc-login/rootfs/var/lib/gridengine/default/common/
root@hpc-master:/var/lib/gridengine/default/common# ln sge_qstat /var/lib/lxc/hpc
    -login/rootfs/var/lib/gridengine/default/common/
```

5.6.3.7. Configuración del scheduler

Configuraremos el scheduler para que siga una estrategia fillup en la asignación de los nodos y para que asigne en último lugar al hpc-node5 en las tareas de la cola main.q. Para ello nos iremos a Scheduler Configuration, luego en la pestaña General Parameters configuraremos Sort by sequence number y en Load Formula indicaremos slots. Puede verse en la figura 5.60.

5.6.3.8. Configuración de JSV

Para finalizar, haremos uso de la funcionalidad JSV que nos permite introducir cierta lógica antes de pasar los trabajos al scheduler. En nuestro caso únicamente haremos que rechace automáticamente los trabajos enviados al entorno paralelo orte-20 si estos trabajos no son múltiplos de 20.

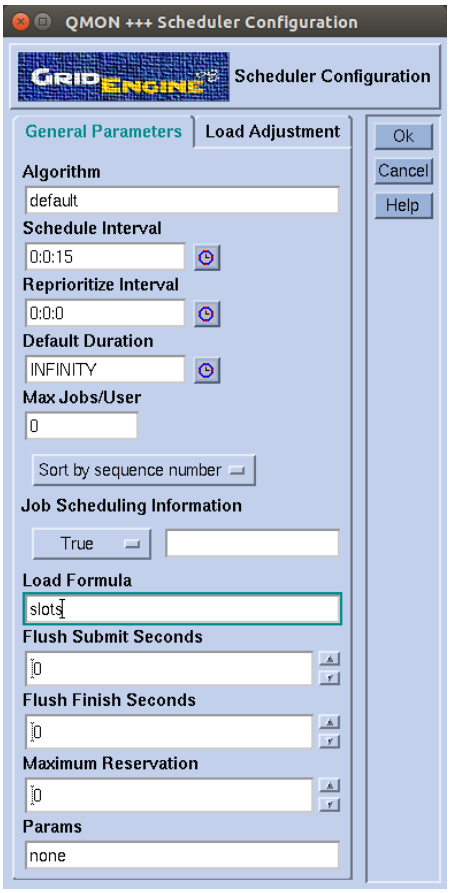


FIGURA 5.60: Configuración del scheduler del Clúster en Qmon

En los paquetes que he modificado existen algunos binarios que son necesarios y los scripts de ejemplo están ahora disponibles en /usr/share/gridengine/jsv. Para ponerlo en marcha:

```
root@hpc-master:~# mkdir -p /var/lib/gridengine/util/resources
root@hpc-master:~# cp /usr/share/gridengine/arch /var/lib/gridengine/util
root@hpc-master:~# cp -r /usr/share/gridengine/jsv /var/lib/gridengine/util/
    resources
```

Luego crearemos el script jsv al que se llamará en cada submit que se realice. En nuestro caso será el que viene a continuación.

```
root@hpc-master:~# cat /var/lib/gridengine/util/resources/jsv/jsv-hpc-cefca.sh
#!/bin/bash
#
# Script basado en jsv.sh, para comentarios ver este fichero
# Si se invoca el entorno paralelo orte-20 se comprueba que
# se solicitan múltiplos de 20

PATH=/bin:/usr/bin


jsv_on_start()
{
```

```
#   jsv_send_env
   return
}


jsv_on_verify()
{
   do_correct="false"
   do_wait="false"

   if [ "`jsv_get_param pe_name`" == "orte-20" ]; then
          slots=`jsv_get_param pe_min`
          i=`echo "$slots % 20" | bc`

          if [ $i -gt 0 ]; then
                 jsv_reject "Parallel job does not request a multiple of 20 slots
    "
                 return
          fi
   fi

   if [ "$do_wait" = "true" ]; then
          jsv_reject_wait "Job is rejected. It might be submitted later."
   elif [ "$do_correct" = "true" ]; then
          jsv_correct "Job was modified before it was accepted"
   else
          jsv_accept "Job is accepted"
   fi
   return
}

. ${SGE_ROOT}/util/resources/jsv/jsv_include.sh

jsv_main
```

Una vez que tenemos todo listo, nos iremos a `Cluster Configuration` en `Host global` pulsaremos en `Modify`. Luego en la pestaña `Advanced Settings` definiremos JSV URL con el valor `/var/lib/gridengine/util/resources/jsv/jsv-hpc-cefca.sh` como puede verse en la figura Qmon - Configuracion avanzada clúster.

5.6.4. Pruebas de ejecución

5.6.4.1. Pruebas realizadas

Se realizaron numerosas pruebas para ver el correcto funcionamiento del gestor de recursos:

1 Asignación correcta de cola y servidor para los trabajos interactivos y batch.

2 Uso de política fillup en la asignación de los trabajos.

3 Distribución de los trabajos según los requisitos de memoria y mantenimiento de los límites de los recursos solicitados.

4 Funcionamiento de los entornos paralelos y rechazo automático de solicitudes no múltiplos de 20 para el entorno orte-20.

Algunos programas de pruebas Estos son los programitas con los que hice las pruebas del consumo de recursos.

```
lguillen@hpc-node1:~/devora$ cat devora_ram.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Tamaño de página 4KB */
#define PAGE_SZ (1<<12)

void print_usage() {
        fprintf(stderr, "devora_ram niteraciones ram_mb waitsecs\n");
}

int main(int argc, char *argv[]) {
        int i, npag, nit, waitsecs;
        int totalmb, mb;

        if(argc<4) {
                print_usage();
                exit(2);
        }

        sscanf(argv[1], "%d", &nit);
        sscanf(argv[2], "%d", &mb);
        sscanf(argv[3], "%d", &waitsecs);

        for(totalmb = 0, i = 1; i <= nit; i++) {
                for (npag = 0; npag < ((unsigned long)mb<<20)/PAGE_SZ ; ++npag) {
                        void *m = malloc(PAGE_SZ);
                        if (!m) {
                                printf("Error al solicitar memoria");
                                break;
                        }
                        memset(m, 0, 1);
                }
                totalmb+= mb;
                printf("Alocados %d MB de RAM\n", totalmb);
                sleep(1);
        }

        printf("Esperando %d segundos\n", waitsecs);
        sleep(waitsecs);
```

```
        return 0;
}
```

```
lguillen@hpc-node1:~/devora$ cat devora_cpu.c
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <errno.h>

#define MAX_CORES       128

void print_usage() {
        fprintf(stderr, "devora_cpu numcpus waitsecs\n");
}

/** calcula primos */
void do_cpu()
{
        unsigned long i, num, primes = 0;
        for (num = 1; num <= INT_MAX; ++num) {
                for (i = 2; (i <= num) && (num % i != 0); ++i);
                if (i == num)
                        ++primes;
        }
}

void sig_handler(int signo) {
        switch(signo) {
                case SIGALRM: break;
                case SIGTERM:
                case SIGINT:
                case SIGQUIT:
                        printf("Interrumpido\n"); break;
        }
}

int main(int argc, char ** argv)
{
        int i, numcpus, waitsecs;
        pid_t pids[MAX_CORES];

        if(argc<3) {
                print_usage();
                exit(2);
        }

        sscanf(argv[1], "%d", &numcpus);
        sscanf(argv[2], "%d", &waitsecs);

        for(i = 0; i < numcpus; i++) {
                printf("Usando CPU %d\n", i+1);
```

```
                if(!(pids[i] = fork())) {
                        do_cpu();
                        exit(0);
                }
                if(pids[i] < 0) {
                        perror("Fork");
                        exit(2);
                }
        }

        signal(SIGALRM, sig_handler);
        signal(SIGTERM, sig_handler);
        signal(SIGINT, sig_handler);
        signal(SIGQUIT, sig_handler);

        printf("Esperando %d segundos\n", waitsecs);
        alarm(waitsecs);
        pause();

        printf("Matando procesos");
        for(i = 0; i < numcpus; i++) {
                kill(pids[i], SIGQUIT);
        }

        for (i = 0; i < numcpus; ++i) {
                waitpid(pids[i], NULL, 0);
        }

        return 0;
}
```

5.6.4.2. Problemas encontrados

Además de los problemas que se ven en el anexo referentes a Qmon y a ficheros que faltan para la correcta implementación de los scripts JSV, encontré un problema de cierta importancia. El planificador, a la hora de ejecutar trabajos que emplean el entorno paralelo, ignora el parámetro load_formula en la asignación de los trabajos. Esto básicamente quiere decir que la política fillup que se realiza automáticamente y que funciona para los batch normales y los array jobs, no funciona correctamente para los trabajos que hacen uso de un entorno paralelo. Pude comprobar que no soy la única persona y que este bug [3] sigue presente en GE 2011.11.

Tras investigar un poco el código fuente, creo que el problema se encuentra en el fichero source/libs/sched/sge_select_queue.c donde se emplea la función parallel_maximize_slots_pe() en el proceso de selección de host y trata de maximizar la ejecución de trabajos sin tener en cuenta la ordenación de los hosts.

[3] http://gridengine.org/pipermail/users/2012-August/004438.html

Solucionar este bug implica un entendimiento muy profundo del código de gridscheduler que requiere una dedicación de muchas horas de las que no se dispone. Como comenté, parece que el problema persiste en GE 2011.11 pero parece ser que está solucionado en la otra implementación libre existente: **Son Of Grid Engine**. Este es uno de los motivos por el cual se valorará la migración a este software en el futuro. Un posible *workaround* para evitar el efecto de que se ocupen nodos vacíos con trabajos paralelos pequeños consistirá en que el usuario indique de manera explícita la instancia de la cola en el submit de los trabajos. Por ejemplo, si vamos a lanzar un trabajo paralelo de 10 y vemos que sólo hay 2 slots ocupados en el nodo hpc-node1 podemos realizar:

```
$  qsub -q main.q@hpc-node1.office.cefca.es orte10.sh
```

5.7. Sistema de monitorización

Ya tenemos a nuestro sistema de clúster HPC funcionando, para tratar de garantizar que esto siga siendo así a continuación implementaremos el sistema de monitorización.

5.7.1. Preparando los dispositivos para la monitorización

En la sección dedicada al proceso de instalación inicial, ya vimos una pequeña parte de monitorización cuando configuramos las alertas por email en la HP MSA y las HP iLO. Sin embargo, esto es del todo insuficiente. El principal motivo es que se trata de un mecanismo de *monitorización pasiva*, lo que quiere decir que es el sistema monitorizado el que envía las alarmas y no es el sistema de monitorización el que comprueba su estado. Evidentemente la monitorización pasiva tiene un problema fundamental: si el sistema cae completamente o los sistemas de comunicación caen, no habrá email que alerte de este hecho y por lo tanto no seremos capaces de detectar el problema (hasta que los usuarios nos alerten).

Sin embargo no es este el único problema, también es muy conveniente integrar estas alarmas con un sistema centralizado de monitorización, de modo que se almacenen en una base de datos común todos los eventos y que nos permita realizar un seguimiento y una correlación de eventos. También podremos configurar el sistema de monitorización para que realice determinadas acciones correctivas de forma automática ante determinados eventos, de forma que eviten la intervención del administrador.

Para la integración con los sistemas de monitorización, se hará uso del protocolo SNMP. Para ello se configurarán los agentes disponibles en los dispositivos de forma que permitan, por un lado, la lectura de la base de datos de estado (MIB) desde el sistema de

monitorización y, por otro lado, envíen un aviso (TRAP) al sistema de monitorización cuando ocurra un evento sensible de atención. De este modo podremos realizar:

- Monitorización activa: desde el sistema de monitorización iremos comprobando por encuesta vía SNMP datos de estado y métricas de los dispositivos.
- Monitorización pasiva: el sistema monitorizado enviará al sistema de monitorización una alarma (TRAP) cuando ocurra un evento sensible de ser atendido

5.7.1.1. Configuración del switch Cisco 3750

Nos conectaremos mediante SSH y realizaremos la configuración. Para ello crearemos una lista de control de acceso con las máquinas que pueden conectarse al servicio SNMP (en nuestro caso únicamente el sistema de monitorización) y configuraremos la clave de comunidad con acceso de sólo lectura. También configuraremos que los números de identificadores asignados en la MIB de las interfaces se mantengan a lo largo del tiempo.

```
cefca-swt-013#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
cefca-swt-013(config)#access-list 50 permit 10.10.10.100
cefca-swt-013(config)#access-list 50 deny   any
cefca-swt-013(config)#snmp-server community CEFC@_RO RO 50
cefca-swt-013(config)#snmp ifmib ifindex persist
cefca-swt-013(config)#end
cefca-swt-013#wr
Building configuration...
[OK]
```

Para la monitorización pasiva habilitaremos que se realicen traps en los eventos que nos interesa que nos avise y el host al que enviará las traps junto con la clave. En nuestro caso habilitaremos las que se enumeran a continuación. Nótese que he incluido las traps config-copy y config, esto se hará para realizar pruebas de que funciona correctamente.

```
cefca-swt-013#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
cefca-swt-013(config)#snmp-server enable traps snmp authentication linkdown
    linkup coldstart warmstart
cefca-swt-013(config)#snmp-server enable traps cef resource-failure peer-state-
    change peer-fib-state-change inconsistency
cefca-swt-013(config)#snmp-server enable traps config-copy
cefca-swt-013(config)#snmp-server enable traps config
cefca-swt-013(config)#snmp-server enable traps cpu threshold
cefca-swt-013(config)#snmp-server enable traps bridge newroot topologychange
cefca-swt-013(config)#snmp-server enable traps stpx inconsistency root-
    inconsistency loop-inconsistency
cefca-swt-013(config)#snmp-server enable traps port-security
cefca-swt-013(config)#snmp-server enable traps envmon fan shutdown supply
    temperature status
```

```
cefca-swt-013(config)#snmp-server enable traps errdisable
cefca-swt-013(config)#snmp-server host 10.10.10.100 version 2c CEFC@_TRAP
cefca-swt-013(config)#end
cefca-swt-013#wr
Building configuration...
[OK]
```

5.7.1.2. Configuración de la MSA

Para la configuración de la MSA, únicamente configuraremos la comunidad SNMP de lectura para la monitorización activa del dispositivo. La monitorización pasiva la realizaremos únicamente vía email como vimos en la sección de instalación. Al no existir en la red mucha información al respecto, por lo que requiere un pequeño desarrollo específico y por ser ser la única MSA de la que se dispone en CEFCA, no van a tratarse en este proyecto el tratamiento de los TRAPs.

5.7.1.3. Configuración de los HP Proliant

Una de las grandes ventajas introducidas con la generación 8 de los servidores Proliant es la capacidad de monitorización sin agentes. Gracias a esta funcionalidad, a la que llama *Agentless Management Service*, será posible monitorizar los datos relativos al estado del servidor sin la necesidad de instalar un agente en el sistema operativo. Esto nos será sumamente útil, ya que uno de nuestros objetivos será maximizar el aprovechamiento de los nodos de cómputo y podremos liberar a las CPUs de la tarea de chequear sensores de temperatura, ventiladores, estados de las memorias, etc. En el gráfico 5.61, podemos ver que en este modo, la comunicación vía SNMP la realiza íntegramente la iLO, que tiene a su disposición toda la información hardware del servidor. Además, mediante un módulo de kernel y un pequeño servicio, podrá recoger información básica del sistema operativo como si se encuentra en ejecución o el el estado del enlace de las nic.

Para ello configuraremos la gestión SNMP e instalaremos en el sistema operativo el paquete `hp-msa` del *Service Pack for Proliant de HP*.

5.7.2. Monitorización de la red con Cacti

5.7.2.1. Monitorización del switch Cisco

Para la monitorización del switch, es necesario tener previamente instalado y configurado un sistema Cacti con su spooler y definiciones RRAs. También será necesario instalar

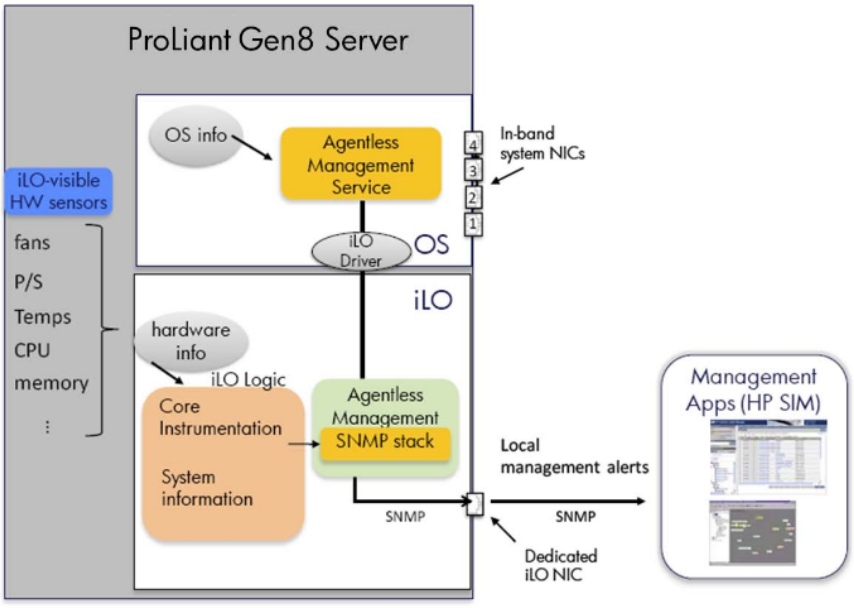


FIGURA 5.61: Diagrama HP Agentless Monitoring System

varias plantillas existentes en la red para la monitorización de equipos Cisco. Sin embargo, dichas plantillas son para otros dispositivos, por lo que será necesario crear una plantilla específica para nuestro modelo de switch a partir de las existentes. Por supuesto, deberemos realizar la configuración del agente SNMP del switch con la comunidad de lectura necesaria y para que atienda a las peticiones del spooler de monitorización.

Pero antes de todo, será necesario realizar la configuración de monitorización en el dispositivo Cisco para ello nos iremos `Host Templates -> Add`. Daremos un nombre a nuestra plantilla y agregaremos los Graph Templates y Data Queries que pueden verse en la figura 5.62.

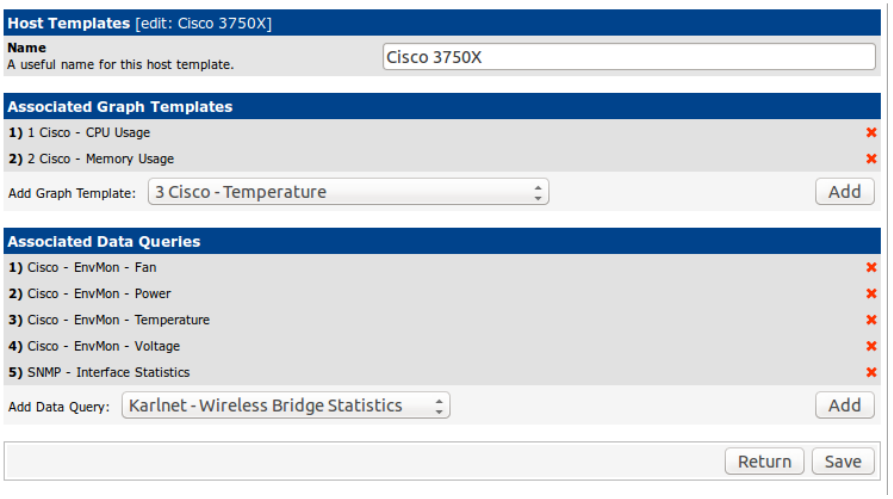


FIGURA 5.62: Creando una plantilla de modelo Cisco 3750 en Cacti

Una vez tenemos la plantilla, crearemos el dispositivo. Para ello nos iremos a `Devices -> Add` y seleccionaremos la plantilla que hemos creado junto con los datos para el switch, tal y como puede verse en la figura 5.63. Cuando grabemos, cacti tratará de conectarse vía snmp y obtendremos información de uptime del switch.

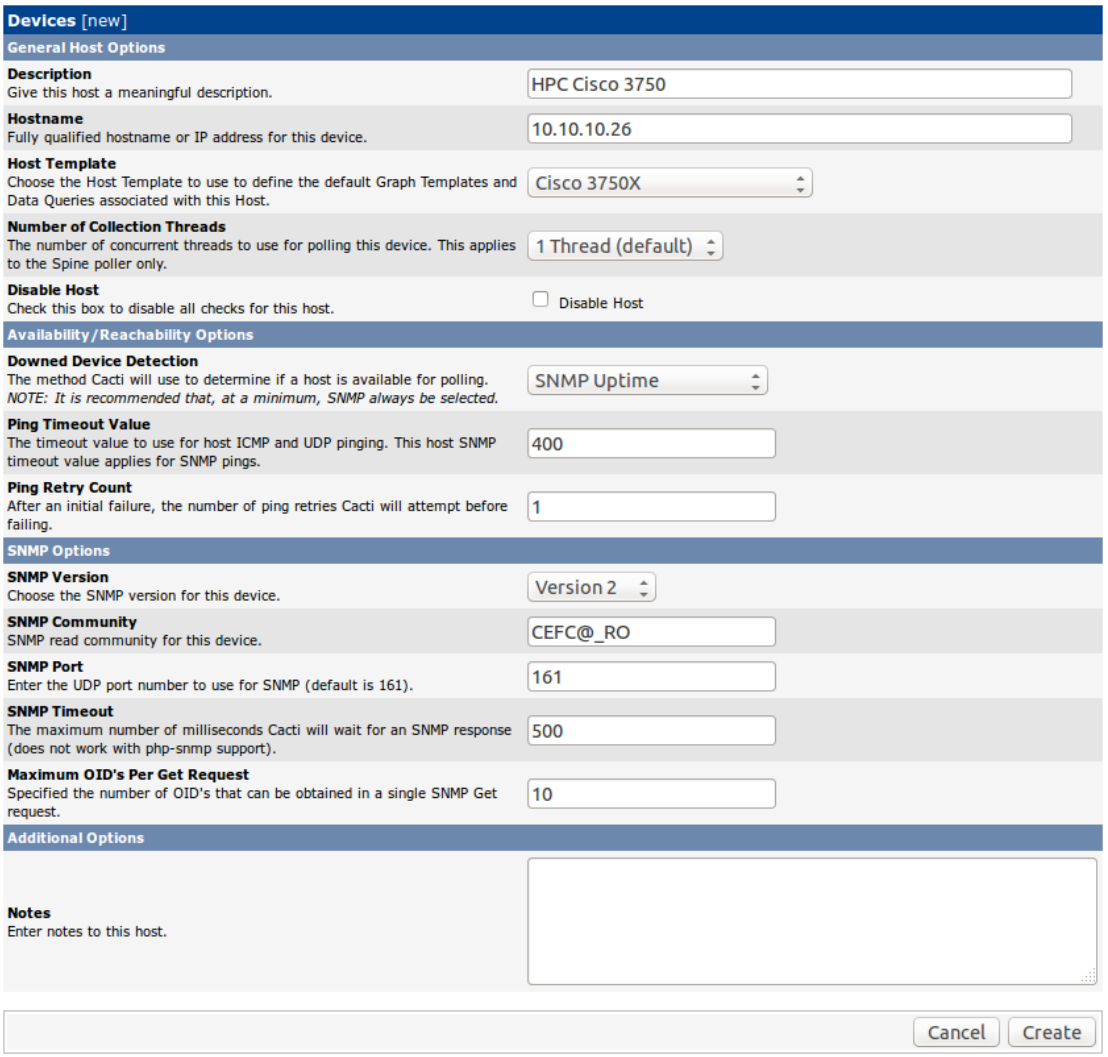


FIGURA 5.63: Creando el dispositivo del switch Cisco en Cacti

Ya tenemos el dispositivo, ahora tendremos que crear los gráficos. Para ello pulsaremos sobre `Create Graphs for this Host`. Aquí seleccionaremos los gráficos que nos interesan, en primer lugar seleccionaremos los gráficos de CPU, memoria y el gráfico de temperatura sobre el sensor. Esto puede verse en la figura 5.64. Tras esto pulsaremos el botón `Create`.

A continuación volveremos a crear gráficos, esta vez para crear gráficos del tráfico en las interfaces de red. Para ello seleccionaremos las interfaces de las que deseamos crear los gráficos, escogeremos el tipo de gráfico `In/Out Bytes (64-bit Counters)` y pulsaremos `Create`. Puede verse en la figura 5.65.

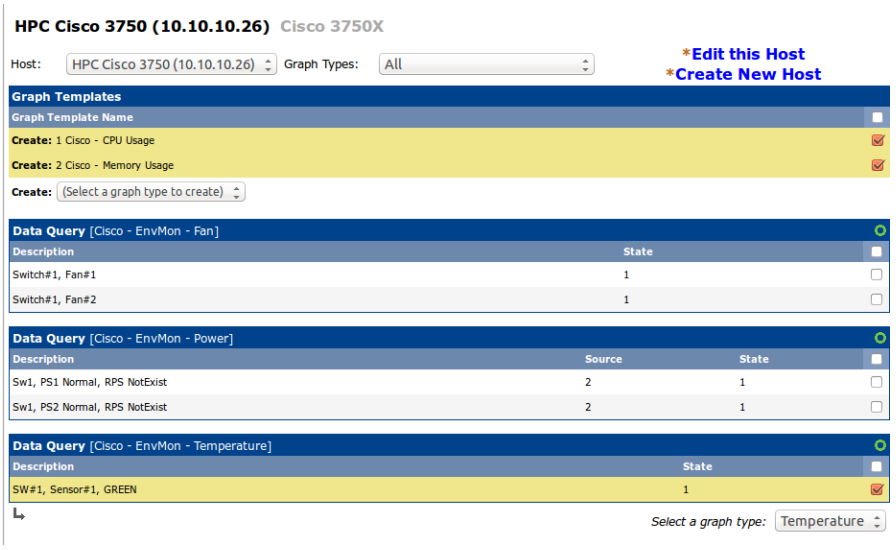


FIGURA 5.64: Creando gráficos generales del switch Cisco en Cacti

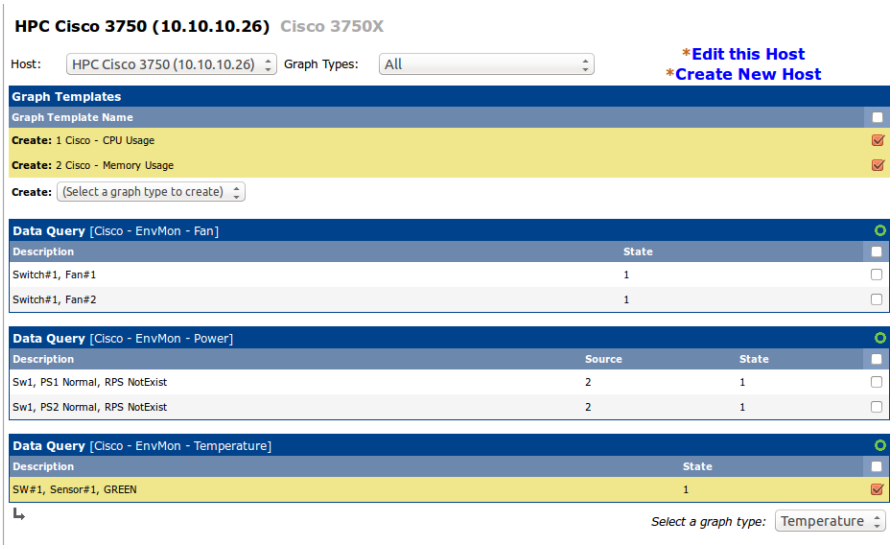


FIGURA 5.65: Creando gráficos de tráfico de interfaces del switch Cisco en Cacti

Una vez que tenemos todos los gráficos, deberemos crear la interfaz para visualizarlos, para ello nos iremos a Graph Trees y seleccionaremos el nodo sobre el que colgaremos nuestros gráficos. En nuestro caso, ya existe un nodo llamado Switches. Sobre dicho nodo pulsaremos sobre Add y crearemos un nuevo item de tipo Host y seleccionaremos nuestro host. Puede verse en la figura 5.66.

Al cabo de unos minutos, si nos vamos a la pestaña Graphs, podremos visualizar los gráficos que hemos creado para nuestro switch, como puede verse en la figura 5.67.

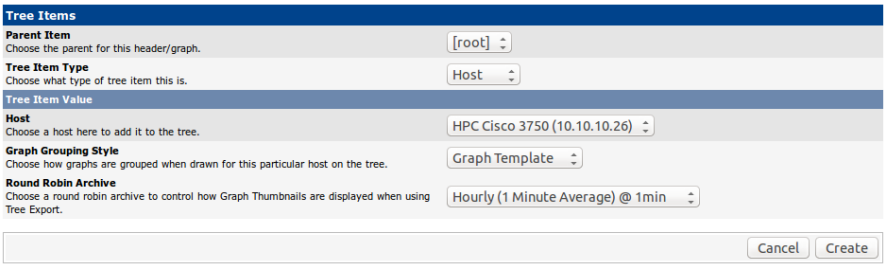


FIGURA 5.66: Agregando switch Cisco al árbol de gráficos en Cacti

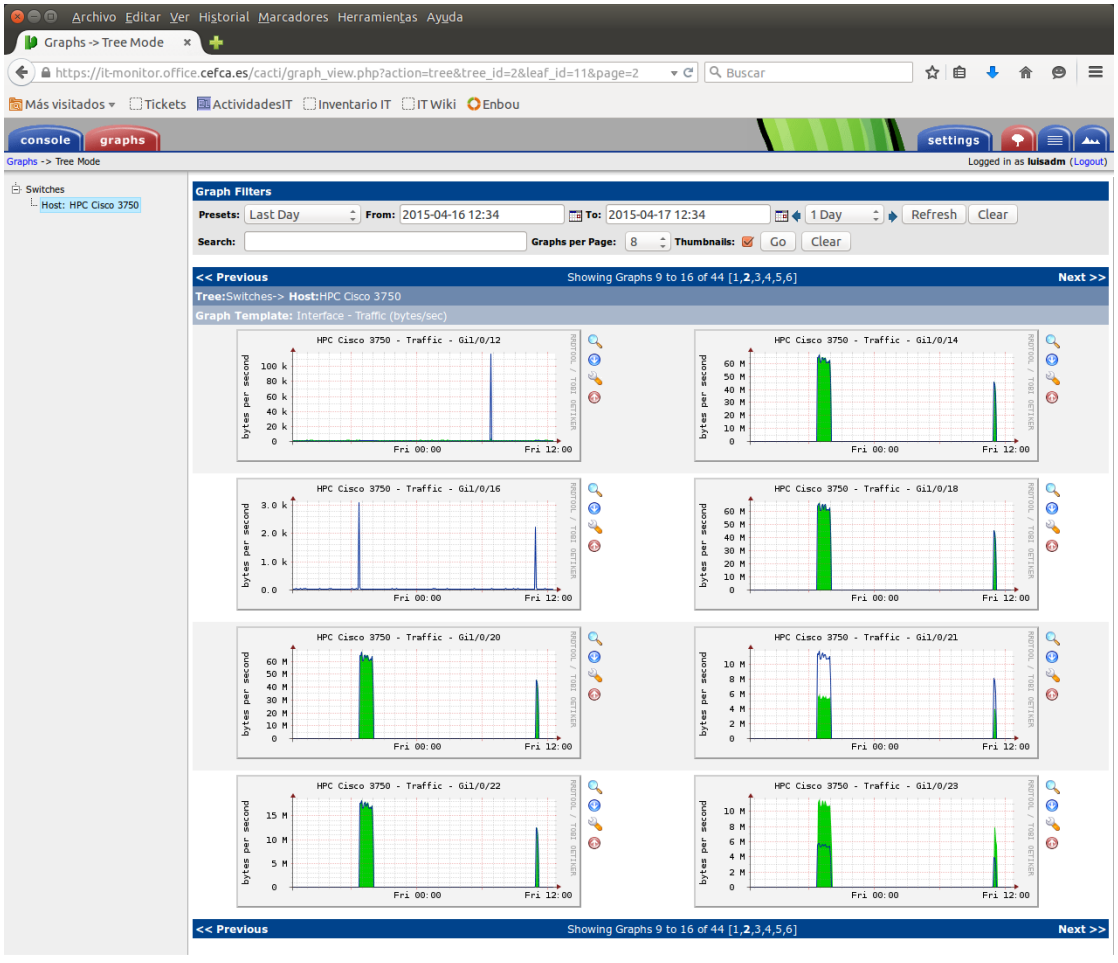


FIGURA 5.67: Viendo los gráficos del switch Cisco en Cacti

5.7.3. Monitorización del estado del clúster con Nagios

Tras el estudio de las funcionalidades y los agentes disponibles para los dispositivos se realizó el diseño de la figura 5.68.

El modelo de datos en Nagios tiene ciertas similitudes con un modelo orientado a objetos. Para el diagrama he usado la notación de estereotipo al objeto usado en Nagios y la cursiva para definir que es una plantilla. Esto es debido a la similitud conceptual existentes de las plantillas de Nagios con las clases abstractas. He empleado el símbolo de la herencia para modelar la relación de herencia de plantilla y el símbolo de dependencia para representar las referencias entre objetos dentro de Nagios.

Antes de entrar a describir la configuración, comentar brevemente algunos conceptos:

- Monitorización activa: es aquella realizada desde el sistema de monitorización hacia el sistema monitorizado.
- Monitorización pasiva: es aquella en la cual el sistema monitorizado avisa de eventos críticos que han podido sucederle al sistema de monitorización.
- Monitorización directa: es aquella en la que el agente final que chequea el estado del servicio se encuentra en el sistema de monitorización.
- Monitorización indirecta: es aquella en la que el agente final se encuentra en el sistema monitorizado o en un sistema monitorizado intermedio. Para esto se utiliza generalmente el protocolo NRPE.

5.7.3.1. Monitorización del switch Cisco

Monitorización activa Descargaremos de la red el plugin `check-cisco.pl` [4] y lo copiaremos en el directorio `/usr/local/lib/nagios-plugins`. Después crearemos un fichero de configuración de Nagios con los comandos necesarios para utilizar el plugin.

```
root@cefca-trm-001:~# cat /etc/nagios-plugins/config/cisco.cfg
define command{
        command_name    check_cisco_temp
        command_line    /usr/local/lib/nagios-plugins/check-cisco.pl -H '
    $HOSTADDRESS$' -t temp -w '$ARG1$' -c '$ARG2$' -C '
    $_HOSTCISCO_SNMP_COMMUNITY$'
        }

define command{
        command_name    check_cisco_fan
```

[4] https://exchange.nagios.org/directory/Plugins/Hardware/Network-Gear/Cisco/Check-Cisco-Catalyst/details

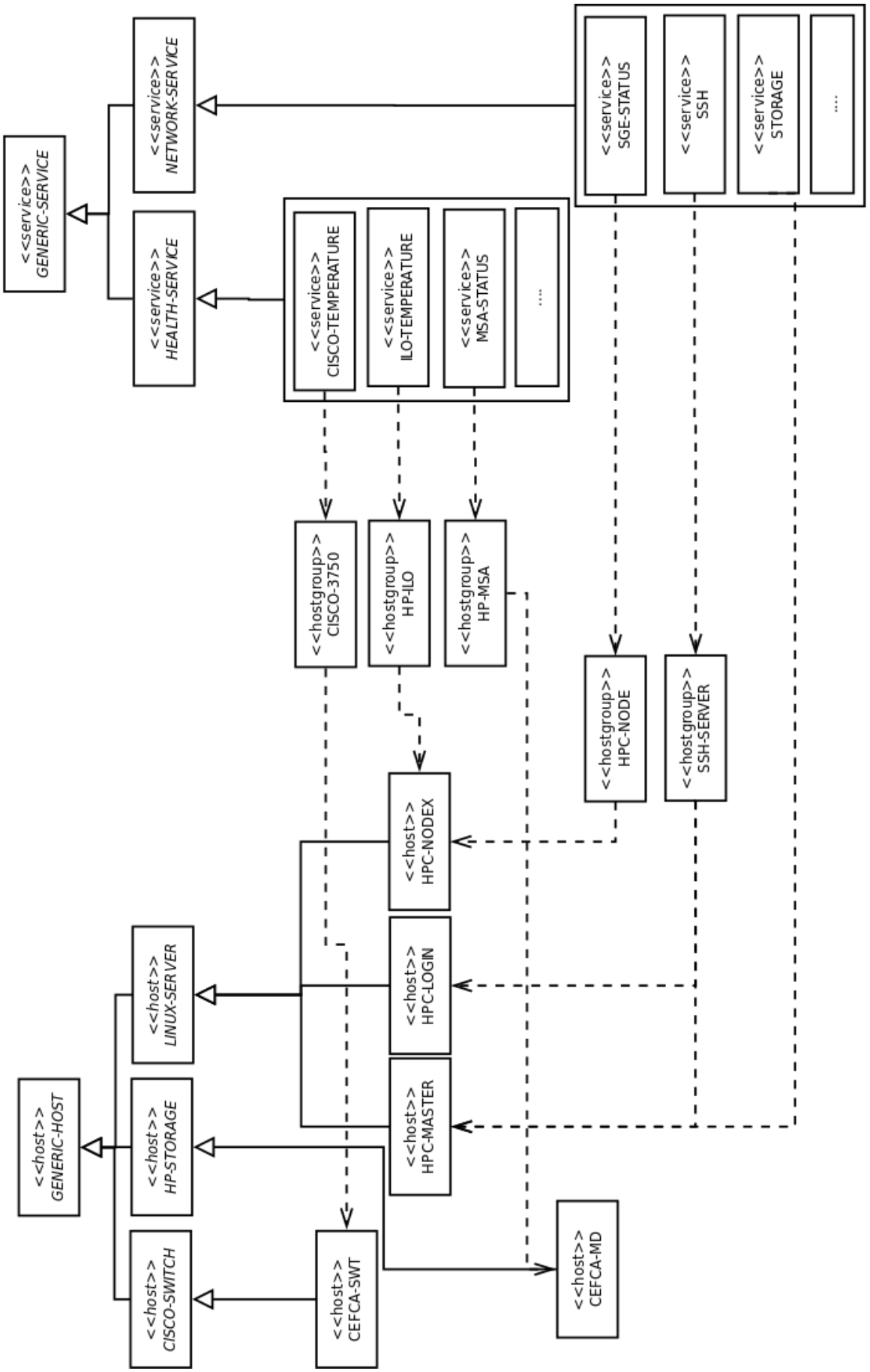


FIGURA 5.68: Modelo de datos para la monitorización con Nagios

```
        command_line    /usr/local/lib/nagios-plugins/check-cisco.pl -H '
    $HOSTADDRESS$' -t fan -C '$_HOSTCISCO_SNMP_COMMUNITY$'
        }

define command{
        command_name    check_cisco_ps
        command_line    /usr/local/lib/nagios-plugins/check-cisco.pl -H '
    $HOSTADDRESS$' -t ps -C '$_HOSTCISCO_SNMP_COMMUNITY$'
        }

define command{
        command_name    check_cisco_cpu
        command_line    /usr/local/lib/nagios-plugins/check-cisco.pl -H '
    $HOSTADDRESS$' -t cpu -w '$ARG1$' -c '$ARG2$' -C '$_HOSTCISCO_SNMP_COMMUNITY$
    '
        }

define command{
        command_name    check_cisco_mem
        command_line    /usr/local/lib/nagios-plugins/check-cisco.pl -H '
    $HOSTADDRESS$' -t mem -C '$_HOSTCISCO_SNMP_COMMUNITY$'
        }

define command{
        command_name    check_cisco_interface
        command_line    /usr/local/lib/nagios-plugins/check-cisco.pl -H '
    $HOSTADDRESS$' -t int -i '$ARG1$' -C '$_HOSTCISCO_SNMP_COMMUNITY$'
        }
```

Una vez que hemos definido los comandos, crearemos la plantilla cisco-switch de la que heredarán nuestros switches y crearemos nuestro host.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hosts/cisco-switch.cfg
define host {
        use                 generic-host
        name                cisco-switch
        check_interval      5
        max_check_attempts  2
        icon_image          cefca/switch.png
        icon_image_alt      Switch
        statusmap_image     cefca/switch.gd2
        register            0
}
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hosts/hpc-cefca/cefca-swt-013.cfg
define host{
        use                     cisco-switch
        host_name               cefca-swt-013
        alias                   Switch del HPC Cisco 3750X
        address                 10.10.10.26
        _CISCO_SNMP_community   CEFC@_RO
}
```

Ahora crearemos un grupo de host para nuestro modelo de switch y agregaremos al switch como miembro del grupo.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hostgroups/cisco-switches.cfg
define hostgroup {
        hostgroup_name  cisco-3750
        alias           Switches Cisco 3750
        members         cefca-swt-013
}
```

Hasta ahora hemos definido los comandos con los que chequeamos el switch, pero no los servicios que harán uso de los mismos y que deberán estar relacionados con los hosts. En nuestro caso, crearemos los servicios y los relacionaremos con el hostgroup cisco-* para que todos los modelos Cisco los utilicen.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/services/health/cisco-switch.cfg
define service{
        use                     health-service
        service_description     Cisco temperatura
        check_command           check_cisco_temp!45!55
        hostgroup_name          cisco-*
}

define service{
        use                     health-service
        service_description     Cisco ventiladores
        check_command           check_cisco_fan
        hostgroup_name          cisco-*
}

define service{
        use                     health-service
        service_description     Cisco alimentacion
        check_command           check_cisco_ps
        hostgroup_name          cisco-*
}

define service{
        use                     health-service
        service_description     Cisco memoria
        check_command           check_cisco_mem
        hostgroup_name          cisco-*
}

define service{
        use                     health-service
        service_description     Cisco cpu
        check_command           check_cisco_cpu!70!90
        hostgroup_name          cisco-*
}
```

Ya tenemos la configuración de monitorización activa, solo falta reiniciar nagios y en unos minutos veremos el chequeo de todos nuestros servicios.

Monitorización pasiva Esta parte de la monitorización es más compleja, haremos uso de las traps enviadas por el switch cisco. Para ello necesitaremos configurar un servicio de monitorización pasiva y que sea el que disparemos cuando nos llegue un TRAP SNMP mediante la interfaz que nos proporciona Nagios.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/services/health/cisco-switch.cfg
        ....
## servicio  monitorizacion pasiva
define service{
        use                     health-service
        service_description     SNMP_TRAP
        active_checks_enabled   0
        hostgroup_name          cisco-*
        check_command           check-host-alive
}
```

Modificaremos el script que hace de interfaz para los chequeos pasivos, indicando que use explícitamente la shell bash, ya que con dash (el que está asociado a sh) pueden existir algunos problemas:

```
root@cefca-trm-001:~# cat /usr/share/nagios3/plugins/eventhandlers/
    submit_check_result
#!/bin/bash
        ....
```

Durante el proceso de instalación de nuestro servidor de monitorización, junto a Nagios se debieron instalar los paquetes `snmpd` y `snmptt`. Procederemos a asegurarnos de que el servicio está configurado correctamente. Para ello el demonio snmptrapd deberá ser lanzado y agregaremos la opción `-n` para que no trate de realizar resoluciones inversas de los host orígenes de las traps.

```
root@cefca-trm-001:~# cat /etc/default/snmpd
# This file controls the activity of snmpd and snmptrapd

# Don't load any MIBs by default.
# You might comment this lines once you have the MIBs downloaded.
#export MIBS=

# snmpd control (yes means start daemon).
SNMPDRUN=yes

# snmpd options (use syslog, close stdin/out/err).
SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -g snmp -I -smux,mteTrigger,mteTriggerConf
    -p /var/run/snmpd.pid'

# snmptrapd control (yes means start daemon).  As of net-snmp version
# 5.0, master agentx support must be enabled in snmpd before snmptrapd
# can be run.  See snmpd.conf(5) for how to do this.
TRAPDRUN=yes

# snmptrapd options (use syslog).
```

```
TRAPDOPTS='-n -Lsd -p /var/run/snmptrapd.pid'
```

Aunque la descarga de mibs de cisco no funciona correctamente, instalaremos el paquete `snmp-mibs-downloader` para nos servirá para descargar algunas mibs adicionales y crear una estructura de directorios para nuestras mibs.

```
root@cefca-trm-001:~# apt-get install snmp-mibs-downloader
```

Ahora descargaremos las mibs de cisco bajándonos el paquete completo con las MIBs que Cisco pone a su disposición en la web, lo desempaquetaremos y moveremos todos los archivos a `/var/lib/mibs/cisco`. Posteriormente crearemos un enlace simbólico en `/usr/share/mibs`.

```
root@cefca-trm-001:~# mkdir /var/lib/mibs/cisco
root@cefca-trm-001:~# mkdir mibs
root@cefca-trm-001:~# cd mibs
root@cefca-trm-001:~/mibs# wget ftp://ftp.cisco.com/pub/mibs/v2/v2.tar.gz
root@cefca-trm-001:~/mibs# tar zxf v2.tar.gz
root@cefca-trm-001:~/mibs# mv auto/mibs/v2/* /var/lib/mibs/cisco
root@cefca-trm-001:~/mibs# ln -s /var/lib/mibs/cisco/ /usr/share/mibs/cisco
```

Una vez que tenemos las mibs, deberemos indicarlo a todas las herramientas de *net-snmp* en la configuración. Para ello agregaremos el nuevo directorio a la configuración.

```
root@cefca-trm-001:~# cat /etc/snmp/snmp.conf
# As the snmp packages come without MIB files due to license reasons, loading
# of MIBs is disabled by default. If you added the MIBs you can reenable
# loading them by commenting out the following line.
mibs :
mibdirs +/usr/share/mibs/cisco
```

Ahora seleccionaremos las mibs que nos interesan a partir del listado de mibs de nuestro switch que existe en la web de Cisco y las copiaremos a un directorio para poder procesarlas más fácilmente. Lo que haremos será utilizar la utilidad `snmpttconvertmib`, que realizará un fichero de configuración de snmptt a partir de los ficheros con la información de las mibs. En dicho fichero se realizará un mapeo de la trap a la ejecución del comando pasado mediante el parámetro `--exec`. En nuestro caso lo que hacemos será llamar a la interfaz shell script que dispone Nagios, pasándole la dirección ip, el nombre del servicio SNMP_TRAP (el que acabamos de crear) y el código de estado 2 (crítico).

```
root@cefca-trm-001:~# ls mibs/3750 | more
BRIDGE-MIB.my
CISCO-CONFIG-COPY-MIB.my
CISCO-CONFIG-MAN-MIB.my
CISCO-ENVMON-MIB.my
CISCO-ERR-DISABLE-MIB.my
CISCO-PORT-SECURITY-MIB.my
CISCO-PROCESS-MIB.my
```

```
CISCO-STP-EXTENSIONS-MIB.my
SNMPv2-MIB.my
root@cefca-trm-001:~# for mib in ~/mibs/3750/*; \
 do snmpttconvertmib --in=$mib --out=/etc/snmp/snmptt-cisco.conf --debug \
  --exec='/usr/share/nagios3/plugins/eventhandlers/submit_check_result $r
    SNMP_TRAP 2' ; done
```

Realizaremos unos ajustes adicionales a la configuración de `snmptt` para que funcione en modo demonio y para que carge el módulo snmp-perl que permitirá la correcta resolución del espacio de nombres de las MIBs de nombres a enteros y viceversa. Además agregaremos el fichero de configuración recién creado.

```
root@cefca-trm-001:~# cat /etc/snmp/snmptt.ini
#
# SNMPTT v1.4 Configuration File
#
# Linux / Unix
#
        ....
mode = daemon
        ....
net_snmp_perl_enable = 1
        ....
snmptt_conf_files = <<END
/etc/snmp/snmptt.conf
/etc/snmp/snmptt-cisco.conf
END
```

Hemos configurado la utilidad snmptt, pero no hemos realizado el enlace con snmptrapd. Para esto, lo que haremos será en primer lugar configurar snmptrad para que acepte las traps con el código de comunidad que hemos especificado y para que logee y ejecute. Además agregaremos que el manejador por defecto de las trap será el comando snmptthandler que se comunicará con el demonio snmptt.

```
root@cefca-trm-001:~# cat /etc/snmp/snmptrapd.conf
authCommunity log,execute CEFC@_TRAP
traphandle default /usr/sbin/snmptthandler
```

5.7.3.2. Monitorización de la matriz de discos HP MSA

Para la monitorización de la matriz de discos, utilizaremos el plugin existente en la red `check_msa_hadware.pl` [5] que realizará un chequeo vía SNMP de los OID relevantes a las alarmas y devolverá su estado. El problema de este plugin es que no permite realizar chequeos individuales, sino que realizará un chequeo a todos los componentes y devolverá error si alguno de ellos presenta un problema. Podría ser interesante en un futuro

[5] https://github.com/david-barbion/monitoring/tree/master/check_msa_hardware

refactorizar este componente de forma que permitiese realizar chequeos individuales por sensor e incluso sacar información de rendimiento para poder graficar con `PNP4Nagios`. Sin embargo, esto se dejará para un funturo.

Tras su descarga lo dejaremos en el directorio de plugins `/usr/local/lib/nagios-plugins`, le daremos permisos de ejecución y definiremos el comando en Nagios. En dicho comando haremos uso, además de la dirección del host para el que se invoque el comando, de una macro residente en la definición del host que se llame `_MSA_SNMP_community` que contenga la clave de comunidad SNMP.

```
root@cefca-trm-001:~# cat /etc/nagios-plugins/config/hp-msa.cfg
define command {
        command_name nagios_msa_status
        command_line /usr/local/lib/nagios-plugins/check_msa_hardware.pl -H
    $HOSTADDRESS$ -C $_HOSTMSA_SNMP_COMMUNITY$ -v 2
        }
```

Definiremos la plantilla genérica, como hicimos para el switch, pero esta vez para los productos de almacenamiento HP.

```
define host {
        use                 generic-host
        name                hp-storage
        check_interval      5
        max_check_attempts  10
        icon_image          cefca/storage.png
        icon_image_alt      HP MSA Storage
        statusmap_image     cefca/storage.gd2
        register            0
        }
```

Ahora definiremos el host con la macro anteriormente citada.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hosts/hpc-cefca/cefca-md-001.cfg
define host {
        use                  hp-storage
        host_name            cefca-md-001
        alias                Matriz discos HPC
        address              10.10.10.81
        _MSA_SNMP_community  CEFC@_RO
        parents              cefca-swt-013
        }
```

No asociaremos los servicios directamente al host, sino que los asociaremos a un hostgroup dedicado. Para ello definiremos el hostgroup `hp-msa` y cuyo único miembro será nuestra matriz de almacenamiento.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hostgroups/hp-msa.cfg
define hostgroup {
        hostgroup_name hp-msa
```

```
        alias HP MSA Storage
        members cefca-md-001
        }
```

Finalmente definiremos el servicio que controlará el estado de la MSA y que asociaremos al hostgroup creado previamente.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/services/health/hp-msa.cfg
define service {
        use                     health-service
        service_description     MSA Status
        check_command           nagios_msa_status
        hostgroup_name          hp-msa
        }
```

5.7.3.3. Monitorización de servidores HP Proliant

Para la monitorización de los servidores Proliant, como vimos, instalaremos el paquete hp-msa en todos los servidores HP. En nuestro sistema de monitorización, instalaremos el plugin para Nagios proporcionado por la misma HP mediante su paquete Debian [6]. Para ello:

```
root@cefca-trm-001:~# apt-get install snmpd nmap
root@cefca-trm-001:~# dpkg -i nagios-plugins-hpilo_1.4.0-9_amd64.deb
```

El plugin lleva un sistema de autodescubrimiento y de automatización de la generación de la configuración. Sin embargo dicho sistema es bastante poco flexible y tiene algunos bugs. Por ello crearemos una configuración manual de la configuración y, como veremos más adelante, modificaremos algunos de los scripts que provee.

Monitorización activa del estado En primer lugar crearemos el comando Nagios con el que realizaremos los chequeos. En dicho comando haremos uso de dos macros que deberán ser definidas a nivel de host. La primera será la macro _iLO_IP que contendrá la dirección IP de la iLO y la segunda será la macro _iLO_SNMP_community que contendrá la clave de comunidad con la que se consultará a la iLO. Además recibirá un parámetro que será el que determine el tipo de chequeo a realizar.

```
root@cefca-trm-001:~# cat /etc/nagios-plugins/config/hp-ilo.cfg
define command {
        command_name nagios_hpilo_engine
        command_line /usr/lib/nagios/plugins/nagios_hpilo_engine -H $_HOSTILO_IP$
     -C  $_HOSTILO_SNMP_COMMUNITY$ -o $ARG1$
        }
```

[6] https://github.com/hposp/nagios-plugins-hpilo/tree/nagios-plugins-hpilo-1.40

Al igual que hicimos con los otros dispositivos, definimos una plantilla llamada `linux-server`.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hosts/linux-server.cfg
define host {
        use                     generic-host
        name                    linux-server
        check_interval          5
        max_check_attempts      2
        icon_image              cefca/rack_linux.png
        icon_image_alt          Linux server
        statusmap_image         cefca/rack_linux.gd2
        register 0
        }
```

Creamos los host de nuestros servidores. Definiremos el servidor hpc-master y a todos los hpc-node heredando la plantilla y con las macros anteriormente indicadas.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hosts/hpc-cefca/hpc-node1.cfg
define host {
        use                     linux-server
        host_name               hpc-node1
        address                 10.50.85.10
        alias                   Nodo de cómputo 1
        _iLO_IP                 10.10.10.83
        _iLO_SNMP_community     CEFC@_RO
        _SGE_MASTER             10.50.85.1
        parents                 cefca-swt-013
        }
```

Para asociar los servicios, crearemos un hostgroup llamado `hp-ilo` cuyos miembros serán todos los servidores HP.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hostgroups/hp-ilo.cfg
define hostgroup {
        hostgroup_name  hp-ilo
        alias           Servidores HP Proliant
        members         hpc-master,hpc-node*
        }
```

Sólo queda crear los servicios, asociándolos con el hostgroup definido anteriormente.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/services/health/hp-ilo.cfg
define service {
        use                     health-service
        service_description     System Status
        check_command           nagios_hpilo_engine!1
        hostgroup_name          hp-ilo
        }

define service {
        use                     health-service
        service_description     Fans
```

```
        check_command         nagios_hpilo_engine!2
        hostgroup_name        hp-ilo
        }

define service {
        use                   health-service
        service_description   Memory
        check_command         nagios_hpilo_engine!3
        hostgroup_name        hp-ilo
        }

define service {
        use                   health-service
        service_description   Network
        check_command         nagios_hpilo_engine!4
        hostgroup_name        hp-ilo
        }

define service {
        use                   health-service
        service_description   Power Supplies
        check_command         nagios_hpilo_engine!5
        hostgroup_name        hp-ilo
        }

define service {
        use                   health-service
        service_description   Processors
        check_command         nagios_hpilo_engine!6
        hostgroup_name        hp-ilo
        }

define service {
        use                   health-service
        service_description   Storage
        check_command         nagios_hpilo_engine!7
        hostgroup_name        hp-ilo
        }

define service {
        use                   health-service
        service_description   Temperatures
        check_command         nagios_hpilo_engine!8
        hostgroup_name        hp-ilo
        }
```

Monitorización pasiva Como comenté al principio de este apartado, para la monitorización pasiva he tenido que realizar algunas modificaciones en el script que viene con el paquete y que es el encargado de gestionar los traps emitidos por los servidores HP. Para ello he cogido el script `/usr/lib/nagios/plugins/nagios_hpilo_traps` y he creado un nuevo

script `/usr/local/lib/nagios-plugins/nagios_hpilo_manage_traps` que puede verse en los fuentes de este PFC.

Este nuevo script soluciona algunos problemas relacionados con la resolución del origen del trap y el mapeo a la dirección de servicio del servidor que lo origina. Para realizar el mapeo entre iLO y host se recurre a un fichero hp_ilo.map de que contiene el mapeo y las claves de comunidad de cada servidor (es necesario porque una vez realizado el trap, el manejador realiza una consulta SNMP para comprobar el OID oportuno antes de elevar el trap hacia la interfaz con Nagios). Por lo tanto, será necesario definir el fichero de mapeo.

```
root@cefca-trm-001:~# cat /etc/nagios3/hp_ilo.map
# nodo;ip ilo;ip cluster;comunidad_snmp
hpc-master;10.10.10.80;10.50.85.1;CEFC@_RO
hpc-node1;10.10.10.83;10.50.85.10;CEFC@_RO
hpc-node2;10.10.10.84;10.50.85.11;CEFC@_RO
hpc-node3;10.10.10.85;10.50.85.12;CEFC@_RO
hpc-node4;10.10.10.86;10.50.85.13;CEFC@_RO
hpc-node5;10.10.10.87;10.50.85.14;CEFC@_RO
```

El tratamiento del trap no se realizará vía el demonio snmptt, sino que lo gestionaremos directamente vía el demonio snmptrapd. Para ello todos los traps con un OID perteneciente a las subramas de las alertas, serán despachados empleando el manejador que hemos definido. Esto será necesario configurarlo en snmptrand.

```
root@cefca-trm-001:~# cat /etc/snmp/snmptrapd.conf
authCommunity log,execute CEFC@_TRAP
traphandle 1.3.6.1.4.1.232.0.*         /usr/local/lib/nagios-plugins/
    nagios_hpilo_manage_traps
traphandle default /usr/sbin/snmptthandler
```

5.7.3.4. Monitorización de los servicios del Cluster HPC

Una vez que tenemos monitorizados todos los dispositivos, pasaremos a la monitorización del estado de los servicios del cluster. Como indicamos, uno de los objetivos es que el sistema de monitorización impacte lo menos posible en los sistemas de cómputo.

Todos los servicios que definiremos a continuación heredarán de la plantilla network service, que definiremos.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/services/network.cfg
define service {
        name            network-service
        use             generic-service
        check_interval  5
        register        0
        }
```

Todos los nodos tienen el servicio SSH instalado, sin embargo únicamente monitorizaremos este servicio en los nodos hpc-master y hpc-login. Para ello definiremos el hostgroup ssh-server en el que incluiremos ambos nodos.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hostgroups/ssh-server.cfg
define hostgroup {
        hostgroup_name  ssh-server
        alias           Servidores SSH
        members         hpc-master,hpc-login
        }
```

Después simplemente definiremos el servicio con el comando del plugin integrado en Nagios y la relación con el hostgroup anteriormente creado.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/services/network/ssh.cfg
define service{
        use                     network-service
        service_description     SSH
        check_command           check_ssh
        hostgroup_name          ssh-server
}
```

Para el chequeo del estado del servidor en la cola en el servidor monitorizado, se recurrirá a la monitorización indirecta (puede verse en la figura 5.69 cómo funciona este tipo de monitorización). En primer lugar será necesario especificar el chequeo del servicio.

```
root@cefca-trm-001:~# cat /etc/nagios-plugins/config/check_nrpe_sge.cfg
define command {
        command_name    check_nrpe_sge
        command_line    /usr/lib/nagios/plugins/check_nrpe -H $_HOSTSGE_MASTER$ -
    c check_sge_node_$HOSTNAME$
        }
```

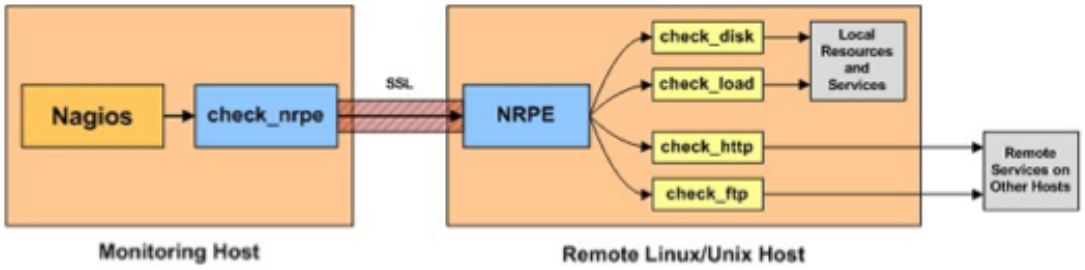


FIGURA 5.69: Chequeos activos indirectos mediante NRPE

Definiremos el host hpc-master junto con los servicios de monitorización directa que chequearán la disponibilidad de discos vía NRPE.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hosts/hpc-cefca/hpc-master.cfg
define host {
        use                     linux-server
        host_name               hpc-master
```

```
        address               10.50.85.1
        _iLO_IP               10.10.10.80
        _iLO_SNMP_community   CEFC@_RO
        parents               cefca-swt-013
        }

define service {
        use                   health-service
        host_name             hpc-master
        service_description   OS disk space
        check_command         check_nrpe_1arg!check_sda1
        }

define service {
        use                   network-service
        host_name             hpc-master
        service_description   Storage scratch disk space
        check_command         check_nrpe_1arg!check_storage_scratch
        }

define service {
        use                   network-service
        host_name             hpc-master
        service_description   Storage users disk space
        check_command         check_nrpe_1arg!check_storage_users
        }
```

Y definiremos el nodo de login.

```
root@cefca-trm-001:~# cat /etc/nagios3/conf.d/hosts/hpc-cefca/hpc-login.cfg
define host {
        use          linux-server
        host_name    hpc-login
        alias        Nodo de login del HPC
        address      10.50.85.2
        _SGE_MASTER  10.50.85.1
        parents      hpc-master
        }
```

Ya tenemos la configuración en el sistema de monitorización, pero es necesario realizar la configuración NRPE del sistema monitorizado. Para ello.

```
root@hpc-master:~# cat /etc/nagios/nrpe_local.cfg
########################################
# Do any local nrpe configuration here
########################################

allowed_hosts=127.0.0.1,10.50.84.100

dont_blame_nrpe=0

command[check_sda1]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -p /dev/sda1
```

Agregaremos los almacenamientos compartidos en un fichero aparte.

```
root@hpc-master:~# cat /etc/nagios/nrpe.d/storage.cfg
command[check_storage_scratch]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -
    p /dev/mapper/vg_scratch-lv0
command[check_storage_users]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -p
    /dev/mapper/vg_users-lv0
```

Y para finalizar, la monitorización del estado de los nodos en la cola.

```
root@hpc-master:~# cat /etc/nagios/nrpe.d/sge_nodes.cfg
command[check_sge_node_hpc-node1]=/usr/local/lib/nagios-plugins/check_sge.py -H
    hpc-node1
command[check_sge_node_hpc-node2]=/usr/local/lib/nagios-plugins/check_sge.py -H
    hpc-node2
command[check_sge_node_hpc-node3]=/usr/local/lib/nagios-plugins/check_sge.py -H
    hpc-node3
command[check_sge_node_hpc-node4]=/usr/local/lib/nagios-plugins/check_sge.py -H
    hpc-node4
command[check_sge_node_hpc-node5]=/usr/local/lib/nagios-plugins/check_sge.py -H
    hpc-node5
```

5.7.3.5. Vista de nuestro sistema de monitorización

Tras el reinicio de nagios tendremos nuestro sistema disponible como pueden verse en las figuras 5.70 y 5.71.

5.7.4. Monitorización del uso de los recursos del clúster con Ganglia

Hasta ahora hemos visto la integración con los sistemas de monitorización existentes en CEFCA. Sin embargo estos sistemas de monitorización carecen de la funcionalidad requerida, arquitectura escalable y bajo consumo de recursos para la monitorización del estado del clúster y el uso de sus recursos en tiempo real. Para esta labor usaremos Ganglia, la herramienta más óptima para este tipo de entornos.

Ganglia tiene una arquitectura de monitores de muy bajo consumo de recursos que recopilan información y van enviando dicha información por multicast o unicast. El componente monitor se llama *gmond*. Algunos de los monitores se encargarán a su vez de recopilar la información y realizar agregados de información que posteriormente serán pooleados por un demonio *gmetad* y que irá archivando en bases de datos RRD. Para finalizar, el componente *webinterface* se encargará de realizar gráficas a partir de dichas bases de datos.

En la figura 5.72 puede verse cómo realizaremos nuestra configuración. Desplegaremos monitores en cada uno de los nodos del clúster y mediante una configuración unicast,

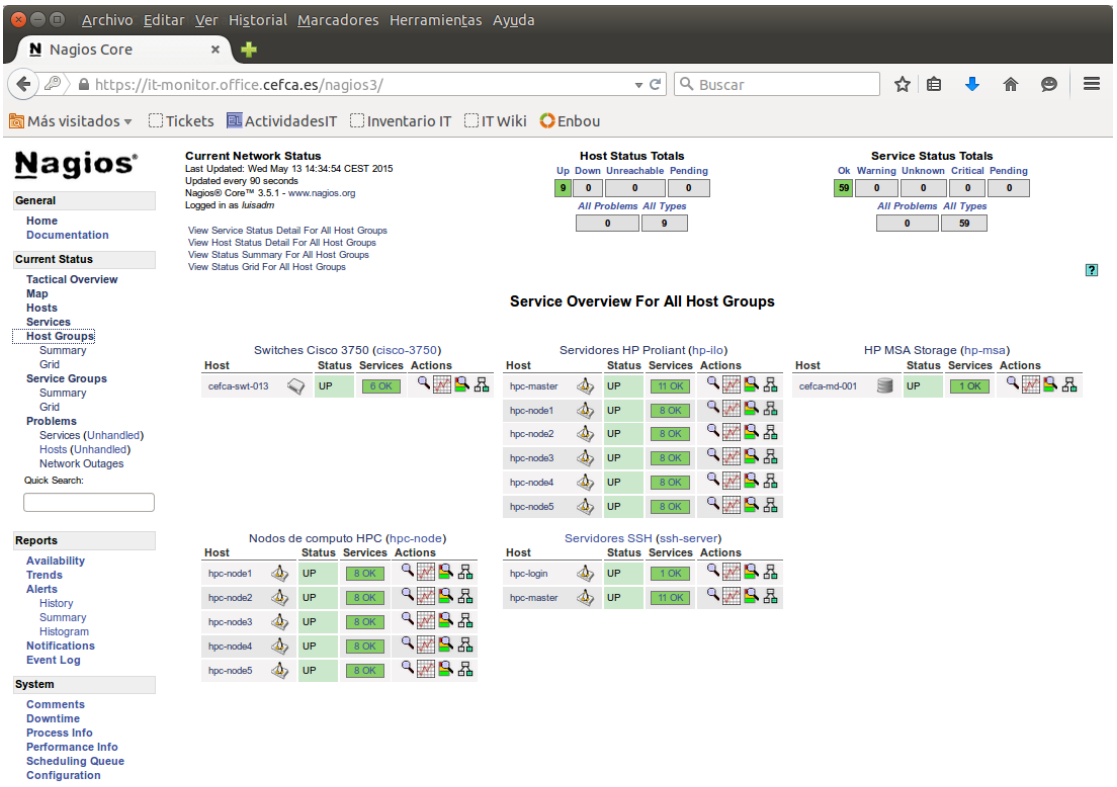


FIGURA 5.70: Vista de servicios monitorizados en Nagios

enviaremos la información al monitor existente en el host *hpc-master*. Posteriormente, desplegaremos el demonio gmetad y la interfaz webinterface en un servidor que se encuentra en la red de infraestructura llamado *cluster-info*.

5.7.4.1. Instalación de los monitores en los nodos de cómputo

Simplemente instalaremos el paquete y configuraremos el nombre del clúster y la configuración unicast.

```
root@hpc-node1:~# apt-get install ganglia-monitor
root@hpc-node1:~# cat /etc/ganglia/gmond.conf
        ....
cluster {
  name = "HPC-CEFCA"
  owner = "unspecified"
  latlong = "unspecified"
  url = "unspecified"
}
        ....
udp_send_channel {
  host = 10.50.85.1
  port = 8649
  ttl = 1
```

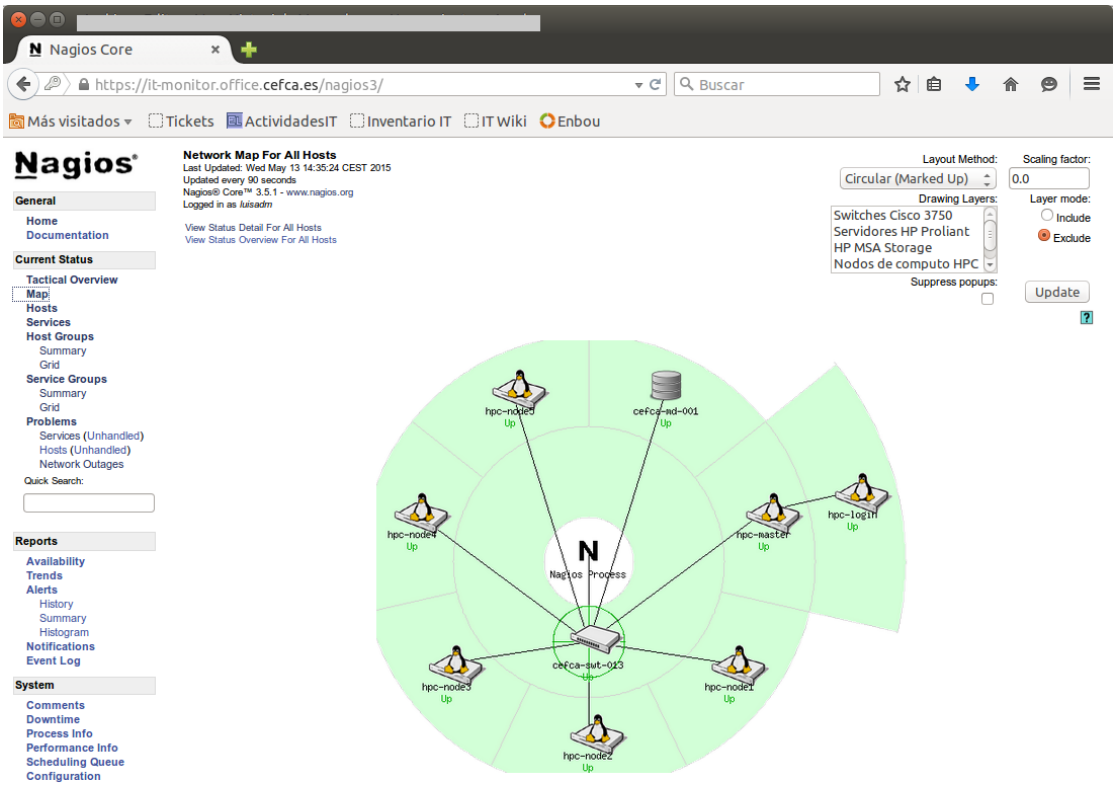


FIGURA 5.71: Mapa de hosts en Nagios

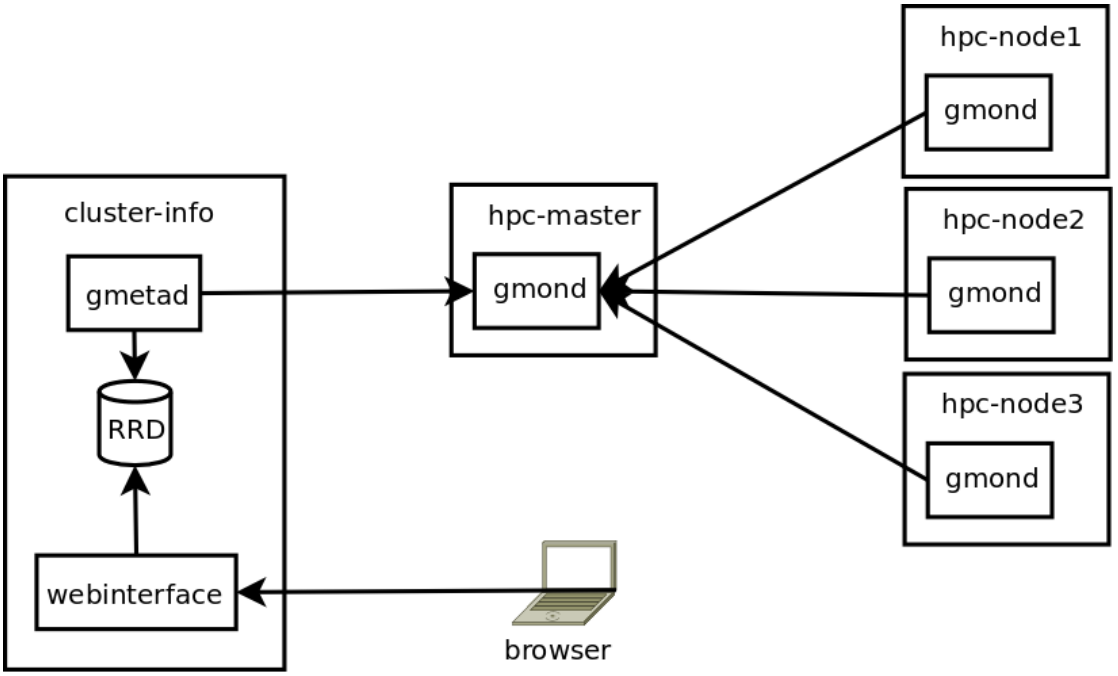


FIGURA 5.72: Diagrama de despliegue de Ganglia

```
}
        ....
```

5.7.4.2. Instalación del monitor en el nodo cabecera

En este caso procederemos del mismo modo, pero además pondremos habilitaremos el socket en el que recibirá los mensajes de los nodos y el socket en el cual recibirá el pooling del componente gmetad.

```
root@hpc-master:~# apt-get install ganglia-monitor
root@hpc-master:~# cat /etc/ganglia/gmond.conf
    ....
cluster {
  name = "HPC-CEFCA"
  owner = "unspecified"
  latlong = "unspecified"
  url = "unspecified"
}
        ....
        udp_send_channel {
          host = 10.50.85.1
          port = 8649
          ttl = 1
        }
        ....
        udp_recv_channel {
          port = 8649
        }
                ....
        tcp_accept_channel {
          port = 8648
        }
                ....
```

Además instalaremos un pequeño script he que desarrollado a partir de [7] y que usará los sensores de ganglia para publicar el estado de los trabajos en ejecución y de los slots consumidos en cada cola.

```
root@hpc-master:~# cat /usr/local/bin/sge-ganglia.sh
#!/bin/bash

publish_jobs_stats() {
        qstat -u '*' | tail -n +3 | awk '
                BEGIN { pending=running=error=0; }
                ($5 ~ /^qw/)   { pending++; }
                ($5 ~ /[rRt]/) { running++; }
                ($5 ~ /E/ )    { error++;   }
        END {
```

[7] https://github.com/ganglia/gmetric/tree/master/hpc/sge_jobs

```
                                cmd="/usr/bin/gmetric --name sge_pending --value
    "pending" --type uint16";
                                system(cmd);
                                cmd="/usr/bin/gmetric --name sge_running --value
    "running" --type uint16";
                                system(cmd);
                                cmd="/usr/bin/gmetric --name sge_error   --value
    "error" --type uint16";
                                system(cmd);
                }'

}


publish_queue_stats() {
        queuename=$1

        qstat -g c -q ${queuename}.q | tail -n +3 | awk -v queuename=$queuename '
                BEGIN { used=res=avail=total=0; }
        { used=$3; }
        { res=$4; }
        { avail=$5; }
        { total=$6; }
        END {
                cmd="/usr/bin/gmetric --name sge_queue_"queuename"_used --value "
    used" --type uint16";
                system(cmd);
                cmd="/usr/bin/gmetric --name sge_queue_"queuename"_res --value "
    res" --type uint16";
                system(cmd);
                cmd="/usr/bin/gmetric --name sge_queue_"queuename"_avail --value
    "avail" --type uint16";
                system(cmd);
                cmd="/usr/bin/gmetric --name sge_queue_"queuename"_total --value
    "total" --type uint16";
                system(cmd);
                }'

}


publish_jobs_stats
publish_queue_stats main
publish_queue_stats interactive
```

Y crearemos un fichero cron para que se ejecute a cada minuto:

```
root@hpc-master:~# cat /etc/cron.d/sge-ganglia
* * * * * root /usr/local/bin/sge-ganglia.sh &>/dev/null
```

5.7.4.3. Instalación del componente gmetad

Como vimos, el componente gmetad será desplegado en el servidor cluster-info. Para ello simplemente instalaremos y configuraremos definiendo el nombre del clúster y el origen de la información. Además alteraremos la configuración por defecto y agregaremos el archivado de los valores MAX y MIN.

```
root@cluster-info:~# apt-get install gmetad
root@cluster-info:~# cat /etc/ganglia/gmetad.conf
        ....
data_source "HPC-CEFCA" 30 10.50.85.1:8648
        ....
RRAs "RRA:AVERAGE:0.5:1:5856" "RRA:MAX:0.5:1:5856" "RRA:MIN:0.5:1:5856" \
        "RRA:AVERAGE:0.5:4:20160" "RRA:MAX:0.5:4:20160" "RRA:MIN:0.5:4:20160"  \
        "RRA:AVERAGE:0.5:40:52704" "RRA:MAX:0.5:40:52704" "RRA:MIN:0.5:40:52704"
        ....
```

5.7.4.4. Instalación del componente ganglia-webfrontend

Para finalizar, instalaremos el frontend web en el servidor cluster-info. Es tan sencillo como instalar el paquete y crear los enlaces adecuados en la configuración de apache para activarlo.

```
root@cluster-info:~# apt-get install ganglia-webfrontend
root@cluster-info:~# cd /etc/apache2/conf-available
root@cluster-info:/etc/apache2/conf-available# ln -s ../../ganglia-webfrontend/
    apache.conf ganglia.conf
root@cluster-info:/etc/apache2/conf-available# cd ../conf-enabled/
root@cluster-info:/etc/apache2/conf-enabled# ln -s ../conf-available/ganglia.conf
     .
root@cluster-info:~# service apache2 restart
```

Ya deberíamos tenerlo funcionando, pero nos falta agregar los reportes sobre gridengine, para ello copiaremos los scripts que se encuentra en el código fuente de la memoria en.

```
/usr/share/ganglia-webfrontend/graph.d/jobqueue_report.php
/usr/share/ganglia-webfrontend/graph.d/queue_main_report.php
/usr/share/ganglia-webfrontend/graph.d/queue_interactive_report.php
```

Y los activaremos en la interfaz.

```
root@cluster-info:~# cat /var/lib/ganglia-web/conf/default.json
{
        "included_reports": ["load_report","mem_report","cpu_report","
    network_report",
"jobqueue_report","queue_main_report","queue_interactive_report"]
}
```

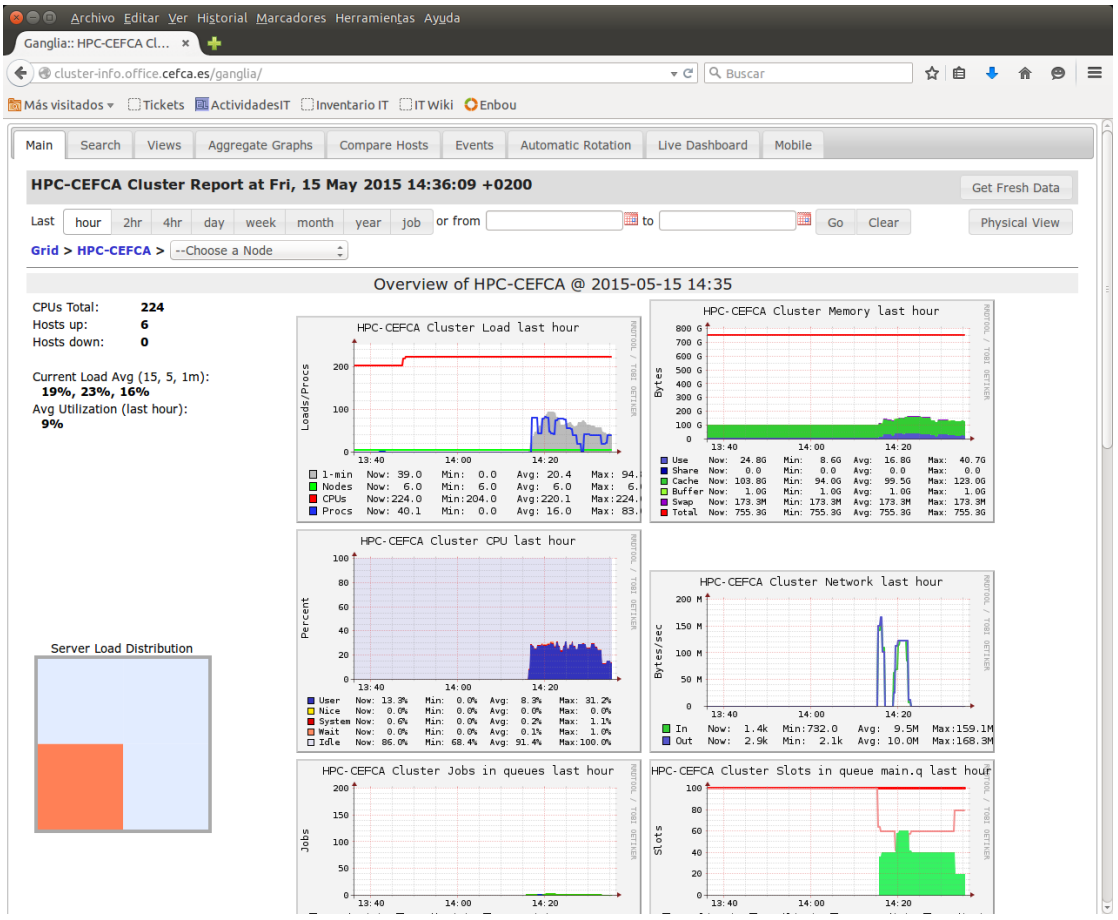


FIGURA 5.73: Vista global del clúster con Ganglia

5.7.5. Información de estado del clúster en el mensaje del día en el nodo de login

Aunque no forma parte del sistema de monitorización en si mismo, vamos a incluir algo de información útil en el MOTD (mensaje del día) que se le da al usuario cuando inicia sesión en el nodo de login. Para ello crearemos un sencillo script que se ejecutará en cada inicio de sesión.

```
lguillen@hpc-login:~$ cat /etc/update-motd.d/10-queues-status
#!/bin/bash

show_jobs_stats() {
        qstat -u '*' | tail -n +3 | awk '
                BEGIN { pending=running=error=0; }
                ($5 ~ /^qw/)    { pending++; }
                ($5 ~ /[rRt]/) { running++; }
                ($5 ~ /E/ )     { error++;   }
        END {
                print running, pending, error;
                }'

}
```

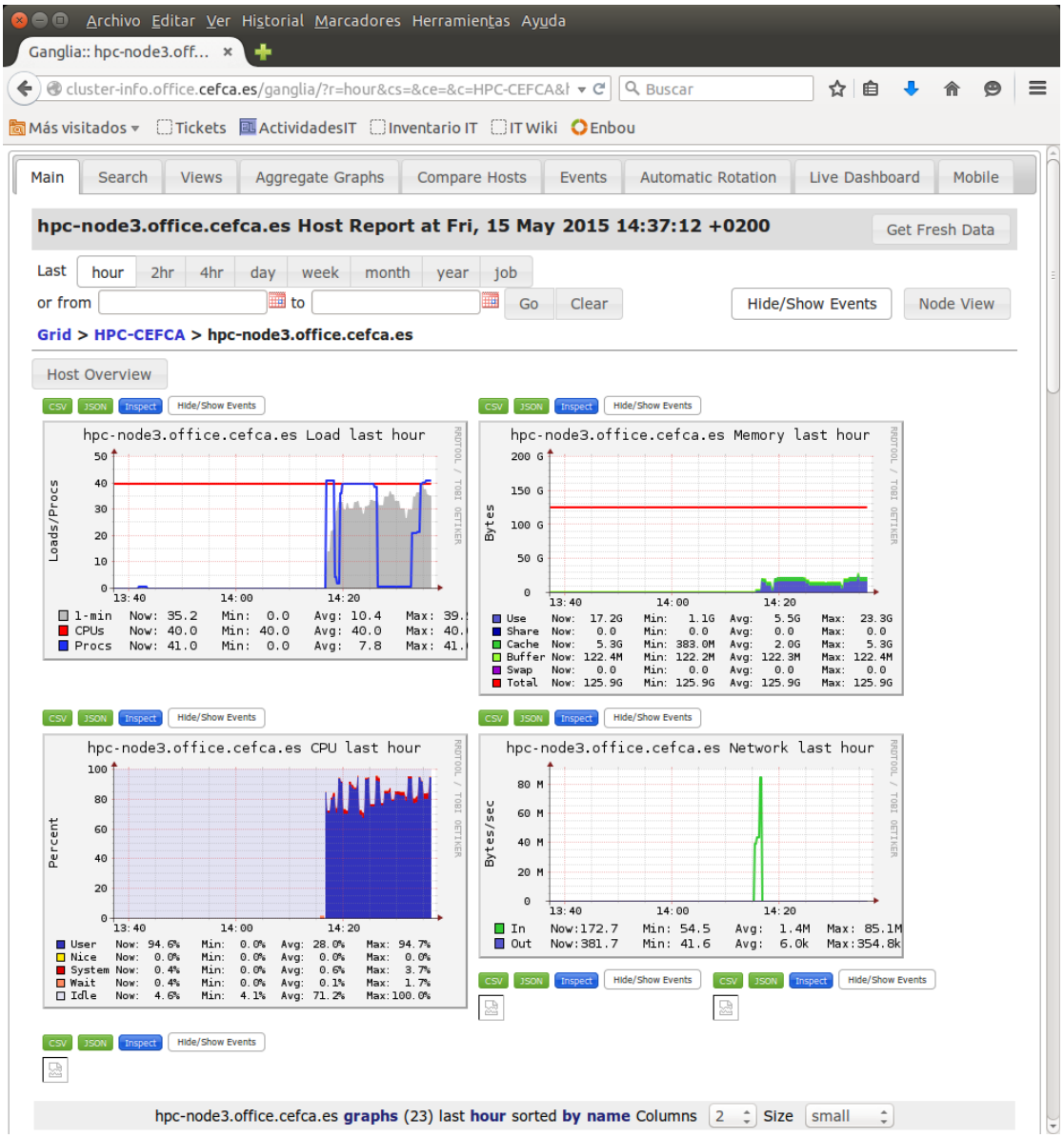


FIGURA 5.74: Vista de un nodo con Ganglia

```
show_queue_stats() {
        queuename=$1

        qstat -g c -q ${queuename}.q | tail -n +3 | awk -v queuename=$queuename '
                BEGIN { used=res=avail=total=0; }
        { used=$3; }
        { res=$4; }
        { avail=$5; }
        { total=$6; }
        END {
                print used, res, avail, total;
                }'

}
```

```
jobs_stats=(‘show_jobs_stats‘)
mainq_stats=(‘show_queue_stats main‘)
interactiveq_stats=(‘show_queue_stats interactive‘)

echo    "Trabajos            Slots|     main.q       |   interactive.q  |"
echo    "--------------      ----------------------------------------|"
printf "Ejecución: %3d           | Usados:     %3d  |   Usados:     %3d|\n"  ${
    jobs_stats[0]} ${mainq_stats[0]} ${interactiveq_stats[0]}
printf "Pendientes:%3d           | Reservados: %3d  |   Reservados: %3d|\n"  ${
    jobs_stats[1]} ${mainq_stats[1]} ${interactiveq_stats[1]}
printf "Error:     %3d           | Disponibles:%3d  |   Disponibles:%3d|\n"  ${
    jobs_stats[2]} ${mainq_stats[2]} ${interactiveq_stats[2]}
printf "                         | Total:      %3d  |   Total:      %3d|\n"  ${
    mainq_stats[3]} ${interactiveq_stats[3]}
echo    "================================================================"
```

Para que esto funcione correctamente, debido a un bug, es necesario realizar un pequeño parche en el fichero /etc/pam.d/sshd. Basta con poner editar las siguientes líneas.

```
lguillen@hpc-login:~$ cat /etc/pam.d/sshd
        ....
# Print the message of the day upon successful login.
# This includes a dynamically generated part from /run/motd.dynamic
# and a static (admin-editable) part from /etc/motd.
session    optional     pam_motd.so  motd=/run/motd.dynamic
session    optional     pam_motd.so  noupdate
        ....
```

Con esta funcionalidad se le presentará al usuario un login como el que se muestra en la figura 5.75 y en el que se le informa acerca de los trabajos y colas del sistema.

5.8. Despliegue automatizado

A continuación trataremos la automatización completa del despliegue de los nodos de cómputo. Esto nos permitirá desplegar nuevas configuraciones y agregar nuevos nodos al clúster en cuestión de minutos.

5.8.1. Implementación del despliegue de la configuración

Como comenté, la parte de la configuración se desarrollará usando LGSetup. Aunque existen paquetes de configuración para todos los tipos de nodo, en este apartado por simplificar, únicamente veremos la parte de creación del paquete `deploy_hpc-exec-node`, que

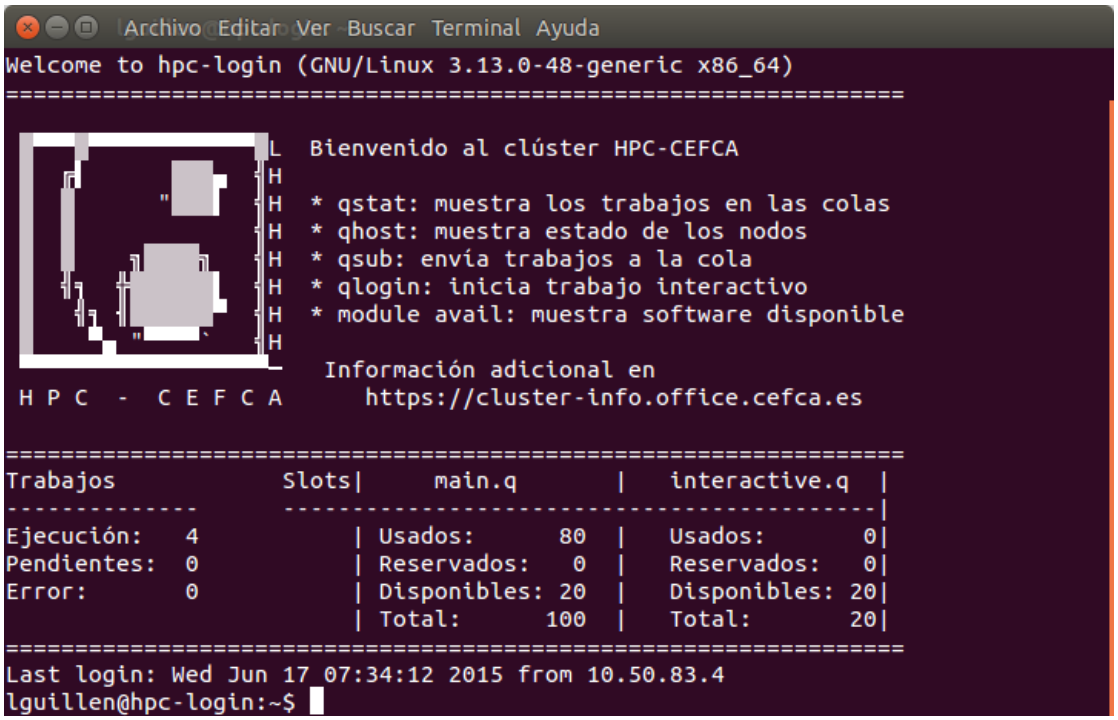


FIGURA 5.75: Mensaje del día del nodo de login del Clúster

es el encargado de realizar la configuración de los nodos de cómputo. describiendo las distintas etapas necesarias y sin entrar en profundidad en su funcionamiento interno. Quien quiera ver los detalles, tiene a su disposición el código fuente adjunto en el proyecto.

5.8.1.1. Creación de la estructura básica del paquete

En primer lugar crearemos la estructura básica del paquete, para ello crearemos el directorio del paquete y el contenedor files

```
$ mkdir deploy_hpc-exec-node
$ mkdir deploy_hpc-exec-node/files
```

En primer lugar crearemos el fichero de información, indicando el nombre del paquete, descripción y versión.

```
$ cat deploy_hpc-exec-node/info
package_name=deploy_hpc-exec-node
package_desc="Despliegue de nodo de ejecucion hpc"
package_ver="0.1"
```

Luego el fichero con los componentes que usará. En nuestro caso hará uso del componente common que contiene multitud de helpers que ayudan en la configuración básica del servidor, el componente gridengine que lleva helpers específicos para la instalación y

configuración de gridengine y el componente ganglia que tiene los suyos con respecto a este sistema de monitorización.

```
$ cat deploy_hpc-exec-node/components
common
gridengine
ganglia
```

Por último, definiremos el escenario que empleamos. En nuestro caso usaremos el escenario *office*. El escenario contiene la configuración genérica común a un determinado escenario como son los servidores dns, nombre de dominio, certificados CA, etc..

```
$cat deploy_hpc-exec-node/scenarios
office
```

Con esto ya podríamos compilar el paquete, pero no haría nada ya que no hemos definido todavía las etapas de configuración.

5.8.1.2. Definición de las etapas

En la definición de nuestro paquete de configuración definiremos cuatro etapas:

- **Etapa 1**: Configuración básica del nodo.
- **Etapa 2**: Configuración del almacenamiento local y NFS
- **Etapa 3**: Despliegue de los componentes del gestor de recursos
- **Etapa 4**: Instalación de software científico local del nodo y configuración de modules

Cada etapa deberá ser definida con un fichero *stageX.run* en el que se indicará la secuencia de helpers de configuración que se ejecutará y un fichero *stageX.cfg* donde se indicarán las variables de configuración que usarán los helpers de la etapa.

5.8.1.3. Implementación de la etapa 1: Configuración básica del nodo

Como su nombre indica, en esta tapa realizaremos la configuración básica. Entenderemos como configuración básica la integración del sistema con la infraestructura: configuración de red, gestor de paquetes, unión de la máquina al dominio, configuración syslog, monitorización, etc.

En nuestro caso tenemos que tener en cuenta que el nodo en el momento de la configuración ya tiene red (la obtuvo por dhcp), por lo que lo primero que haremos será la configuración del gestor de paquetes.

La secuencia de helpers se definirá en el fichero `deploy\_hpc-exec-node/files/etc/stage1.run` y será la siguiente:

- **config_apt_proxy.sh**: configurará el proxy apt de cefca.
- **config_apt_software_repo.sh**: agrega el repositorio apt de ubuntucefca14 y agrega la key.
- **config_apt_pinning_release.sh**: prioriza los paquetes existentes en ubuntucefca14 sobre los de la distribución general.
- **update_packages.sh**: actualiza los orígenes de los repositorios.
- **create_admins.sh**: crea los usuarios pasados y los agrega al grupo sudo para ser administradores locales.
- **enable_bonding.sh**: instala los paquetes y carga los módulos necesarios para uso de bonding.
- **config_net_ifaces.sh**: realiza la configuración de las interfaces de red y la recarga.
- **config_hostname.sh**: configura correctamente el hostname en el sistema con la dirección ip por defecto de la máquina.
- **config_motd.sh**: instala un fichero de plantilla motd si existe.
- **config_cacert.sh**: instala el certificado de la CA a nivel de sistema.
- **config_ntpdate.sh**: configura el cliente ntpdate para sincronización ntp y lo programa en cron.hourly.
- **config_ssh.sh**: instala el servidor ssh, realiza la configuración tanto del cliente como del servidor y agrega las keys ssh pasadas al usuario root.
- **config_syslog_client.sh**: configura el cliente syslog para que use el servidor pasado.
- **config_kerberos_client_dnslookup.sh**: instala y configura el cliente kerberos usando para la resolución de los KDC el servicio DNS.
- **config_samba4_client_ad.sh**: configura el cliente samba de active directory y agrega la máquina al dominio.

- **config_samba4_service_principals.sh**: agrega nuevos principales kerberos de tipo servicio en el directorio y los agrega al keytab a la máquina.
- **config_sssd_ad.sh**: configura el demonio sssd para la integración nsswitch y pam con active directory y así poder usar los usuarios del dominio.
- **config_nfs_client.sh**: instala y configura la parte cliente de nfs, realizando la configuración adecuada si se usa kerberos.
- **config_lgdeploy_client.sh**: descarga del servidor lgdeploy la parte cliente que permite la ejecución remota. Además registra un nuevo agente cron en el servidor lgdeploy y lo instala en el cron del sistema.
- **install_ganglia_monitor.sh**: instala los componentes de monitorización del software de monitorización ganglia,
- **config_ganglia_monitor_unicast.sh**: configura el monitor ganglia para que envíe la información por unicast a un servidor.

El fichero de configuración para esta etapa será el siguiente:

```
$ cat deploy_hpc-exec-node/files/etc/stage1.cfg
#!/bin/bash

source $HelperConfigDir/global.cfg


ip_iface1=`LANG=C ifconfig em1 2>/dev/null|awk '/inet addr:/ {print $2}'|sed 's/
    addr://'` \
        || __die "Error obteniendo ip"
[ -z "$ip_iface1" ] && __die "No se pudo obtener ip"

last_ip=`echo $ip_iface1 | cut -f4 -d"."`

### config_hostname.cfg
cfg_hostname=`hostname`
cfg_defaultIp="$ip_iface1"

### config_net_ifaces.cfg
cfg_netIface[0]="em1"
cfg_netType[0]="static"
cfg_netAddress[0]="$ip_iface1"
cfg_netNetmask[0]="255.255.255.0"
cfg_netGateway[0]="10.50.85.254"
cfg_netDnsNameservers[0]="$cfg_dnsServers"
cfg_netDnsSearch[0]="$cfg_dnsSearch"

cfg_netIface[1]="em2"
cfg_netType[1]="static"
cfg_netAddress[1]="192.168.14.${last_ip}"
cfg_netNetmask[1]="255.255.255.0"
```

```
cfg_netIface[2]="bond0"
cfg_netType[2]="static"
cfg_netAddress[2]="192.168.13.${last_ip}"
cfg_netNetmask[2]="255.255.255.0"
cfg_netBonding[2]=1
cfg_mtu[2]="9000"

cfg_bondMode[1]="802.3ad"
cfg_bondMiimon[1]="100"
cfg_bondDownDelay[1]="200"
cfg_bondUpDelay[1]="200"
cfg_bondSlaves[1]="em5 em6"

cfg_netIface[3]="em5"
cfg_netType[3]="manual"
cfg_netBondMaster[3]="bond0"

cfg_netIface[4]="em6"
cfg_netType[4]="manual"
cfg_netBondMaster[4]="bond0"


## config_samba4
cfg_sambaAddDomain=$__TRUE
cfg_sambaAddDomainUser="useraddtodomain"
cfg_sambaAddDomainPasswd=$(cat $HelperConfigDir/samba_add_domain.passwd) \
|| __die "Error leyendo samba_add_domain.passwd"

cfg_sambaAddSpn="nfs"

## config_nfs_client
cfg_nfsKerberosEnable=$__FALSE
cfg_nfsBlackListGssKrb5=$__TRUE

### config_ganglia
cfg_gangliaCluster="HPC-CEFCA"
cfg_gangliaSendIp="10.50.85.1"
cfg_gangliaSendPort="8649"


### config_lgdeploy
cfg_lgdeployCronAddUser="cron_admin"
cfg_lgdeployCronAddPasswd=$(cat $HelperConfigDir/lgdeploy_add_cron.passwd) \
                || __die "Error leyendo lgdeploy_add_cron.passwd"
cfg_lgdeployCronSuffix="hourly"
```

Como podemos ver, el fichero incluirá la carga de datos desde el fichero `global.cfg`. Este fichero global.cfg viene distribuido en el escenario e incluye todos los parámetros de configuración comunes del escenario.

Después tenemos la configuración para el helper `config\_net\_ifaces.sh` que realizará una configuración estática del `/etc/network/interfaces`. En esta parte cogemos la ip que nos

suministró el dhcp para dejarla de forma estática y obtener el último octeto para fijar la configuración de red del resto de interfaces.

A continuación tenemos la parte de configuración que permite agregar la máquina al dominio. En este caso pondremos el usuario con el que agregamos las máquinas al nuestro dominio el password se almacenará en un fichero aparte protegido para el usuario root y que se excluirá de los commits del git. Además, aunque deshabilitamos explícitamente el uso de kerberos en el nfs y metemos el módulo en blacklist, agregamos el SPN nfs, por si queremos en un futuro habilitar el montaje en el futuro al servidor NFS Kerberos general de CEFCA.

Para finalizar, se define nombre del clúster en ganglia y el servidor que gestionará los agregados de la monitorización junto con el usuario, password y el tipo de cron que se usará en la integración cron de LGDeploy.

5.8.1.4. Implementación de la etapa 2: Configuración del almacenamiento local y NFS

En esta etapa configuraremos la parte referente al almacenamiento. En este caso, dada la particularidad de la configuración del almacenamiento del clúster, será necesario escribir algunos helpers específicos que no están disponibles en los componentes. Los helpers especializados se dejarán en el directorio `deploy\_hpc-exec-node/files/lib/helpers`.

La secuencia de helpers se definirá en el fichero `deploy\_hpc-exec-node/files/etc/stage2.run` y será la siguiente:

- **config_storage.sh**: realizará la creación del filesystem en la partición definida y creará los puntos de montaje necesarios.
- **config_storage_local.sh**: creará la estructura de montajes `/local`. Este helper es uno de los helpers especializados.
- **config_autofs_hpcstorage.sh**: instalará el autofs y realizará la configuración especializada del almacenamiento compartido en el clúster (los home, software, etc.). Este helper también es un helper especializado.

El fichero de configuración en esta etapa será el siguiente:

```
$ cat deploy_hpc-exec-node/files/etc/stage2.cfg
#!/bin/bash

source $HelperConfigDir/global.cfg
```

```
### config_storage
cfg_storageDevice="/dev/sda2"
cfg_storageFSLabel="sd_local"
cfg_storageFSMountOpt="defaults,noatime"


### config_autofs_hpcstorage
cfg_storageHPCServer="192.168.14.1"
```

En este caso volvemos a cargar la configuración global del escenario y definimos las variables para la configuración del almacenamiento local y la dirección del servidor NFS que usará el automontador.

5.8.1.5. Implementación de la etapa 3: Despliegue de los componentes del gestor de recursos

En esta etapa se realizará la instalación y configuración del software gestor de recursos para el nodo de cómputo.

La secuencia de helpers se definirá en el fichero `deploy\_hpc-exec-node/files/etc/stage3.run` y será la siguiente:

- **install_mta_gridnode.sh**: realizará la configuración del MTA que se instala en cada nodo de cómputo y que usará una configuración smarthost, definiendo como servidor de relay el servidor de email de CEFCA.
- **install_gridengine_exec.sh**: realizará la instalación y configuración del demonio de ejecución y la parte cliente necesaria para la ejecución de trabajos paralelos.

El fichero de configuración en esta etapa será el siguiente:

```
$ cat deploy_hpc-exec-node/files/etc/
#!/bin/bash

source $HelperConfigDir/global.cfg

## mta
cfg_postfixRelayHost="correo.cefca.es"

## gridengine
cfg_gridengineMaster="hpc-master.office.cefca.es"
```

De nuevo definimos la configuración global del escenario y definimos el host que hará relay y el servidor master de nuestro clúster.

5.8.1.6. Implementación de la etapa 4: Instalación de software científico local del nodo y configuración de modules

Esta será la última etapa de ejecución en la configuración. Para esta etapa también se desarrollará un helper especializado.

La secuencia de helpers se definirá en el fichero `deploy\_hpc-exec-node/files/etc/stage4.run` y será la siguiente:

- **install_packages.sh**: instalará los paquetes que se definen en el fichero de configuración. Estos paquetes serán todo el software científico empaquetado en la distribución y que estarán localmente instalados en cada nodo de cómputo.
- **config_hpcmodules.sh**: realizará la instalación y configuración en el nodo del entorno modules para que realice la carga de los modules de `/usr/local/Cluster-software`. Este será un helper especializado.

$ cat deploy_hpc-exec-node/files/etc/stage4.cfg

```
#!/bin/bash

source $HelperConfigDir/global.cfg


cfg_packages="ubuntucefca-scientific,openmpi-bin,libopenmpi-dev,environment-
    modules\
,libssl-dev,libxp6,mercurial,gdb,libgl1-mesa-dev,libxt-dev,ipython"
```

5.8.1.7. Compilación e instalación del paquete

Una vez que tenemos el paquete listo lo compilamos mediante el comando del `make\_package.sh` mediante el toolkit LGSetup en el directorio tmp.

```
# sudo ./make_package.sh packages/office/deploy_hpc-exec-node/ /tmp/
$ ls -lh /tmp/deploy_hpc-exec-node_0.1.tar.gz
-rw-r--r-- 1 root root 62K may  6 12:28 /tmp/deploy_hpc-exec-node_0.1.tar.gz
```

Podemos copiar y lanzar la configuración sobre cualquier nodo de cómputo de manera sencilla

```
# tar xvf deploy_hpc-exec-node_0.1.tar.gz
# deploy_hpc-exec-node/bin/lgsetup.sh
```

5.8.2. Implementación del despliegue del sistema operativo

Para este desarrollo implementé un pack de recetas en LGDeploy llamado cefcahpc. Para el correcto funcionamiento del pack de recetas es necesario realizar

5.8.2.1. Despliegue del servidor LGDeploy

El software de instalación de LGDeploy va en un fichero `installer.tar.gz`. Tras descomprimir el tar.gz hay que personalizar los ficheros `services.cfg` y `config.cfg`.

LGDeploy es un sistema distribuido que admite tener instalados diversos roles. Dichos roles se especifican en el fichero `services.cfg`. En nuestro caso seleccionamos los siguientes roles:

```
#!/bin/bash

DHCP_SERVICE=N
NBD_SERVICE=Y
DATABASE_SERVICE=Y
TFTP_SERVICE=Y
SAMBA_SERVICE=Y
UDPCAST_SERVICE=N
GPXE_SERVICE=N
REVERSETUNNEL_SERVICE=N
WEBAPP=Y
```

Una breve descripción de los roles activos:

- **NBD_Service**: es un sencillo protocolo de ficheros que trabaja a nivel de bloque. Es una alternativa a NFS para montar sistemas sin disco y lo usamos en LGDeploy para la integración con la herramienta SystemrescueCD.
- **Database_Service**: es el servidor de base de datos Mysql que usa LGDeploy para su funcionamiento. La base de datos puede estar en otro host pero en nuestro despliegue la instalamos localmente.
- **TFTP_Service**: se instala un servidor TFTP desde el que se servirán los archivoís va PXE para los arranques de los equipos. Sin embargo, no es un TFTP cualquiera, realiza un punto de montaje FUSE servido por un demonio FTS. Realicé e implementé un módulo para dicho demonio que sirviese de integración con LGDeploy.
- **SAMBA_Service**: es un servidor de ficheros de red que pueden montar los clientes LGDeploy y se usa para almacenar imágenes completas de los sistemas y poder realizar despliegues basados en imágenes.

- **WebAPP**: es la aplicación web de LGDeploy, el corazón del sistema. Se instala y configura un servidor Apache y todo su entorno.

Después se configuraron todos los valores relevantes en el fichero `config.cfg`. En este fichero se configuran especialmente usuarios y passwords que se usarán y orígenes desde los que descargar el software necesario.

Tras configurar esto basta ejecutar el instalador.

```
# installer/bin/install_all.sh
```

5.8.2.2. Instalación del libro de recetas CEFCAHPC

Para el desarrollo de este proyecto, se realizó un libro de recetas LGDeploy que engloba las recetas con las que gestionaremos la automatización. Este libro de recetas tiene como dependencia el libro de recetas Ubuntu, que durante la instalación prepara todo el entorno necesario para el despliegue de servidores y estaciones de trabajo Ubuntu.

Las recetas que se desarrollaron fueron las siguientes:

- **CefcaHpcPushCommand**: Realiza la ejecución de un comando en todos los nodos del clúster.
- **CefcaHpcPushScript**: Realiza la ejecución de un script dinámico en todos los nodos del clúster.
- **CefcaHpcCronScript**: Pone un script dinámico a disposición de todos los agentes cron instalados en los nodos.
- **CefcaHpcDeployNode**: Realiza un despliegue completo del sistema operativo mas la configuración en un nodo.
- **CefcaHpcDeployNodes**: Realiza un despliegue completo de sistema operativo mas configuración en todos los nodos.

Posteriormente ampliaremos la información en el uso de las recetas.

Los libros de recetas vienen en la propia distribución de LGDeploy, por lo que simplemente habrá que ejecutar los instaladores de los libros de recetas.

```
# installer/bin/install_recipe_ubuntu.sh
# installer/bin/install_recipe_cefcahpc.sh
```

5.8.2.3. Alta de nodos en servidores DHCP

Como hemos visto, entre los roles instalados no se encuentra el rol de servidor DHCP. Esto es porque usaremos el clúster ya existente en CEFCA de servidores DHCP para servir las concesiones. La pega de no instalarlo es que tendremos que dar de alta los equipos manualmente en el servicio y realizar la configuración del parámetro `next-server` en los servidores DHCP para que apunten al servidor TFTP integrado en nuestro servidor LGDeploy.

La configuración es muy sencilla. En nuestro caso, generamos un fichero de concesiones estática dedicado en el que haremos a nivel de grupo los parámetros filename y next-server. Una vez integrada y replicada la configuración en el clúster DHCP, recargaremos el servicio.

```
# cat /etc/dhcp/dhcpd/leases/group-hpc_servers.conf
group {
        next-server lgdeploy.office.cefca.es;
        filename "pxelinux.0";

        ## cefca-sv-013
        host hpc-node1 {
                hardware ethernet 9c:b6:54:0f:e6:98;
                fixed-address hpc-node1.office.cefca.es;
        }
        ....
```

Pero esto no es suficiente, por el diseño de red visto, el clúster DHCP se encuentra en una red distinta que la red del clúster. Por lo tanto, será necesario instalar un servidor DHCP Relay en el cortafuegos que hace de pasarela a los nodos del clúster.

5.8.2.4. Inicio de sesión en el sistema

Tras la instalación del sistema, iniciaremos sesión con el usuario creado en el proceso de instalación usando la URL de la aplicación. Puede verse en la figura 5.76.

5.8.2.5. Creación de las entradas de arraque

En primer lugar hay que crear los *Arranques de los equipos,* para ello nos iremos a `Arranque de equipos -> Agregar entrada` y rellenaremos los datos de los equipos. Rellenaremos la mac, el nombre del host, el elemento de arranque, la dirección ip y la etiqueta. El elemento de arranque se trata de la configuración pxelinux con que arrancará por defecto el equipo. En el caso de estos servidores HP, el arranque `localboot` daba problemas, por lo que hay que seleccionar el arranque `localboot\_hd0`. En el caso de los nodos de cómputo

FIGURA 5.76: Inicio de sesión en LGDeploy

FIGURA 5.77: Crear entrada de arranque en LGDeploy

daremos el nombre de etiqueta *hpc-exec-node*. Esto es importante, porque en las recetas nos referiremos a las etiquetas. En la figura 5.77 puede verse la creación de un arranque.

Una vez que hemos creado los arranques, podremos modificarlos o eliminarlos. En la figura 5.78 podemos ver los arranques de nuestro clúster y sus acciones por defecto.

5.8.2.6. Crear las instancias Push

LGDeploy puede enviar enviar en cualquier momento un trabajo a cualquier dispositivo realizando un push si existe un agente para ello. En la versión de LGDeploy hay dos tipos de agentes: local y ssh. En la última fase del despliegue automatizado de los nodos con LGDeploy, se instalará una key pública ssh con la que posteriormente podremos

FIGURA 5.78: Listado de entradas de arranque en LGDeploy

FIGURA 5.79: Crear instancia SSH en LGDeploy

enviar trabajos. Para su uso es necesario que creemos las instancias push necesarias. Para ello nos iremos a `Instancias push -> Crear instancia SSH`, e indicaremos los datos tal y como se ve en la figura 5.79.

En la figura 5.80 puede verse un listado.

5.8.2.7. Etiquetado de las instancias Cron

Ya vimos en la etapa de configuración que los nodos de cómputo se registraban en lgdeploy. Esto lo hacían para crear una instancia cron en el sistema y negociar una

FIGURA 5.80: Listado de instancias push en LGDeploy

FIGURA 5.81: Modificar instancia cron en LGDeploy

clave compartida. Si la instancia cron ya existía, tan sólo se genera una nueva key compartida. El problema es que no tenemos las instancias etiquetadas y la primera vez hay que hacerlo, para ello nos iremos a Instancias Cron y pulsaremos sobre el botón de modificar y rellenaremos en el formulario la etiqueta, como puede verse en la figura 5.81.

5.8.2.8. Uso de las recetas

La unidad de ejecución de LGDeploy son las tareas. Existen distintos tipos de tareas y cada una tiene sus propias características y funcionalidad. Además de esto, para llevar

FIGURA 5.82: Listado de recetas en LGDeploy

su comentido, las distintas tareas usan diferentes ficheros y plantillas. Aunque es posible definir tareas desde la interfaz de LGDeploy, es un proceso muy engorroso y repetitivo. Con el fin de simplificar los procedimientos más comunes de administración se definieron las recetas.

Podemos definir una receta en LGDeploy como un conjunto de clases y de ficheros asociados que proporcionan al administrador una funcionalidad de alto nivel y que implementan una lógica particular en la creación de tareas para llevar a cabo su cometido. A partir del concepto de receta, podemos decir que los libros de recetas no son más que un conjunto de recetas relacionado.

En el servidor únicamente instalamos los libros de recetas de *Ubuntu* y *CefcaHpc*. En la figura 5.82 podemos ver todas las recetas que contienen.

5.8.2.9. Receta CefcaHpcDeployNode

Con esta receta instalaremos completamente el sistema en un nodo: despliegue del sistema operativo y posterior configuración.

Para ello la receta genera dos tareas: una de tipo `Unattend` y otra de tipo `Script`. La tarea de tipo `Unattend` será la encargada de desplegar el sistema operativo en el nodo mediante el uso del debian installer y un preseed diñamico generado por LGDeploy a partir de los elementos de dicha tarea. Al final del propio preseed se instalará un script de primer arranque que hará que en el siguiente arranque se dispare una tarea de tipo Script.

Esta nueva tarea de tipo Script descargará el script asociado a la tarea cuya lógica es bajarse un paquete de despliegue que hayamos realizado previamente con LGSetup y lo ejecutará. Una vez finalizado todo el proceso, notificará su estado y enviará el log de todo el proceso a LGDeploy.

La entrada de datos de la receta es:

- **Hostname**: nombre del nodo. Deberá coincidir con el Hostname dado en Arranque de equipos.
- **Interfaz de red**: será el nombre de la interfaz de red principal que se usará para el correcto arranque del debian installer. En el caso de estos HP Proliant, el nombre de interfaz principal es `em1` mientras que en las máquinas virtuales de el entorno de pruebas es `eth0`. Para solucionar este problema, se ha dejado como parámetro de la receta y si no se pone nada usará por defecto `em1`.
- **Nombre completo del usuario**: será el nombre completo del usuario administrador con el que realizará la instalación debian installer.
- **Usuario**: será el nombre del usuario del administrador.
- **Password**: será la contraseña que se establecerá durante la instalación.
- **Usar apt-proxy**: si activo, el preseed será configurado para que use el apt-cacher interno de CEFCA.
- **Paquetes adicionales**: todos los paquetes especificados separados por espacio se instalarán de manera adicional junto con el despliegue.
- **Copiar key ssh**: si lo activamos, se copiará la clave pública RSA de lgdeploy para que podamos posteriormente gestionar su instancia push.
- **Paquete de despligue**: será el nombre del paquete de tipo LGSetup que descargará del servidor LGDeploy. Por defecto el formulario fija su valor a deploy_hpc-exec-node, sin embargo puede cambiarse por otro para realizar despliegues de configuración diferentes o de pruebas. Los paquetes de despliegue deberán ubicarse en el directorio `/usr/local/lgdeploy/var/securefiles` del servidor LGDeploy.

En la figura 5.83 puede verse una captura de pantalla de la receta.

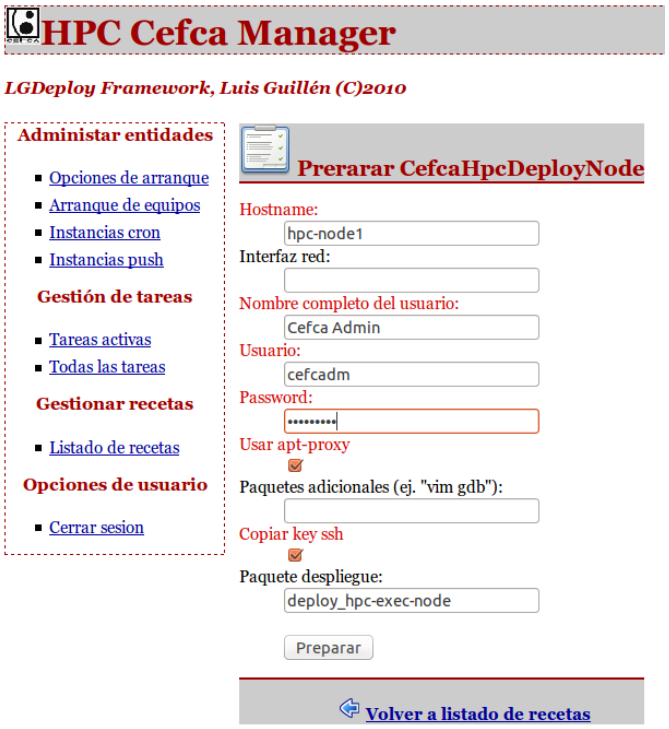


FIGURA 5.83: Receta CefcaHpcDeployNode en LGDeploy

5.8.2.10. Receta CefcaHpcDeployNodes

Con esta receta instalaremos completamente el sistema en todos los nodos: despliegue del sistema operativo y posterior configuración.

El funcionamiento de esta receta es idéntico al visto para la receta *CefcaHpcDeployNode*. La diferencia es que en lugar de hacerlo por hostname, lo hace por etiqueta. Esto hará que obtenga todos las entradas de arranque asociadas a dicho tag y para cada una de ellas genere un par de trabajos idénticos a los comentados en la receta anterior.

La entrada de datos de la receta es idéntica a la anterior salvo:

- **Tag**: nombre de etiqueta de los nodos. Deberá coincidir con las etiquetas definidas en los arranques.

En la figura 5.84 puede verse una captura de pantalla de la receta.

5.8.2.11. Receta CefcaHpcPushCommand

Con esta receta se lanzará inmediatamente un comando a todos los nodos que tengan la etiqueta pasada.

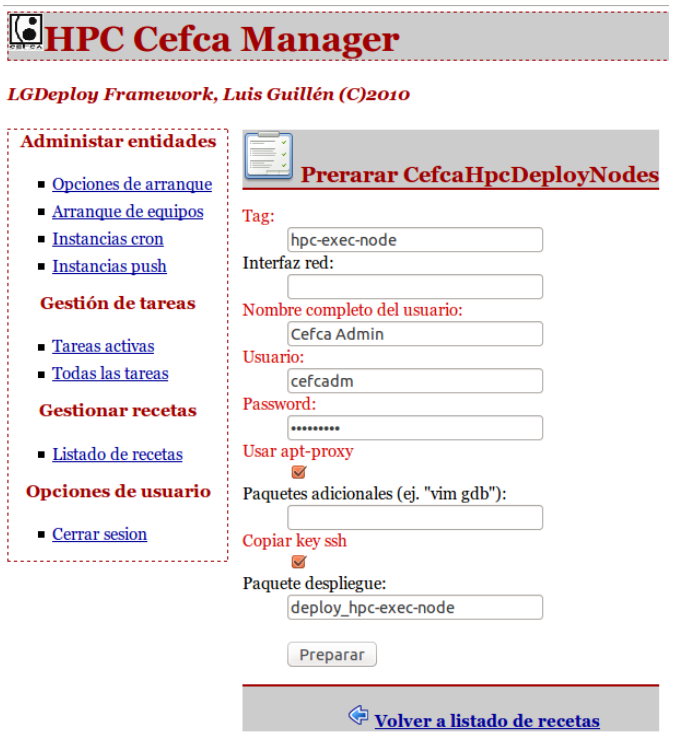


FIGURA 5.84: Receta CefcaHpcDeployNodes en LGDeploy

Para ello la receta obtendrá todas las instancias push que tengan la etiqueta pasada y generará una tarea de tipo `Command` con el comando especificado en la receta para cada instancia y realizará el push. El comando se ejecutará en todos los nodos y obtendremos el estado y el log de salida obtenida en cada una de las tareas generadas.

La entrada de datos de la receta será:

- **Etiqueta**: nombre de etiqueta de los nodos. Deberá coincidir con las etiquetas definidas en las instancias push.
- **Comando**: el comando que ejecutará.

En la figura 5.85 puede verse una captura de pantalla de la receta.

5.8.2.12. Receta CefcaHpcPushScript

Con esta receta se lanzará inmediatamente un script dinámico generado desde el servidor LGDeploy a todos los nodos que tengan la etiqueta pasada.

Para ello la receta obtendrá todas las instancias push que tengan la etiqueta pasada y generará una tarea de tipo `Script` con el script especificado en la receta para cada

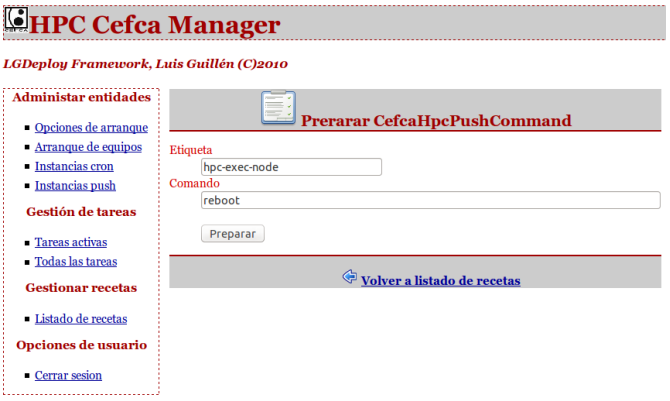


FIGURA 5.85: Receta CefcaHpcPushCommand en LGDeploy

FIGURA 5.86: Receta CefcaHpcPushScript en LGDeploy

instancia y realizará el push. El script se ejecutará en todos los nodos y obtendremos el estado y el log de salida obtenida en cada una de las tareas generadas.

La entrada de datos de la receta será:

- **Etiqueta**: nombre de etiqueta de los nodos. Deberá coincidir con las etiquetas definidas en las instancias push.
- **Script**: el comando que ejecutará. Los scripts dinámicos que ejecutará deberán encontrarse en el directorio `/usr/local/lgdeploy/var/scripts` del servidor.

En la figura 5.86 puede verse una captura de pantalla de la receta.

5.8.2.13. Receta CefcaHpcCronScript

Con esta receta se dejará un script listo para que en la próxima ejecución de todos los agentes cron registrados con la etiqueta definida descarguen el script dinámico desde el servidor LGDeploy y lo ejecuten.

FIGURA 5.87: Receta CefcaHpcCronScript en LGDeploy

Para ello la receta obtendrá todas las instancias cron que tengan la etiqueta pasada y generará una tarea de tipo `Script` con el script especificado en la receta para cada instancia. Los agentes cron instalados en los nodos (y con su correspondiente key de registro) descargaran el script dinámico del servidor LGDeploy y reportarán el estado y el log de salida obtenido.

La entrada de datos de la receta será:

- **Etiqueta**: nombre de etiqueta de los nodos. Deberá coincidir con las etiquetas definidas en las instancias cron.
- **Script**: el comando que ejecutará. Los scripts dinámicos que ejecutará deberán encontrarse en el directorio `/usr/local/lgdeploy/var/scripts` del servidor.

En la figura 5.87 puede verse una captura de pantalla de la receta.

5.8.2.14. Vídeo demostración de despliegue completo del clúster

Realicé un screencast en el que muestro cómo se reinstala el clúster completamente. Es vídeo, que está colgado en Youtube con el título *Demostración despliegue cluster HPC con LGDeploy*, puede verse en [8].

5.9. Mediciones y ajustes del sistema

Ya hemos visto hasta ahora la implantación de la solución. Pero antes de poner en marcha nuestro sistema en producción, pasaremos a caracterizarlo de manera general. Sin llegar

[8] https://www.youtube.com/watch?v=923_oFBYoFE

al detalle, trataremos de detectar los límites y posibles cuellos de botella existentes en nuestro sistema. Conocer los límites nos será muy útil para saber qué tamaños y tipos de problemas podremos resolver con nuestra solución HPC.

Aprovecharemos la tarea de caracterización para llevar a cabo ajustes en el sistema con el fin de lograr una optimización genérica de nuestra solución. En el mundo HPC este proceso lo realizan personas muy especializadas en este campo. Este proceso es un proceso complejo que abarca la optimización a nivel de sistema, a nivel de software y a nivel microarquitectura y es un proceso que se realiza sobre aplicaciones concretas que han de ejecutarse en HPCs órdenes de magnitud superiores al nuestro. Además, tal y como vimos durante el proceso de especificación de la solución, si bien existían determinados tipos de problemas que deberían poder resolverse, nuestra solución debería ser lo suficientemente genérica. Por todo esto, el proceso de optimización que llevaremos a cabo entrará dentro de unos límites razonables y abordables por un PFC.

Para describir el proceso, usaremos la clasificación que empleamos durante el estado del arte para describir un clúster HPC: nodos, interconexionado, almacenamiento y software, e iremos viendo uno a uno las características y ajustes más significativos. Para finalizar veremos el rendimiento del clúster en su conjunto, para lo que ejecutaremos algunos benchmarks. Todo el código fuente de scripts realizados y resultados se entregan junto al DVD adjunto a este PFC.

5.9.1. Nodo

5.9.1.1. Especificación y ajustes

CPU Desde el punto de vista del nodo, vamos a centrarnos en analizar el sistema CPU y memoria. Como vimos, cada nodo de cómputo está compuesto de dos procesadores E5-2670v2. Puede verse la tabla de especificaciones de este procesador en la página en la web de Intel [9].

Como vimos durante el Estado del Arte, la forma en que se mide el rendimiento en cómputo es en Gigaflops. Un Flop significa una unidad de coma flotante por segundo, lo cual nos indica cúantas operaciones de coma flotante por segundo pueden realizarse. Para saber cúantas unidades de coma flotante podemos realizar durante un segundo en un core, deberemos saber cúantos ciclos puede realizar el core en un segundo y cuántas operaciones de coma flotante puede realizar en un ciclo. Cuántas operaciones de coma flotante en un ciclo lo determina la arquitectura del procesador. En el caso de la

[9] http://ark.intel.com/products/75275/Intel-Xeon-Processor-E5-2670-v2-25M-Cache-2_50-GHz

familia Ivy Bridge a la que pertenece nuestro procesador, gracias a las extensiones AVX podemos realizar 8 operaciones de coma flotante de doble precisión en un ciclo (suma sobre 4 números en registro AVX y multiplicación sobre 4 números en registro AVX) [10].

Dado que nuestros cores tienen 2.5GHz podemos realizar el cálculo: 8 x 2.5GHz = 20 GFLOPS. Si multiplicamos por 10 cores tendremos 200 GFLOPS por procesador y si multiplicamos por 2 procesadores tendremos un total de 400GFLOPS por nodo.

En lo que respecta al procesador, se realizaron los siguientes ajustes en la BIOS del sistema:

- Habilitamos HyperThreading (aunque en algunos tests deshabilitaremos para comparar). Existen pros y contras en esta opción. El Hyperthreading o EMT duplica algunas unidades funcionales de un core entre las que se encuentra los registros de estado y permite la ejecución concurrente de dos procesos en un mismo core. El sistema operativo muestra estos "hardware threads" como un procesadores lógicos adicionales y planifica procesos (o hilos) como si de un procesador más se tratase. La pega es que estos hardware threads comparten algunas unidades funcionales críticas como son las unidades de coma flotante o las caches L1 y L2. Esto hace que se compita por estos recursos, creando contención en la unidad de coma flotante o reemplazando memoria de cache del otro hardware thread y con ello probocando fallos de caché. Para un caso de uso general el hyperthreading es recomendable ya que en aritmética de enteros o para procesos que hacen mucho uso de IO como conexiones de red, acceso a disco, etc ofrece un rendimiento de un superior. Sin embargo en el ámbito HPC, debido a las pegas enumeradas, sus beneficios no están tan claros y se suele deshabilitar (de hecho a continuación veremos en algún benchmark ofrece un peor rendimiento). Pero todo depende del problema. En nuestro caso el motivo principal para habilitarlo es que se realiza bastante procesamiento de imágenes y por lo tanto el sistema puede beneficiarse de esta característica.

- Turbo Boost: es una tecnología que sube la frecuencia del reloj cuando detecta que hay mucha exigencia de cómputo en un core, otros procesadores no están haciendo nada y existen las condiciones de temperatura adecuadas. Esto beneficiará principalmente a programas seriales y con escasa paralelización que puedan ocupar algún nodo.

- Seleccionaremos Máximo rendimiento: seleccionaremos el perfil de alto rendimiento. Aunque esté seleccionada esta opción, permitiremos que el sistema operativo realice escalado de frecuencia de manera automática por lo que no tuneáramos la

[10] http://www.penguincomputing.com/how-to-calculate-hpc-efficiency/

política del escalado de frecuencia de todos los cores para ponerla en *performance* (sólo lo haremos en los benchmarks). Esto permitirá el mejor funciomamiento del Turbo Boost y de paso ahorrará algo de energía cuando no se use.

Memoria Ya comentamos cuando modelamos la configuración para el gestor de recursos que nuestro sistema es un sistema NUMA, lo que quiere decir que cada procesador tiene su zona de memoria RAM local y puede acceder a la memoria del otro procesador (memoria remota) mediante un bus de alta velocidad (bus QPI). Además cada procesador tiene su L3 de memoria de 25 MB comartido por todos los cores y cada core L2 de 256KB y L1 de instrucciones y datos de 32KB. En la figura Hwloc - Vista cpus puede verse la organización de los cores y memoria de nuestro sistema.

Intel a partir de *Nehalem* empezaró a incluir el controlador de memoria dentro del mismo procesador, esto incrementó sustancialmente las velocidades de acceso a memoria debido a que se usaba un bus dedicado no compartido con otros recursos. Como vimos en la especificación del procesador los tipos de memoria que son capaces de soportar nuestros nodos son DDR3. Todos nuestros nodos incluyen 8 módulos de tipo HP SmartMemory 16GB HP 16GB 2RX4 PC3-14900R-13. Este tipo de módulos de memoria son de tipo RIMM y soportan una frecuencia de 1866 MHz, sin embargo con esta configuración sólo soporta 1600MHz (según la calculadora de HP [11].

Un aspecto importante del rendimiento del sistema CPU - Memoria es la organización de la misma. Como comenté, es una arquitectura NUMA y cada procesador debe disponer de su propia memoria disponible. Además la memoria se organiza en canales que pueden ser accedidos en paralelo, por lo que para un óptimo rendimiento se busca distribuir los DIMMS de forma que se ocupe el mayor número de canales posibles, debido a esto y siguiendo el paper [[12], se instalaron los módulos en P1-1, P1-3, P1-6, P1-8 y P2-1, P2-3, P2-6, P2-8 tal y como aparece en la figura 5.88.

Además se realizaron los siguientes ajustes en la BIOS del sistema:

- Máximo voltaje: esto permite, a costa de un mayor consumo, que funcione a la mayor velocidad posible. Existen algunos estudios acerca del voltaje.

5.9.1.2. Mediciones

Benchmark Linpack para el nodo Para medir el rendimiento de un nodo usaremos el benchmark Linpack. Este benchmark se ha convertido en el estándar con el

[11] http://h22195.www2.hp.com/MemoryTool/Home/SelectServer
[12] http://h20565.www2.hp.com/hpsc/doc/public/display?docId=emr_na-c03293145

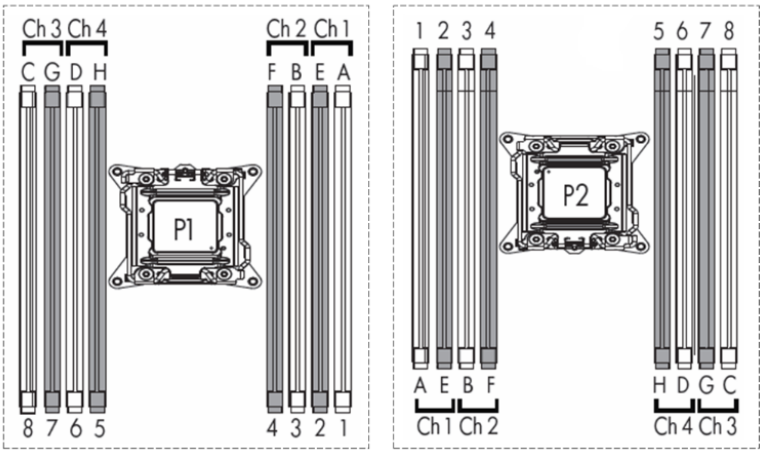


FIGURA 5.88: Organización de los módulos de RAM

que se mide el rendimiento de los HPC (es con el que se obtiene el ranking de los 500 supercomputadores más grandes) aunque también puede usarse obviamente para medir el rendimiento de un único nodo. Realiza un uso muy intensivo de coma flotante y usa rutinas de las librerías de BLAS (Basic Linear Algebra Subroutines), por lo que su rendimiento también dependerá de la implementación de esta. Este benchmark la mayor parte del tiempo utiliza la rutina DAXPY que realiza la operación c[i] = a[i] + b[i] * k.

La implementación del benchmark que usaremos es la HPL. Esta implementación es una implementación escrita en C que genera un sistema de ecuaciones lineal de orden n y lo soluciona usando una descomposición LU con pivote parcial. Además de librerías de BLAS y dado que obviamente el problema está paralelizado, el benchmark usará MPI para la comunicación de procesos. Como obviamente el sistema de ecuaciones que puede resolver un supercomputador no es el mismo que puede resolver, por ejemplo, una raspberry pi (sí, también es posible ejecutar el benchmark en una), es necesario parametrizar el benchmark de acuerdo a las dimensiones de la matriz a resolver, procesos paralelos, etc.

Para la compilación del benchmark usaremos el mismo binario que compilamos mediante EasyBuild en el apartado dedicado al software científico. Para la compilación del binario, se indica al compilador que optimice para la arquitectura del nodo, esto hará que el programa haga uso de extensiones del procesador como AVX. Usaremos la versión compilada con el toolchain `goolf` que incluye a `OpenBLAS` como implementación de BLAS, que a su vez fue compilado específicamente para la arquitecutra del nodo. En cuanto a los parámetros de entrada que usaremos, utilizaremos una calculadora de las diversas que existen en la red [13] para generar el fichero de entrada HPL.dat con los parámetros necesarios.

[13] http://www.advancedclustering.com/act-kb/tune-hpl-dat-file/

Antes de la ejecución cambiaremos la política de escalado de frecuencia de todos los cores para que estén al máximo rendimiento.

```
\#  echo performance | tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor >/dev/null
```

Pondremos el fichero HPL.dat en el mismo directorio y realizaremos un script para realizar la ejecución (en este caso no lo haremos a través del gestor de recursos).

```
#!/bin/bash

OPENBLAS_NUM_THREADS=1
export OPENBLAS_NUM_THREADS

module load HPL/2.0-goolf-1.5.14-no-OFED
mpirun -np 20 /home/CEFCA/lguillen/xhpl-final/mpi-ST-20-1/xhpl.openblas
```

Como vemos en el script del ejemplo, indicaremos a las librerías openblas que use un único thread y llamamos a mpirun indicando que queremos 20 procesos (que es el número de cores que tiene el nodo). Tras la ejecución tenemos la siguiente salida:

```
================================================================================
HPLinpack 2.0  --  High-Performance Linpack benchmark  --   September 10, 2008
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :  107136
NB     :     192
PMAP   : Row-major process mapping
P      :       4
Q      :       5
PFACT  :   Right
NBMIN  :       4
NDIV   :       2
RFACT  :   Crout
BCAST  :  1ringM
DEPTH  :       1
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
```

```
ALIGN  : 8 double precision words

--------------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
          ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be               1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0

================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4      107136   192     4     5            2048.96              4.001e+02
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0027286 ...... PASSED
================================================================================

Finished      1 tests with the following results:
                         1 tests completed and passed residual checks,
                         0 tests completed and failed residual checks,
                         0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```

Como vemos en esta ejecución estamos obteniendo el máximo posible (400 Gflops). Se realizaron diversas pruebas de ejecución utilizando dos variables: hilos configurados en librería openblas y parámetros del sistema. En lo que respecta a parámetros del sistema definimos tres:

- ST: en este caso deshabilitaremos el Hyper Threading.
- HT: en este caso habilitamos el Hyper Threading.
- HT (bindtocore): en este caso habilitamos el Hyper Threading llamamos a `mpirun` con el parámetro `bind-to-core`.

En los ficheros incluídos con el código fuente puede verse la invocación de los benchmark, los ficheros HPL.dat y la salida de la ejecución de los mismos. Tras la ejecución del benchmark obtuvimos los resultados mostrados de la figura 5.89.

Como puede verse, el mejor resultado posible del benchmark lo obtuvimos con el Hyper Threading deshabilitado. En este modo el benchmark nos está dando un 100 % del rendimiento teórico. Sin embargo, si agregamos hilos de ejecución a la librería OpenBLAS para que paralice alguno de sus cálculos, el rendimiento cae hasta casi el 50 %.

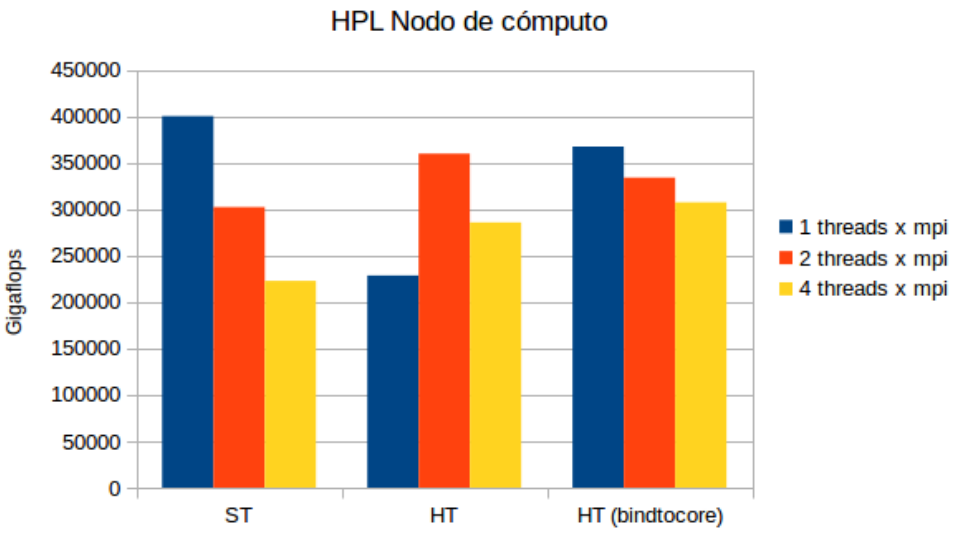


FIGURA 5.89: Ejecución de hpl en nodo de cómputo

Si observamos los resultados obtenidos con el Hyper Threading, llama especialmente la atención el bajísimo rendimiento del benchmark con un único hilo, que es ligeramente superior 50 %. Sin embargo, cuando usamos 2 hilos al sistema, el rendimiento mejora sustancialmente, estando en el 90 % del rendimiento posible. Agregando más threads vuelve a caer.

El último de los modos se hizo tras ver el bajísimo rendimiento del benchmark con el Hyper Threading habilitado para un único hilo. Mi hipótesis era que o bien el sistema operativo (supuestamente si no se explicita en mpirun deja al sistema operativo) estaba planificando procesos en hardware threads del mismo core, o bien en cada replanificación asignaba el proceso a un core distinto, inutilizando los datos que pudiera haber en caché (aunque si esto estaba ocurriendo, sus efectos no se veían con el Hyper Threading deshabilitado). Por supuesto, también podía ser una combinación de ambos factores. Por ello utilicé el parámetro bind-to-core. Con este parámetro cuando mpirun levanta los procesos necesarios, modifica la afinidad de cada proceso (mediante una llamada al sistema) para vincularla a un core físico (OpenMPI está compilado con las librerías hwloc, que le indican sobre la topología del nodo). Los datos obtenidos parecen validar mi hipótesis, pero como puede verse obtiene un rendimiento inferior usando 2 hilos. Aquí dejé mi investigación ya que continuar las dos líneas posibles llevaría tiempo y saldríamos del contexto del PFC. Las líneas posibles que barajaba serían: comprobar la paralelización que OpenBLAS lleva a cabo en las rutinas que se emplean en el benchmark para ver si el tiempo adicional que requiere la de paralelización es mayor que la mejora obtenida por esa paralelización sobre el algoritmo serial. Si realmente existiese tal mejora, sería un problema de la planificación de los hilos llevada a cabo por la librería OpenMP (OpenBLAS se basa en esta librería), que no están siendo alocados de forma óptima.

Durante la ejecución de los benchmarks se estuvo observando las gráficas de rendimiento proporcionadas por Ganglia. Por un lado tenemos gráficas de procesos y por otro lado gráficas de utilización de la CPU. El primer grupo de imágenes 5.91 muestra la ejecución con el Hyper Threading habilitado usando 1 hilo, 2 hilos y 4 hilos. En el segundo 5.93 se muestra cómo se desactiva el hyperthreading y vemos en la misma gráfica la ejecución de 4 hilos con hyper threading y la ejecución con el hyper threading deshabilitado y 1 hilo de ejecución. Estas gráficas explican algunas de las diferencia de rendimiento medida. Sin embargo, hay que entender previamente a las gráficas uno de los problemas existentes con el Hyper Threading a la hora de visualizar gráficas e interpretar resultados. Como vimos, con el Hyper Threading el sistema operativo ve el doble de cores y por lo tanto contabiliza su uso del mismo modo que si de cores reales se tratase. Por ello no hay que caer en el error de pensar que un menor uso de cpu implica un menor rendimiento (aunque normalmente sea así). Esto se ejemplifica muy bien con la primera gráfica de carga de CPU, en ella vemos un uso de la mitad de la CPU, pero como vimos en los resultados con la opción bind-to-core obtuvimos un mayor rendimiento. Las diferencias de rendimiento a las que me refiero es lo que ocurre cuando se agregan hilos de ejecución y se produce una sobresubscripción de las tareas en ejecución. Puede verse cómo el tiempo dedicado a sistema se incrementa sustancialmente y roba tiempo de usuario (que es el que realmente realiza las tareas de cómputo). Esto es debido al tiempo invertido en cambios de contexto, gestión de concurrencia, etc. En la última de las gráficas puede verse la ejecución con el HT deshabilitado cómo el tiempo de usuario ocupa casi el 100 % del tiempo de CPU y por lo tanto obtenemos el mayor rendimiento.

Benchmark STREAM de memoria El ancho de banda de la memoria es un factor muy importante ya que al final determina el tiempo que se tarda en traer información de la memoria a nuestra CPU. Como podemos ver en la figura 5.94 obtenida del magnífico libro "Optimizing HPC Applications with Intel Cluster Tools", el ancho de banda con la memoria principal es varios órdenes de magnitud inferior al ancho de banda obtenido sobre la memoria caché. Dependiendo del tipo de problema, puede ser que la eficiencia de las cachés sea muy baja y por lo tanto ser un factor limitante al cómputo.

Uno de los benchmarks más usados para medir el ancho de banda de la memoria es el STREAM [14]. Es un benchmark sintético que mide el ancho de banda sostenido para diversas operaciones computacionales:

- copy: a(i) = b(i)
- scale: a(i) = q*b(i)

[14] http://www.cs.virginia.edu/stream/

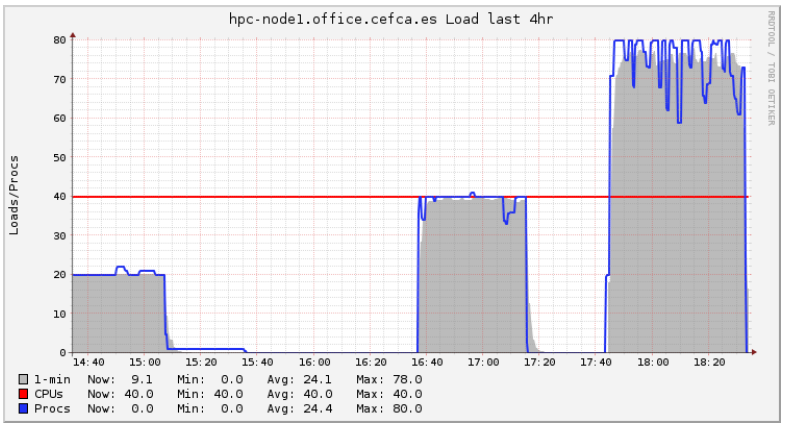


FIGURA 5.90: Estadística de procesos durante ejecución xhpl con HT

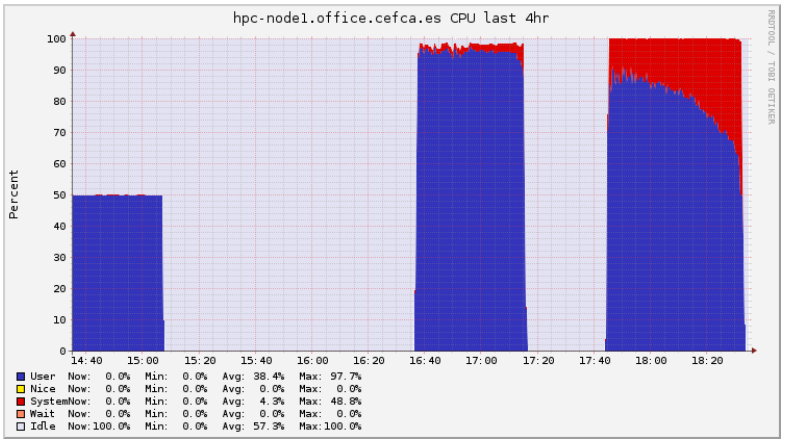


FIGURA 5.91: Estadística de CPU durante ejecución xhpl con HT

- sum: a(i) = b(i) + c(i)
- triad: a(i) = b(i) + q*c(i)

El benchmark tiene dos parámetros a configurar, el tamaño de los arrays sobre los que realizará el benchmark y el número de iteraciones que realizará. Para definir el tamaño de los arrays el creador del benchmark dice que hay que asignar un tamaño al menos 4 veces el tamaño de la caché de nivel 3. En nuestro caso tenemos una caché L3 de 25M, por lo que si los elementos del array son de tipo double (64 bits) que ocupan 8 bytes, tendremos que la caché podrá alojar 3125000 elementos. A continuación multiplicamos por cuatro y obtenemos que son necesarios 12500000 elementos. Como vamos a medir en paralelo y tenemos dos cachés de nivel 3, daremos 25000000 elementos al array. En cuanto el número de iteraciones, para facilitar la observación de la ejecución del benchmark pondremos 200. También para simplificar los parámetros en tiempo de ejecución de OpenMP del benchmark (las últimas versiones del benchmark incluyen este soporte [https://sites.utexas.edu/jdm4372/2013/01/17/stream-version-5-10-released/]) y para interpretar mejor los resultados, durante todas estas pruebas deshabilitaremos el hyperthreading.

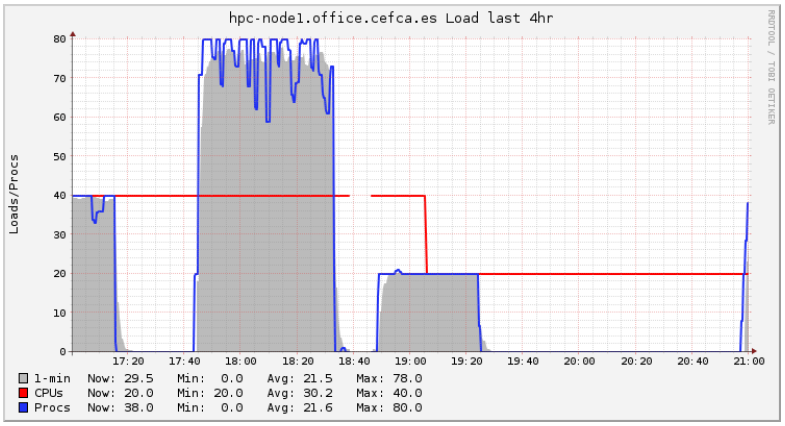


FIGURA 5.92: Estadística de procesos durante ejecución xhpl con HT

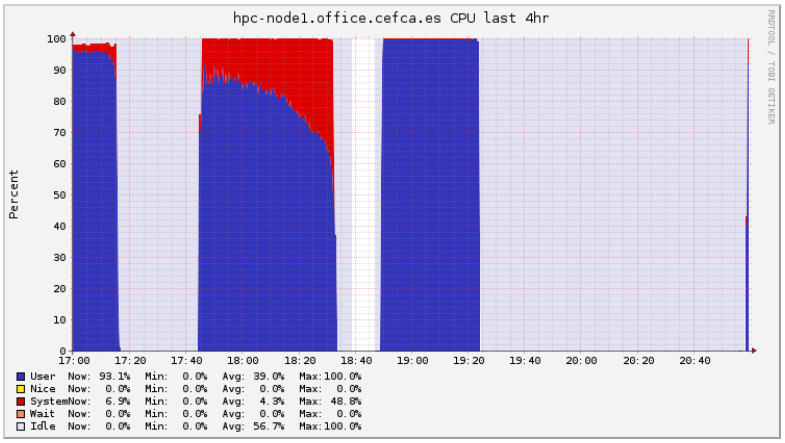


FIGURA 5.93: Estadística de CPU durante ejecución xhpl sin HT

Para ello compilaremos el benchmark con:

```
gcc -O3 -DSTREAM_ARRAY_SIZE=25000000 -DNTIMES=200 -fopenmp  -o stream stream.c
```

También compilaremos una versión single core para estas comparaciones iniciales:

```
gcc -O3 -DSTREAM_ARRAY_SIZE=12500000 -DNTIMES=200 -o stream.mono stream.c
```

Como vimos durante el proceso de configuración, habilitamos los contadores hardware que proporciona Intel. Usaremos dichos contadores para monitorizar el estado de la ejecución del benchmark. Para ello deberemos descargar el software Intel Performance Counter Monitor [https://software.intel.com/en-us/articles/intel-performance-counter-monitor]. Este software proporciona unas librerías que podremos incluir en nuestros programas para acceder a los contadores hardware y además unos programas que hacen uso de los mismos. Una vez descargado el software lo compilaremos y realizaremos las operaciones indicadas para su uso:

```
# make
# echo 0 > /proc/sys/kernel/nmi_watchdog
# modprobe msr
```

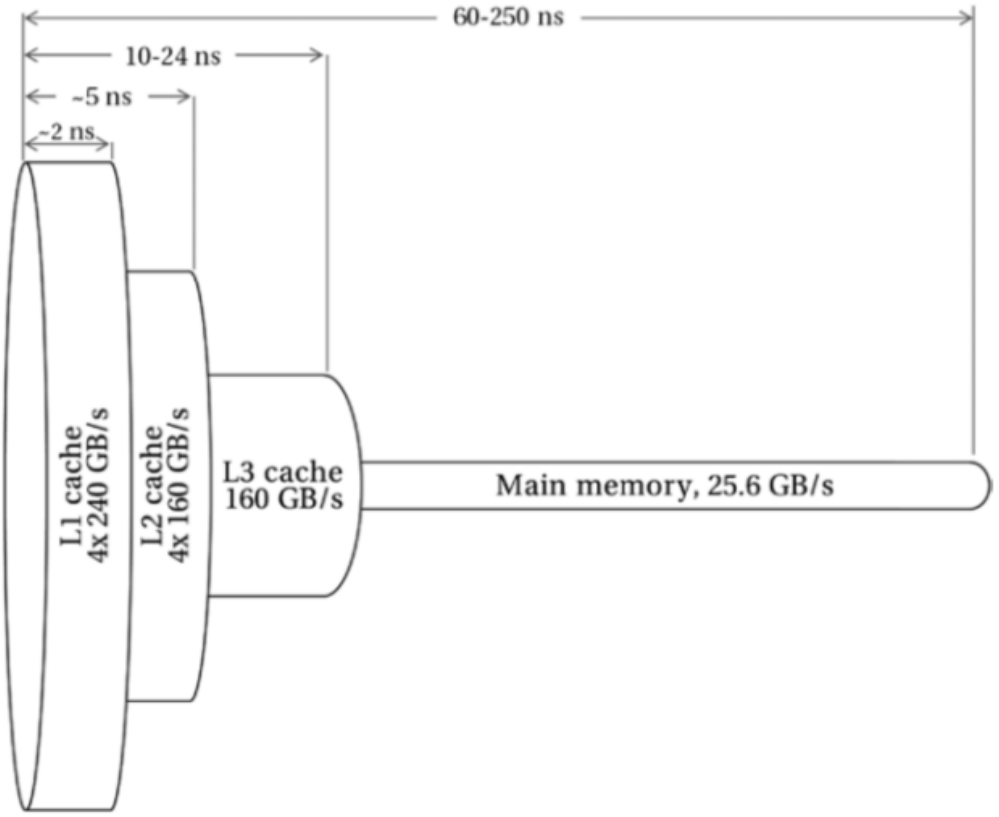


FIGURA 5.94: Ancho de banda y latencia en un procesador Haswell

Lanzaremos los ejecutables del software stream y además ejecutaremos en otras ventanas (en diversas ejecuciones) los programas pcm-memory.x pcm-numa.x. A continuación puede verse la salida del benchmark stream (usando openmp) y capturas de pantalla de las utilidades de Intel en las figuras `pcm-memory counters` y `pcm-numa counters`.

```
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 25000000 (elements), Offset = 0 (elements)
Memory per array = 190.7 MiB (= 0.2 GiB).
Total memory required = 572.2 MiB (= 0.6 GiB).
Each kernel will be executed 200 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Number of Threads requested = 20
Number of Threads counted = 20
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 5154 microseconds.
   (= 5154 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
```

```
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           61199.4     0.006651     0.006536     0.007141
Scale:          59735.2     0.006820     0.006696     0.007239
Add:            68190.8     0.008887     0.008799     0.009306
Triad:          69258.7     0.008786     0.008663     0.009287
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

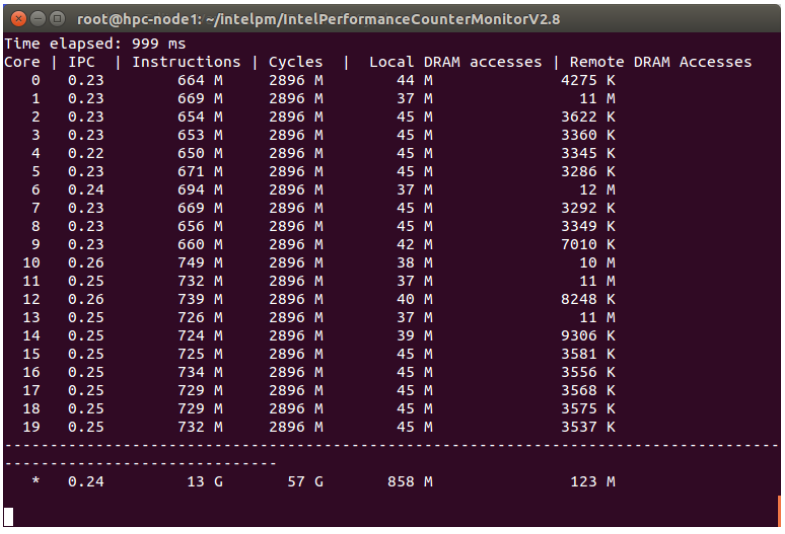


FIGURA 5.95: Medición con numa-memory.X

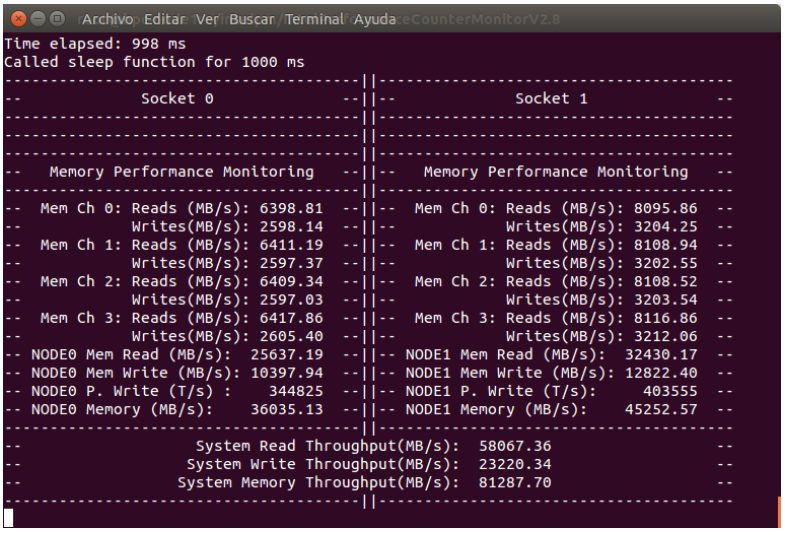


FIGURA 5.96: Medición con pcm-memory.X

Se realizaron diversas pruebas de ejecución, jugando con los parámetros de la librería openmp [https://gcc.gnu.org/onlinedocs/libgomp.pdf] y con taskset para la versión de sólo un core a la vez que se analizaba cómo se repartía el tráfico de memoria con las

utilidades de Intel. Finalmente realizamos una gráfica de la figura 5.97 en la que se compara la ejecución en 1 core y en 20 cores.

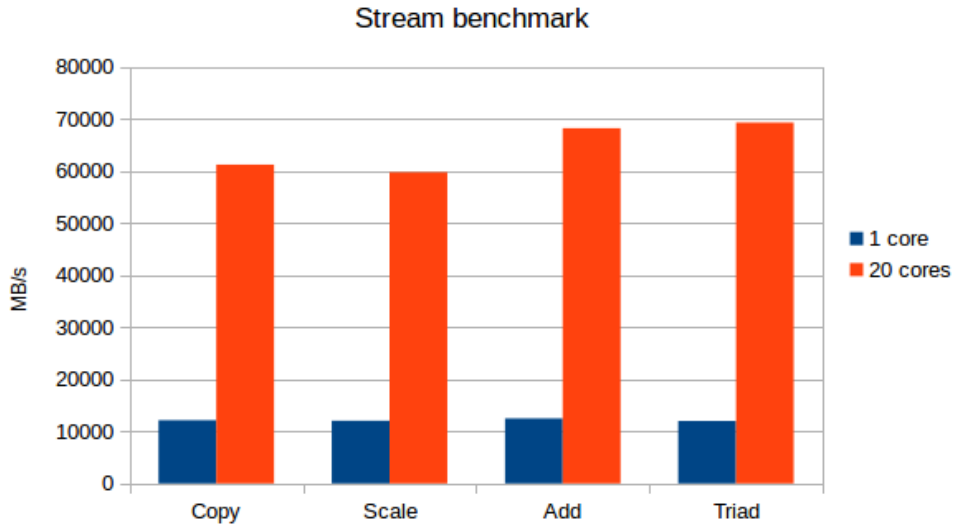


FIGURA 5.97: Resultados de Stream Benchmark

Con 20 cores el sistema alcanza un ancho de banda sostenido de 69GB/s. Aunque hay que tener en cuenta que la configuración de memoria es de 1600MHz en lugar de los 1866MHz que puede llegar a soportar el procesador, esta medición se queda lejos del ancho de banda máximo especificado por Intel de de 59.7 GB/s por procesador y de otras mediciones STREAM vistas en la red [15]. Como puede verse en la figura `pcm-memory counters`, durante la medición se observaron picos con mayores velocidades de transferencias. El compilador usado ha sido el gcc 4.8.2 distribuido en ubuntu 14.04, quedará pendiente para el futuro realizar pruebas con una trial del compilador de Intel y comparar resultados.

Como puede verse en la gráfica `Stream Benchmark`, el ancho de banda total que puede obtenerse depende obviamente del número de cores que estén accediendo en paralelo al sistema de memoria. También puede apreciarse que 20 cores no suponen un factor 20 sobre el ancho de banda obtenido en un core. Por ello, vamos a realizar un pequeño estudio de cómo nuestro sistema es dependiente del número de cores y de cómo la distribución de estos cores influirá en el sistema de memoria.

Para realizar este estudio jugaremos con los parámetros de OpenMP que pasaremos vía variables de entorno:

- OMP_NUM_THREADS: determina el número de hilos que lanzará.

[15] http://www.admin-magazine.com/HPC/Articles/Finding-Memory-Bottlenecks-with-Stream

- OMP_PROC_BIND: determina que el hilo permanecerá en el core (evitando que el planificador del sistema operativo lo reasigne a otro core). También la afinidad por defecto con la que asignará los nuevos hilos será al mismo procesador.
- GOMP_CPU_AFFINITY: especificaremos un listado de cores en los que se designará la afinidad de los hilos. Con el hyperthreading desactivado, los cores numerados del 0 al 9 pertenecerán al procesador 1 y los cores numerados del 10 al 19 al procesador 2. Por ejemplo si hacemos GOMP_CPU_AFFINITY="0,1,10,11", determinaremos la afinidad de los hilos a los dos primeros cores de cada cpu.

Las pruebas consisten en ir agregando cores por un lado de manera serial y por otro lado de forma simétrica. De forma serial agregaremos cores al benchmark indicando OMP_PROC_BIND de manera que la afinidad sea en el mismo socket (hasta que superados los 10 cores empiecen a asignarse en el otro). La ejecución de forma simétrica la haremos usando el parámetro GOMP_CPU_AFFINITY agregando cores a una y a otra cpu de la siguiente forma: 0; 0,10; 0,10,1; 0,10,1,11... El resultado de la ejecución de las pruebas puede verse en la gráfica `Stream Triad`. Hemos empleado los resultados de la operación *triad* por ser la operación más significativa.

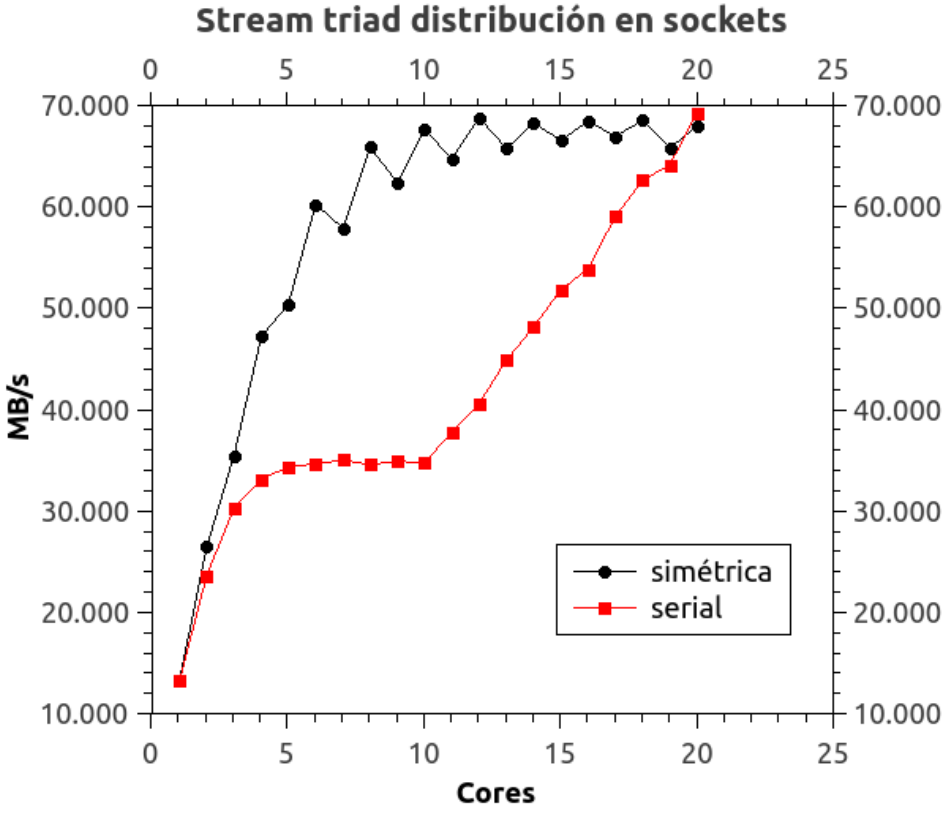


FIGURA 5.98: Stream Benchmark distribuido entre los sockets

Como podemos comprobar en la gráfica de la figura 5.98, el uso distribuido de procesos en las distintas CPUs proporciona un ancho de banda agregado muy superior. En la forma serial vemos cómo a partir de los 5 cores se converge hacia el ancho de banda total que puede soportar una única CPU y cuando empieza a usarse la segunda CPU de nuevo comienza a crecer hasta el máximo soportado. En la forma simétrica vemos cómo el ancho de banda agregado se dispara, obteniendo en los 10-12 cores (5-6 en cada CPU) el máximo del rendimiento.

Como complemento a nuestra observación vamos a tomar los valores de ejecución simétrica de los cores y vamos a observar el ancho de banda disponible a cada core, para ello bastará coger el ancho de banda y dividirlo por el número de cores.

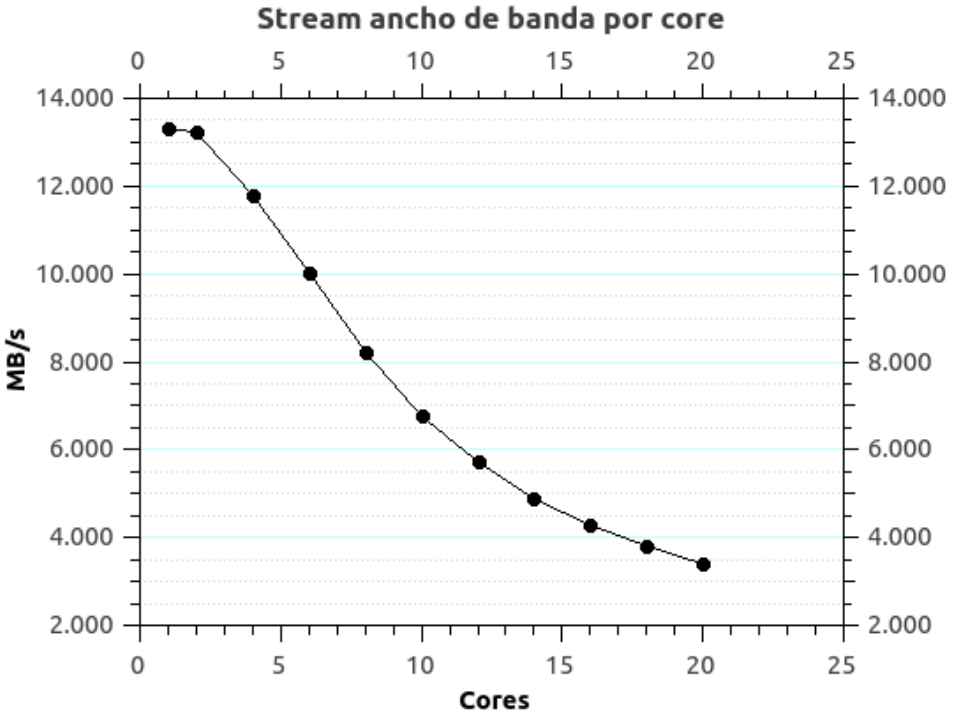


FIGURA 5.99: Stream Benchmark ancho de banda por core

Como vemos en la figura 5.99, el ancho de banda disponible para cada core se reduce significativamente. Esto indica que agregar más cores a nuestro sistema empeoraría todavía más el ancho de banda disponible por core y que por lo tanto el sistema de memoria y su tecnología es el factor limitante.

IMPORTANTE: puede extraerse de este benchmark la errónea conclusión de que es más eficiente distribuir los hilos de un mismo proceso entre los distintos procesadores. Nada más lejos de la realidad, el objetivo de este benchmark es medir el acceso a memoria principal tratando de evitar el uso de la memoria caché. De manera general hay que lograr la mayor afinidad posible en nuestros programas para explotar al máximo el uso

de las memorias caché. Solamente ante procesos independientes o procesos con múltiples hilos en los que se hace uso de matrices muy grandes (como hace el benchmark) en la que la distribución de los datos esté muy dispersa o que necesiten computar dichos datos de manera independiente, sólo en esos casos conviene distribuir los hilos entre procesadores.

5.9.2. Interconexión

5.9.2.1. Especificación y ajustes

Vamos a centrar nuestro estudio en las redes que requieren mayor rendimiento: almacenamiento y cómputo. Como vimos, todas las conexiones de red se realizan mediante el switch Cisco 3750X [16]. Dicho switch se compone de 48 puertos de 1G y usa una política store-and-forward.

Aunque el switch ofrece capa 3 y viene con una licencia IP Base, como vimos, no utilizamos las características de capa 3. Las conexiones que realizamos se hicieron de la siguiente manera:

- Red de almacenamiento:
  - Nodo cabecera: bonding de 4 interfaces
  - Nodo cómputo: 1 interfaz
- Red de cómputo:
  - Nodo de cómputo: bonding de 2 interfaces

MTU El primero de los ajustes que realizamos fue el uso de los jumbo frames en las redes de almacenamiento y cómputo, para lo que se definió una MTU de 9000. El uso de una MTU superior a la predefinida de 1500 bytes proporciona una mayor eficiencia al sistema ya que permite transportar una mayor carga útil (menos paquetes a enviar implican menor cantidad de información de cabeceras). Varios son los inconvenientes de una mayor MTU. El primero de ellos se da si el medio en el cual se transporta la información no es muy fiable (desde el punto de vista de la integridad, no de la seguridad) ya que una mayor longitud de MTU aumenta la probabilidad de que un paquete pueda ser descartado y deba ser vuelto a enviar. Otro inconveniente es que, al circular paquetes más grandes, la contención en la entrada de paquetes es mayor y por lo tanto puede verse aumentada la latencia cuando se dan múltiples conexiones. Pero el principal inconveniente

[16] http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3750-x-series-switches/data_sheet_c78-584733.html

en la práctica de usar una MTU grande es la posible fragmentación que se da cuando un paquete tiene que atravesar distintos saltos (routers) y, aunque existen mecanismos de descubrimiento de la MTU, no pueden usarse ya que suele estar mal configurado equipamiento intermedio [https://blog.cloudflare.com/path-mtu-discovery-in-practice/]. En nuestro caso las redes están aisladas por lo que no habrá ningún problema de fragmentación, el medio es muy buena calidad y, dado el bajo número de conexiones, el incremento general en la latencia será despreciable.

Esta MTU nos dará un máximo de ancho de banda teórico de 123 MB/s [17]. Ya vimos cómo se configuraba y que era necesario configurarla tanto en los nodos como en el switch de comunicaciones.

En el apartado de mediciones ya veremos el uso de iperf, de momento nos bastará con saber que es una herramienta con la que podremos realizar mantener una o varias conexiones TCP cliente-servidor, transmitir un flujo ininterrumpido de datos por ellas. Usando una de las muchas funcionalidades que nos ofrece (que es la medición de la mtu), podemos realizar una rápida comprobación de que estamos haciendo uso de los jumbo frames en todos los hosts mediante esta utilidad. Para ello lanzaremos iperf en el nodo maestro en modo servidor (opción -s) especificando que indique la mtu de la conexión (opción -m) y lanzaremos iperf en los nodos de cómputo (opción -c ipservidor). Tras la ejecución podemos comprobar que los jumboframes se están usando correctamente:

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.14.1 port 5001 connected with 192.168.14.11 port 59441
[  5] local 192.168.14.1 port 5001 connected with 192.168.14.14 port 36191
[  6] local 192.168.14.1 port 5001 connected with 192.168.14.10 port 38308
[  7] local 192.168.14.1 port 5001 connected with 192.168.14.12 port 35661
[  8] local 192.168.14.1 port 5001 connected with 192.168.14.13 port 57571
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-16.7 sec   992 MBytes   499 Mbits/sec
[  4] MSS size 8948 bytes (MTU 8988 bytes, unknown interface)
[  6]  0.0-16.7 sec  1011 MBytes   508 Mbits/sec
[  6] MSS size 8948 bytes (MTU 8988 bytes, unknown interface)
[  5]  0.0-16.7 sec  1.92 GBytes   990 Mbits/sec
[  5] MSS size 8948 bytes (MTU 8988 bytes, unknown interface)
[  7]  0.0-16.7 sec   959 MBytes   482 Mbits/sec
[  7] MSS size 8948 bytes (MTU 8988 bytes, unknown interface)
[  8]  0.0-16.7 sec   978 MBytes   492 Mbits/sec
[  8] MSS size 8948 bytes (MTU 8988 bytes, unknown interface)
```

Bondings de almacenamiento Durante la parte de diseño de red definimos que sólo usaríamos una interfaz de almacenamiento en los nodos de cómputo en vez de

[17] http://rickardnobel.se/actual-throughput-on-gigabit-ethernet/

usar dos interfaces que teníamos disponibles. El motivo por el que no se establecieron ambas interfaces es porque no iban a poder aprovecharse en este switch. Para entender esto es necesario comprender cómo funciona el bonding de interfaces: un bonding de interfaces no duplica "mágicamente" el ancho de banda disponible que puede lograrse en una conexión. Lo que hace es repartir las conexiones entre las interfaces que forman el bonding en función de un criterio y este reparto hace que pueda lograrse el ancho de banda agregado de las interfaces. En los bondings entre switches y hosts tendremos dos actores: el host que decide en qué interfaz de su bonding pone un paquete y el switch que decide en qué interfaz de un bonding debe poner el paquete.

Empecemos por la parte más restrictiva: el switch. Cuando un switch necesita poner un paquete dirigido a una determinada mac cuyo destino está en una interfaz en un bonding debe seleccionar el puerto en el cual poner el paquete. Dónde lo pone dependerá de las posibilidades que ofrezca el switch. En caso de nuestro 3750 existen varias políticas posibles:

- política src-mac: realizará una operación xor y un módulo sobre la mac origen del paquete para seleccionar el puerto. Es la política por defecto
- política ip-src: realizará una operación xor y un módulo sobre la ip origen del paquete para seleccionar el puerto
- política src-dst-ip: realizará una operación xor y un módulo sobre las direcciones ip origen y destino del paquete para seleccionar el puerto
- politica src-dst-mac: realizará una operación xor y un módulo sobre las direcciones mac origen y destino del paquete para seleccionar el puerto
- politica src-dst: realizará una operación xor y un módulo sobre las mac destino del paquete para seleccionar el puerto

Debido a que este modelo de switch sólo ofrece estas políticas y a que la comunicación con la red de almacenamiento siempre se realizará entre el nodo de cómputo y el nodo cabecera, en toda comunicación entre ambos hosts, cuando el switch tenga que hacer un forwarding de un paquete siempre lo pondrá en la misma interfaz, obteniendo en la práctica una única interfaz de red. En otros switches que ofrecen políticas más avanzadas como por ejemplo el número de puerto, sí sería posible obtener mediante varias conexiones el agregado de las interfaces entre nodo y servidor de almacenamiento (pero nunca el agregado en una única conexión tcp).

Ya hemos visto la importancia de la política de bonding, deberemos por lo tanto optimizar el agregado de las cuatro interfaces del servidor cabecera de forma que aproveche

en la mayor medida posible el agregado de las interfaces cuando estén todos los hosts enviándole información. Para ello ejecutamos un servidor iperf en el servidor maestro y lanzamos concurrentemente un iperf cliente desde cada nodo de cómputo experimentando con cada una de las políticas. Además de guardar los datos de iperf, se observó en la utilidad Cacti el tráfico que pasaba por las interfaces del bonding.

Las políticas que se probaron y el orden fueron:

1 política por defecto src-mac

2 política ip-src

3 política src-dst-ip

4 politica src-dst-mac

5 politica src-dst

Se obtuvieron las siguientes gráficas 5.101 en Cacti.

Como puede observarse en Cacti, con ninguna de las políticas es posible aprovechar el agregado de las cuatro interfaces. Por cómo el switch realiza el hash para seleccionar la interfaz, las que mejor resultado presentan son las políticas ip-src y src-dst-ip.

Por ello, se define una política general ip-src en el switch. Siguiendo el mismo principio, se define una política layer2+3 en la configuración del bonding del servidor cabecera para la selección del origen en sus paquetes.

Bondings de cómputo Por lo visto en las pruebas anteriores con el switch, parece que los hash generados con las políticas por ip distribuyen mejor la carga para cuatro interfaces, pero esto no quiere decir que sea igual de eficiente para dos interfaces. Aunque quisiéramos seleccionar otra política para el bonding de cómputo, no sería posible en este switch ya que la política es global.

Sin embargo sí podremos configurar el bonding en origen en la política usada por los servidores. Aunque existen muchas políticas de bonding en Linux nos centraremos en:

1 layer2: usa las direcciones mac origen y destino. Es la usada por defecto

2 layer2+3: usa las direcciones mac y ip.

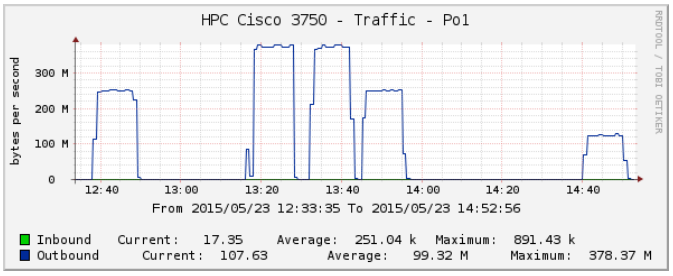


FIGURA 5.100: Estadística tráfico agregado del bonding de almacenamiento

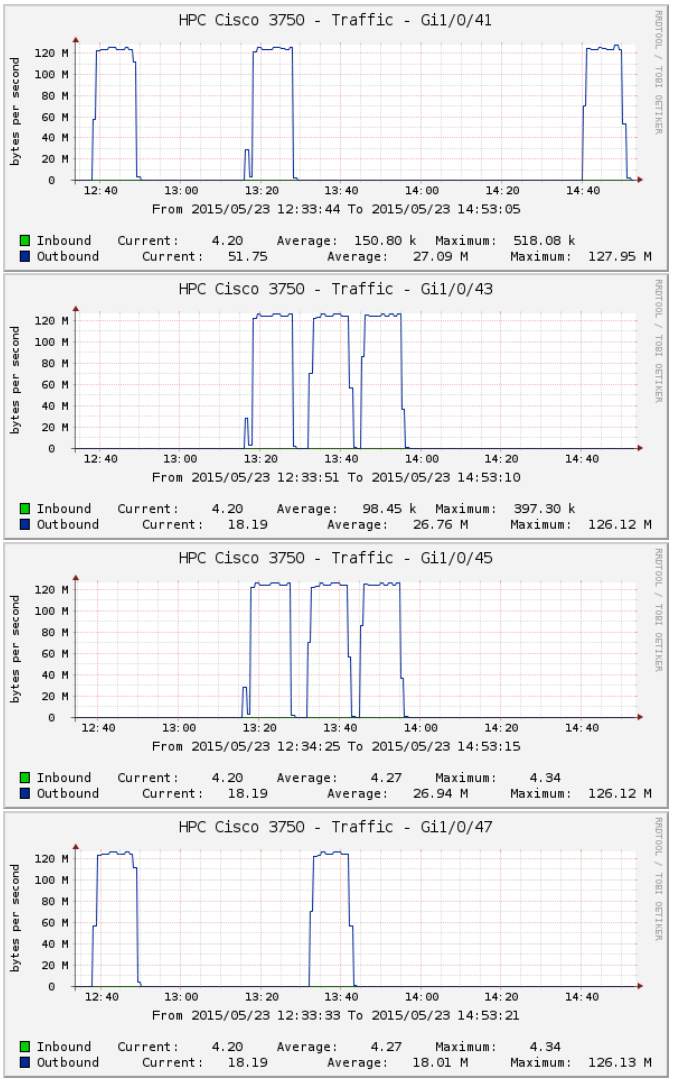


FIGURA 5.101: Estadística tráfico de interfaces del bonding de almacenamiento

En este caso lanzamos de manera concurrente un iperf servidor en cada uno de los nodos y lanzamos un proceso iperf cliente para cada uno del resto de los nodos de forma que obtengamos conexiones de todos los nodos hacia todos los nodos. Esto se hizo mediante scripts y se guardaron resultados de iperf y se observó con Cacti.

Como pueden verse en las gráficas 5.102, en este caso no existían diferencias significativas entre ambas configuraciones, por lo que se optó por la configuración por defecto de layer2.

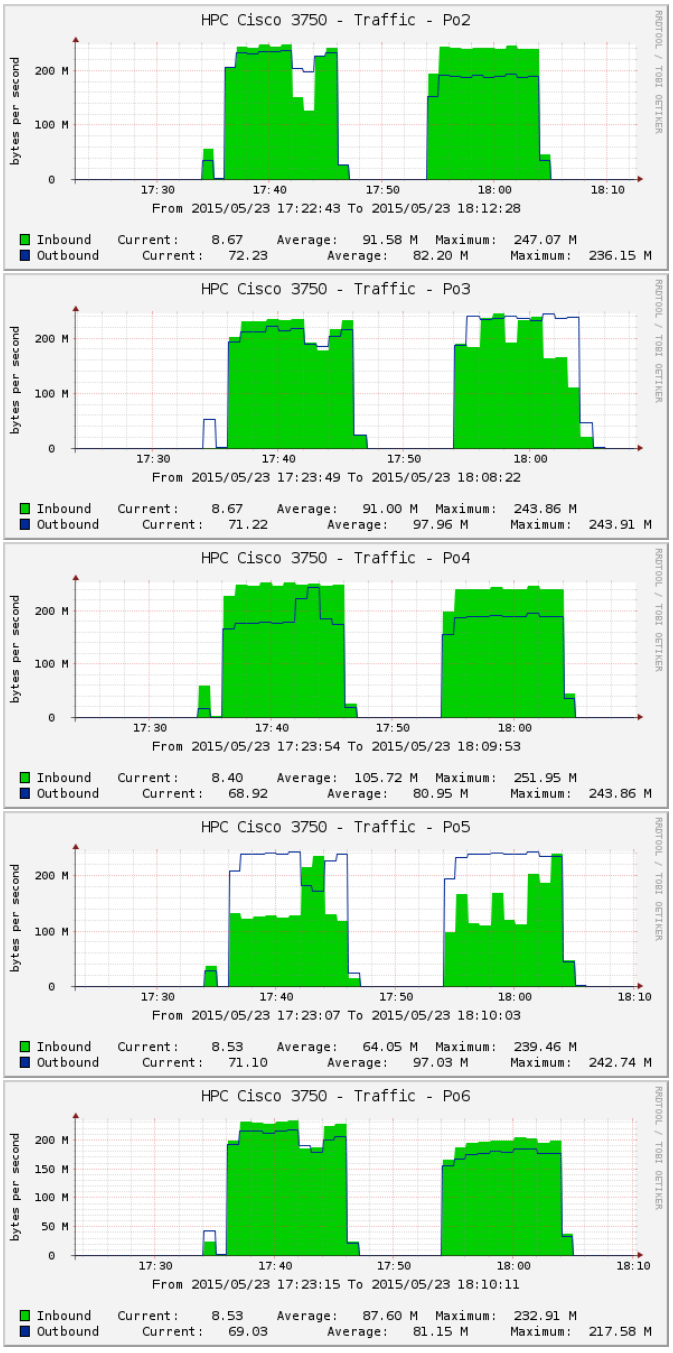


FIGURA 5.102: Estadística de tráfico de los bondings de la red de cómputo

5.9.2.2. Mediciones

Benchmark iperf Ya introdujimos anteriormente algunas funcionalidades de Iperf. Iperf es una herramienta multiplataforma que es capaz de realizar las más diversas pruebas con los protocolos TCP y UDP, admitiendo conexiones concurrentes. Suele utilizarse para medir prestaciones, especialmente para medir throughput aunque también puede usarse para buscar límites en electrónica de red buscando la saturación UDP, mediciones de latencia, etc. En nuestro análisis con iperf nos centramos únicamente en throughput,

en concreto en el throughput que podemos alcanzar concurrentemente. Basándonos en este análisis fue como realizamos los ajustes en el bonding de almacenamiento. A continuación explico en que consisten las pruebas, muestro unas gráficas con los resultados obtenidos y describo brevemente las conclusiones.

Como comenté durante los ajustes de los bondings, esta prueba consiste en un iperf lanzado en modo servidor y un iperf lanzado en modo cliente desde todos los nodos de forma concurrente e indicando en intervalos de un segundo información sobre el ancho de banda. La transferencia de la información se origina desde el cliente al servidor con el fin de observar el efecto del forwarding del switch en el bonding. La prueba se realiza durante 10 minutos y se observó con Cacti tal y como vimos. Además usamos la información de iperf para elaborar la gráfica de la figura 5.103. En primer lugar vemos que el nodo 5 está usando completamente una interfaz en el bonding de almacenamiento, dando los 118MB/s en lugar de los 123MB/s esperados, esto llama la atención ya que es el máximo de 1G con una MTU de 1500 *(NOTA: este resultado se me pasó por alto hasta que comprobé las gráficas y esto lo hice cuando ya había migrado completamente a la red de 10G por lo que ya no pude trabajar en resolver o explicar este problema). El resto de interfaces se reparten dos interfaces de red del bonding. Llama la atención cómo funciona el control de flujo en TCP, dando lugar a esa forma en la cual cuando una conexión pierde velocidad es "robada" por otra conexión pero concentrándose todas en un promedio de 60MB/s dando los 240MB/s que proporciona el agregado de dos interfaces.

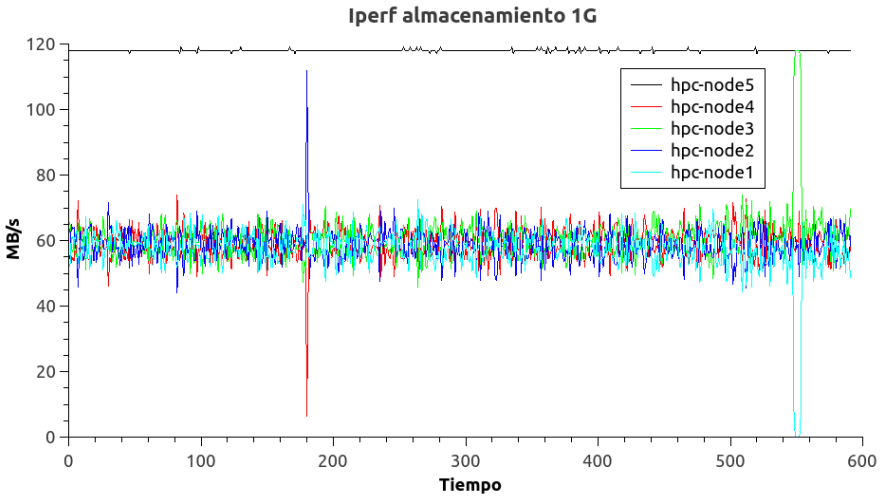


FIGURA 5.103: Iperf red de almacenamiento 1G

La siguiente prueba es la que se vio también en los ajustes de bondings referente al la selección layer2 en el lado del servidor de cómputo y ip-src en el lado del switch. Puede verse la gráfica en la figura 5.104. En este caso se analiza el envío de conexiones desde el nodo hpc-node1 hacia todos los demás de manera concurrente. Mientras tanto todos los

demás también están conectándose entre sí, enviando y recibiendo conexiones de todo el mundo. En general se observa un agregado de los 2Gbps y que entre los 350 y 475 segundos en la ejecución existe una caída de algunos de los flujos de comunicación. Esto se observa también en la asimetría entre Inbound y Outbound en algunos momentos en las gráficas del bonding. Estos cortes en el flujo pueden verse también en las mediciones realizadas desde los otros nodos de cómputo pero el corte en la conexión no se produce en ningún momento y sólo afecta al flujo. Como comenté anteriormente, cuando realicé estas pruebas observé únicamente con cacti el ancho de banda ya que el objetivo era ver el uso agregado de los bondings de interfaces. En el momento en que redacto esto soy consciente de que pude haber observado contadores en busca de descarte de paquetes y capturas de tráfico para ver el comportamiento del control de flujo de TCP (aunque esto último hubiera afectado la observación) y no me es posible volver a reproducir estas pruebas ya que está instalada la red 10G. Mi hipótesis es que tenemos un descarte de paquetes debido a la saturación provocada por la concurrencia, pero no soy capaz de determinar si es en la salida, en el forwarding, en la entrada o afecta a todo.

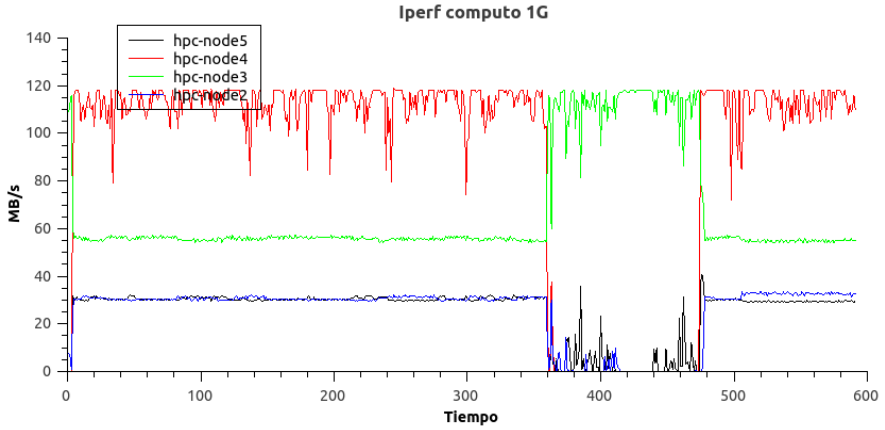


FIGURA 5.104: Iperf red de cómputo 1G

Benchmark netpipe La utilidad netpipe (Network Protocol Independent Performance Evaluator) es una utilidad que se utiliza para medir prestaciones en diversos protocolos. Para todos los protocolos soportados va incrementando el tamaño del mensaje que se envía y mide ancho de banda y lantencia. En este caso los tests se ejecutaron sobre la interfaz de red general. Pueden verse los resultados en la figura 5.106.

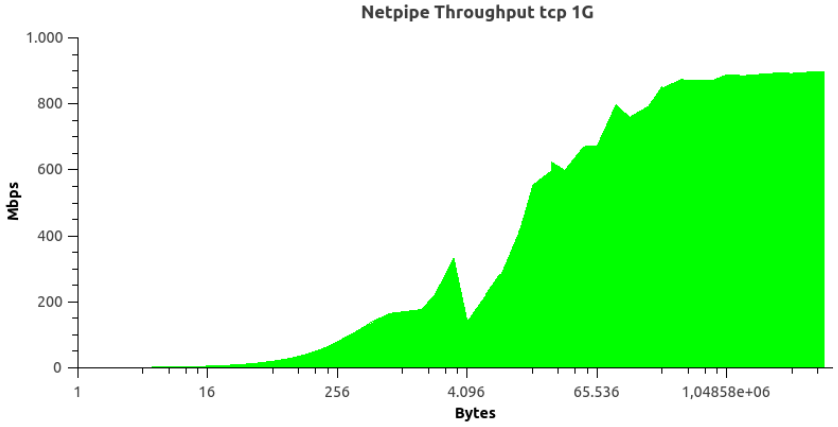


FIGURA 5.105: Netpipe medición de throughput TCP red 1G

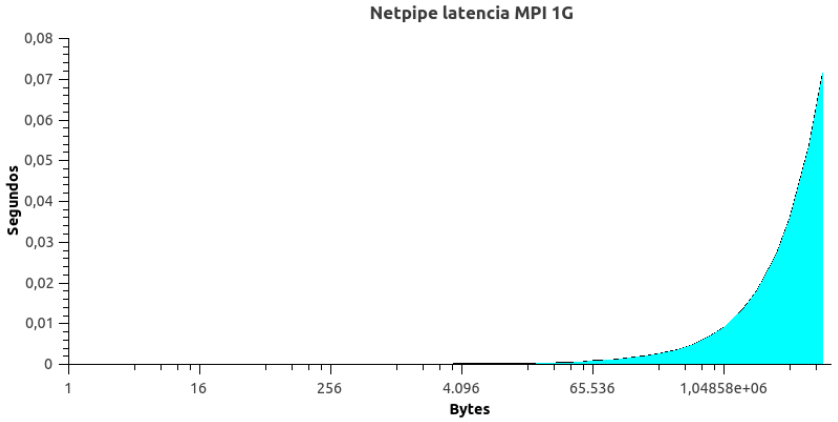


FIGURA 5.106: Netpipe medición de latencia MPI red 1G

5.9.3. Almacenamiento

5.9.3.1. Especificación y ajustes

El sistema de almacenamiento es el más complejo de analizar ya que múltiples sistemas se ven involucrados. Por un lado tenemos el sistema formado por la matriz de discos y sus volúmenes. A este sistema agregaremos la conexión con el servidor de cabecera, el multipathing, el lvm y el sistema de archivos, formando un nuevo sistema. Finalmente compartiremos el almacenamiento del servidor usando usando la red y el protocolo NFS formando un nuevo sistema que analizar. A todo esto también deberemos de agregar el sistema de almacenamiento local que tienen los nodos.

Matriz de discos La especificación y los ajustes referentes a la distribución de los discos en arrays se discutió en la parte dedicada al diseño del sistema de almacenamiento. En este momento llegamos a la siguiente configuración:

- Almacenamiento usuario:

  - Número de discos: 14
  - Tipo de RAID: RAID6
  - Tamaño chunk: 512KB

- Almacenamiento scratch
  - Número de discos: 8
  - Tipo de RAID: RAID 10
  - Tamaño chunk: 512KB

El tamaño de chunk que se eligió fue el valor por defecto de 512KB. Cuando tuve que definir esto tenía que poner el sistema en marcha lo antes posibles y probar distintas configuraciones de chunk sería muy costoso en tiempo (realizar RAIDs de tantos discos puede llevar más de un día de tiempo). Un tamaño grande de chunk, contra lo que parece ser más intuitivo, favorece los accesos aleatorios. Por lo tanto, aunque el tamaño de chunk grande penalizara los accesos secuenciales, esa ganancia no se vería en la práctica ya que el cuello de botella estaba claro que iba a ser la red. Todo esto sumado a que no encontré recomendaciones ni análisis por parte de HP hicieron que dejase el chunk en su valor por defecto.

Otro de los ajustes que había que realizar era la asignación de la controladora que iba a gestionar cada uno de los volúmenes. Aunque la controladora ofrezca una configuración de sistema activo-activo tal y como luego veremos en las observaciones no se trata de un sistema dual activo [18] y cada volúmen debe ser gestionado por una controladora (si esa controladora falla, la otra controladora asume automáticamente todos los volúmenes de gestionaba la otra controladora). Por lo tanto en nuestra configuración, como teníamos dos volúmenes y dos controladoras, asignamos un volumen a cada controladora (y así lo hacía el sistema por defecto).

Finalmente hay que tratar el sistema de memoria caché que ofrece la controladora. Este sistema afecta mucho al rendimiento proporcionado por el sistema. Los ajustes que se realizaron fueron:

- hacer que el modo de caché sea inmediato, esto hace que retorne inmediatamente el ok ante un sync del sistema operativo
- deshabilitar que el host tenga el control write-back de la caché. La matriz de discos lleva una batería y una compact flash para que en caso de fallo vuelque la información de la caché, de este modo podremos confiar en la integridad de los

[18] http://gestaltit.com/all/tech/storage/stephen/multipath-activepassive-dual-active-activeactive/

datos y hacer que de modo predeterminado todas las escrituras sean write-back. El write-through se habilitará automáticamente ante fallo en la alimentación de caché y ante fallo en la compact flash.

Conexión a nodo cabecera El primer elemento es la conexión con la matriz. Como ya se indicó, conectamos un cable desde cada controladora hacia un puerto de la HBA SAS del servidor. Para hacerla funcionar correctamente instalamos un driver más actualizado (en anexos) con el que pudiéramos configurar el multipath correctamente en nuestra controladora.

El siguiente elemento a tratar es el multipath. Como se observará durante las pruebas mediante el comando `iostat`, la comunicación con cada volumen realmente no se hace alternando entre dispositivos como cabría de esperar con una política round robin. Por lo tanto parecería tener más sentido una configuración activo activo con prioridad el dispositivo asignado en la controladora. Sin embargo mantendremos la configuración round robin porque es la indicada por el fabricante.

La siguiente capa con la que nos encontramos es el LVM. El principal motivo por el que elegimos configurar LVM fue por la flexibilidad que nos ofrece a costa de un overhead despreciable. El particionado lo realizamos usando la utilidad `parted` de forma automática ya que de este modo la propia utilidad hace el correcto alineado de las particiones.

En este apartado finalmente tenemos al sistema de archivos. Como ya se comentó, debido a que los volúmenes eran relativamente pequeños, no existía un caso de uso muy concreto sobre el que realizar un estudio de las alternativas y el tiempo disponible para hacerlo era escaso, se optó por EXT4 por ser un sistema de archivos muy generalista y el sistema oficialmente soportado con la distribución con las opciones por defecto en su creación. Lo que sí tunearemos un poco es en las opciones de montaje del sistema de archivos.

- vg_users: defaults,usrquota. Usamos la opciones por defecto y especificamos el uso de cuotas de usuario en el volumen.
- vg_scratch: noatime. En este caso configuramos para que no se actualice el inodo en cada acceso al disco lo cual penaliza el rendimiento.

Compartición NFS La versión del protocolo que usamos es la NFSv4 que hace uso de TCP como protocolo de transporte. Ya vimos que no agregamos la capa de autenticación y los mecanismos adicionales de integridad y encriptación que puede agregar kerberos ya que nos encontramos en nodos confiables, la red de almacenamiento es privada y además este mecanismo supondría un overhead adicional.

Los únicos ajustes que realizaremos en el servidor serán:

- subiremos el número de hilos en `/etc/default/nfs-kernel-server` la variable de entorno `RPCNFSDCOUNT=16`.
- fijaremos el modo de exportación de tipo async para el recurso scratch. En este modo no se realizarán las escrituras contra el almacenamiento en el caso de una solicitud sync de un cliente y por lo tanto acelerará el almacenamiento de ficheros de trabajo temporales. Desde el punto de vista del cliente, dejaremos que negocie automáticamente los valores de conexión ya que como vemos asigna los tamaños de búffers

En el lado del cliente, dejaremos que el cliente negocie los valores adecuados. Como vemos él solo fija los valores máximos a rsize y wsize:

```
lguillen@hpc-node1:~$ cat /proc/mounts
        ....
192.168.14.1:/homes/lguillen /home/CEFCA/lguillen nfs4 rw,relatime,vers=4.0,rsize
    =1048576,
wsize=1048576,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,
    clientaddr=192.
168.14.10,local_lock=none,addr=192.168.14.1 0 0
```

5.9.3.2. Mediciones

Para nuestro análisis, debido a que se va a trabajar mayormente sobre imágenes grandes, vamos a centrarnos exclusivamente en pruebas de acceso secuencial y a quedarnos en el nivel de filesystem. Podría llevarse un análisis más riguroso y exhaustivo pero desbordaría el alcance de este PFC.

Mediciones con hdparm La utilidad hdparm tiene por principal objetivo ajustar los parámetros de los discos IDE y SATA. Sin embargo, esta utilidad también tiene unas opciones de medición de rendimiento de lectura secuencial que el software realiza a nivel de bloque y que puede usarse para cualquier tipo de disco.

Los resultados obtenidos pueden verse en la figura hdparm. Como puede observarse, el rendimiento del volumen de usuarios es superior al del resto de los volúmenes. Es mayor que el volumen de scracth ya que tiene casi el doble de discos y en RAID6 lo penalizado son las escrituras. Llama la atención el elevado rendimiento del disco SSD que, con un único disco y en lectura secuencial, tenemos un rendimiento parecido a los 8 discos en RAID10. El último es el RAID1 del sistema operativo del nodo cabecera que se ha medido para tener una referencia más.

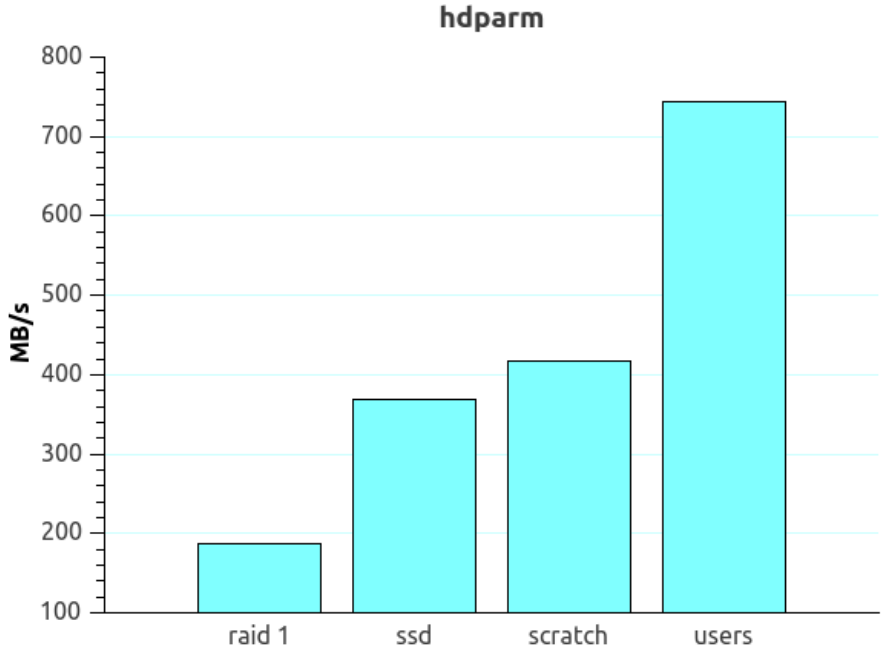


FIGURA 5.107: Mediciones con hdparm

Mediciones con dd La utilidad dd es una sencilla y muy conocida utilidad que realiza copia de bytes entre ficheros. Con esta utilidad realizaremos unas mediciones muy básicas de rendimiento de escritura y lectura secuencial. Para la escritura jugaremos con dos tipos: fdatasync y dsync. Con fdatasync al finalizar todo el proceso de copia se lanzará una llamada al sistema sync para forzar el volcado de la información que queda en búffers intermedios a disco mientras que con dsync se realizará un sync del búffer definido en dd en cada iteración de escritura.

En las pruebas de escritura definiremos un tamaño de búffer de 1M y realizaremos de forma secuencia la escritura de ese búffer 1024, 2048, 4096 y 8192 veces. El objetivo de coger valores tan grandes es tratar de llenar las cachés de las controladoras ya que para valores pequeños las escrituras son muy rápidas. Cada una de las pruebas la realizamos 5 veces y cogemos una media. Entre medición y medición esperamos unos minutos para que las controladoras escriban de manera asíncrona.

Podemos ver en la gráfica de la figura 5.108 que conforme escribimos más información en los arrays de discos de las matrices, estos tienen a bajar su throughput. Esto probablemente se deba a que se mitiga el efecto de las cachés de las controladoras.

En la figura 5.109, los arrays de discos se ven seriamente penalizados y están en un rendimiento similar al disco SSD. Como comenté al explicar la opción dsync, en cada iteración (en la que se realiza una escritura del búffer de 1M) se realiza una operación sync de forma que los datos deben ser escritos en disco. Cuando tratamos las cachés de

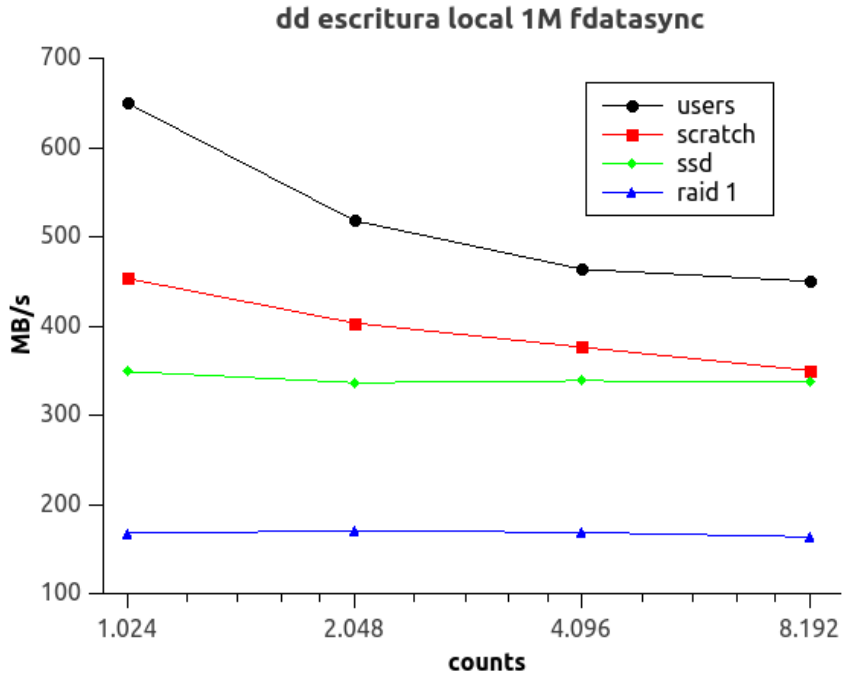


FIGURA 5.108: Escritura local con dd fdatasync

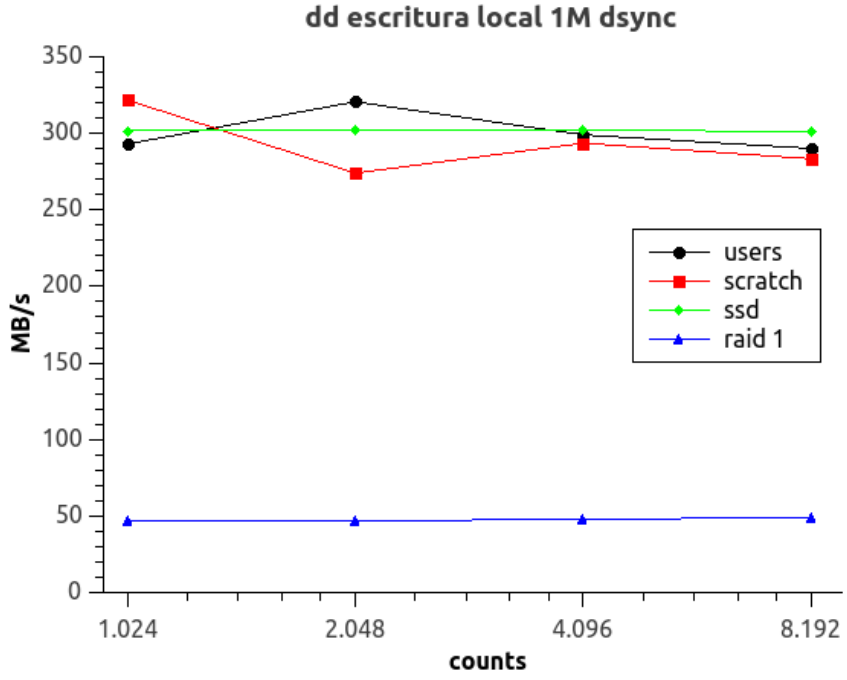


FIGURA 5.109: Escritura local con dd dsync

las controladoras una opción que mejoraba el rendimiento y que marcamos (debido a que disponemos del hardware necesario) era la de realizar escrituras asíncronas aunque el sistema pidiese una escritura síncrona. Sin embargo, para el número de iteraciones tan grande, y ante los resultados parece que esta opción no tiene efecto. Como vimos, este bajo rendimiento podría mejorarse redimensionando el tamaño del chunk de los arrays.

Las pruebas de lectura son un poco más complejas debido a todas las cachés que hay que tener en cuenta. En este caso lo que hacemos es escribir un fichero que será el que leamos posteriormente. A continuación escribimos otro fichero de 16G para tratar de llenar las cachés de las controladoras, realizamos la operación sync y esperamos unos minutos para esperar la escritura asíncrona de las controladoras y finalmente forzaremos al kernel a borre todos los búffers y cachés. En ese momento ya podemos medir la lectura secuencial con dd. En este caso también se realiza 5 veces y se realiza usando un búffer de lectura de 1M y el número de iteraciones visto para la escritura.

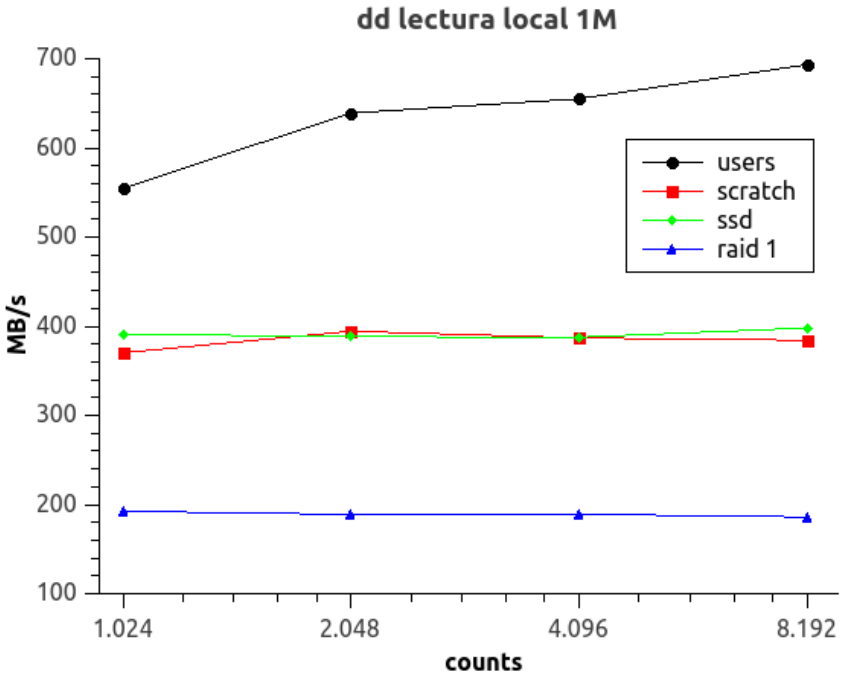


FIGURA 5.110: Lectura local con dd

Como podemos ver en la gráfica de la figura 5.110, para los arrays obtenemos unos resultados peores que con hdparm. Aunque estamos midiendo diferentes cosas ya que con el dd estamos pasando por la capa del filesystem, podemos comparar relativamente con los resultados con los obtenidos con las otras mediciones. Necesitaríamos realizar un estudio más exhaustivo para poder identificar la causa.

Finalmente mediremos el acceso al almacenamiento vía NFS desde uno de los nodos de cómputo. En este caso vamos a mostrar únicamente las escrituras.

Si observamos las figuras 5.111 y 5.109, observaremos en ambas cómo la escritura sobre el volumen scratch es mejor que la escritura sobre el volumen de usuarios. Esto se debe a la opción async definida en la exportación del sistema de archivos en el servidor cabecera. Podemos comprobar que el rendimiento en la escritura nfs en el caso fdatasync es el

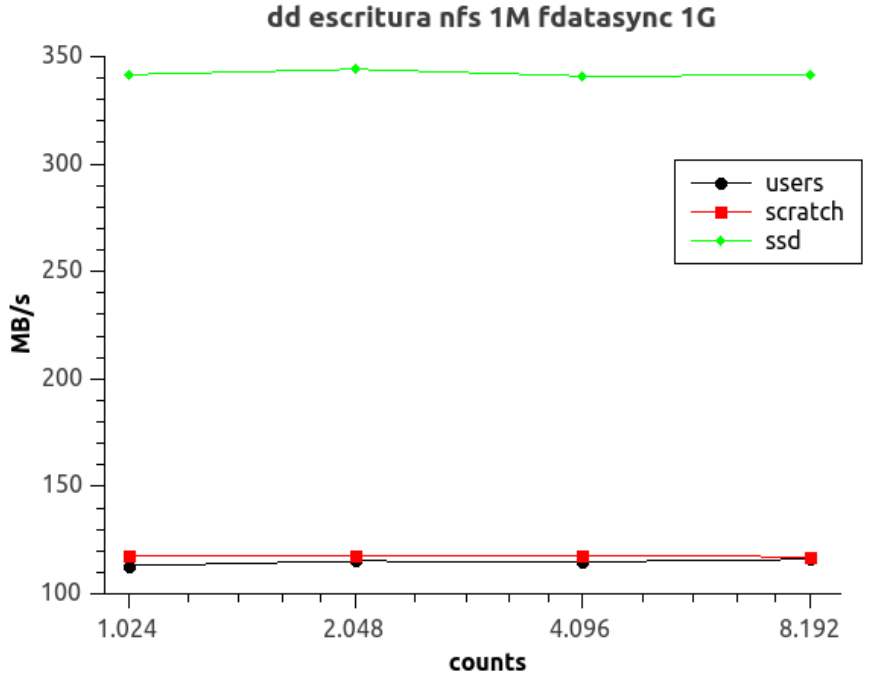


FIGURA 5.111: Escritura nfs con dd fdatasync

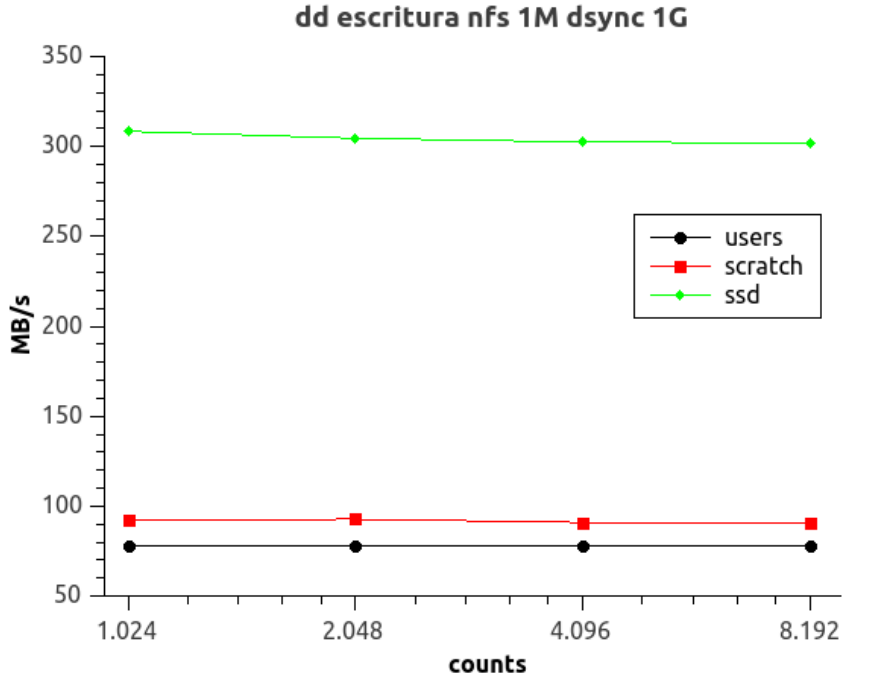


FIGURA 5.112: Escritura nfs con dd dsync

rendimiento que ofrece la red y mientras que en el caso de dsync es inferior debido al tráfico NFS generado adicionalmente por las operaciones de sincronización.

5.9.4. Software

5.9.4.1. Especificación y ajustes

Como vimos durante el proceso de especificación definimos que, además de las librerías científicas incluidas en el sistema disponibles en nuestra distribución *ubuntucefca14*, ofreceríamos diferentes alternativas de librerías que estarían optimizadas para el hardware del clúster. Ya vimos que el software EasyBuild agrupaba el conjunto compiladores y librerías en lo que denominaba *toolchain*.

Toolchain goalf

En el caso del toolchain goalf (ATLAS, BLACS, FFTW, GCC, OpenMPI, ScaLAPACK) la versión es la 1.5.12-no-OFED.

- GCC: 4.8.1
- OpenMPI: 1.6.5
- ATLAS: 3.10.1
- LAPACK: 3.4.2
- FFTW: 3.3.3
- ScaLAPACK: 2.0.2

Toolchain goolf

En el caso del toolchain goolf (BLACS, FFTW, GCC, OpenBLAS, OpenMPI, ScaLAPACK) la versión es la 1.5.14-no-OFED que incluye:

- GCC: 4.8.2
- hwloc: 1.8.1
- OpenMPI: 1.6.5
- OpenBLAS: 0.2.9
- FFTW: 3.3.4
- ScaLAPACK: 2.0.2

Para la compilación de ambos toolchains se dejaron los ajustes predefinidos en cada uno de los ficheros de EasyBuild usados (salvo la compilación de OpenMPI que fue necesario agregar un flag para el soporte del gestor de recursos SGE). En estos ajustes se llama siempre al compilador GCC con la opción `-march=native`. Esta opción indica al compilador GCC que detecte la arquitectura y características que ofrece (extensiones SSE, AVX...) y genera código específico.

Antes de realizar las mediciones cabe destacar que para la compilación de las librerías ATLAS en el toolchain goalf, los makefiles de instalación realizan un montón de pruebas para ver los tipos de caché, etc. De este modo se definen parámetros necesarios en tiempo de compilación especialmente optimizados para la máquina. Esto provoca que la ejecución de programas linkados contra esta librería sólo pueden realizarse en las máquinas con una arquitectura y organización idéntica a la que se compiló el software. Esto hace que no puedan ejecutarse usando la versión optimizada en el nodo de login. Otro apunte a destacar es que la librería OpenBLAS es más moderna que las ATLAS y como se puede ver, hace uso de la librería hwloc que se usa para obtener información sobre la organización de cores y memoria de la máquina y fijar la afinidad de los hilos en tiempo de ejecución.

5.9.4.2. Mediciones

Benchmarks librerías Python Vamos a realizar una pequeña comparación entre las librerías disponibles para el lenguaje python disponibles en nuestro clúster más significativas: numpy y scipy. Para ello nos valdremos de unos pequeños problemas encontrados por la red que pueden usarse a modo de benchmark [19].

El primero de los benchmark utiliza dos matrices de orden 10000 y realiza las operaciones de producto de matrices. Puede observarse en las figuras 5.113 y 5.114 cómo existe una diferencia significativa entre las versiones suministradas por la librerías optimizadas y la librería proporcionada por el sistema.

En el segundo de los benchmarks también se realiza un producto de matrices, pero en este caso se ve cómo afecta el orden de las matrices involucradas al tiempo de resolución de los mismos. Este benchmark realiza un cálculo en gigaflops a partir de los tiempos obtenidos. Vemos en la figura test_numpy2 la gráfica comparativa entre las distintas versiones.

En último de estos benchmarks de librerías de python, realizaremos un par de pruebas de la librería scipy sobre una matriz cuadrada de orden 1000. La gráfica de la figura

[19] http://gromgull.net/blog/2013/07/multithreaded-scipynumpy-with-openblas-on-debian/

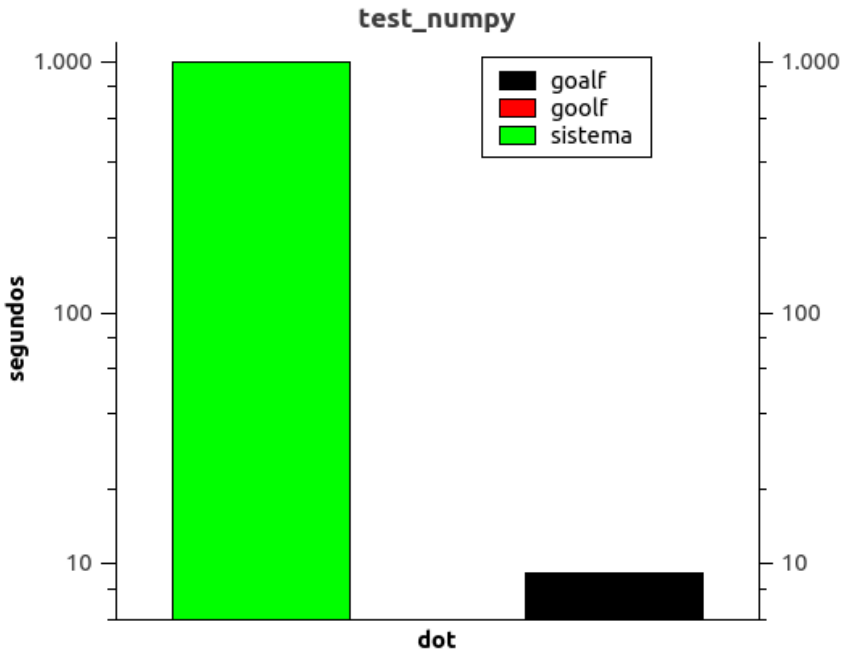


FIGURA 5.113: Benchmark test_numpy dot

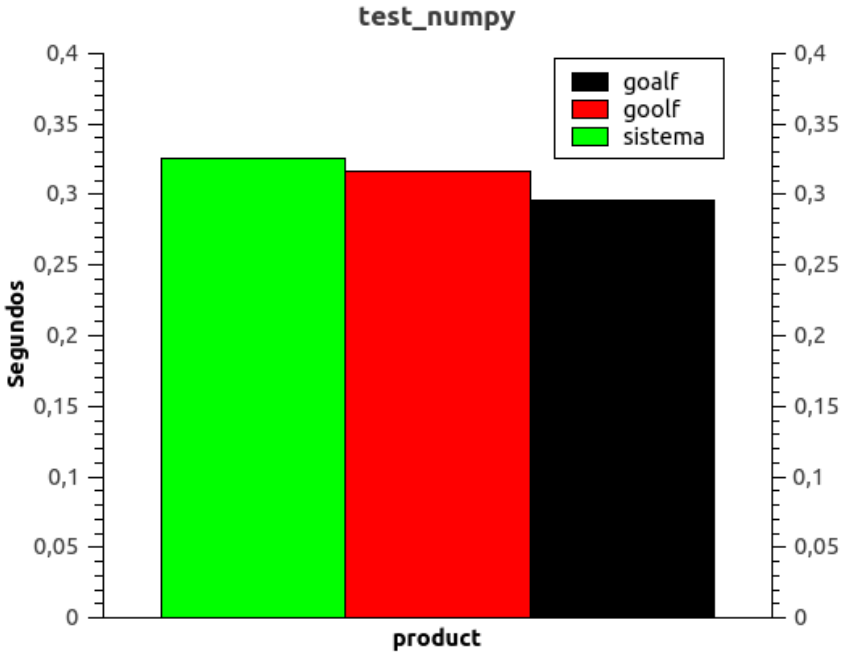


FIGURA 5.114: Benchmark test_numpy product

5.116 es una descomposición de sobre dicha matriz cholesky y la de la figura 5.117 una descomposición en valores singulares.

Benchmark R A continuación realizaremos un benchmark que hay para R [20] y compararemos la eficiencia de las librerías en goolf contra las del sistema. A continuación

[20] http://r.research.att.com/benchmarks/

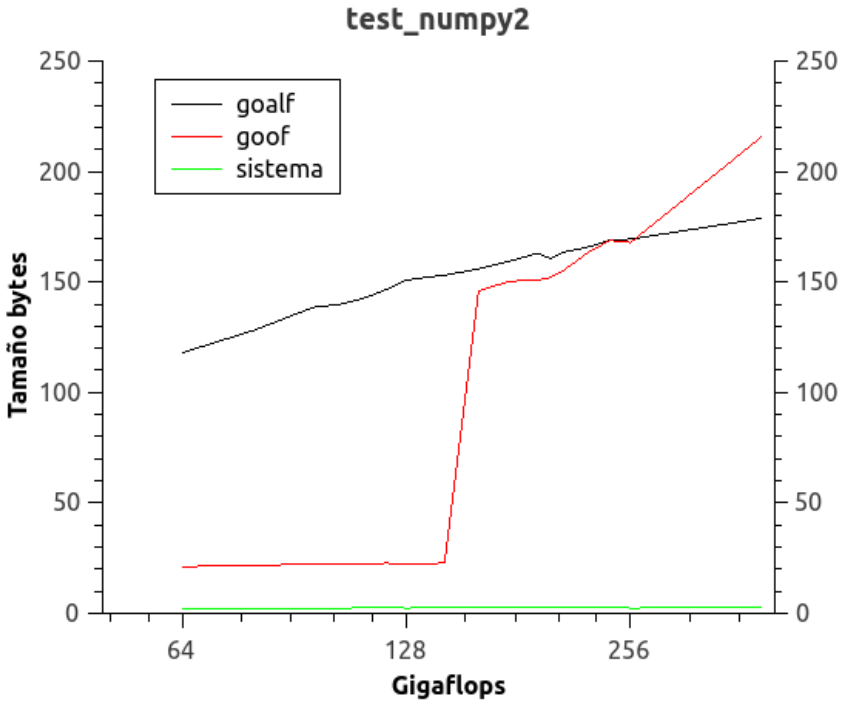


FIGURA 5.115: Benchmark test_numpy2

muestro los resultados obtenidos.

Resultados para las librerías del sistema

```
   R Benchmark 2.5
   ===============
Number of times each test is run___________________________:  3

   I. Matrix calculation
   ---------------------
Creation, transp., deformation of a 2500x2500 matrix (sec):  0.876
2400x2400 normal distributed random matrix ^1000____ (sec):  0.606333333333333
Sorting of 7,000,000 random values__________________ (sec):  0.694333333333333
2800x2800 cross-product matrix (b = a' * a)_________ (sec):  10.9736666666667
Linear regr. over a 3000x3000 matrix (c = a \ b')___ (sec):  5.18966666666666
                     ------------------------------------------
                 Trimmed geom. mean (2 extremes eliminated):  1.46691132601269

   II. Matrix functions
   --------------------
FFT over 2,400,000 random values____________________ (sec):  0.283000000000006
Eigenvalues of a 640x640 random matrix______________ (sec):  0.849666666666669
Determinant of a 2500x2500 random matrix____________ (sec):  3.752
Cholesky decomposition of a 3000x3000 matrix________ (sec):  4.31233333333333
Inverse of a 1600x1600 random matrix________________ (sec):  2.90433333333333
                     ------------------------------------------
                 Trimmed geom. mean (2 extremes eliminated):  2.0998388019711

   III. Programmation
   ------------------
```

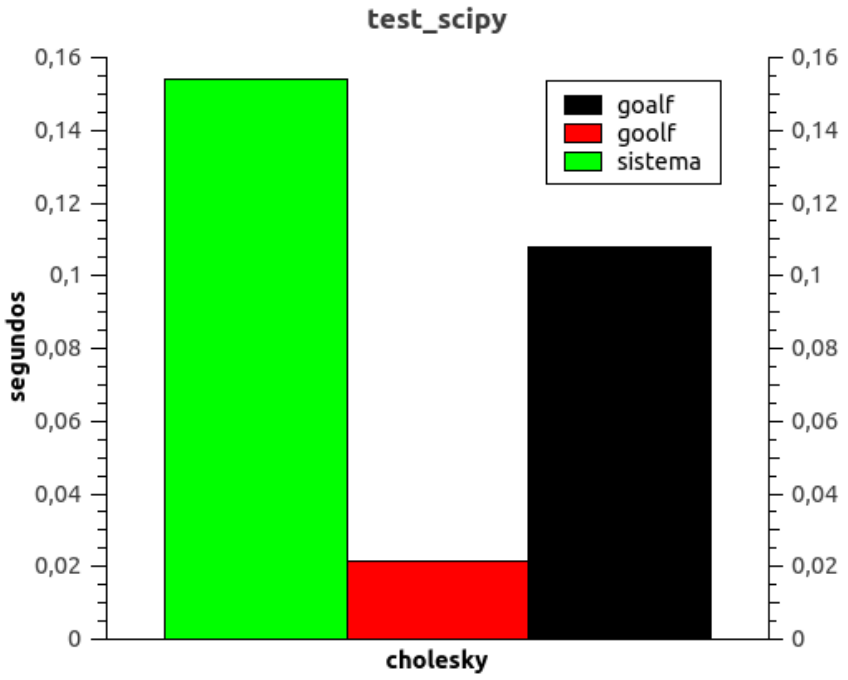


FIGURA 5.116: Benchmark test_scipy cholesky

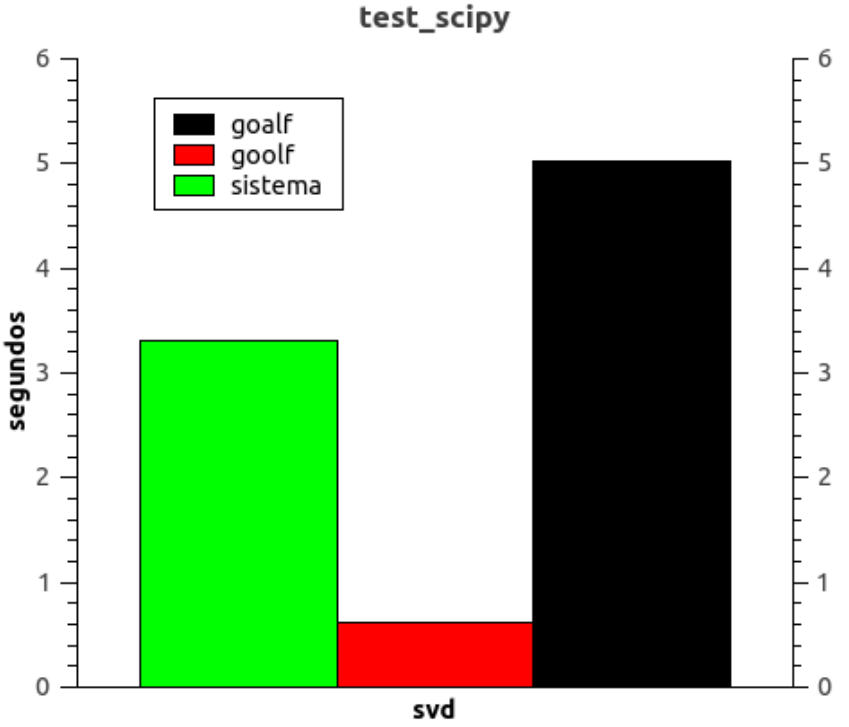


FIGURA 5.117: Benchmark test_scpy svd

```
3,500,000 Fibonacci numbers calculation (vector calc)(sec):  0.692666666666658
Creation of a 3000x3000 Hilbert matrix (matrix calc) (sec):  0.265666666666656
Grand common divisors of 400,000 pairs (recursion)__ (sec):  1.00833333333333
Creation of a 500x500 Toeplitz matrix (loops)_______ (sec):  0.63100000000001
Escoufier's method on a 45x45 matrix (mixed)________ (sec):  0.477000000000004
                    ---------------------------------------------
```

```
                Trimmed geom. mean (2 extremes eliminated):  0.592958103940229


Total time for all 15 tests_____________________________ (sec):  33.516
Overall mean (sum of I, II and III trimmed means/3)_ (sec):  1.22237542208301
                      --- End of test ---
```

Resultados para el toolchain goolf

```
   R Benchmark 2.5
   ===============
Number of times each test is run____________________________:  3

   I. Matrix calculation
   ---------------------
Creation, transp., deformation of a 2500x2500 matrix (sec):  0.823666666666667
2400x2400 normal distributed random matrix ^1000____ (sec):  0.609333333333334
Sorting of 7,000,000 random values__________________ (sec):  0.76
2800x2800 cross-product matrix (b = a' * a)_________ (sec):  0.104333333333334
Linear regr. over a 3000x3000 matrix (c = a \ b')___ (sec):  0.136666666666666
                      ---------------------------------------------
                 Trimmed geom. mean (2 extremes eliminated):  0.398514116842327

   II. Matrix functions
   --------------------
FFT over 2,400,000 random values____________________ (sec):  0.270666666666666
Eigenvalues of a 640x640 random matrix______________ (sec):  1.711
Determinant of a 2500x2500 random matrix____________ (sec):  0.198000000000003
Cholesky decomposition of a 3000x3000 matrix________ (sec):  0.125000000000001
Inverse of a 1600x1600 random matrix_____________ (sec):  0.453000000000001
                      ---------------------------------------------
                Trimmed geom. mean (2 extremes eliminated):  0.289556102490364

   III. Programmation
   ------------------
3,500,000 Fibonacci numbers calculation (vector calc)(sec):  0.594666666666664
Creation of a 3000x3000 Hilbert matrix (matrix calc) (sec):  0.205333333333333
Grand common divisors of 400,000 pairs (recursion)__ (sec):  0.893333333333336
Creation of a 500x500 Toeplitz matrix (loops)_______ (sec):  0.350999999999999
Escoufier's method on a 45x45 matrix (mixed)________ (sec):  0.353999999999999
                      ---------------------------------------------
                Trimmed geom. mean (2 extremes eliminated):  0.419624971349528


Total time for all 15 tests_____________________________ (sec):  7.59
Overall mean (sum of I, II and III trimmed means/3)_ (sec):  0.364484657844005
                      --- End of test ---
```

5.9.5. Sistema clúster HPC

5.9.5.1. Mediciones de rendimiento

Benchmark Linpack Ya usamos la implementación HPL del benchmark de Linpack para medir el rendimiento de un nodo. A continuación mediremos el rendimiento de nuestro clúster. Para la definición del fichero de parámetros HPL.dat, usaremos de nuevo la calculadora online que usamos para el nodo pero especificando que esta vez tenemos 5 nodos. En este caso ejecutaremos únicamente el benchmark con el HT activo (sin ningún parámetro adicional) para una configuración de OpenBLAS de 1, 2 y 4 threads.

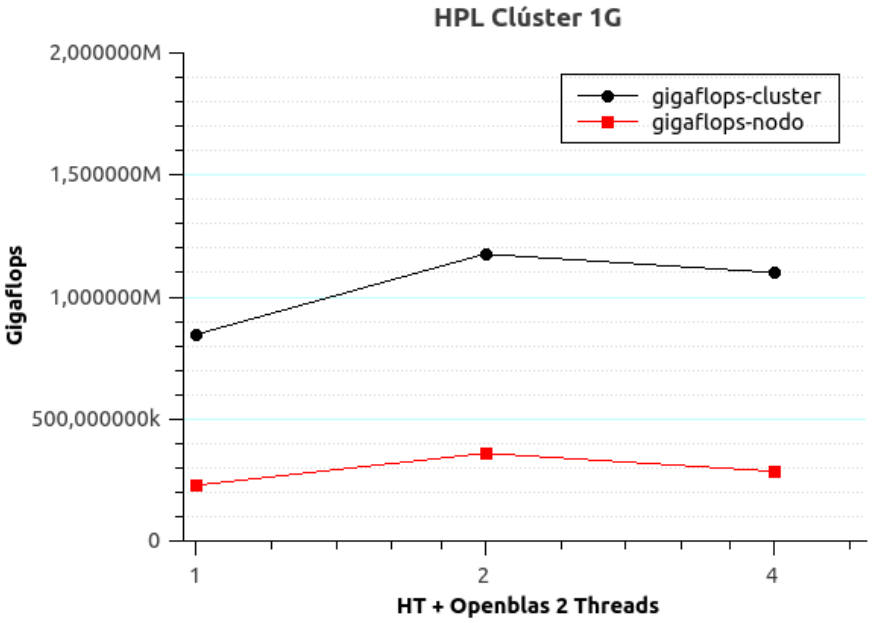


FIGURA 5.118: Ejecución hpl en clúster 1G

Tras la ejecución, podemos ver en la gráfica 5.118 una comparativa entre lo que se obtuvo para un nodo y para los 5 nodos del clúster. Como puede observarse el rendimiento dista de ser un factor 5 sobre el que obtuvimos para un nodo. Obtenemos un máximo de 1'176 Teraflops que es un 58 % sobre el rendimiento máximo de 2 Teraflops que podría dar el clúster.

Benchmark gadget-3 Para finalizar el estudio del rendimiento del clúster usaremos un benchmark especializado similar al tipo de problemas que van a usar las personas que trabajan en el área de cosmología. El benchmark es *gadget-3*, que es una versión más nueva sobre la que existente en la red [21] y en la que colaboró uno de los científicos que se encuentran entre el personal de CEFCA. Este benchmark es un benchmark

[21] http://www.mpa-garching.mpg.de/gadget/

que realiza simulaciones cosmológicas y está diseñado para ser usado en computadores masivamente paralelizados con memoria distribuida. El problema que resuelve este benchmark está muy bien paralelizado, por lo que agregar nuevas unidades de cómputo al proceso hace que el tiempo total necesario para resolver el problema disminuya (obviamente hasta el límite definido en la ley de Amdhal), por eso es también un buen medidor de la escalabilidad del clúster.

Existen dos componentes en el benchmark que vamos a usar, el primero es N-GenIC con el que se genera el universo de partida y el segundo es P-Gadget3 que es el software que realiza las simulaciones. No voy a entrar en cómo se parametriza el benchmark, simplemente comentaré que con el componente N-GenIC se se generaron dos universos: small (16 ficheros de 61 MB) y medium (16 ficheros de 512 MB). Luego se compiló usando OpenMPI, GSL y FFTW 2 el componente P-Gadget3 para su uso de 1, 2, 4 y 8 PosixThreads. Todos los benchmarks se realizarán con el hyperthreading habilitado.

La primera de las pruebas que haremos será ejecutar el benchmark small agregando nuevos nodos, añadiendo con cada nodo 20 procesos MPI al cómputo. Además se probarán las distintas versiones compiladas, aumentando el número de hilos PosixThreads usados por cada proceso MPI. Como puede observarse en la gráfica de la figura 5.119, en lo que respecta a los threads el uso de 2 y 4 hilos generan los mejores resultados. Lo más importante es observar cómo se comporta el problema ante la agregación de nuevos procesos MPI que se encuentran en nodos distintos y que por lo tanto generan comunicación. Vemos que cuando agregamos el segundo nodo (40 MPI), el problema tarda menos en resolverse, pero cuando agregamos más nodos el tiempo de resolución de el mismo problema se dispara.

A la vista de los resultados anteriores, en la siguiente prueba trataremos de centrarnos aún más en analizar el factor de comunicación. Para ello usaremos el mismo problema para los mismos 20 procesos MPI, pero en este caso distribuiremos los procesos entre los nodos, esto es: 1 nodo (20 procesos), 2 nodos (10 procesos y 10 procesos) y 4 nodos (5 procesos, 5 procesos, 5 procesos y 5 procesos). De este modo podemos comparar cómo afecta la comunicación al proceso. Los resultados obtenidos en esta prueba los analizaremos y veremos gráficamente cuando analizaremos la comparación con la actualización de la red a 10G.

Para finalizar las pruebas, realizamos el benchmark de tamaño medium empleando un único thread y usando todos los nodos del clúster. Para el último paso del benchmark obtuvimos los siguientes resultados:

```
Step 3, Time: 0.0204637, MPI-Tasks: 100  Threads: 1
total            1358.62  100.0%
treegrav          391.48   28.8%
```

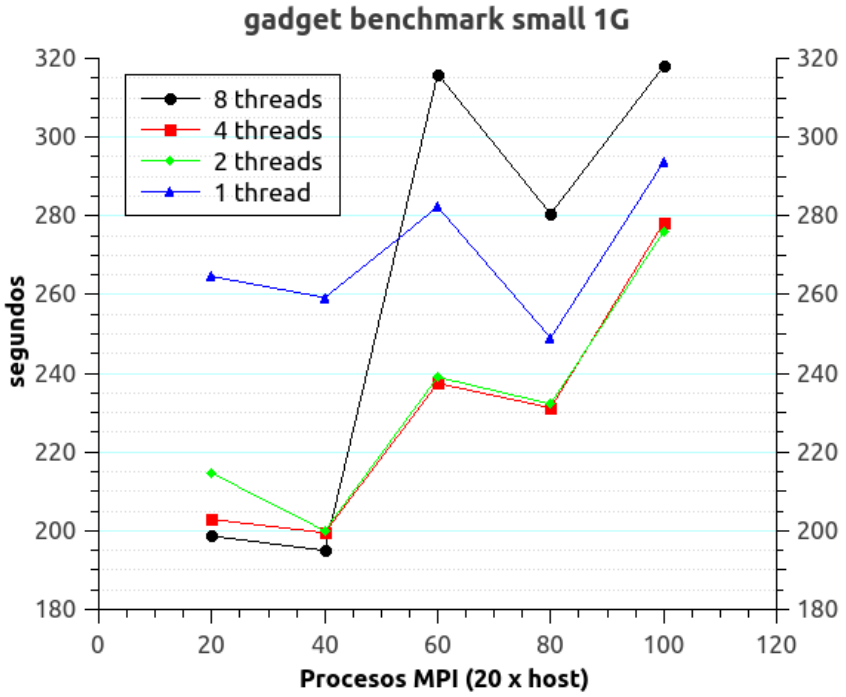


FIGURA 5.119: Ejecución gadget3 small en clúster 1G

```
   treebuild        13.06    1.0%
   treeupdate        0.00    0.0%
   treewalk        186.58   13.7%
   treecomm         55.54    4.1%
   treeimbal       127.64    9.4%
pmgrav             452.53   33.3%
sph                239.63   17.6%
   density          41.51    3.1%
   denscomm         32.92    2.4%
   densimbal        25.26    1.9%
   hydrofrc         49.48    3.6%
   hydcomm          46.78    3.4%
   hydmisc           5.10    0.4%
   hydnetwork        0.00    0.0%
   hydimbal         33.76    2.5%
   hmaxupdate        0.48    0.0%
domain             264.93   19.5%
potential            0.00    0.0%
predict              0.00    0.0%
kicks                0.70    0.1%
i/o                  0.00    0.0%
peano                3.06    0.2%
sfrcool              0.00    0.0%
blackholes           0.00    0.0%
fof/subfind          0.00    0.0%
smoothing            0.00    0.0%
hotngbs              0.00    0.0%
weights_hot          0.00    0.0%
enrich_hot           0.00    0.0%
weights_cold         0.00    0.0%
```

```
enrich_cold          0.00     0.0%
cs_misc              0.00     0.0%
misc                 6.29     0.5%
```

5.10. Actualización de la solución a Ethernet 10G

5.10.1. Especificación

5.10.1.1. Necesidades de actualización

Como pudo verse durante las mediciones del sistema, el talón de Aquiles del clúster era la red. Como vimos no era algo inesperado, sino que ya se conocía cuando se diseñó el sistema y se preparó para una posible actualización.

La baja velocidad de la red no sólo influía al cómputo de problemas de cosmología que hacían uso de MPI, también vimos que limitaba el acceso al almacenamiento. En la imagen 5.120 puede verse un instante de ejecución de un problema de análisis de imágenes cómo el tiempo que roba la espera de acceso al almacenamiento a la CPU pueden llegar hasta el 20 %.

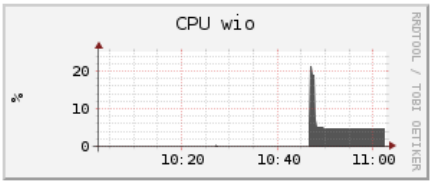


Figura 5.120: Contención al almacenamiento

Por ello y ante la disponibilidad de algunos recursos se decidió que se realizaría una actualización del clúster. El objetivo principal sería la actualización de la red aunque también había más necesidad de cómputo. Además se agregaba la circunstancia de que el chassis tenía todavía dos slots libres y si no se completaba con equipos compatibles (generación 8 de hp) transcurrido el tiempo no podría ampliarse y sería más cara la adquisición de otro equipamiento. Debido a esta circunstancia el proveedor de la solución debía ser exclusivamente HP o sus partners, por lo que la actualización incluiría dos partidas:

- Adquisición de equipo de comunicaciones y cableado necesario
- Adquisición de los nodos restantes para completar el chasis y tarjetas de networking para los nodos existentes

Para la definición de las necesidades de comunicaciones se estuvo barajando entre Infiniband o 10G. La tecnología que mejores prestaciones ofrecía era Infiniband, pero se estuvo sondeando el mercado y se observó que tenía un coste superior. Además del coste de adquisición habría que sumar otros costes como el de formación técnica en dicha tecnología que únicamente para ser capaz de realizar una instalación básica podría llevar entre 30 y 40 horas. A esto habría que sumar la configuración y recompilación de todo el software que podría llevar unas 10 horas de trabajo.

Para complementar la toma de decisiones, se hicieron unas pruebas de benchmarking de problemas de cosmología en otros HPC que disponían de Infiniband para ver cuánto era la mejora. El rendimiento era muy superior al que se obtenía en el clúster con 1G pero dado que la mayoría de la comunicación eran transferencias de memoria y la latencia no era un elemento tan crítico y se estimó que la actualización a 10G sería suficiente.

Otro elemento a resolver era el tema del cableado. En tecnología 10G los adaptadores suelen venir con formato SFP+, esto hace que sea necesario recurrir a transceivers que hagan la conversión de medio. Dado que el switch iba a estar en el mismo rack que el equipamiento y la adquisición de dichos transceivers incrementa sustancialmente el precio, se decidió que se adquiriría cableado directo.

5.10.1.2. Proceso de selección y adquisición del equipamiento

Una vez decidido que se optaría por 10G se estuvieron viendo equipos que pudiesen satisfacer los requerimientos y se actuó conforme a la ley, pidiendo presupuestos a varios proveedores de Cisco, Mellanox y HP. En lo que respecta al cableado, Mellanox ofrecía puertos QSF+ para los que era necesario cables splitters a SFP+ mientras que Cisco y HP ofrecían puertos SFP+ lo que permitía cable directo. Tras analizar los presupuestos obtenidos se observó que:

- El switch Cisco era un factor 2 el precio y encima con un menor número de puertos.
- El switch de Mellanox era superior en prestaciones (especialmente en cuanto a latencia y ofrecía nivel 3) pero tenía un coste superior al de HP.
- El switch de HP era el más barato y además se trataba del mismo proveedor (evitando problemas de compatibilidades).

Finalmente el precio fue el factor determinante en el proceso, por lo que se adquirió la propuesta de HP.

En lo que respecta a los nodos, se definieron los mismos modelos HP Proliant SL230 gen8. En cuanto a las tarjetas se decidió que se adquirirían tarjetas en formato LOM que sustituirían a las de 4 puertos de Gigabyte, de modo que se dejaría el slot pci express libre por si hubiese ampliaciones en un futuro y deberían tener dos puertos, uno para la red de cómputo y otro para la red de almacenamiento.

5.10.1.3. Especificaciones finales del hardware

Tarjetas de red

- Nodo cabecera: HP NC552SFP 10GbE 2P Svr Adapter
- Nodos de cómputo: HP Ethernet 10Gb 2P 530FLR-SFP+ Adapter

Equipo comunicaciones HP FexFabric 5700-40XG-2QSFP+ Switch

- 40 puertos 10G formato SFP+
- 2 puertos QSFP 40G de uplinks
- 1 puerto gestión 1G
- Fuente redundante

Figura 5.121: HP FexFabric 5700-40XG-2QSFP+ Switch

Cableado Cables DAC HP X240 10G SFP+ SFP+ 3m DAC Cable

Figura 5.122: Cables DAC SFP+

5.10.2. Diseño de la red

El switch 10G será utilizado para las redes de almacenamiento y cómputo. Como las redes ambas redes estarán completamente implementadas en este switch no será necesaria la conexión de uplinks con otros switch. La única conexión existente será la conexión de la interfaz de gestión de 1G existente en el switch a una boca en el switch Cisco 3750X.

Las redes general y de gestión, todo quedará como está cableado al switch 3750X. En lo que respecta a las redes de almacenamiento y cómputo procederemos a quitar todo el cableado y se eliminar su configuración del switch Cisco 3750X.

En el nuevo switch definiremos las dos vlanes:

- vlan 14: hpc_almacenamiento
- vlan 15: hpc_mpi

Como se vio durante la parte de especificación, las tarjetas disponen de dos puertos. Dichos puertos se conectarán y configurarán en el switch del siguiente modo:

- Nodo cabecera: Puertos 1 y 2 en red de almacenamiento con Bonding.

- Nodos de cómputo: Puerto 1 red de cómputo y puerto 2 red de almacenamiento

En lo que respecta a las direcciones de red, todo quedará como estaba, únicamente será necesario agregar la configuración ip en la red de gestión del switch:

- IP Gestión switch HP: 10.10.10.27

5.10.3. Instalación y configuración

5.10.3.1. Enrackado y cableado

Como se comentó durante el proceso de instalación hubo algunos problemas en cuanto al acondicionamiento del chasis en el rack. Debido a que era necesario utilizar el rack Dell para migrar a él los servidores de la infraestructura CEFCA, se adquirió finalmente un rack HP en el cual se instalaría el clúster y sería el lugar para futuras ampliaciones. Por lo tanto se aprovechó la parada para migrar al nuevo rack, para lo que hubo que realizar una planificación del rack previa.

En la figura 5.123 se ve el proceso de migración del y la instalación y cableado del nuevo switch.

5.10.3.2. Switch 10G

Interfaz de configuración HP La interfaz de configuración de HP es una interfaz CLI. Para el acceso a la misma puede realizarse vía un puerto de consola serie o por red empleando algún protocolo de terminal remota. Como el sistema no tiene configurada la interfaz de gestión es necesario realizar la primera parte de la configuración vía consola serie. Para ello se utilizó el software *minicom* de linux con los parámetros de configuración presentes en la documentación de HP.

Una vez iniciada sesión vía consola serie o vía red la interfaz presenta un prompt. Dispone de dos modos, el modo de vista de usuario y el modo de vista de sistema. En el modo de vista de usuario sólo puede verse el estado del switch y de la configuración. Para cambiar al modo de vista de sistema recurriremos al comando `system-view` y para grabar la configuración al comando `save`.

```
<cefca-swt-014>system-view
System View: return to User View with Ctrl+Z.
[cefca-swt-014]save
The current configuration will be written to the device. Are you sure? [Y/N]:y
Please input the file name(*.cfg)[flash:/startup.cfg]
```

Figura 5.123: Migración de rack y upgrade a 10G

```
(To leave the existing filename unchanged, press the enter key):
Validating file. Please wait...
Saved the current configuration to mainboard device successfully.
```

Configuración del hostname Configuraremos el nombre del equipo.

```
<HP>system-view
System View: return to User View with Ctrl+Z.
[HP]sysname cefca-swt-014
```

Configuración del ventilador El sistema daba errores, como puede verse a continuación.

```
%Jan  2 16:51:31:695 2011 cefca-swt-014 DEV/1/FAN_DIRECTION_NOT_PREFERRED: Fan 2
    airflow direction.
%Jan  2 16:52:51:694 2011 cefca-swt-014 DEV/1/FAN_DIRECTION_NOT_PREFERRED: Fan 1
    airflow direction.
%Jan  2 16:52:51:695 2011 cefca-swt-014 DEV/1/FAN_DIRECTION_NOT_PREFERRED: Fan 2
    airflow direction.
<cefca-swt-014>display Fan
 Slot 1
          FAN    1
          State    : FanDirectionFault
          Wind Direction    :Port-to-Power
          Prefer Wind Direction    :Power-to-Port
          FAN    2
          State    : FanDirectionFault
          Wind Direction    :Port-to-Power
          Prefer Wind Direction    :Power-to-Port
```

Para solucionarlo configuraremos la dirección del ventilador adelante atrás.

```
<cefca-swt-014>system-view
System View: return to User View with Ctrl+Z.
[cefca-swt-014]fan prefer-direction slot 1 port-to-power
```

Configuración gestión Para poder gestionar y monitorizar el switch es necesario que configuremos la red de gestión.

```
<cefca-swt-014>system-view
[cefca-swt-014]interface M-GigabitEthernet 0/0/0
[cefca-swt-014]ip address 10.10.10.27
[cefca-swt-014]undo shutdown
```

Configuración del ntp Habilitaremos la configuración ntp, haciendo que sincronicemos el switch contra el servidor de tiempo de la red de gestión.

```
[cefca-swt-014]ntp-service enable
[cefca-swt-014]ntp-service unicast-server 10.10.10.50
[cefca-swt-014]display ntp-service status
 Clock status: synchronized
 Clock stratum: 5
 System peer: 10.10.10.50
 Local mode: client
 Reference clock ID: 10.10.10.50
 Leap indicator: 00
 Clock jitter: 0.000031 s
 Stability: 0.000 pps
 Clock precision: 2^-17
 Root delay: 58.71582 ms
 Root dispersion: 161.04126 ms
 Reference time: d919519f.8667d760  Wed, Jun  3 2015 10:12:47.525

[cefca-swt-014]clock protocol ntp
```

Configuración servicio ssh Habilitaremos la configuración del servicio ssh para acceder a la gestión del switch,

```
[cefca-swt-014]ssh server enable
[cefca-swt-014]public-key local create rsa
The range of public key modulus is (512 ~ 2048).
If the key modulus is greater than 512, it will take a few minutes.
Press CTRL+C to abort.
Input the modulus length [default = 1024]:
Generating Keys...
....................o
....................+
....................++++
..................++++++
Create the key pair successfully.
[cefca-swt-014]user-interface vty 0 15
[cefca-swt-014-line-vty0-15]authentication-mode scheme
[cefca-swt-014-line-vty0-15]protocol inbound ssh
[cefca-swt-014-line-vty0-15]quit
```

Creación de usuarios locales de administración Crearemos un usuario local de administración y le daremos acceso ssh.

```
[cefca-swt-014]local-user adminuser
New local user added.
[cefca-swt-014-luser-manage-adminuser]password simple lapassword
[cefca-swt-014-luser-manage-adminuser]service-type ssh
[cefca-swt-014-luser-manage-adminuser]authorization-attribute user-role network-
    admin
[cefca-swt-014-luser-manage-adminuser]quit
```

Configuración agente SNMP Configuraremos el agente SNMP para que podamos realizar la monitorización.

```
[cefca-swt-014]snmp-agent community read CEFC@_RO
[cefca-swt-014]snmp-agent sys-info version v2c
```

Configuración VLAN Realizaremos la configuración de las vlanes que tendrá el switch.

```
<cefca-swt-014>system-view
System View: return to User View with Ctrl+Z.
[cefca-swt-014]vlan 13
[cefca-swt-014-vlan13]description Red_MPI
[cefca-swt-014-vlan13]quit
[cefca-swt-014]vlan 14
[cefca-swt-014-vlan14]description Red_Storage
[cefca-swt-014-vlan14]quit
```

Configuración de los puertos de red En primer lugar configuraremos los puertos, para ello será necesario darle una descripción, habilitar el puerto y los jumboframes.

```
[cefca-swt-014]interface Ten-GigabitEthernet 1/0/2
[cefca-swt-014-Ten-GigabitEthernet1/0/2]description ; cefca-sv-012 ; te0/1 ;
    Almacenamiento hpc-master
[cefca-swt-014-Ten-GigabitEthernet1/0/2]undo shutdown
[cefca-swt-014-Ten-GigabitEthernet1/0/2]jumboframe enable
```

Después en la configuración de vlan agregaremos los puertos de red corresondientes.

```
[cefca-swt-014]vlan 13
[cefca-swt-014-vlan13]port Ten-GigabitEthernet 1/0/6 Ten-GigabitEthernet 1/0/10
    Ten-GigabitEthernet 1
/0/14 Ten-GigabitEthernet 1/0/26 Ten-GigabitEthernet 1/0/30
```

Configuración del bonding de almacenamiento Es necesario crear una interfaz de tipo Bridge-Aggregation.

```
[cefca-swt-014]interface Bridge-Aggregation 1
[cefca-swt-014-Bridge-Aggregation1]description Bonding almacenamiento hpc-master
[cefca-swt-014-Bridge-Aggregation1]quit
```

Agregaremos los puertos que pertenecen al bonding en su configuración.

```
[cefca-swt-014]interface Ten-GigabitEthernet 1/0/2
[cefca-swt-014-Ten-GigabitEthernet1/0/2]port link-aggregation group 1
[cefca-swt-014-Ten-GigabitEthernet1/0/2]quit
[cefca-swt-014]interface Ten-GigabitEthernet 1/0/4
[cefca-swt-014-Ten-GigabitEthernet1/0/4]port link-aggregation group 1
[cefca-swt-014-Ten-GigabitEthernet1/0/4]quit
```

Configuraremos el bonding con el protocolo LACP.

```
[cefca-swt-014]interface Bridge-Aggregation 1
[cefca-swt-014-Bridge-Aggregation1]link-aggregation mode dynamic
[cefca-swt-014-Bridge-Aggregation1]lacp edge-port
[cefca-swt-014-Bridge-Aggregation1]quit
```

Finalmente agregaremos el bonding a la vlan de almacenamiento

```
[cefca-swt-014]vlan 14
[cefca-swt-014-vlan14]port Bridge-Aggregation 1
[cefca-swt-014-vlan14]quit
```

Ajuste del bonding Como ya vimos con el switch anterior, será necesario configurar el modo con el que el switch realizará el forwarding de paquetes en sus interfaces. Para ello podremos ver el estado del bonding y el modo que usa.

```
<cefca-swt-014>display link-aggregation summary
Aggregation Interface Type:
BAGG -- Bridge-Aggregation, RAGG -- Route-Aggregation
Aggregation Mode: S -- Static, D -- Dynamic
Loadsharing Type: Shar -- Loadsharing, NonS -- Non-Loadsharing
Actor System ID: 0x8000, bcea-fa13-ca3c

AGG         AGG   Partner ID              Selected  Unselected  Individual  Share
Interface   Mode                          Ports     Ports       Ports       Type
--------------------------------------------------------------------------------
BAGG1       D     0xffff, d0bf-9cf1-80c4  2         0           0           Shar
<cefca-swt-014>display link-aggregation verbose
Loadsharing Type: Shar -- Loadsharing, NonS -- Non-Loadsharing
Port Status: S -- Selected, U -- Unselected, I -- Individual
Flags:  A -- LACP_Activity, B -- LACP_Timeout, C -- Aggregation,
               D -- Synchronization, E -- Collecting, F -- Distributing,
               G -- Defaulted, H -- Expired

               Aggregate Interface: Bridge-Aggregation1
               Aggregation Mode: Dynamic
               Loadsharing Type: Shar
               System ID: 0x8000, bcea-fa13-ca3c
               Local:
                 Port             Status  Priority Oper-Key  Flag

    --------------------------------------------------------------------------------

                 XGE1/0/2         S       32768    1         {ACDEF}
                 XGE1/0/4         S       32768    1         {ACDEF}
               Remote:
                 Actor            Partner Priority Oper-Key  SystemID
        Flag

    --------------------------------------------------------------------------------
```

```
                XGE1/0/2          2        255       33         0xffff, d0bf-9cf1
    -80c4 {ACDEF}
                XGE1/0/4          1        255       33         0xffff, d0bf-9cf1
    -80c4 {ACDEF}

<cefca-swt-014>display link-aggregation load-sharing mode interface Bridge-
    Aggregation 1
Bridge-Aggregation1 Load-Sharing Mode:
Layer 2 traffic: packet type-based sharing
Layer 3 traffic: packet type-based sharing
```

Aunque puede hacerse la configuración del modo a nivel global, pero lo haremos a nivel de interfaz.

```
<cefca-swt-014>system-view
System View: return to User View with Ctrl+Z.
[cefca-swt-014]interface bridg
[cefca-swt-014]interface Bridge-Aggregation 1
[cefca-swt-014-Bridge-Aggregation1]link-aggregation load-sharing mode source-ip
[cefca-swt-014-Bridge-Aggregation1]quit
```

5.10.3.3. Nodos del clúster

Instalación de las tarjetas

Configuración kernel Una vez instaladas las tarjetas y comprobado que el módulo de kernel necesario es cargado correctamente y presenta las interfaces correspondientes, será necesario tunear valores del kernel para aprovechar las tarjetas 10G. Por defecto los valores asignados a buffers y a otras estructuras de datos que utiliza el kernel para la gestión de las interfaces de red y la pila tcp/ip son óptimos para interfaces de 1G pero son insuficientes para el correcto aprovechamiento de interfaces 10G. Por ello tunearemos estos valores y haremos que se fijen automáticamente durante el inicio del sistema.

```
# socket buffers
sysctl -w net.core.rmem_max=16777216
echo "net.core.rmem_max = 16777216" >> /etc/sysctl.d/local.conf
sysctl -w net.core.wmem_max=16777216
echo "net.core.wmem_max = 16777216" >> /etc/sysctl.d/local.conf

# autotunning tcp buffer
sysctl -w net.ipv4.tcp_moderate_rcvbuf=1
echo "net.ipv4.tcp_moderate_rcvbuf = 1" >> /etc/sysctl.d/local.conf
sysctl -w net.ipv4.tcp_rmem="4096 87380 16777216"
echo "net.ipv4.tcp_rmem = 4096 87380 16777216" >> /etc/sysctl.d/local.conf
sysctl -w net.ipv4.tcp_wmem="4096 87380 16777216"
echo "net.ipv4.tcp_wmem = 4096 87380 16777216" >> /etc/sysctl.d/local.conf

# tcp backlog queue
```

```
sysctl -w net.ipv4.tcp_max_syn_backlog=4096
echo "net.ipv4.tcp_max_syn_backlog = 4096" >> /etc/sysctl.d/local.conf
sysctl -w net.core.somaxconn=1024
echo "net.core.somaxconn = 1024" >> /etc/sysctl.d/local.conf

# device backlog
sysctl -w net.core.netdev_max_backlog=10000
echo "net.core.netdev_max_backlog = 10000" >> /etc/sysctl.d/local.conf
```

Configuración de las interfaces La configuración será casi idéntica a la que vimos en el apartado de configuración de los nodos. La única opción que deberemos agregar es la configuración referente a la longitud de la cola de transmisión de la interfaz, que haremos que sea de 10000 en lugar de los 1000 por defecto. Por lo demás únicamente se realizará una reasignación de interfaces.

En el nodo cabecera tendremos.

```
root@hpc-master:~# cat /etc/network/interfaces
# Fichero generado mediante config_net_ifaces.sh

auto lo
iface lo inet loopback

auto em1
iface em1 inet static
        address 10.50.85.1
        netmask 255.255.255.0
        gateway 10.50.85.254
        dns-nameservers 10.50.84.1 10.50.84.2
        dns-search office.cefca.es cefca.es

auto em2
iface em2 inet manual

auto bond0
iface bond0 inet static
        address 192.168.14.1
        netmask 255.255.255.0
        bond-mode 802.3ad
        bond-miimon 100
        bond-downdelay 200
        bond-updelay 200
        bond-xmit-hash-policy layer2+3
        bond-slaves p1p1 p1p2
        mtu 9000

auto p1p1
iface p1p1 inet manual
        post-up ifconfig p1p1 txqueuelen 10000
        bond-master bond0

auto p1p2
```

```
iface p1p2 inet manual
        post-up ifconfig p1p2 txqueuelen 10000
        bond-master bond0
```

En los nodos de cómputo configuraremos el interfaces y eliminaremos el driver bonding de la carga de módulos.

```
root@hpc-node1:~# cat /etc/network/interfaces
# Fichero generado mediante config_net_ifaces.sh

auto lo
iface lo inet loopback

auto em1
iface em1 inet static
        address 10.50.85.10
        netmask 255.255.255.0
        gateway 10.50.85.254
        dns-nameservers 10.50.84.1 10.50.84.2
        dns-search office.cefca.es cefca.es

auto em2
iface em2 inet manual
        dns-nameservers 10.50.84.1 10.50.84.2
        dns-search office.cefca.es cefca.es

auto em3
iface em3 inet static
        address 192.168.13.10
        netmask 255.255.255.0
        post-up ifconfig em3 txqueuelen 10000
        mtu 9000

auto em4
iface em4 inet static
        address 192.168.14.10
        netmask 255.255.255.0
        post-up ifconfig em4 txqueuelen 10000
        mtu 9000
```

5.10.3.4. Monitorización

Dejaremos para más adelante la monitorización integral del switch. De momento únicamente configuraremos la monitorización Cacti para obtener el tráfico de red de las interfaces y poder realizar pruebas de rendimiento. Para ello simplemente:

- crearemos un dispositivo nuevo como vimos en la parte dedicada a monitorización, pero en este caso lo haremos de tipo Generic SNMP-enabled.

- los nombres de interfaz en el switch HP son muy largos, por lo que es necesario irse a `Settings -> Visual` y modificar el campo `Data Queries Maximum Field Length` a 25.
- crearemos los gráficos de dispositivo como vimos para el switch Cisco.
- agregaremos los gráficos del dispositivo al árbol de visualización.

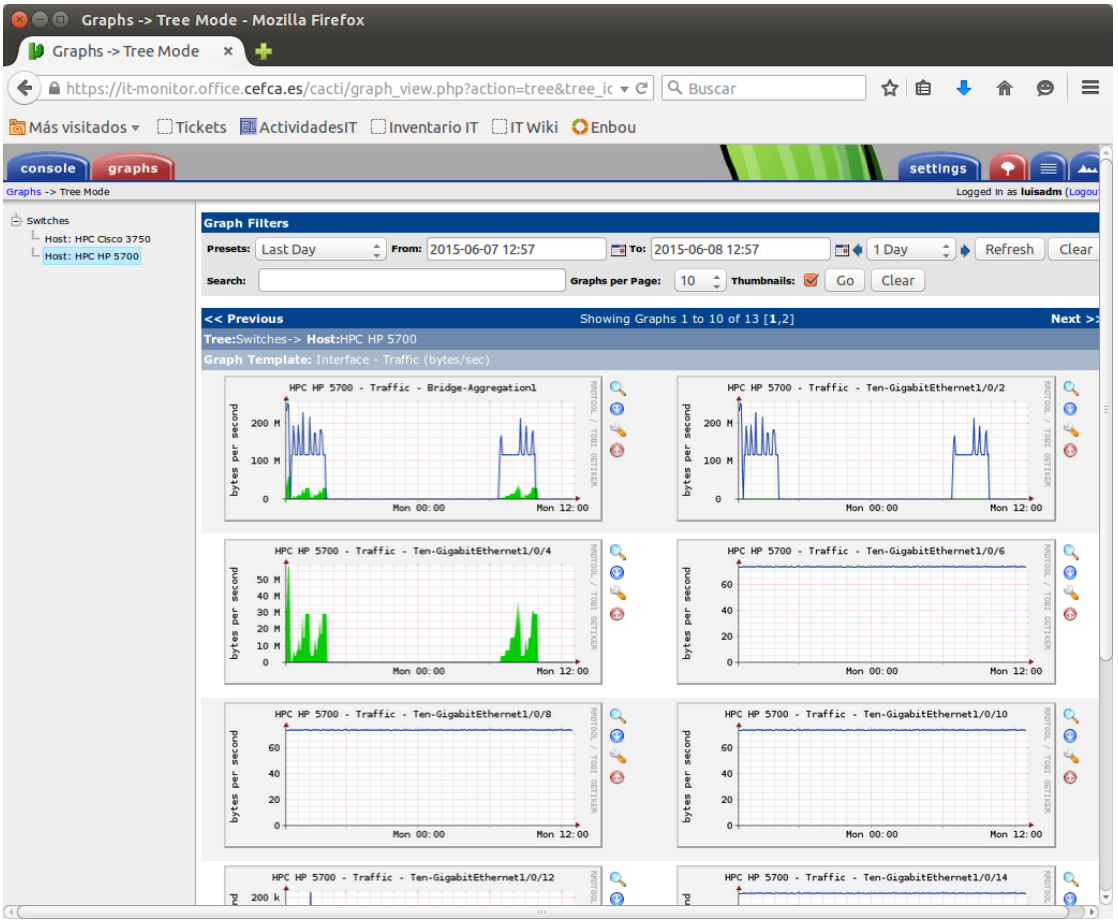


FIGURA 5.124: Monitorización de switch HP en Cacti

5.10.3.5. Despliegue automatizado

Para la automatización simplemente cambiaremos el fichero `stage1.run` en el que eliminaremos el helper `enable_bonding.sh` y agregaremos el helper:

- **config_kernel_values_10g.sh** que realizará el tuneo de valores del kernel para uso de interfaces 10G visto anteriormente.

Del fichero `stage1.cfg` únicamente modificaremos la sección referente a las interfaces de red:

```
### config_net_ifaces.cfg
cfg_netIface[0]="em1"
cfg_netType[0]="static"
cfg_netAddress[0]="$ip_iface1"
cfg_netNetmask[0]="255.255.255.0"
cfg_netGateway[0]="10.50.85.254"
cfg_netDnsNameservers[0]="$cfg_dnsServers"
cfg_netDnsSearch[0]="$cfg_dnsSearch"

cfg_netIface[1]="em2"
cfg_netType[1]="manual"

cfg_netIface[2]="em3"
cfg_netType[2]="static"
cfg_netPostUp[2]="ifconfig em3 txqueuelen 10000"
cfg_netAddress[2]="192.168.13.${last_ip}"
cfg_netNetmask[2]="255.255.255.0"
cfg_mtu[2]="9000"

cfg_netIface[3]="em4"
cfg_netType[3]="static"
cfg_netPostUp[3]="ifconfig em4 txqueuelen 10000"
cfg_netAddress[3]="192.168.14.${last_ip}"
cfg_netNetmask[3]="255.255.255.0"
cfg_mtu[3]="9000"
```

Actualizaremos la versión del paquete a 0.2 y la subiremos al servidor *lgdeploy*.

5.10.4. Mediciones y comparación con 1G

5.10.4.1. Especificación y ajustes

Nodos Ya vimos que realizamos algunos ajustes a nivel de kernel y en la carga de las interfaces para un uso óptimo de las interfaces 10G. Además de esto, las tarjetas 10G adquiridas [22] tienen soporte hardware TCP Offload Engine. Este soporte hace que algunas de las operaciones relativas a chequeos de checksum, fragmentación, comprobaciones de estado, etc se realicen el hardware, dotando al sistema de una menor latencia y un mejor rendimiento. Verificamos que las opciones disponibles están habilitadas mediante el comando `ethtool -k interfaz`.

Switch Acabamos de ver que el switch es un HP FexFabric 5700 [23]. El switch ofrece 40 puertos 10G, tiene 2 puertos QSFP para uplinks y usa una política *cut-through* lo que le permite tener una menor latencia.

[22] http://www8.hp.com/h20195/v2/getpdf.aspx/c04111479.pdf?ver=3
[23] http://www8.hp.com/h20195/v2/getpdf.aspx/c04347352.pdf?ver=1

Acabamos de ver que las conexiones se realizaron de la siguiente manera:

- Red de almacenamiento:
  - Nodo cabecera: bonding de 2 interfaces
  - Nodo cómputo: 1 interfaz
- Red de cómputo:
  - Nodo de cómputo: 1 interfaz

Bondings de almacenamiento Tal y como hicimos en la red 10G será necesario que ajustemos los parámetros de bonding del switch. En esta ocasión usaremos sólo dos modos para las pruebas:

1 Política por defecto

2 Basado en ip-src

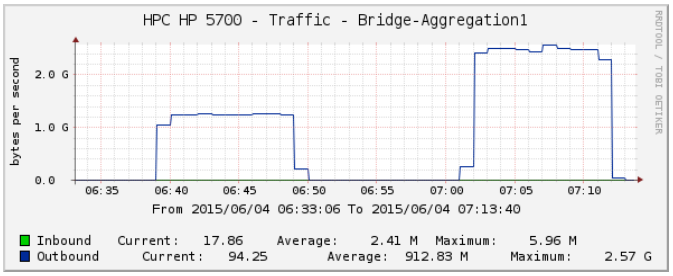


FIGURA 5.125: Estadística tráfico del bonding de almacenamiento 10G

Como puede verse de la gráfica 5.125 obtenida mediante Cacti, el modo por defecto tan sólo aprovecha una interfaz mientras que el modo basado en ip-src aprovecha el agregado.

Pruebas enlaces de cómputo En este caso al no existir bondings para el cómputo no será necesario realizar ningún ajuste. Sin embargo comprobamos con la misma prueba que hicimos que todos los enlaces funcionan correctamente.

Lo que vemos en la figura 5.126 es que se aprovecha el full-duplex de todas las tarjetas y no se ve ningún tipo de caída como las que observamos en los bondings realizados sobre el anterior equipamiento de 1G.

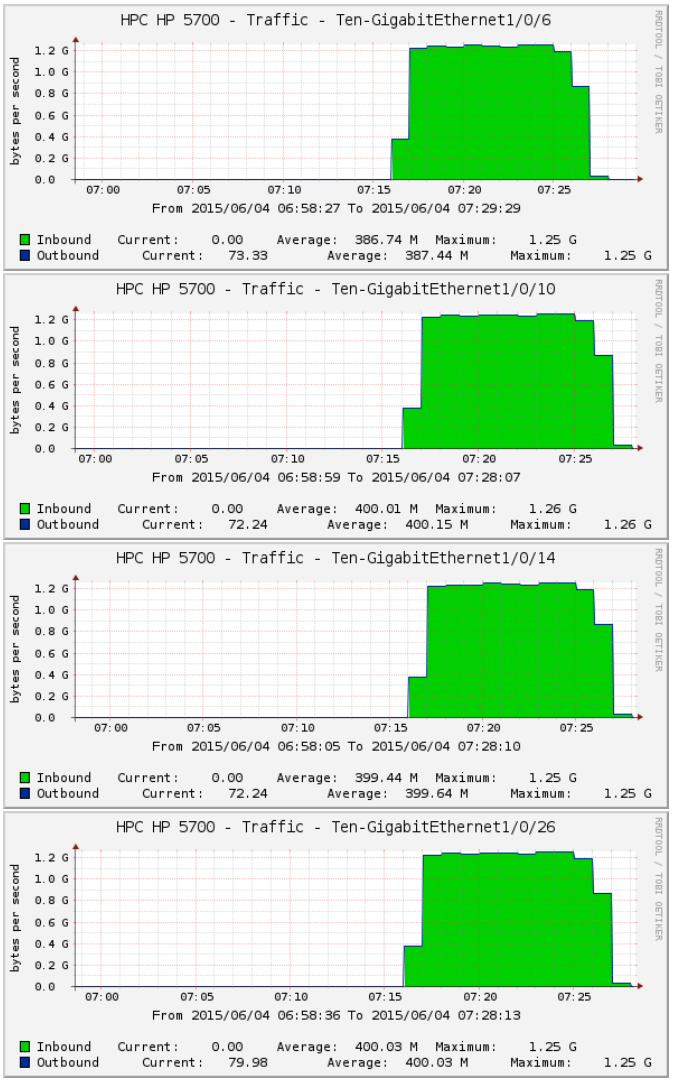


FIGURA 5.126: Estadística de tráfico de las interfaces de la red de cómputo 10G

5.10.4.2. Mediciones

Benchmark iperf Como hemos visto en el ajuste del bonding, realizamos exactamente los mismos benchmarks que para 1G y también en este caso analizamos los resultados. Pueden verse en las figuras 5.127 y 5.128 cómo se observa el comportamiento del control de flujo de TCP pero en este caso no vemos las caídas en el flujo que vimos con los dispositivos de networking 1G.

Benchmark netpipe Se hicieron las pruebas de throuhput TCP y latencia MPI tal y como puede verse en la figura 5.130. A continuación se comparó con los datos obtenidos para 1G para ver la mejora. Además de la mejora obvia en el throughput, como podemos ver en la figura 5.132 obtenemos una mejora en latencia de un factor 2 para los mensajes más pequeños.

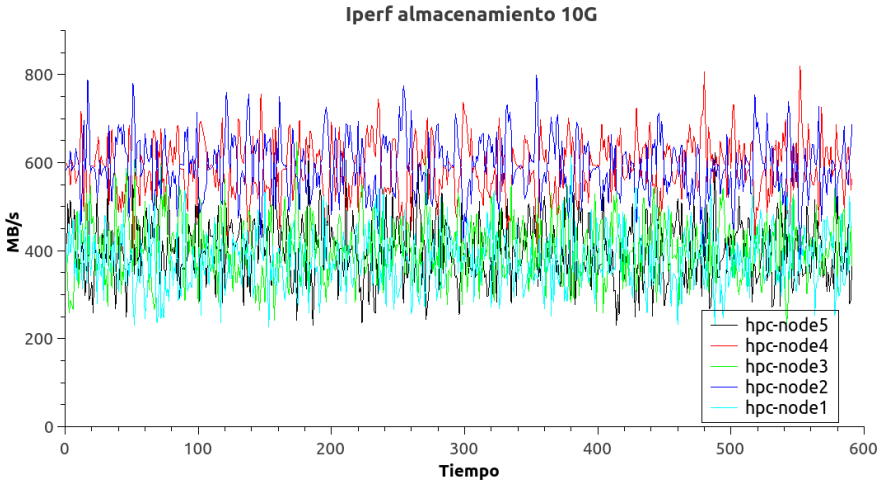


FIGURA 5.127: Iperf red de almacenamiento 10G

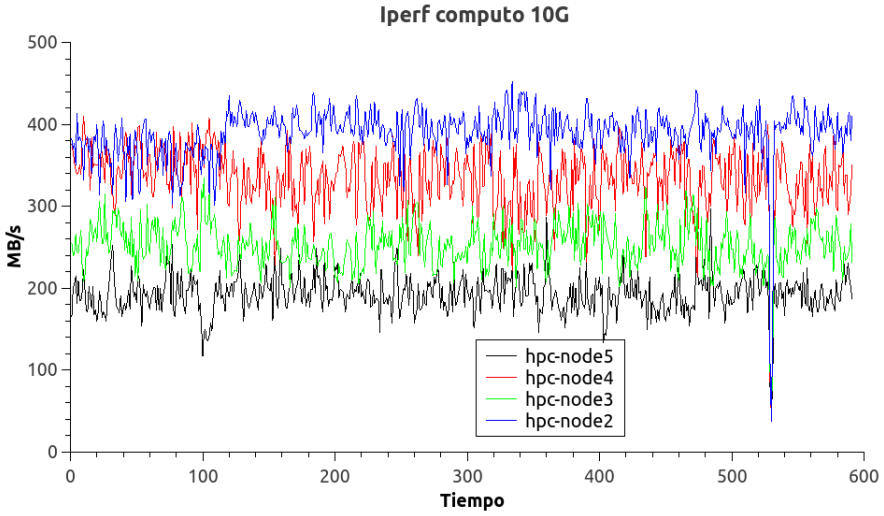


FIGURA 5.128: Iperf red de cómputo 10G

Mediciones dd Por los resultados vistos que vimos anteriormente, la red ya no va a ser el factor limitante. Lanzaremos de nuevo las mismas pruebas vistas de la sección anterior para el sistema NFS.

En las pruebas de escritura NFS podemos ver en las figuras 5.133 y 5.134 que el rendimiento que ofrece el volumen de scratch es muy superior en el modo fdatasync al resto y ligeramente superior al scratch en modo dsync. Estos buenos resultados se deben al modo async usado en NFS. Por otro lado, si observamos al volumen users, el rendimiento en escritura se va reduciendo conforme se guardan más datos mi hipótesis se debe a que es un efecto de las cachés de las controladoras. Sin embargo, el rendimiento es inferior al que obtuvimos en el acceso directo del host y en el caso de escritura síncrona continua del búffer de 1M el rendimiento es todavía peor. Para esto último ya vimos el impacto sustancial en las escrituras síncronas continuadas en 1G que podría ser debido

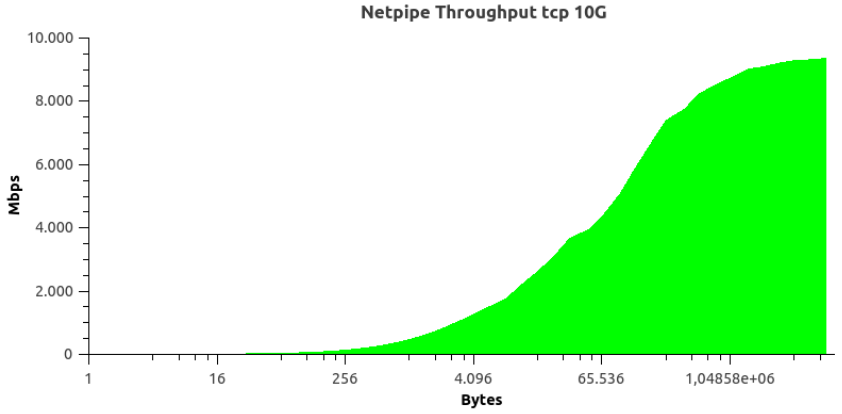


FIGURA 5.129: Netpipe medición de throughput TCP red 10G

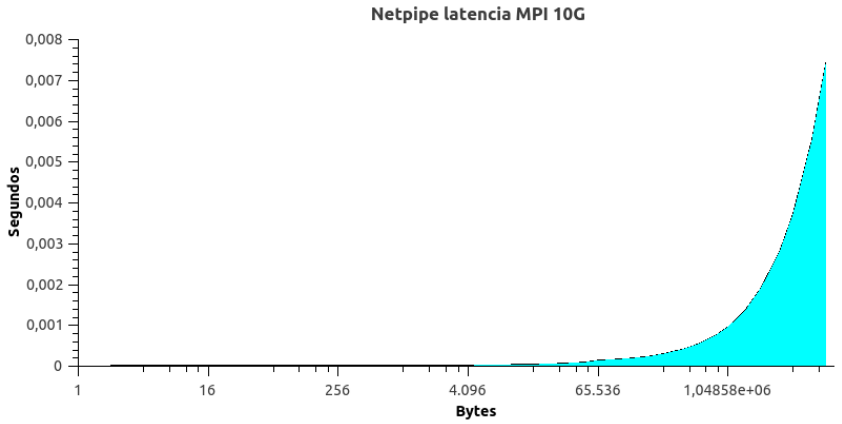


FIGURA 5.130: Netpipe medición de latencia MPI red 10G

al overhead del protocolo, sin embargo habría que llevar un estudio más a fondo para poder dar respuesta a estas cuestiones.

Ya que está claro que el límite no es la red, en este caso sí tiene sentido realizar pruebas de lecturas desde un nodo de cómputo usando NFS. Para las pruebas se procedió de un modo similar a las lecturas locales en el nodo cabecera solo que además se incluyó un comando por ssh para que vaciara las cachés y búffers del nodo cabecera. Los resultados obtenidos pueden verse en la gráfica 5.135. Pueden observarse rendimientos similares pero inferiores a los vistos en el nodo cabecera.

Finalmente con los datos obtenidos construimos la gráfica comparativa de la figura 5.136 en la que vemos una comparativa entre los valores obtenidos en 1G y los obtenidos en 10G.

Benchmark Linpack Para ver el rendimiento global el clúster volveremos a medir usando el benchmark Linpack con el hyperthreading habilitado y 2 hilos de OpenBLAS.

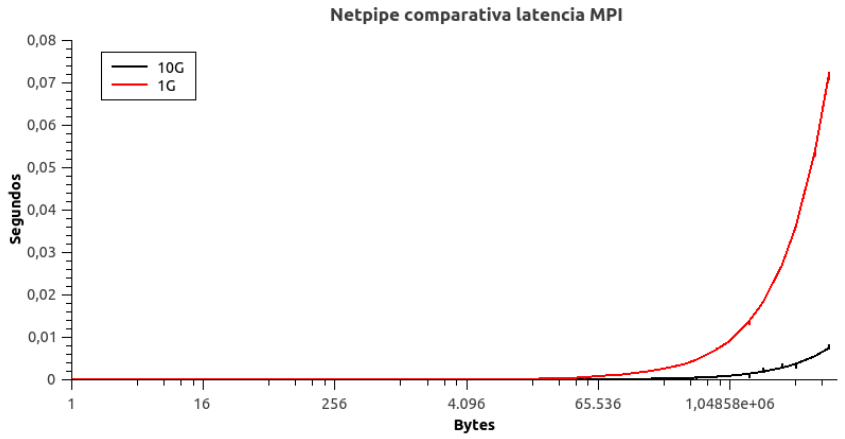


FIGURA 5.131: Netpipe comparativa de latencia

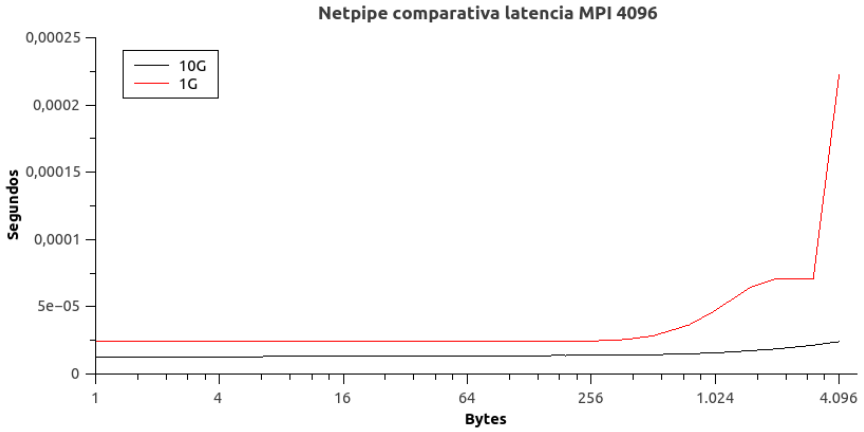


FIGURA 5.132: Netpipe comparativa de latencia mensajes menores de 4096 bytes

El resultado obtenido en este caso es de 1'62 Teraflops lo que supone un 82 % del rendimiento de pico. En la figura 5.137 puede verse una comparativa entre los valores obtenidos.

Benchmark gadget-3 Volveremos a ejecutar las mismas pruebas que vimos para 1G. En primer lugar tenemos la ejecución del benchmark small aumentndo el número de procesos MPI y con distintos PosixThreads. Vemos en la figura 5.138 como en este caso el problema sí que escala conforme agregamos procesos.

A continuación hacemos la prueba que vimos de separar el mismo problema con los mismos procesos MPI pero en distintos nodos. Ahora podemos ver en la gráfica comparativa en la figura 5.139 cómo en ambos casos el tiempo en comunicación que se añade al mover los procesos MPI a otros nodos aumenta. Sin embargo la diferencia en el aumento entre una ejecución con 1G y una 10G es notable.

Para finalizar, volvemos a ejecutar el problema medium que vimos para 1G. En la figura 5.140 puede verse una comparativa entre el tiempo que se toma en obtener el resultado

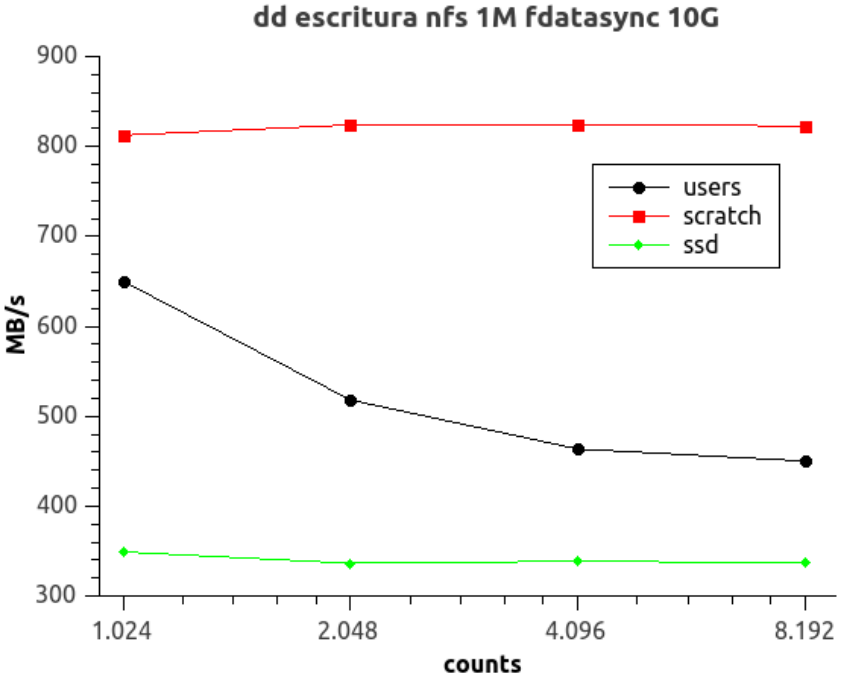


FIGURA 5.133: Escritura nfs con dd fdatasync 10G

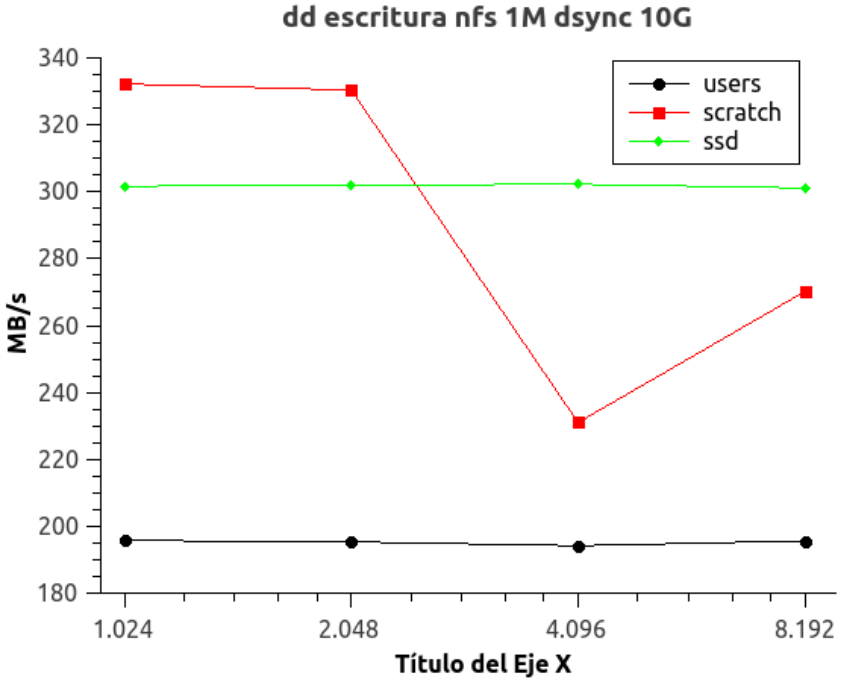


FIGURA 5.134: Escritura nfs con dd dsync 10G

que muestro a continuación comparado con el resultado obtenido con el networking de 1G.

```
Step 3, Time: 0.0204637, MPI-Tasks: 100   Threads: 1
total              601.95   100.0%
treegrav           333.80    55.5%
   treebuild         4.12     0.7%
```

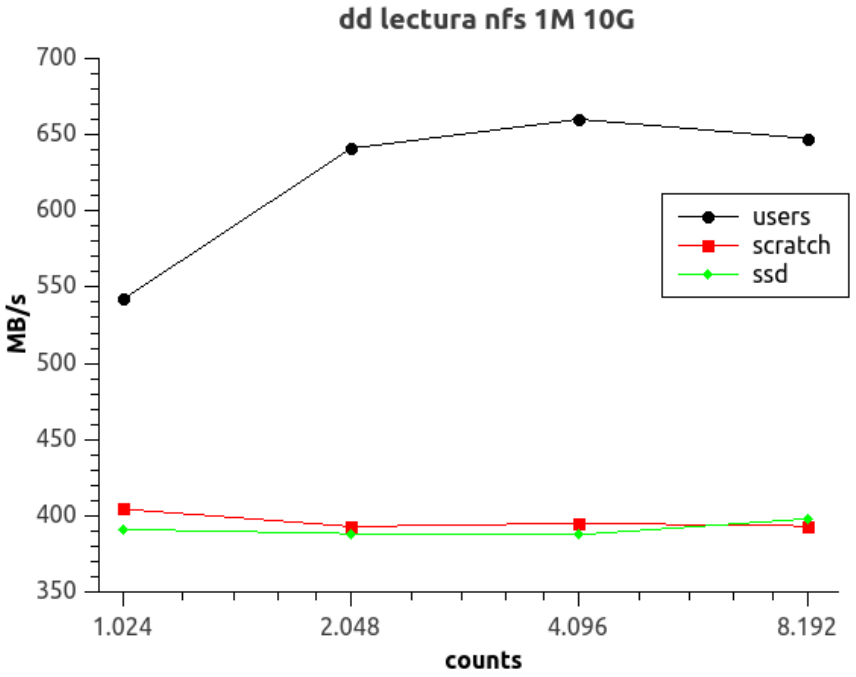


FIGURA 5.135: Lectura nfs con dd en 10G

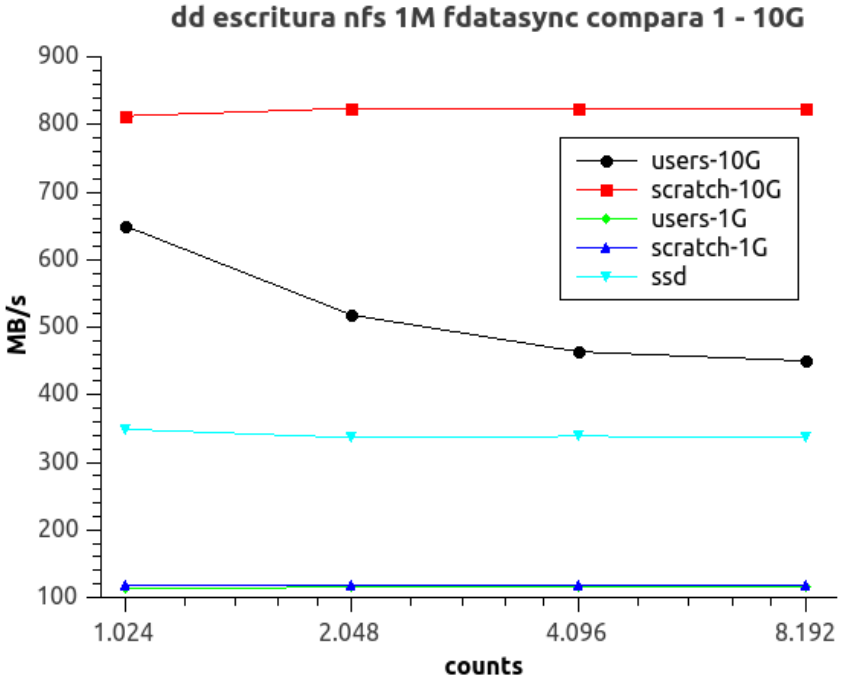


FIGURA 5.136: Comparativa escritura nfs con dd fdatasync en 10G

```
   treeupdate        0.00    0.0%
   treewalk        193.75   32.2%
   treecomm          3.28    0.5%
   treeimbal       130.45   21.7%
pmgrav              67.02   11.1%
sph                155.14   25.8%
```

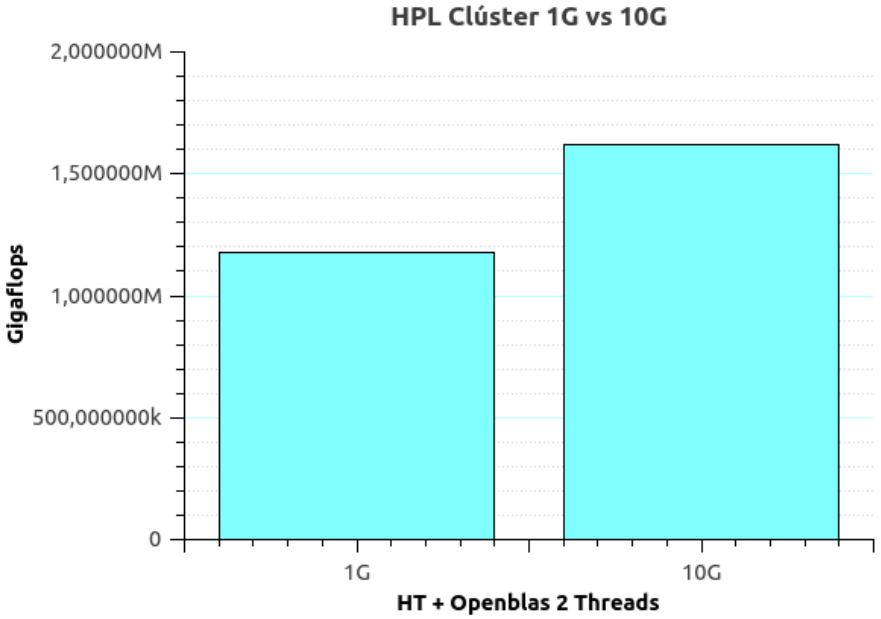


FIGURA 5.137: Comparativa benchmark Linpack 1G vs 10G

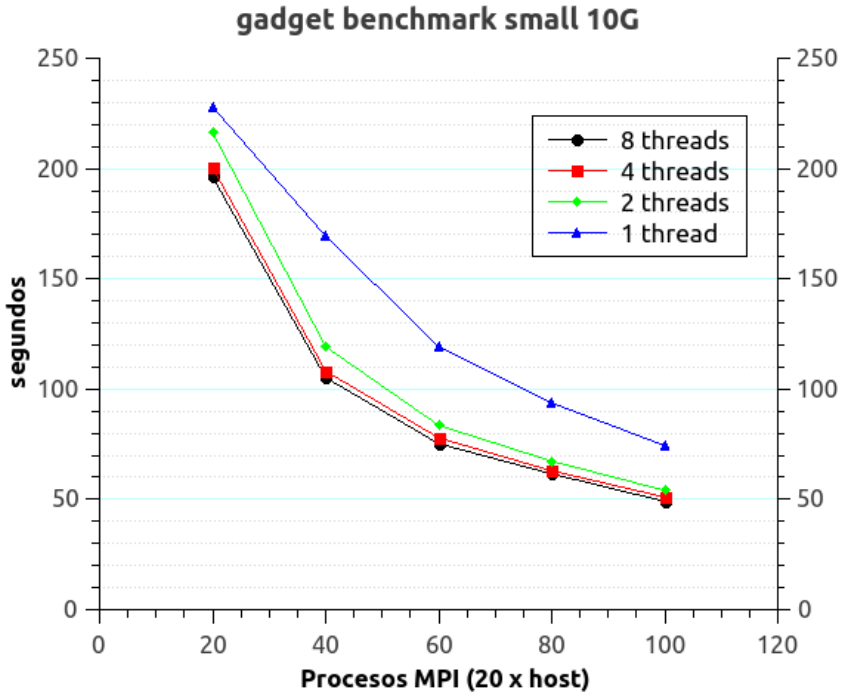


FIGURA 5.138: Ejecución gadget3 small en clúster 10G

```
density          43.01     7.1%
denscomm          1.90     0.3%
densimbal        21.32     3.5%
hydrofrc         51.23     8.5%
hydcomm           2.66     0.4%
hydmisc           0.85     0.1%
hydnetwork        0.00     0.0%
hydimbal         32.81     5.5%
```

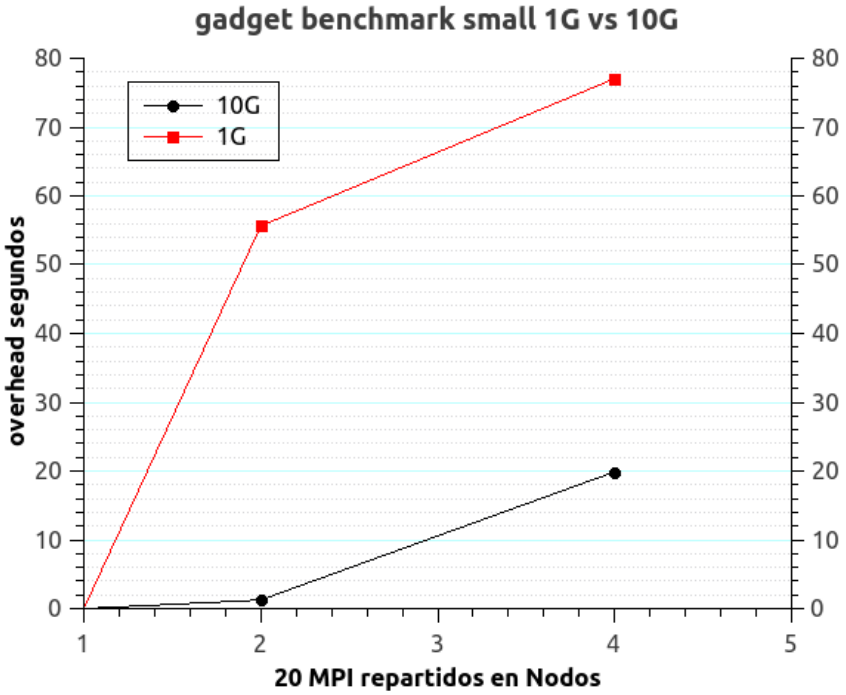


FIGURA 5.139: Comparativa efecto comunicación gadget3 small en 1G vs 10G

```
   hmaxupdate          0.47     0.1%
domain                36.51     6.1%
potential              0.00     0.0%
predict                0.00     0.0%
kicks                  0.72     0.1%
i/o                    0.00     0.0%
peano                  3.11     0.5%
sfrcool                0.00     0.0%
blackholes             0.00     0.0%
fof/subfind            0.00     0.0%
smoothing              0.00     0.0%
hotngbs                0.00     0.0%
weights_hot            0.00     0.0%
enrich_hot             0.00     0.0%
weights_cold           0.00     0.0%
enrich_cold            0.00     0.0%
cs_misc                0.00     0.0%
misc                   5.66     0.9%
```

5.11. Soporte al usuario

5.11.1. Breve análisis

Por fin llegamos al final de esta etapa de desarrollo. En esta parte definiremos brevemente las herramientas con las que se dará soporte al usuario.

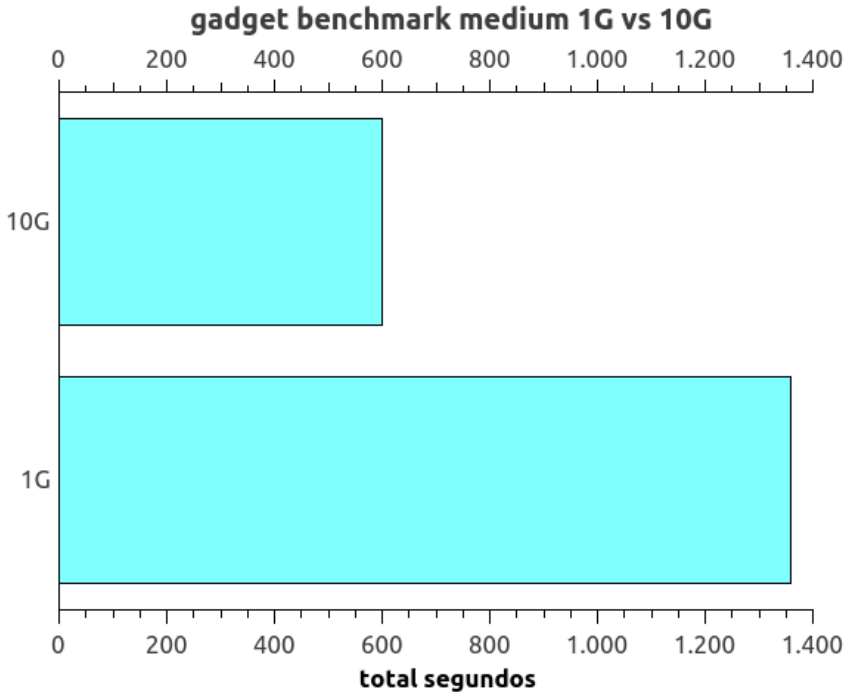


FIGURA 5.140: Comparativa gadget benchmark medium 1G vs 10G

Las necesidades básicas serán:

- Definir un sitio web dedicado en el cual alojar la documentación de usuario y en el que a su vez los usuarios puedan proporcionar nueva documentación.
- Tener un medio de comunicación en el que podamos informar a los usuarios de incidencias, novedades y en el que los usuarios puedan lanzar preguntas.
- Ofrecer una herramienta a través de la cual se gestionarán las incidencias, peticiones, etc.

5.11.2. Herramientas de soporte

5.11.2.1. Portal web cluster-info

La herramienta que se seleccionó para alojar la documentación de usuario fue un Wiki. Se optó finalmente por MediaWiki debido a su madurez, a su facilidad de instalación y a que ya se estaba usando en otros ámbitos en CEFCA.

El despliegue se realizó en el servidor *cluster-info*, servidor en el que instalamos el componente web de monitorización de Ganglia. De este modo teníamos todo alojado en un servidor (este servidor se trata de un servidor virtualizado usando VMware ESXi). Se

estableció que el wiki era de acceso público (dentro de CEFCA) y se usaría autenticación para cambiar o agregar contenido. El motivo no era obviamente por un motivo de falta de confianza, sino simplemente para tener una relación de los usuarios y sus contribuciones. El único elemento un poco peleagudo de la configuración es la parte de autenticación de usuarios contra el dominio. Se solucionó agregando la máquina al dominio, usando un plugin Mediawiki que autocreaba y gestionaba la autorización y la autenticación integrada de Kerberos en Apache para el proceso de autenticación. No voy a describir aquí los pasos de integración, pueden encontrarse en el código fuente de la infraestructura adjunta en el proyecto.

Una vez creado el wiki se generó:

- una página básica de entrada al sistema en la que se describe brevemente el clúster y las colas disponibles
- un tutorial de uso del gestor de recursos
- un tutorial de uso del entorno modules

Toda esta documentación generada se encuentra en el anexo referente a la Documentación de Usuario. En la figura 5.141 puede verse una captura de pantalla de la misma.

5.11.2.2. Lista de distribución de correo interna

Como medio de comunicación para la notificación de incidencias, noticias, etc. se definió el método más simple: una lista de distribución interna. Para ello se creó una lista con la dirección de correo *hpc_users@cefca.es* que tendría como miembros a los usuarios del clúster.

Se valoró el uso de un foro tipo phpBB para el planteamiento de cuestiones de usuarios. Sin embargo, por simplicidad y dado el pequeño volúmen se adoptó que la lista también sería el canal mediante el cual se podrían lanzar cuestiones.

5.11.2.3. Helpdesk

Por último, la herramienta de gestión de incidencias sería la herramienta helpdesk ya existente en CEFCA. Puede verse la figura 5.142.

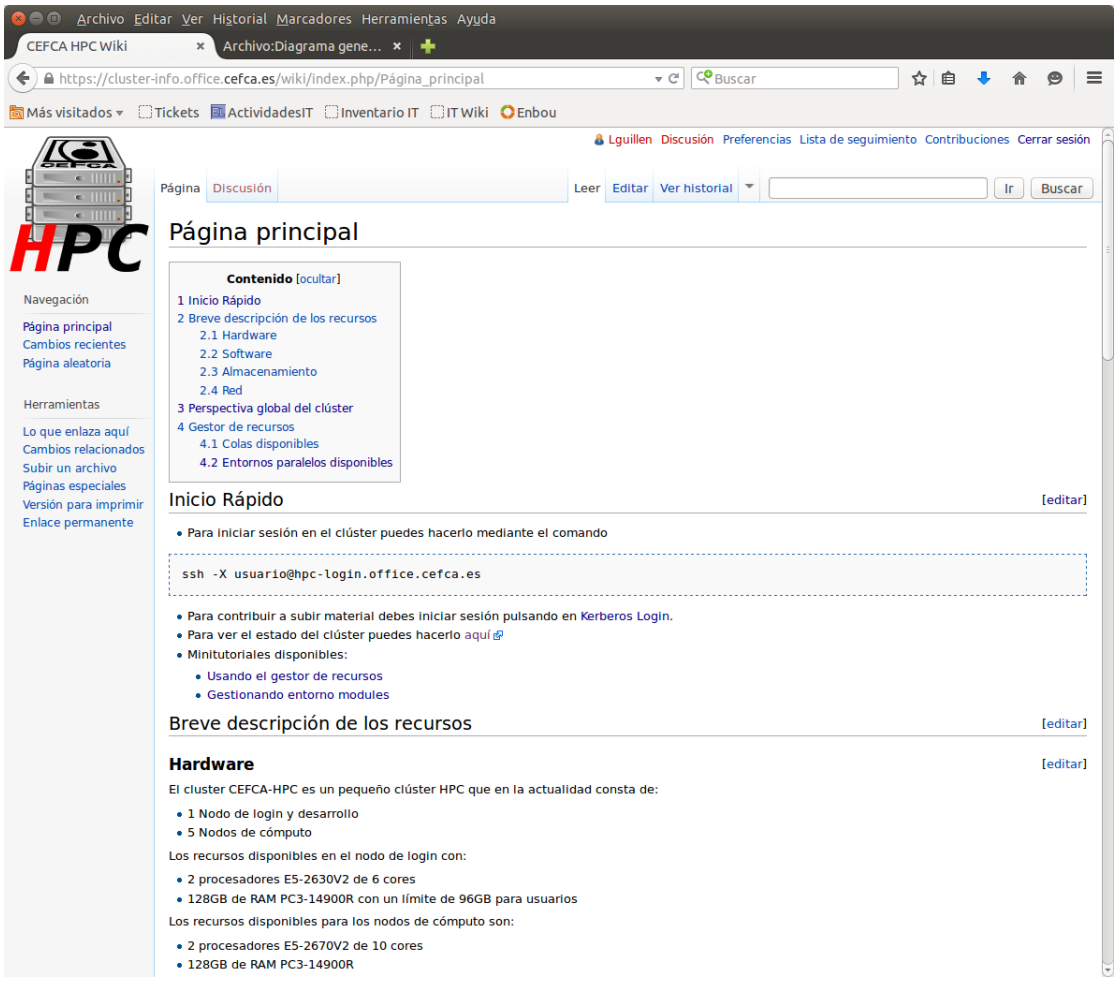


FIGURA 5.141: Web de documentación de usuario

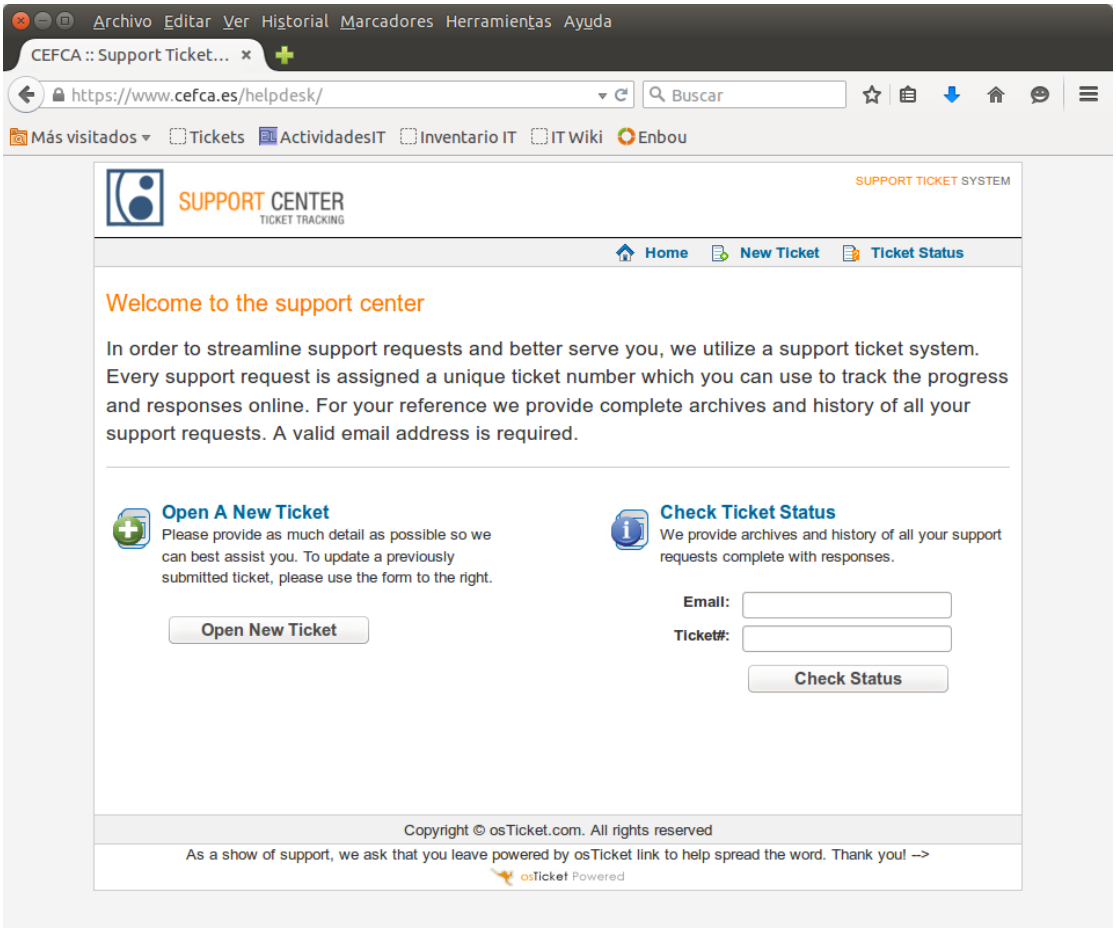


FIGURA 5.142: Helpdesk de usuarios

Capítulo 6

Conclusiones y futuras líneas de trabajo

En este último capítulo de la memoria, daremos un rápido vistazo al desarrollo del proyecto y trataremos de extraer algunas conclusiones. Finalmente enumeraremos algunas de las líneas en las que se trabajarán tras la conclusión del mismo.

6.1. Conclusiones

Tras la especificación de los requerimientos, el siguiente paso a dar en nuestro proyecto fue la adquisición del equipamiento. Como no podía ser de otro modo, el objetivo marcado fue el de obtener la mejor solución al menor coste posible. Debido a la titularidad pública de la institución, la adquisición debía realizarse mediante un proceso de licitación pública, para el cual fue necesario definir los requerimientos específicos de la solución hardware de forma lo suficientemente genérica como para no excluir a ningún proveedor. Como vimos, la oferta ganadora resultante fue la solución de HP, que superaba algunos de los requisitos y a un precio inferior al presentado por el resto de competidores. En este aspecto podemos concluir que se cumplió con creces el objetivo marcado, ya que obtuvimos una solución que no sólo superaba las especificaciones, sino que contaba con el aval de una primera marca y a un precio muy competitivo.

Una vez se completó el proceso de adquisición y quedaron concretados todos los aspectos técnicos del hardware, fue posible pasar a las fases posteriores de diseño e implementación. Uno de los objetivos en este ámbito fue la integración del clúster en la infraestructura de red y en los servicios ya existentes en CEFCA. Como se recogió en la memoria, la solución se integró en la infraestructura de red satisfaciendo los requerimientos de uso,

seguridad y conectividad fijados. En lo que respecta a los servicios, únicamente se implementaron los servicios internos del clúster correspondientes al almacenamiento y a la gestión y monitorización específica de los recursos. Para el resto de servicios necesarios, el clúster se integró con los ya existentes, cumpliendo de este modo los requerimientos marcados.

Desde el punto de vista del administrador del sistema, dicha integración implica que que no es necesario duplicar servicios con todas las ventajas que ello le reporta. Como se recoge en los anexos y en el código fuente del DVD, el despliegue de la infraestructura fue codificado en software. Por un lado, esto permitió acelerar todo el proceso de desarrollo y pruebas de configuración e integración de los sistemas sin interferir en ningún momento con el entorno de producción y, por otro lado, simplificó la tarea de desarrollo de la automatización del despliegue de los nodos de cómputo. Para un administrador, esta posibilidad de replicar la infraestructura es un aspecto muy valorable, ya que le permite, además de comprender mejor la infraestructura, tener un escenario sobre el que probar con antelación cambios en la configuración o la integración de nuevos servicios. Por si esto fuera poco, el administrador del sistema dispone de la presente memoria como documentación de la solución.

Desde el punto de vista del usuario, la integración realizada permite que el clúster sea un recurso más a su disposición y muy parecido al que tiene en su workstation. Desde su primera versión, los usuarios trabajan con el sistema tal y como se especificó: inician sesión en un nodo, pueden ver el estado del clúster y, si tienen recursos a su disposición, lanzan sus trabajos. Esto lo hacen sin importarles aspectos como en qué nodos se van a ejecutar sus trabajos. Ven al sistema como un conjunto agregado de recursos con un software y una jerarquía de directorios homogénea que tienen a su disposición. Con todo esto, podemos afirmar que hemos cumplido con el objetivo de visión de "caja negra" marcado al inicio del proyecto. Además, esta "caja negra" se encuentra complementada de una documentación accesible, especialmente enfocada en conocer su utilización en pocos minutos y dinámica, ya que está abierta a que los usuarios hagan sus propias aportaciones de forma fácil y cómoda.

Desde un punto de vista del sistema, se realizaron algunos ajustes con el fin de obtener una configuración óptima para los tipos de problemas más habituales. Por ello se realizaron ajustes a nivel de sistema, a nivel de red y de almacenamiento para posteriormente compilar y optimizar las librerías científicas más empleadas. Tras las mediciones realizadas, pudimos ver la importancia de emplear algunas de las optimizaciones que realiza el compilador así como la importancia de la planificación de los procesos que realiza el sistema operativo sobre los cores. Por último, tras la actualización de la red a 10G,

quedó de manifiesto la importancia de la tecnología de interconexión en los problemas que hacen uso del paradigma de memoria distribuida.

Por hacer unos breves comentarios sobre el desarrollo del proyecto, diré que el principal obstáculo con el que me encontré en el desarrollo del mismo fue la no exclusividad, esto es: el proyecto debía realizarse en paralelo con otros proyectos ya en marcha y sin dejar de lado las tareas habituales de mantenimiento y soporte. En este aspecto el enfoque de planificación por hitos y el proceso de maduración de la solución entre versiones, permitió una mayor concentración del tiempo en determinados momentos del desarrollo a la vez que permitió seguir sacando adelante otros proyectos en el tiempo transcurrido entre iteraciones. Evidentemente esto no estuvo exento de dificultades, siendo el mayor problema el tiempo adicional que se requería al inicio de cada iteración en retomar y volver al contexto del proyecto. A pesar de esto, en retrospectiva, creo que el enfoque tomado fue un acierto ya que otros proyectos pudieron salir adelante, el clúster estuvo operativo la mayor parte del tiempo y el tiempo de maduración entre versiones permitió obtener una solución más sólida.

Del mismo modo que los obstáculos forman parte de los proyectos reales, también lo forman los cambios que pueden darse durante el desarrollo de los mismos. Tal y como pudimos ver al final de nuestro proyecto, la disponibilidad de recursos adicionales permitieron una actualización de la conectividad interna del clúster a una tecnología superior. La facilidad con la que encajó dicha actualización y los resultados obtenidos por la misma nos sirvió para constatar que habíamos realizado tanto una correcta definición de la solución a nivel de equipamiento como un diseño válido en lo que respecta a las posibilidades de ampliación que existían al inicio del proyecto. Por otro lado, nos permitió comprobar el retorno de la inversión de los esfuerzos dedicados en la automatización, quedando de manifiesto que la automatización no sólo nos permite agregar nuevos nodos al clúster en cuestión de minutos, sino también realizar cambios significativos de forma eficaz.

Finalmente comentaré un par de ejemplos de las mejoras obtenidas con el clúster HPC. El primero de ellos está relacionado con el campo de Evolución de Galaxias y es un caso particular de SED-fitting de galaxias (recuperación de parámetros fundamentales de sus poblaciones estelares). El rendimiento en una máquina estándar de CEFCA (Dell T1650, 4 cores, 16Gb de RAM) da un promedio de una galaxia cada 14.67s (usando multiprocesado), mientras que en el clúster (usando los 100 cores) este tiempo se reduce a que se realiza el análisis de una galaxia cada 0.70s. Esto nos da una ganancia en tiempo de cómputo de un factor 20.92 (un 2100 %). El segundo de los ejemplos está relacionado con el campo de la Cosmología y se trata de un caso particular de BAO peak reconstruction (reconstrucción del pico de las BAO). Consiste en mover las galaxias de

su posición actual (Eulerian position) a su posición inicial (Lagrangian position). Es un problema que hay que resolver obligatoriamente si queremos calcular correctamente parámetros cosmológicos (el principal objetivo de J-PAS). Para resolver el problema utilizamos transformadas de Fourier en 3D en una malla. Al margen de que el tiempo de ejecución se reduce linealmente con el número de procesadores, al escalar el número de celdas de la malla con el cubo de las subdivisiones de la malla, rápidamente aumentamos el consumo de memoria de modo que es inviable en una workstation (requiriendo un problema típico alrededor de 100GB de memoria).

Podemos concluir, tras los casos de éxito mostrados, que la solución realizada se consolida como una potente herramienta que permite al personal científico de CEFCA en algunos casos obtener resultados mucho más rápido de lo que lo hacían antes (un 2100 % en el ejemplo visto) y en otros casos resolver problemas que sencillamente eran irresolubles con las herramientas que tenían a su disposición.

Para finalizar, me gustaría enlazar esto último con una reflexión fruto de un sencillo ejercicio: comprobar los Gigaflops que obtuvimos con nuestro clúster en el test Linpack con el Top500 de hace exactamente 15 años. Pues bien, nuestro modestísimo clúster ocuparía el puesto número 3 de los supercomputadores más potentes del mundo. Esto da para pensar... ¿como serán las máquinas de dentro de 15 años?, ¿dónde estarán los límites?... e ilusionarse... ¿cuántos avances en nuestra incesante búsqueda de respuestas podrán darse gracias a esta revolución tecnológica?.

6.2. Futuras líneas de trabajo

Aunque el proyecto lo hayamos dado por finalizado, el clúster HPC es una herramienta que estará en constante evolución. Prueba de ello es la pequeña lista de tareas que se realizarán a corto y medio plazo.

- Se está a la espera de la recepción de los dos nuevos nodos de cómputo adquiridos con los que se completará el chasis. Al encontrarse el sistema automatizado, el despliegue de los dos nuevos nodos y agregarlos a la configuración del gestor de colas no debería de llevar más de unos minutos. Con estos dos nuevos nodos el clúster pasará de tener 2 teraflops de capacidad de cómputo a tener 2.8 teraflops.
- Como se vio en el apartado a la actualización a 10G, el clúster fue trasladado a un rack con unas nuevas PDUs HP. Estas PDUs son bastante básicas, pero permiten la monitorización de consumo de los equipos. Esto nos permitirá centralizar y medir el consumo agregado del clúster, pudiendo realizar ajustes en el mismo para reducir su impacto sobre el consumo.

- En el proyecto no hemos realizado la integración con el sistema de backup, esto se debe a que durante la ejecución del mismo no disponíamos de un sistema de backup capaz de realizar copias del almacenamiento disponible. Recientemente se adquirió una librería de cintas con la capacidad suficiente para realizar backups del clúster y de otros servidores que también se encontraban sin un medio de respaldo aceptable. En dicho servidor se ha desplegado la solución Bacula, por lo que se instalará el agente necesario en el nodo cabecera.

- Como se observó, existe cierto margen de mejora con la red 10G en lo que al rendimiento de almacenamiento se refiere, por lo que se trabajará en mejorar este sistema.

- Está pendiente una evaluación de las utilidades de desarrollo de Intel para clústers. Se estudiarán también herramientas de perfilado y debuggeado de aplicaciones paralelas.

- Finalmente, de manera adicional se está trabajando en el despliegue de una solución HTCondor en las estaciones de trabajo de modo que puedan aprovecharse también los ciclos de CPU ociosos en las mismas. Para esto será necesario portar la gestión de la configuración de las estaciones a la utilidad Puppet. Una vez completado, se trabajará en portar también la configuración de los nodos de cómputo del clúster a esta utilidad.

Anexo A

Documentación de usuario

A.1. Introducción al clúster

A.1.1. Inicio rápido

- Para iniciar sesión en el clúster puedes hacerlo mediante el comando

```
ssh -X usuario@hpc-login.office.cefca.es
```

- Para contribuir a subir material debes iniciar sesión pulsando aquí.
- Para ver el estado del clúster puedes verlo aquí
- Minitutoriales disponibles:
  - Usando el gestor de recursos
  - Gestionando el entorno modules

A.1.2. Breve descripción de los recursos

A.1.2.1. Hardware

El cluster CEFCA-HPC es un pequeño clúster HPC que en la actualidad consta de:

- 1 Nodo de login y desarrollo
- 5 Nodos de cómputo

Los recursos disponibles en el nodo de login con:

- 2 procesadores E5-2630V2 de 6 cores
- 128GB de RAM PC3-14900R con un límite de 96GB para usuarios

Los recursos disponibles para los nodos de cómputo son:

- 2 procesadores E5-2670V2 de 10 cores
- 128GB de RAM PC3-14900R
- 1 disco duro de tipo SSD

Esto hace un agregado de 100 cores disponibles para las tareas de cómputo y un promedio de 6.4GB de RAM por core. Todos los nodos tienen el hyperthreading habilitado.

A.1.2.2. Software

El sistema operativo desplegado tanto en los nodos de cómputo como en el nodo de login es Ubuntu Server 14.04LTS. Además tiene instalado el paquete ubuntucefca14-scientific que contiene todo el software que se ha definido como estándar de software CEFCA para las estaciones de trabajo.

Además del software citado, existe software especialmente compilado y optimizado a disposición de los usuarios a través del comando module.

A.1.2.3. Almacenamiento

Tenemos dos tipos de almacenamiento:

- **Compartido**: es visible y accesible en todo el clúster.
  - Datos de usuarios: 12 TB
  - Scratch y catálogos: 4 TB
- **Local**: es visible y accesible únicamente en el nodo en el que se ejecuta una tarea.
  - Scratch y catálogos: 188GB

A.1.2.4. Red

Toda la conectividad de red se realiza mediante conexiones de 1G.

A.1.3. Perspectiva global del clúster

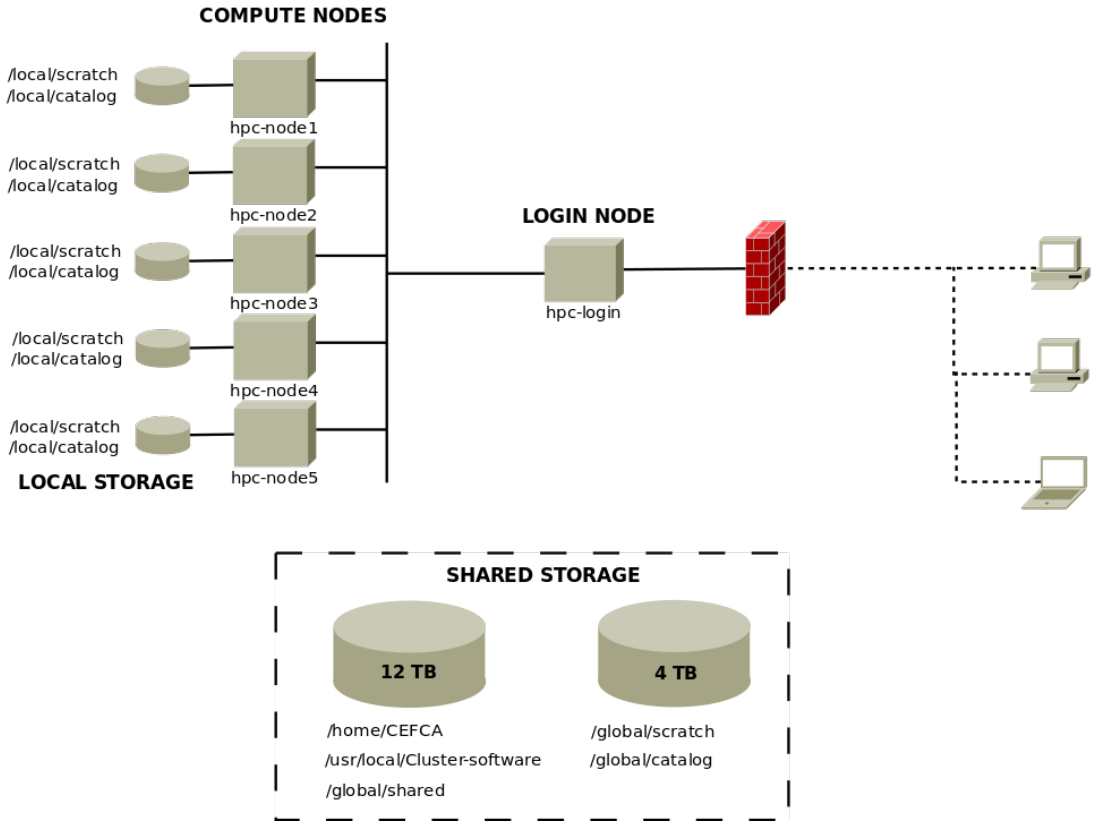


FIGURA A.1: Perspectiva global del clúster

En el diagrama mostrado en la figura A.1 se muestra una aproximación conceptual del clúster desde el punto de vista de los usuarios. Los usuarios deberán conectarse al nodo de login y desde ahí utilizando el gestor de recursos enviarán los trabajos a los nodos de cómputo.

También pueden verse las trayectorias de los recursos de almacenamiento y su alcance.

A.1.4. Gestor de recursos

- Un slot de ejecución equivale a un trabajo y todos los nodos tienen tantos slots de ejecución como cores: 20.
- Si se desean trabajos con más slots de ejecución disponibles, será necesario especificar un entorno paralelo.

- Las colas tienen un tiempo de ejecución por trabajo limitado, si el trabajo sobrepasa el tiempo límite, será eliminado.
- Existe un límite de memoria virtual máximo establecido por slot de 6G de RAM y de escritura en fichero de 20GB. Si se pasa de dichos límites, el trabajo será eliminado. Es posible sobrepasar cualquiera de estos límites, pero para ello será necesario indicarlo explícitamente en la definición del trabajo.

A.1.4.1. Colas disponibles

Existen dos colas disponibles en el clúster:

- **main.q**:
  - Irán todos los trabajos excepto los de tipo interactivo.
  - Se establecerá una duración máxima de los trabajos de 48 horas.
  - En esta cola estarán disponibles todos los nodos de computación del clúster.
  - Esta cola tendrá los entornos paralelos definidos que describiremos posteriormente.
- **interactive.q**:
  - Irán todos los trabajos de tipo interactivo.
  - Se establecerá una duración máxima de 6 horas por trabajo.
  - Esta cola tendrá un único nodo disponible.
  - Esta cola tendrá los entornos paralelos definidos que describiremos posteriormente.

A.1.4.2. Entornos paralelos disponibles

En cuanto a los entornos de paralelización disponibles son tres:

- **parallel**: este entorno localizará un nodo que pueda satisfacer la demanda de slots del trabajo y lo ejecutará.
- **orte**: este entorno, del mismo modo que el anterior, buscará un nodo con los slots libres solicitados y realizará la ejecución en ese nodo. Sin embargo, a diferencia del anterior, se empleará la integración con ORTE, necesaria para las tareas MPI. Permitirá la ejecución de programas que usen OpenMPI pero que demanden un número de slots menor que el tamaño disponible en el nodo.

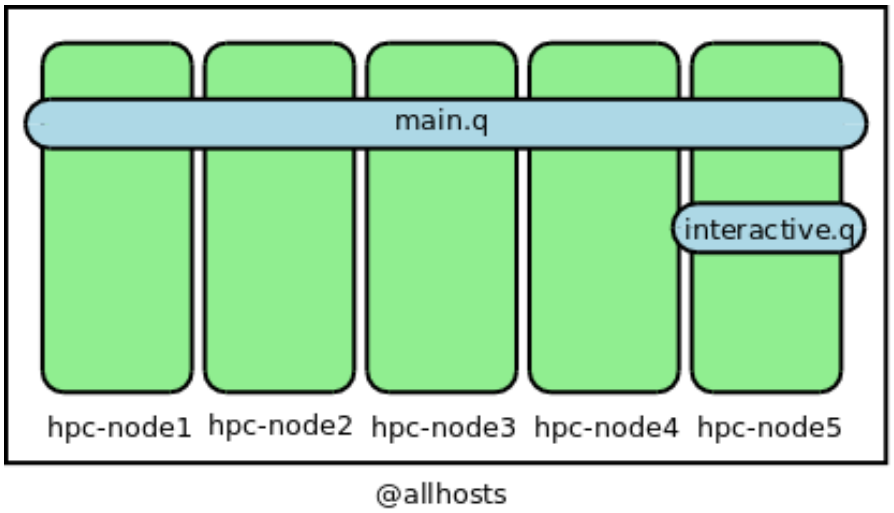


FIGURA A.2: Distribución de las colas de usuario en el clúster HPC

- **orte-20**: este entorno también estará integrado con ORTE, pero en este caso para la solicitud de slots de ejecución deberán suministrarse números múltiplos de 20. Esto obligará a llenar nodos enteros y maximizar el uso y rendimiento de los trabajos de este tipo.

Los entornos de ejecución parallel y orte estarán disponibles tanto en la cola main.q como en interactive.q. El entorno orte-20 sólo estará disponible para la cola main.q.

A.2. Usando el gestor recursos

A.2.1. Introducción

El objetivo del nodo de login es servir de entorno de desarrollo sobre el cual desarrollar nuestros programas, probar, etc. Pero no es un nodo en el cual podamos llevar a cabo ejecuciones de los programas más alla de pequeñas ejecuciones de prueba. Todo trabajo de cómputo deberá pasar por el software de gestor de recursos.

Los trabajos que podemos enviar pueden ser de cuatro tipos:

- Trabajos Batch: es el trabajo más simple, que bien podrá ser una secuencia de comandos.
- Trabajos Array: grupos de trabajos simpilares que pueden ejecutarse en paralelo pero que son completamente independientes uno de otro.
- Trabajos Paralelos: trabajos compuestos de tareas cooperativas y que deben ser ejecutados todos al mismo tiempo.

- Trabajos Interactivos: trabajos que proveen al usuario un login interactivo y que le permite ejecutar código que no puede ser enviado de manera sencilla como trabajo batch.

A.2.2. Funcionamiento básico

A.2.2.1. Enviando un trabajo

Salvo el envío de los trabajos interactivos, se usará el comando qsub. Junto a dicho comando pueden ir una serie de directivas particulares de nuestro tabajo y un fichero batch de definición del trabajo (también es posible llamar directamente a un binario pero está desaconsejado).

En dichos ficheros batch podrán indicarse también las directivas a gridengine empezando por # $. Un fichero básico sería:

```
lguillen@hpc-login:~/tutorial$ cat primertrabajo.sh
#!/bin/bash

#$ -N duerme
#$ -cwd

echo "Me duermo en ‘hostname‘"
sleep 100
echo "Me despierto"
```

Ahora podríamos ejecutarlo simplemente con:

```
lguillen@hpc-login:~/tutorial$ qsub primertrabajo.sh
Your job 1399 ("duerme") has been submitted
```

A.2.2.2. Comprobando el estado de los trabajos

Podemos ver el estado de los trabajos en la cola mediante qstat.

```
lguillen@hpc-login:~/tutorial$ qstat
job-ID  prior   name       user         state submit/start at     queue
                    slots ja-task-ID
-----------------------------------------------------------------------------------------------

   1399 0.50000 duerme     lguillen     r     05/06/2015 06:42:15 main.q@hpc-
    node1.office.cefca.     1
```

Vemos que el trabajo ha sido enviado a la instancia de cola `main.q@hpc-node1.office.cefca.es` y ocupa un slot de ejecución.

Los trabajos pueden presentar los siguientes estados:

- `qw`: `Queue waiting`. Está esperando a ser asignado a alguna cola.
- `t`: `Transferring`. El trabajo ha sido asignado y está siendo transferido a uno o varios nodos.
- `r`: `Running`. El trabajo está ejecutándose en el host de ejecución.
- `E`: `Error`. El trabajo ha fallado por algún motivo y continúa en ejecución en ejecución.
- `h`: `Hold`. El trabajo está siendo retenido por algún motivo. Lo más normal es que esté esperando a que otro trabajo termine.
- `R`: `Restarted`. El trabajo se ha reiniciado por algún motivo. El motivo más común son errores en el host de ejecución y el trabajo es enviado a otro host para ser procesado de nuevo.

Podemos ver más información sobre dicho trabajo.

```
lguillen@hpc-login:~/tutorial$ qstat -j 1399
==============================================================
job_number:                 1399
exec_file:                  job_scripts/1399
submission_time:            Wed May  6 06:42:13 2015
owner:                      lguillen
uid:                        10005
group:                      domain
gid:                        10000
sge_o_home:                 /home/CEFCA/lguillen
sge_o_log_name:             lguillen
sge_o_path:                 /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
    sbin:/bin:/usr/games:/usr/local/games
sge_o_shell:                /bin/bash
sge_o_workdir:              /home/CEFCA/lguillen/tutorial
sge_o_host:                 hpc-login
account:                    sge
cwd:                        /home/CEFCA/lguillen/tutorial
mail_list:                  lguillen@hpc-login.office.cefca.es
notify:                     FALSE
job_name:                   duerme
priority:                   -100
jobshare:                   0
env_list:
script_file:                primertrabajo.sh
usage    1:                 cpu=00:00:00, mem=0.00000 GBs, io=0.00003, vmem
    =32.469M, maxvmem=32.469M
scheduling info:            There are no messages available
```

Durante y tras la ejecución tendremos dos ficheros.

```
lguillen@hpc-login:~/tutorial$ cat duerme.e1399
lguillen@hpc-login:~/tutorial$ cat duerme.o1399
Me duermo en hpc-node1
Me despierto
```

Una vez se ha ejecutado no podemos ver

```
lguillen@hpc-login:~/tutorial$ qstat
lguillen@hpc-login:~/tutorial$
lguillen@hpc-login:~/tutorial$ qstat -j 1399
Following jobs do not exist:
1399
```

A.2.2.3. Información de contabilidad de nuestros trabajos

Podemos ver la información de contabilidad de todos nuestros trabajos con el comando `qacct`. Además podemos especificar el número de trabajo con el parámetro `-j`.

```
lguillen@hpc-login:~/tutorial$ qacct -j 1399
==============================================================
qname        main.q
hostname     hpc-node1.office.cefca.es
group        domain
owner        lguillen
project      NONE
department   defaultdepartment
jobname      duerme
jobnumber    1399
taskid       undefined
account      sge
priority     0
qsub_time    Wed May  6 06:42:13 2015
start_time   Wed May  6 06:42:15 2015
end_time     Wed May  6 06:43:55 2015
granted_pe   NONE
slots        1
failed       0
exit_status  0
ru_wallclock 100
ru_utime     0.008
ru_stime     0.010
ru_maxrss    1772
ru_ixrss     0
ru_ismrss    0
ru_idrss     0
ru_isrss     0
ru_minflt    1990
ru_majflt    0
ru_nswap     0
ru_inblock   16
ru_oublock   24
ru_msgsnd    0
ru_msgrcv    0
ru_nsignals  0
ru_nvcsw     42
ru_nivcsw    6
cpu          0.017
mem          0.000
```

```
io            0.000
iow           0.000
maxvmem       32.469M
arid          undefined
```

A.2.2.4. Eliminando un trabajo

Podremos eliminar un trabajo nuestro mediante el comando `qdel` seguido del identificador del trabajo.

A.2.2.5. Redirigiendo la salida

Suele ser habitual que queramos que los ficheros de salida en unos directorios específicos, para ello crearemos los directorios en los cuales se estraerá e indicaremos las directivas `e` y `o`:

```
lguillen@hpc-login:~/tutorial$ cat primertrabajo.sh
#!/bin/bash

#$ -N duerme
#$ -cwd
#$ -e $HOME/tutorial/error
#$ -o $HOME/tutorial/output

echo "Me duermo en ‘hostname‘"
sleep 100
echo "Me despierto"
```

A.2.2.6. Enviando un email

También suele ser habitual que queramos enviarnos un email cuando se inicie el trabajo, cuando se acabe o en cualquier estado de error. Para ello indicaremos el email mediante la directiva M y los estados en los que se nos notificará con la directiva m. Los estados pueden ser:

- `b`: `Begin`. Se envía el email cuando el proceso entra en el estado r.
- `e`: `End`. Se envía el email cuando el trabajo se desregistra del scheduler y no reporta ningún error (no hay estado `E`).
- `a`: `Aborted` or `rescheduled`. Se envía el email si el trabajo es abortado o replanificado (aparece el estado `E`).

- `s`: `Suspended`. Se envía el email si el trabajo ha sido suspendido (aparece el estado `s`).

```
lguillen@hpc-login:~/tutorial$ cat primertrabajo.sh
#!/bin/bash

#$ -N duerme
#$ -cwd
#$ -e $HOME/tutorial/error
#$ -o $HOME/tutorial/output
#$ -M lguillen@cefca.es
#$ -m bea

echo "Me duermo en ‘hostname‘"
sleep 100
echo "Me despierto"
```

A.2.2.7. Aumentando la prioridad

Todos los trabajos se envían por defecto con una prioridad -100 (así está configurado en el nodo de login). Esto permite que el usuario pueda modificar la prioridad hasta la prioridad 0. Si deseamos ejecutar el trabajo con la máxima prioridad permitida ejecutaremos:

```
lguillen@hpc-login:~/tutorial$  qsub -p 0 primertrabajo.sh
```

Nótese que hemos especificado la prioridad como parámetro del trabajo aunque podríamos haber metido esta directiva como `\#$ -p 0` en el fichero del trabajo.

A.2.2.8. Opciones por defecto

Puede ser conveniente crearnos un fichero .sge_request en nuestro HOME con algunos parámetros por defecto para no estar especificándolos siempre en nuestros `trabajos`. Por ejemplo:

```
lguillen@hpc-login:~$  cat ~/.sge_request
-M lguillen@cefca.es
-m bea
```

A.2.2.9. Monitorizando los host

Podemos ver el estado de carga de los host del cluster mediante el comando `qhost`.

```
lguillen@hpc-login:~/tutorial$ qhost
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
hpc-node1.office.cefca.es lx26-amd64     40  0.01  125.9G    1.0G     0.0     0.0
hpc-node2.office.cefca.es lx26-amd64     40  0.01  125.9G 1018.5M     0.0     0.0
hpc-node3.office.cefca.es lx26-amd64     40  0.01  125.9G    1.1G     0.0     0.0
hpc-node4.office.cefca.es lx26-amd64     40  0.01  125.9G    1.0G     0.0     0.0
hpc-node5.office.cefca.es lx26-amd64     20  0.01  125.9G    1.0G     0.0     0.0
```

También podemos ver el reparto de los trabajos entre los host y los slots consumidos mediante `qstat -f`.

```
lguillen@hpc-login:~/tutorial$ qstat -f
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
interactive.q@hpc-node5.office IP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node1.office.cefca. BP    0/1/20          0.07     lx26-amd64
   1399 0.50000 duerme     lguillen     r     05/06/2015 06:42:15     1
---------------------------------------------------------------------------------
main.q@hpc-node2.office.cefca. BP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node3.office.cefca. BP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node4.office.cefca. BP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node5.office.cefca. BP    0/0/20          0.01     lx26-amd64
```

A.2.2.10. Especificando recursos

En nuestro cluster se asignan por defecto 6G de RAM por slot de ejecución. Esto quiere decir que si deseamos usar más memoria deberemos solicitarlo a explícitamente. Esto debe solicitarse con el parámetro `-l` seguido del recurso, que en nuestro caso es `h_vmem` quedando:

```
lguillen@hpc-login:~/tutorial$  qsub -l h_vmem=10G primertrabajo.sh
```

En el cluster se ha especificado también un límite de escritura para los ficheros de 20G si nuestro trabajo va a escribir ficheros más grandes, deberemos especificarlo.

Nótese que estamos pidiendo trabajos de un slot de ejecución o un procesador. Para pedir más procesadores necesitaremos especificar un entorno paralelo, como se verá posteriormente. En el caso de usar un entorno paralelo, el consumo de h_vmem que especifiquemos será por slot y no el total que queremos del trabajo.

A.2.2.11. Dependencia entre trabajos

Podemos establecer dependencia entre trabajos mediante el parámetro -hold_jid. Por ejemplo crearemos un trabajo que dependerá del anterior de la siguiente manera:

```
lguillen@hpc-login:~/tutorial$ cat segundotrabajo.sh
#!/bin/bash

#$ -N depende
#$ -hold_jid duerme
#$ -cwd
#$ -e $HOME/tutorial/error
#$ -o $HOME/tutorial/output

echo "Por fin me ejecuto"
```

Si lanzamos ambos trabajos, vemos el estado `h` en el trabajo *depende*.

```
lguillen@hpc-login:~/tutorial$ qstat
job-ID  prior   name       user         state submit/start at     queue
                     slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------

   1403 0.50000 duerme     lguillen     r     05/06/2015 07:23:45 main.q@hpc-
    node1.office.cefca.     1
   1404 0.00000 depende    lguillen     hqw   05/06/2015 07:24:00
                         1
```

A.2.2.12. Ejecutando un trabajo interactivo

En ocasiones querremos lanzar un trabajos interactivos, para ello bastará con ejecutar el comando `qlogin`.

```
lguillen@hpc-login:~$ qlogin
local configuration hpc-login.office.cefca.es not defined - using global
    configuration
Your job 1410 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 1410 has been successfully scheduled.
Establishing builtin session to host hpc-node5.office.cefca.es ...
groups: cannot find name for group ID 65436
```

Tendremos a nuestra disposición una shell interactiva y un trabajo asignado. Si deseamos tener más memoria deberemos especificarlo con -l h_vmem=XXG. Puede existir también la posibilidad de querer lanzar programas interactivos gráficos. Para ello debemos habernos conectado al nodo de login por ssh especificando la opción -X.

```
$ ssh -X -C lguillen@hpc-login.office.cefca.es
lguillen@hpc-login:~$ echo $DISPLAY
hpc-login:10.0
```

Deberemos exportar la variable DISPLAY que obtenemos en el nodo de login en el entorno creado por qlogin si deseamos utilizar programas que hagan uso de X11. Por ejemplo:

```
lguillen@hpc-login:~$ qlogin -l h_vmem=10G
local configuration hpc-login.office.cefca.es not defined - using global
    configuration
Your job 1409 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 1409 has been successfully scheduled.
Establishing builtin session to host hpc-node5.office.cefca.es ...
groups: cannot find name for group ID 65435
lguillen@hpc-node5:~$ xterm
xterm: Xt error: Can't open display:
xterm: DISPLAY is not set
lguillen@hpc-node5:~$ export DISPLAY=hpc-login:10.0
lguillen@hpc-node5:~$ xterm
```

A.2.2.13. Ejecutando arrayjobs

Los arrayjobs nos permiten ejecutar y distribuir programas cuya paralelización la hagamos por la entrada de datos y sean independientes los unos de los otros. Para ello usamos el parámetro `-t` e indicaremos el rango de `SGE\_TASK\_ID` que se generarán.

```
lguillen@hpc-login:~/tutorial$ cat arrayjob.sh
#!/bin/bash

#$ -N procesa_array
#$ -t 1-40
#$ -cwd
#$ -e $HOME/tutorial/error
#$ -o $HOME/tutorial/output

echo "Me Ejecuto en `hostname` y soy $SGE_TASK_ID"
sleep 10
```

En el ejemplo estamos generando 40 trabajos que generarán un TASK_ID independiente en cada proceso de 1 a 40. Esto puede ser muy útil para especificar ficheros de entrada diferentes a nuestro programa. Si ejecutamos qstat vermos que ejecuta múltiples tareas en dos nodos, realizando una política fillup.

```
lguillen@hpc-login:~/tutorial$ qstat
job-ID  prior   name       user         state submit/start at     queue
                    slots ja-task-ID
-----------------------------------------------------------------------------------------------

   1405 0.50000 procesa_ar lguillen     r     05/06/2015 07:30:00 main.q@hpc-
    node2.office.cefca.     1 1
   1405 0.50000 procesa_ar lguillen     t     05/06/2015 07:30:00 main.q@hpc-
    node2.office.cefca.     1 2
```

```
   1405 0.50000 procesa_ar lguillen     t     05/06/2015 07:30:00 main.q@hpc-
    node2.office.cefca.     1 3
        ....
```

A.2.2.14. Ejecutando en entornos paralelos

Para ejecutar trabajos que requieran múltiples unidades de procesamiento deberemos usar un entorno paralelo. En nuestro clúster tenemos tres disponibles: parallel, orte y orte-20. Todos los trabajos de tipo OpenMP, multihilo, que usen librerías paralelas de python, etc. deberán usar el entorno parallel.

Para solicitar un entorno paralelo se realiza con el parámetro `-pe` seguido del nombre del entorno paralelo y de los slots pedidos de ejecución. Además tendremos en la variable `NSLOTS` el número que hemos pedido que podrá ser de utilidad por ejemplo para definir la variable `OMP_NUM_THREADS` o el número de procesos pasados a un `mpirun`.

Este script en bash nos servirá de ejemplo para el uso del parámetro pe y de la variable NSLOTS. NO es un ejemplo de cómo deben ser nuestros programas paralelos.

```
lguillen@hpc-login:~/tutorial$ cat paralelo.sh
#!/bin/bash

#$ -N paralelo
#$ -pe parallel 4
#$ -cwd
#$ -e $HOME/tutorial/error
#$ -o $HOME/tutorial/output

date
for i in `seq 1 $NSLOTS`; do
        sleep 20 &
done
for pid in `jobs -p`; do
        echo "Ejecutando en paralelo pid $pid"
done

for pid in `jobs -p`; do
        wait $pid
done

date
```

Podemos ver cómo se han distribuido los slots. En caso de los entornos parallel y orte irán todos al nodo. En orte-20 distribuirá los trabajos entre nodos completos.

```
lguillen@hpc-login:~/tutorial$ qstat -f
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
```

```
interactive.q@hpc-node5.office IP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node1.office.cefca. BP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node2.office.cefca. BP    0/4/20          0.01     lx26-amd64
   1407 0.50000 paralelo   lguillen     r     05/06/2015 07:46:00     4
---------------------------------------------------------------------------------
main.q@hpc-node3.office.cefca. BP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node4.office.cefca. BP    0/0/20          0.01     lx26-amd64
---------------------------------------------------------------------------------
main.q@hpc-node5.office.cefca. BP    0/0/20          0.01     lx26-amd64
```

Podemos ver la salida de la ejecución.

```
lguillen@hpc-login:~/tutorial/output$ cat paralelo.*
Wed May  6 07:46:00 CEST 2015
Ejecutando en paralelo pid 41621
Ejecutando en paralelo pid 41622
Ejecutando en paralelo pid 41623
Ejecutando en paralelo pid 41624
Wed May  6 07:46:20 CEST 2015
```

A.2.2.15. Ejecutando en entorno paralelo orte-20

El uso del entorno orte-20 requiere un trato especial. Este entorno está diseñado para poder pedir nodos enteros y lanzar trabajos grandes que requieran del paradigma de memoria distribuida. Si deseamos un nodo pediremos 20 slots, si deseamos dos nodos pediremos 40, etc.

En el siguiente ejemplo crearemos un trabajo en el cual se utilizarán dos nodos y distribuirá 20 procesos mpi en cada uno de los nodos.

```
lguillen@hpc-login:~/tutorial$ cat orte20.sh
#!/bin/bash

#$  -N hello_orte20
#$  -pe orte-20 40
#$  -cwd
#$  -e $HOME/tutorial/error
#$  -o $HOME/tutorial/output



mpirun -np  $NSLOTS $HOME/tutorial/mpihello
```

A.3. Gestionando el entorno con modules

A.3.1. Introducción

Durante su inicialización la shell cargará una serie de variables de entorno, algunas de estas variables estarán relacionadas con cómo se encontrarán los binarios o cómo el enlazador dinámico encontrará las librerías. Será posible tener múltiples versiones en el sistema de un binario o librería y, mediante la gestión de dichas variables, indicar qué versión deseamos utilizar. Es posible que los usuarios gestionen esto a mano, pero suele ser habitual dotar de un software específico para realizar este tipo de tarea. A este software se le suele denominar "módulos de entornoz es muy popular en los HPC. Este tipo de software permite definir módulos que contienen la información necesaria para configurar el entorno de la shell para determinadas aplicaciones, librerías y versiones. Existen varias implementaciones:

- environment-modules-c
- environment-modules-tcl
- lmod

Aunque cada una tiene su propio lenguaje de definición de módulos, todas ofrecen la misma interfaz de uso desde el punto de vista del usuario.

En nuestro cluster tenemos instalada y configurada la implementación de environment-modules-tcl que permite el uso de scripts tcl.

A.3.2. Uso

A.3.2.1. Listado de módulos

Si tecleamos el comando module sin parámetros veremos todas las opciones que nos permite. Podremos listar los módulos disponibles en el clúster mediante el comando `module avail`.

Como podemos ver, los módulos en nuestro clúster se encuentran agrupados por el nombre del software. Podemos limitar la búsqueda a un software en particular.

```
lguillen@hpc-login:~$ module avail FFTW

----- /usr/local/Cluster-software/modules/all -----
FFTW/2.1.5-gompi-1.5.12-no-OFED
```

```
FFTW/3.3.3-gompi-1.5.12-no-OFED
FFTW/3.3.4-gompi-1.5.14-no-OFED
```

A.3.2.2. Carga de módulos

Vemos que existen varias versiones disponibles de FFTW. Podemos ver que, además de la versión, se indican las versiones del compilador o toolchain con el que ha sido compilado.

```
lguillen@hpc-login:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
    local/games
lguillen@hpc-login:~$ module load FFTW
lguillen@hpc-login:~$ echo $PATH
/usr/local/Cluster-software/software/FFTW/3.3.4-gompi-1.5.14-no-OFED/bin:/usr/
    local/Cluster-software/software/OpenMPI/1.6.5-GCC-4.8.2-no-OFED/bin:/usr/
    local/Cluster-software/software/hwloc/1.8.1-GCC-4.8.2/bin:/usr/local/Cluster-
    software/software/GCC/4.8.2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr
    /bin:/sbin:/bin:/usr/games:/usr/local/games''
```

Podemos cargar un módulo usando module load. Si no especificamos una versión, cogerá la última de la que dispone. Además, si el módulo requiere a su vez de que otros módulos estén cargados, cargará todos los módulos necesarios.

A.3.2.3. Listado de módulos cargados

Podemos listar los módulos cargados

```
lguillen@hpc-login:~$ module list
Currently Loaded Modulefiles:
  1) GCC/4.8.2
  2) hwloc/1.8.1-GCC-4.8.2
  3) OpenMPI/1.6.5-GCC-4.8.2-no-OFED
  4) gompi/1.5.14-no-OFED
  5) FFTW/3.3.4-gompi-1.5.14-no-OFED
```

A.3.2.4. Descargar un módulo

Podemos descargar un módulos con unload.

```
lguillen@hpc-login:~$ module unload FFTW
lguillen@hpc-login:~$ module list
Currently Loaded Modulefiles:
  1) GCC/4.8.2                          3) OpenMPI/1.6.5-GCC-4.8.2-no-OFED
  2) hwloc/1.8.1-GCC-4.8.2              4) gompi/1.5.14-no-OFED
```

A.3.2.5. Reemplazar un módulo

Podemos reemplazar un módulo con switch, teniendo en cuenta que el nuevo módulo cargará también todas sus dependencias.

```
lguillen@hpc-login:~$ module load FFTW
lguillen@hpc-login:~$ module list
Currently Loaded Modulefiles:
  1) GCC/4.8.2                          3) OpenMPI/1.6.5-GCC-4.8.2-no-OFED   5)
    FFTW/3.3.4-gompi-1.5.14-no-OFED
  2) hwloc/1.8.1-GCC-4.8.2              4) gompi/1.5.14-no-OFED
lguillen@hpc-login:~$ module switch FFTW/2.1.5-gompi-1.5.12-no-OFED
lguillen@hpc-login:~$ module list
Currently Loaded Modulefiles:
  1) GCC/4.8.2                          4) gompi/1.5.14-no-OFED              7)
    gompi/1.5.12-no-OFED
  2) hwloc/1.8.1-GCC-4.8.2              5) GCC/4.8.1                         8)
    FFTW/2.1.5-gompi-1.5.12-no-OFED
  3) OpenMPI/1.6.5-GCC-4.8.2-no-OFED    6) OpenMPI/1.6.5-GCC-4.8.1-no-OFED
```

A.3.2.6. Cargar módulos automáticamente al inicio de sesión

Para ello bastará crear un fichero .modulerc en nuestro home.

```
lguillen@hpc-login:~$ cat .modulerc
#%Module
module load FFTW/3.3.4-gompi-1.5.14-no-OFED
```

A.3.2.7. Usando módulos en trabajos enviados al gestor de recursos

Aunque el comando qsub mantiene el entorno que se tenía en el trabajo, para evitar problemas es recomendable realizar la carga de manera explícita en el fichero del trabajo.

```
#$ -cwd
#$ -N mitrabajo
#$ -S /bin/bash

module load FFTW/3.3.4-gompi-1.5.14-no-OFED
$HOME/mitrabajo
```

A.3.2.8. Creando nuestros propios módulos

Puede ser útil para el usuario crear módulos propios para la carga automatizada de variables de entorno o el uso de software compilado por el usuario. Para ello simplemente:

1 Haremos un `export MODULEPATH=\$ HOME/modules:\$ {MODULEPATH}` y lo añadiremos al nuestro `.bashrc`.

2 Crearemos un directorio modules en nuestro home en el que dejaremos nuestros propios módulos.

Para escribir nuestro propio módulo podemos inspirarnos en alguno de los módulos que hay en `/usr/local/Cluster-software/modules/all` .

http://modules.sourceforge.net/

Anexo B

Problema con el driver hpsa y MSA 2040 SAS

Tras la instalación del sistema operativo del nodo cabecera y la creación de la configuración en la MSA, no se veían los dispositivos de bloque duplicados, por lo que no estaban viendose las dos rutas posibles a cada LUN.

```
root@hpc-master:~# lsscsi
[2:0:0:0]    disk    HP       LOGICAL VOLUME   6.00  /dev/sda
[2:3:0:0]    storage HP       P420i            6.00  -
[3:0:0:1]    disk    HP       MSA 2040 SAS     G105  /dev/sdb
[3:0:0:2]    disk    HP       MSA 2040 SAS     G105  /dev/sdc
[3:3:0:0]    storage HP       P431             2.04  -
```

Tras investigar, parecía que el driver sí que veía cuatro dispositivos, pero a la hora de asignar el dispositivo lo hacía por duplicado, como podía verse de la salida del dmesg:

```
[    5.519222] scsi3 : hpsa
[    5.522078] hpsa 0000:07:00.0: RAID              device c3b3t0l0 added.
[    5.522079] hpsa 0000:07:00.0: Direct-Access     device c3b0t0l1 added.
[    5.522080] hpsa 0000:07:00.0: Direct-Access     device c3b0t0l2 added.
[    5.522081] hpsa 0000:07:00.0: Direct-Access     device c3b0t0l1 added.
[    5.522082] hpsa 0000:07:00.0: Direct-Access     device c3b0t0l2 added.
[    5.522167] scsi 3:3:0:0: RAID              HP       P431             2.04 PQ:
    0 ANSI: 5
[    5.522310] scsi 3:0:0:1: Direct-Access     HP       MSA 2040 SAS     G105 PQ:
    0 ANSI: 5
[    5.522456] scsi 3:0:0:2: Direct-Access     HP       MSA 2040 SAS     G105 PQ:
    0 ANSI: 5
[    5.522602] scsi 3:3:0:0: Attached scsi generic sg2 type 12
[    5.522739] sd 3:0:0:1: Attached scsi generic sg3 type 0
[    5.522817] sd 3:0:0:1: [sdb] 23417339904 512-byte logical blocks: (11.9 TB
   /10.9 TiB)
[    5.522858] sd 3:0:0:2: Attached scsi generic sg4 type 0
[    5.523007] sd 3:0:0:2: [sdc] 7805034496 512-byte logical blocks: (3.99 TB
   /3.63 TiB)
```

Tras obtener la versión del driver:

```
root@hpc-master:~#  cat /sys/module/hpsa/version
3.4.0-1
```

E investigar, el problema se encontraba en el fichero hpsa.c, concretamente en las líneas donde se definían los parámetros:

```
1797         /* It's a logical device */
1798         if (is_ext_target(h, device)) {
1799                 /* external target way, put logicals on bus 1
1800                  * and match target/lun numbers box
1801                  * reports, other smart array, bus 0, target 0, match lunid
1802                  */
1803                 hpsa_set_bus_target_lun(device,
1804                         1, (lunid >> 16) & 0x3fff, lunid & 0x00ff);
1805                 return;
1806         }
1807         hpsa_set_bus_target_lun(device, 0, 0, lunid & 0x3fff);
```

La función `is_ext_target` se alimentaba de una estructura de literales donde debería encontrarse el modelo de nuestra cabina.

```
1756 static unsigned char *ext_target_model[] = {
1757         "MSA2012",
1758         "MSA2024",
1759         "MSA2312",
1760         "MSA2324",
1761         "P2000 G3 SAS",
1762         NULL,
1763 };
```

En versiones posteriores parecía que el problema estaba resuelto con la inclusión del mismo, así que descargué la última versión de hpsa, la 3.4.8 y procedí a compilar el driver utilizando dkms, de forma que se compilara automáticamente con cada actualización de kernel y empaquetarlo en un deb.

Instalación de dkms

```
root@hpc-master:~#  apt-get install debhelper dkms build-essential linux-headers-‘uname -r‘
```

Descarga del driver y creación de módulo dkms.

```
root@hpc-master:~# cp hpsa-3.4.8-140.tar.bz2 /usr/src/
root@hpc-master:~# cd /usr/src/
root@hpc-master:/usr/src# tar jxf hpsa-3.4.8-140.tar.bz2
root@hpc-master:/usr/src/hpsa-3.4.8# cat dkms.conf
MAKE="make -C drivers/scsi KERNELDIR=/lib/modules/${kernelver}/build"
CLEAN="make -C drivers/scsi clean"
BUILT_MODULE_NAME=hpsa
BUILT_MODULE_LOCATION=drivers/scsi
DEST_MODULE_LOCATION="/updates"
```

```
PACKAGE_NAME=hpsa-dkms
PACKAGE_VERSION=3.4.8
REMAKE_INITRD="yes"
AUTOINSTALL="yes"
```

Agregación del módulo dkms y construcción.

```
root@hpc-master:/usr/src/hpsa-3.4.8# dkms add -m hpsa -v 3.4.8

Creating symlink /var/lib/dkms/hpsa/3.4.8/source ->
                                 /usr/src/hpsa-3.4.8

DKMS: add completed.
root@hpc-master:/usr/src/hpsa-3.4.8# dkms status
hpsa, 3.4.8: added
vboxhost, 4.3.14, 3.13.0-32-generic, x86_64: installed
vboxhost, 4.3.14, 3.13.0-48-generic, x86_64: installed
root@hpc-master:/usr/src/hpsa-3.4.8# dkms build -m hpsa -v 3.4.8

Kernel preparation unnecessary for this kernel.  Skipping...

Building module:
cleaning build area....
make KERNELRELEASE=3.13.0-48-generic -C drivers/scsi KERNELDIR=/lib/modules
    /3.13.0-48-generic/build....
cleaning build area....

DKMS: build completed.
```

Creación del paquete deb.

```
root@hpc-master:/usr/src/hpsa-3.4.8# dkms mkdsc -m hpsa -v 3.4.8 --source-only
Using /etc/dkms/template-dkms-mkdsc
copying template...
modifying debian/changelog...
modifying debian/compat...
modifying debian/control...
modifying debian/copyright...
modifying debian/dirs...
modifying debian/postinst...
modifying debian/prerm...
modifying debian/README.Debian...
modifying debian/rules...
copying legacy postinstall template...
Copying source tree...
Building source package... dpkg-source --before-build hpsa-dkms-3.4.8
 debian/rules clean
 dpkg-source -b hpsa-dkms-3.4.8
dpkg-source: warning: no source format specified in debian/source/format, see
    dpkg-source(1)
 dpkg-genchanges -S >../hpsa-dkms_3.4.8_source.changes
dpkg-genchanges: including full source code in upload
 dpkg-source --after-build hpsa-dkms-3.4.8
```

```
DKMS: mkdsc completed.
Moving built files to /var/lib/dkms/hpsa/3.4.8/dsc...
Cleaning up temporary files...
root@hpc-master:/usr/src/hpsa-3.4.8# dkms mkdeb -m hpsa -v 3.4.8 --source-only
Using /etc/dkms/template-dkms-mkdeb
copying template...
modifying debian/changelog...
modifying debian/compat...
modifying debian/control...
modifying debian/copyright...
modifying debian/dirs...
modifying debian/postinst...
modifying debian/prerm...
modifying debian/README.Debian...
modifying debian/rules...
copying legacy postinstall template...
Copying source tree...
Building binary package...dpkg-buildpackage: warning: using a gain-root-command
    while being root
 dpkg-source --before-build hpsa-dkms-3.4.8
 fakeroot debian/rules clean
 debian/rules build
 fakeroot debian/rules binary
 dpkg-genchanges -b >../hpsa-dkms_3.4.8_amd64.changes
dpkg-genchanges: binary-only upload - not including any source code
 dpkg-source --after-build hpsa-dkms-3.4.8


DKMS: mkdeb completed.
Moving built files to /var/lib/dkms/hpsa/3.4.8/deb...
Cleaning up temporary files...
```

Instalación del deb.

```
root@hpc-master:/usr/src/hpsa-3.4.8# cp /var/lib/dkms/hpsa/3.4.8/deb/hpsa-dkms_3
    .4.8_all.deb ~
root@hpc-master:~# rm -rf /var/lib/dkms/hpsa/
root@hpc-master:~# dpkg -i hpsa-dkms_3.4.8_all.deb
Seleccionando el paquete hpsa-dkms previamente no seleccionado.
(Leyendo la base de datos ... 98377 ficheros o directorios instalados actualmente
    .)
Preparing to unpack hpsa-dkms_3.4.8_all.deb ...
Unpacking hpsa-dkms (3.4.8) ...
Configurando hpsa-dkms (3.4.8) ...
Loading new hpsa-3.4.8 DKMS files...
First Installation: checking all kernels...
Building only for 3.13.0-48-generic
Building for architecture x86_64
Building initial module for 3.13.0-48-generic
Done.

hpsa:
Running module version sanity check.
 - Original module
```

```
 - Installation
   - Installing to /lib/modules/3.13.0-48-generic/updates/dkms/

depmod....

Backing up initrd.img-3.13.0-48-generic to /boot/initrd.img-3.13.0-48-generic.old
    -dkms
Making new initrd.img-3.13.0-48-generic
(If next boot fails, revert to initrd.img-3.13.0-48-generic.old-dkms image)
update-initramfs....

DKMS: install completed.
Processing triggers for initramfs-tools (0.103ubuntu4.2) ...
update-initramfs: Generating /boot/initrd.img-3.13.0-48-generic
```

Tras reiniciar el sistema... ¡voilà!

```
root@hpc-master:~# lsscsi
[2:0:0:0]    disk    HP       LOGICAL VOLUME   6.00  /dev/sda
[2:3:0:0]    storage HP       P420i            6.00  -
[3:1:1:1]    disk    HP       MSA 2040 SAS     G105  /dev/sdb
[3:1:1:2]    disk    HP       MSA 2040 SAS     G105  /dev/sdc
[3:1:2:1]    disk    HP       MSA 2040 SAS     G105  /dev/sdd
[3:1:2:2]    disk    HP       MSA 2040 SAS     G105  /dev/sde
[3:3:0:0]    storage HP       P431             2.04  -
```

Posteriormente se agregó el paquete al repositorio de software APT con `reprepro`.

Referencia empaquetado dkms [1]

[1] http://www.xkyle.com/building-linux-packages-for-kernel-drivers/

Anexo C

Problema con los paquetes del gestor de recursos gridengine

C.1. Problema con interfaz gráfica Qmon

El primer problema que existe con qmon está bien documentado es simplemente la necesidad de la existencia de unas fuentes en el servidor X al que vayamos a redirigir la interfaz. La salida que obtenemos al ejecutar qmon es la siguiente:

```
lguillen@hpc-login:~$ qmon
Warning: Cannot convert string "-adobe-courier-medium-r-*--14-*-*-*-m-*-*-*" to
    type FontStruct
Warning: Cannot convert string "-adobe-courier-bold-r-*--14-*-*-*-m-*-*-*" to
    type FontStruct
Warning: Cannot convert string "-adobe-courier-medium-r-*--12-*-*-*-m-*-*-*" to
    type FontStruct
X Error of failed request:  BadName (named color or font does not exist)
  Major opcode of failed request:  45 (X_OpenFont)
  Serial number of failed request:  384
  Current serial number in output stream:  395
```

Para solucionarlo simplemente:

```
# apt-get install xfstt
# apt-get install  xfonts-base xfonts-75dpi xfonts-100dpi
```

Pero el problema que da origen a este anexo es el siguiente: al realizar la instalación de la interfaz gráfica qmon mediante el paquete gridengine-qmon incluido en Ubuntu 14.04LTS presentaba un problema, y es que no conseguía dibujar los iconos de la aplicación. Puede verse en la figura como se puede ver en la figura C.1.

La salida que se obtenía por la shell era la siguiente:

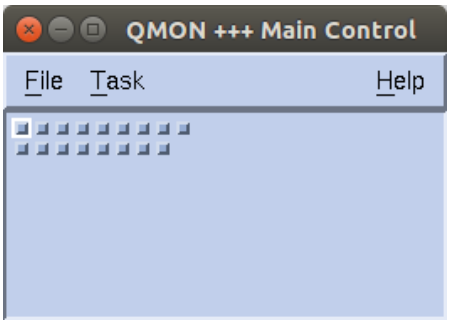


FIGURA C.1: Interfaz Qmon sin iconos

```
lguillen@hpc-login:~$ qmon
Warning: Cannot convert string "intro" to type Pixmap
Warning: Cannot convert string "toolbar_job" to type Pixmap
Warning: Cannot convert string "toolbar_queue" to type Pixmap
Warning: Cannot convert string "toolbar_submit" to type Pixmap
Warning: Cannot convert string "toolbar_cplx" to type Pixmap
Warning: Cannot convert string "toolbar_host" to type Pixmap
Warning: Cannot convert string "toolbar_cluster" to type Pixmap
Warning: Cannot convert string "toolbar_sched" to type Pixmap
Warning: Cannot convert string "toolbar_calendar" to type Pixmap
Warning: Cannot convert string "toolbar_user" to type Pixmap
Warning: Cannot convert string "toolbar_pe" to type Pixmap
Warning: Cannot convert string "toolbar_ckpt" to type Pixmap
Warning: Cannot convert string "toolbar_ticket" to type Pixmap
Warning: Cannot convert string "toolbar_prj" to type Pixmap
Warning: Cannot convert string "toolbar_rqs" to type Pixmap
Warning: Cannot convert string "toolbar_ar" to type Pixmap
Warning: Cannot convert string "toolbar_browser" to type Pixmap
Warning: Cannot convert string "toolbar_exit" to type Pixmap
```

Sin embargo, el programa funcionaba perfectamente, pero era especialmente molesto en algunas interfaces como en la que puede verse en la figura C.2.

Estuve mirando el problema en Internet y no había solución, todo el mundo indicaba que con 12.04 funcionaba y con 14.04 no. Tras investigar un poco, pude hacerlo funcionar compilándolo a mano con los mismos fuentes, por lo que parecía ser un problema del empaquetado.

Finalmente di, con el problema. En la versión 12.04 se usaba la librería lesstiff2 que es una implementación libre de Motiff. Sin embargo, con la liberación de Motiff, se empaquetó dicha librería y todos los programas que usaban Motiff y eran compilados contra Lesstiff se reempaquetaron. Así se hizo con la versión de qmon existente en 14.04, sin embargo no se eliminó un parche existente en el paquete que se aplicaba sobre los fuentes y que realizaba algunos cambios que permitían la compatibilidad con lesstiff. Por esto, para solucionar el problema, simplemente hay que eliminar el parche, realizar unas pocas modificaciones y generar el paquete.

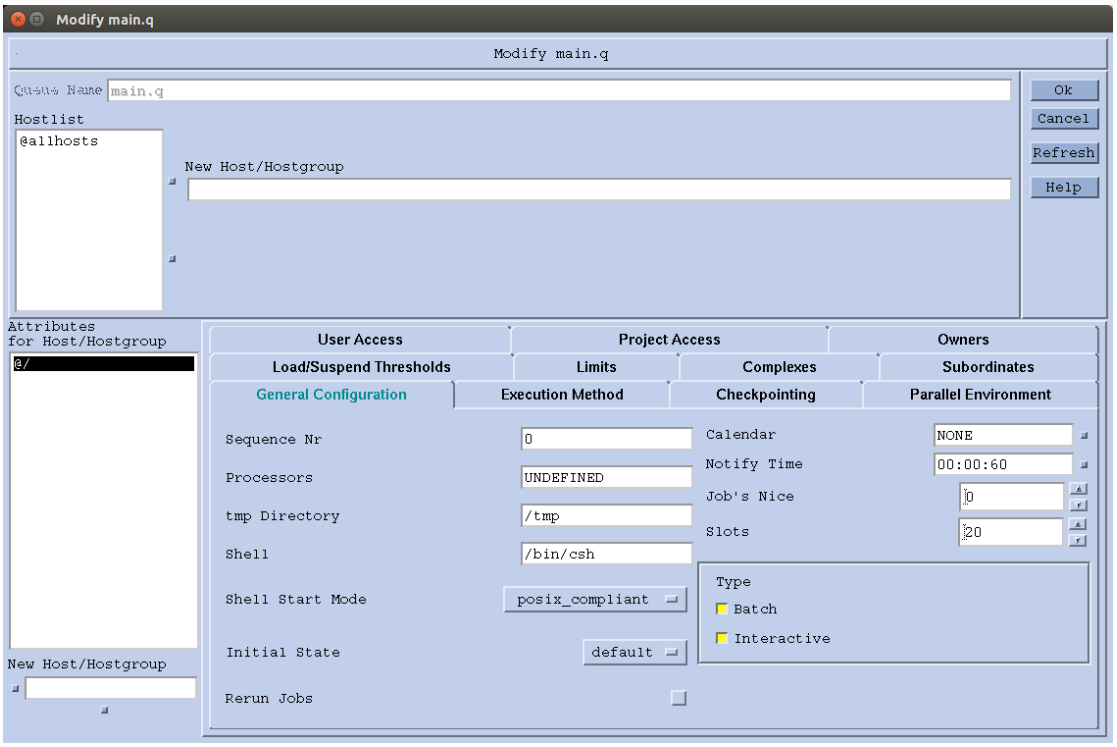


FIGURA C.2: Interfaz Qmon vista de colas iconos

En primer lugar instalaremos las dependencias de construcción del software.

```
# apt-get install debhelper csh groff libdb-dev libssl-dev
 libncurses5-dev libpam0g-dev libxt-dev libmotif-dev libxpm-dev libxmu-dev
 po-debconf quilt default-jdk ant ant-optional junit javacc libxft-dev
```

Después crearemos un directorio de trabajo y bajaremos las fuentes.

```
$ mkdir pkg
$ cp pkg
$ apt-get source gridengine-qmon
```

Procederemos a realizar el parcheado de fuentes necesario.

```
$ cd gridengine-6.2u5
$ rm debian/patches/030-qmon-lesstiff.diff
$ sed -i '/030-qmon-lesstiff.diff/d' debian/patches/series
$ export QUILT_PATCHES=debian/patches
$ quilt new 030-qmon-pixmaps.diff
El parche 030-qmon-pixmaps.diff está ahora arriba
$ quilt add source/clients/qmon/Q
El archivo source/clients/qmon/Q ha sido añadido al parche 030-qmon-pixmaps.diff
$ cat source/clients/qmon/Q
        ....
        Qmon*pixmapFilePath:   /usr/share/gridengine/pixmaps/%N.xpm
        ....
$quilt refresh
Parche actualizado 030-qmon-pixmaps.diff
```

En este punto habremos eliminado el parche conflictivo y creado un nuevo parche con el enlace correcto a los iconos, como el siguiente.

```
$ cat debian/patches/030-qmon-pixmaps.diff
Index: gridengine-6.2u5/source/clients/qmon/Q
===================================================================
--- gridengine-6.2u5.orig/source/clients/qmon/Q 2010-06-06 18:20:18.000000000
    +0200
+++ gridengine-6.2u5/source/clients/qmon/Q      2015-04-21 13:46:16.907407232
    +0200
@@ -305,7 +305,8 @@
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 !! remove comments for big toolbar icons and comment the following line
 !! Qmon*pixmapFilePath:  %R/qmon/PIXMAPS/big/%N.xpm:%R/qmon/PIXMAPS/%N.xpm
-Qmon*pixmapFilePath:   %R/qmon/PIXMAPS/%N.xpm
+!! Qmon*pixmapFilePath:   %R/qmon/PIXMAPS/%N.xpm
+Qmon*pixmapFilePath:   /usr/share/gridengine/pixmaps/%N.xpm
 Qmon*contextHelpFile:  qmon_help

 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Además será necesario que se instale la librería `libXltree.so` en el sistema.

```
$ cat debian/gridengine-qmon.install
....
source/3rdparty/qmon/LINUXAMD64_26/libXltree.so /usr/lib/
```

C.2. Problema con soporte JSV

No se incluyen los scripts de ejemplo que vienen en las fuentes y faltan algunos binarios que son necesarios para hacer funcionar el soporte JSV para bash.

```
$ cat debian/gridengine-master.install
....
debian/tmp/usr/utilbin/read_raw     /usr/lib/gridengine
debian/tmp/usr/utilbin/echo_raw     /usr/lib/gridengine

$ cat debian/gridengine-common.install
....
source/dist/util/resources/jsv /usr/share/gridengine
source/dist/util/arch          /usr/share/gridengine
```

C.3. Generación de los paquetes

Ya tenemos listo el paquete, ahora simplemente ejecutaremos `dch` para que cree una versión local del paquete con el sufijo cefca y `debuild` para generarlo.

```
$ dch --local cefca
$ debuid -us -uc
```

Ya tendremos los paquetes en el directorio pkg. Subiremos a nuestro repositorio local con Reprepro para que pueda instalarse mediante apt los siguientes paquetes:

- `gridengine-common_6.2u5-7.3cefca1_all.deb`
- `gridengine-master_6.2u5-7.3cefca1_amd64.deb`

Tras la instalación tendremos un Qmon con el problema solucionado, tal y como puede verse en la figura C.3.

FIGURA C.3: Interfaz Qmon con iconos

Y comprobamos que los scripts jsv están

```
root@hpc-master:~# ls /usr/share/gridengine/jsv/
jjsv.sh  jsv_include.sh  jsv_include.tcl  jsv.pl  JSV.pm  jsv.sh  jsv.tcl
    logging.properties
root@hpc-master:~# ls /usr/lib/gridengine/*raw
/usr/lib/gridengine/echo_raw  /usr/lib/gridengine/read_raw
```

Anexo D

Un vistazo a las LGTools

D.1. Lgvbox

D.1.1. Qué es lgvbox

Lgvbox es un conjunto de scripts que he desarrollado para simplificar el desarrollo de infraestructuras virtuales de pruebas. Se basa en la definición ficheros de especificación de máquina virtual en los cuales definimos las características de la máquina virtual y en ficheros de definición de infraestructuras con el listado de máquinas que las componen.

Mediante los ficheros de definición se podrán crear rápidamente las máquinas, indicando las interfaces de red necesarias, memoria, discos duros, etc. Para la definición de los discos se hace uso de un mecanismo de `disk builders` que ofrece lgvbox. Actualmente hay dos disk builders implementados: uno que hace clonaciones de discos duros virtuales existentes y otro que crea discos duros vacíos. Además proporciona un mecanismo de `autodeploy` básico (la imagen debe de tener soporte) que permite realizar de manera automática la configuración completa de la máquina tras su primer inicio.

D.1.2. Instalación y configuración inicial

Para la instalación de lgvbox será necesario tener previamente instaladas las lgbashlib bien en `/usr/local/lgbashlib` o bien en `/usr/lib/lgbashlib`. Por supuesto, será necesario tener instalado VirtualBox en el equipo. La presente versión funciona perfectamente con Virtualbox 4.3, así que se recomienda esta versión o superior. Además será necesaria la instalación del Extension Pack si deseamos utilizar la conexión a la consola.

Una vez tenemos instalado virtualbox, configuraremos un directorio en el que residirán las máquinas virtuales. En este ejemplo, se ha creado un directorio `/var/virtualmachines` en el que residirán las máquinas y en el que el usuario tiene privilegios de lectura y escritura.

```
# mkdir /var/virtualmachines
# mkdir /var/virtualmachines/.disks
# chown -R luis /var/virtualmachines
```

Se configurará virtualbox para que use ese directorio, tal y como aparece en la figura D.1.

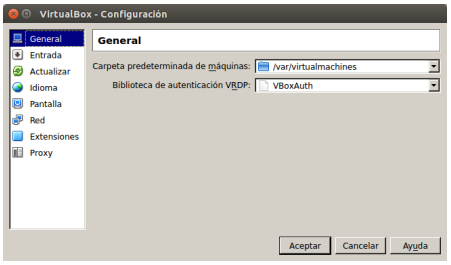


FIGURA D.1: Configuración de la ubicación de máquinas en VirtualBox

Una vez tenemos esto realizado, descomprimiremos lgvbox en la carpeta /usr/local/lgvbox y crearemos un enlace simbólico de `/usr/local/lgvbox/bin/lgvbox.sh` a `/usr/local/bin/lgvbox.sh`.

```
# tar zxf lgvbox.tar.gz
# mv lgvbox /usr/local
# cd /usr/local/bin
# ln -s ../lgbvox/lgvbox.sh lgvbox.sh
```

Después en el directorio del usuario crearemos un fichero de configuración

```
luis@oaj-ws-004:~$ cat .lgvboxconf

## lugar en el que almacenan las máquinas virtuales (crea directorio .disks)
VirtualMachinesDir=/var/virtualmachines

## lugar especificación de máquinas virtuales
VirtualMachinesSpecDir=$HOME/pfc-full/machines

## lugar especificación de escenarios
ScenariosSpecDir=$HOME/pfc-full/scenarios

## segundos a esperar entre inicio de máquina y máquina
ScenariosStartWait=15

## segundos a esperar entre apagado de máquina y máquina
ScenariosShutdownWait=10
```

D.1.3. Creación de máquinas virtuales plantillas

LGVbox se basa en plantillas de máquinas virtuales para el desarrollo de las infraestructuras. Una plantilla no es más que una máquina virtual con un sistema operativo base instalado. Por lo tanto

Hay que configurar la carpeta de almacenamiento de las máquinas virtuales.

En este caso vamos a crear una máquina virtual con ubuntu 14.04 server:

- Crearemos la máquina virtual, para ello le daremos al icono Nueva e indicaremos el nombre `ubuntuserver64` y tipo Ubuntu 64 bits.
- Le daremos la 512MB de RAM. Luego en las máquinas que creemos podremos modificar este parámetro.
- Diremos que crear un disco virtual de tipo VDI, Reservado dinámicamente, y muy importante: en la ubicación especificaremos que lo haga en el directorio `/var/virtualmachines/.disks`, como puede verse en la imagen D.2.
- Iniciaremos la máquina e instalaremos el sistema operativo con las opciones por defecto.

FIGURA D.2: Configuración del disco duro virtual de la plantilla en VirtualBox

- Tras crear la máquina virtual actualizaremos todo el sistema e instalaremos las Guest Additions.
- Reiniciaremos y borraremos la cache con apt-get clean.
- Nos aseguraremos de que no hay ningún cd virtual insertado.
- Instalaremos las keys públicas ssh que deseemos en los usuarios.
- Si hemos instalado el servidor ssh, deberemos detener el servicio ssh y eliminar las claves rsa del host. En el despliegue posterior de máquinas basadas en esa plantilla, deberemos asegurarnos de regenerar las keys ssh.

- Agregaremos el siguiente código al fichero `/etc/rc.local`.

```
if [ -f /media/sf_autodeploy/autodeploy.sh ]; then

        [ -f /root/.autodeploy ] && exit 0
        touch /root/.autodeploy
        sh /media/sf_autodeploy/autodeploy.sh >/root/autodeploy.log 2>&1
        echo "$?" > /root/autodeploy.exit

fi

exit 0
```

- Luego en el fichero `/etc/sudoers` modificaremos la línea de sudo: `\%sudo ALL=(ALL:ALL ) NOPASSWD: ALL`
- Apagaremos la máquina.

Cuando hemos llegado a este punto, ya podemos usar el disco creado de la máquina como plantilla. Podremos realizar plantillas de los distintos sistemas operativos. En el caso de Windows, deberemos de asegurarnos de ejecutar `sysprep`.

D.1.4. Definición de máquinas y escenarios

D.1.4.1. Ejemplo de definición de máquina

El modo más sencillo de ver cómo se definen las máquinas y ver las opciones es con un ejemplo como el que sigue.

```
luis@oaj-ws-004:~$ cat pfc-full/machines/router.cfg

## configuración general
config_machine() {
        ## nombre de la máquina en virtualbox
        cfg_machine_name="router-pfc-full"
        ## tipo de sistema operativo, ver documentación virtualbox cadenas
    aceptadas
        cfg_machine_so="Ubuntu_64"
        ## memoria RAM en MB
        cfg_machine_memory="512"

        ## tipo de arranque
        cfg_machine_start_type="headless"
#       cfg_machine_start_type="gui"

        ## puerto para conexión vrdp con máquina usado por attach_console
        cfg_machine_vrdeport=5001
```

```
        ## si arranca por pxe
        cfg_machine_pxeboot="false"

        ## usuario dentro de la máquina virtual usado por exec_in_vm
        cfg_machine_execUser="cefcadm"
        cfg_machine_execPasswd="cefcadm"
}

## configuración de interfaces de red
config_ifaces() {
        #iface 1
        cfg_iface_mode[1]="nat"

        #iface 2
        cfg_iface_mode[2]="intnet"
        cfg_iface_modeOpt[2]="services_net"

        #iface 3
        cfg_iface_mode[3]="intnet"
        cfg_iface_modeOpt[3]="machines_net"

        #iface 4
        cfg_iface_mode[4]="intnet"
        cfg_iface_modeOpt[4]="hpc_net"
        cfg_iface_mac[4]="08002773AC69"
}

## configuración de redirección de puertos desde el host
## el tipo de interfaz asociado debe ser obligatoriamente de tipo nat
config_port_forwarding() {
        #forward 1
        ## numero de la interfaz de red
        cfg_forward_iface[1]=1
        ## nombre de la redireccion
        cfg_forward_name[1]="guestssh"
        ## protocolo
        cfg_forward_protocol[1]="tcp"
        ## puerto en el anfitrión
        cfg_forward_hostPort[1]="2222"
        ## puerto en la máquina virtual
        cfg_forward_vmPort[1]="22" #opcional
        cfg_forward_hostIp[1]="" # opcional
        cfg_forward_vmIp[1]="" # opcional
}

## controladora de almacenamiento
config_storage() {
        #storage1
        cfg_storage_name[1]="SATA"
        cfg_storage_type[1]="sata"
        cfg_storage_bootable[2]="on"
}

## discos duros que se crearán en la máquina
config_hdisks() {
```

```
        ## ejemplo que crea un disco duro a partir de un disco existente
        #disk1
        cfg_hdisk_name[1]="${cfg_machine_name}_disk1.vdi"
        cfg_hdisk_storageName[1]="SATA"
        cfg_hdisk_port[1]=0
        cfg_hdisk_device[1]=0
        cfg_hdisk_builder[1]="clone_disk.sh ubuntuserver64.vdi ${cfg_machine_name
    }_disk1.vdi"

        ## ejemplo que crea un disco duro vacío de 10GB
        #disk2
        cfg_hdisk_name[2]="${cfg_machine_name}_disk2.vdi"
        cfg_hdisk_storageName[2]="SATA"
        cfg_hdisk_port[2]=1
        cfg_hdisk_device[2]=0
        cfg_hdisk_builder[2]="make_empty_disk.sh ${cfg_machine_name}_disk2.vdi
    10000"
}

## configura carpetas compartidas con el host
config_shared_folders() {
        ## ejemplo que monta la carpeta especial autodeploy
        cfg_shared_name[1]="autodeploy"
        cfg_shared_hostPath[1]="$HOME/deployments/${cfg_machine_name}"
        cfg_shared_automount[1]="true"
        cfg_shared_readonly[1]="false"
}
```

D.1.4.2. Ejemplo de definición un escenario

Una infraestructura es un simple listado de máquinas.

```
luis@oaj-ws-004:~$ cat pfc-full/scenarios/test-scenario.list
router
dc1
dc2
dhcp1
dhcp2
```

D.1.5. Uso de lgvbox

El uso de lgvbox debe ir seguido de la acción que deseamos realizar y los argumentos requeridos para dicha acción.

```
lgvbox.sh accion argumentos
```

A continuación se definirán las acciones, pero antes cabe remarcar que hay que distinguir entre nombre de máquina virtual en lgvbox del nombre de máquina virtual en Virtualbox.

El nombre de máquina virtual para lgvbox es el nombre del fichero usado en la definición. El fichero del ejemplo tenía el nombre de `router.cfg` y la variable `cfg_machine_name` con el nombre `router-pfc-full`. Pues bien, el nombre de máquina que usaremos en lgvbox y que por lo tanto utilizaremos es `router`, mientras que para virtualbox es `router-pfc-full`.

Las acciones que se podrán realizar son las siguientes:

- **create_vm**: crea una máquina virtual. Ejemplo: `lgvbox.sh create_vm router`
- **create_scenario**: crea el escenario. Ejemplo: `lgvbox.sh create_scenario test-scenario`
- **purge_vm**: elimina máquina virtual y discos duros asociados. El fichero con la definición NO se elimina. Ejemplo: `lgvbox.sh purge_vm router`
- **purge_scenario**: elimina todas las máquinas virtuales del escenario. Los ficheros con las definiciones NO se eliminan. Ejemplo: `lgvbox.sh purge_scenario test`
- **start_vm**: inicia la máquina virtual en el modo especificado en la definición.
- **start_scenario**: incia todas las máquinas del escenario en la secuencia indicada.
- **attach_console**: si se ha configurado el vrdport, lanzará un rdesktop contra el puerto especificado en la definición de la máquina virtual. Hay que tener cuidado de no tener vrdport repetidos.
- **exec_in_vm**: ejecuta el comando pasado en la máquina virtual como el usuario de la definición.
- **shutdown_vm**: realiza un apagado ordenado de la máquina virtual.
- **poweroff_vm**: realiza un apagado forzoso de la máquina virtual.
- **shutdown_scenario**: realiza un apagado ordenado de las máquinas virtuales del escenario en el orden inverso en el que aparecen en la definición.
- **take_snapshot_vm**: toma una snapshot de la máquina virtual. Ejemplo: `lgvbox.sh take_snapshot_vm router INICIAL.`
- **take_snapshot_scenario**: toma un snapshot de todas las máquinas virtuales del escenario: `lgvbox.sh take_snapshot_scenario router INICIAL.`
- **restore_snapshot_vm**: restaura el snapshot de la máquina virtual con el nombre pasado.
- **restore_snapshot_scenario**: restaura el snapshot de todas las máquinas virtuales del escenario.

- **delete_snapshot_vm**: elimina el snapshot de la máquina virtual con el nombre pasado.
- **delete_snapshot_scenario**: elimina el snapshot de todas las máquinas virtuales del escenario con el nombre pasado.

D.2. Lgsetup

D.2.1. Qué es lgsetup

Lgsetup proporciona un marco de trabajo sobre el que desarrollar automatizaciones basadas en secuencias de scripts. Aunque puede ser utilizado para desarrollar instaladores de propósito general, se ha concebido especialmente para realizar configuraciones de despliegues de sistemas.

Para llevar a cabo su trabajo define una unidad de automatización a la que se llamada *helper* que no es más que un script con una interfaz determinada. Los helpers son scripts que deben hacer una única cosa y que pueden ser parametrizados siguiendo la interfaz definida de helper. Un helper puede estar en cualquier lenguaje, pero se proporciona un mecanismo sencillo para la creación de helpers en bash.

El marco Lgsetup incluye el instalador (también llamado lgsetup) que es el que se encargará de ejecutar la secuencia de helpers que compondrá nuestra automatización. Para ello se alimentará de dos tipos de ficheros: ficheros de secuencia y ficheros de configuración. En el proceso de ejecución, lgsetup leerá los ficheros de secuencia e irá ejecutando uno a uno los helpers indicados en él. Además llamará a la interfaz del helper indicando el fichero de configuración correspondiente a la secuencia. Si durante la ejecución de la secuencia de helpers ocurriese un error, el instalador detendrá su ejecución. Si se volviese a invocar, volvería la ejecución de la secuencia a partir del helper en el cual dio el error.

Los helpers para una automatización pueden a su vez necesitar de plantillas, ficheros de certificados, etc e incluso puede existir la necesidad de compartir lógica entre helpers. Para eso se define el concepto de *componente*. Un componente es un conjunto de helpers relacionados que incluye librerías comunes, plantillas, etc.

Existirán ficheros de configuración, plantillas e incluso helpers que no sean lo suficientemente genéricos para incluirse en un componente pero sí que serán genéricos para un conjunto de instaladores. Para ello se defininen los *escenarios*.

Los instaladores se crearán a partir de lo que definimos como *paquetes*. Un paquete contendrá información acerca de su nombre, versión, etc, los componentes que incluirá,

los escenarios que incluirá y posibles ficheros de certificados, plantillas e incluso helpers específicos que no se podrán generalizar en un componente o en un escenario.

D.2.2. Creación de un paquete de instalación

Ya hemos visto en la parte de la memoria dedicada a la automatización del despliegue del clúster la creación de un paquete.

D.2.3. Ejecución de un instalador

Para instalar un paquete descomprimiremos el tar.gz. En el ejemplo hemos descomprimido sobre `/root/deployment` y tenemos:

```
root@hpc-node1:/root/deployment# ls
deploy_hpc-exec-node-pfc-full
info
```

El directorio contendrá el instalador en si y en el fichero info encontraremos información sobre el instalador:

```
root@hpc-node1:/root/deployment# cat info
package_name=deploy_hpc-exec-node-pfc-full
package_desc="Despliegue de node de ejecucion hpc de pruebas"
package_ver="0.1"
package_build="20150505-175458"
```

Si hacemos un listado del directorio etc contenido en el instalador vemos:

```
root@hpc-node1:/root/deployment# ls deploy_hpc-exec-node-pfc-full/etc/
global.cfg
lgdeploy_add_cron.passwd
samba_add_domain.passwd
stage1.cfg
stage1.run
stage2.cfg
stage2.run
stage3.cfg
stage3.run
```

Podemos comprobar que allí tenemos los dos tipos de ficheros de los que hemos hablado: ficheros de secuencia (los .run) y ficheros de configuración (los .cfg).

Si ejecutamos el instalador sin parámetros:

```
root@hpc-node1:/root/deployment# deploy_hpc-exec-node-pfc-full/bin/lgsetup.sh
```

Procesará todos los ficheros de secuencia existentes por orden alfabético y para cada fichero de secuencia utilizará el fichero de configuración con el mismo nombre para alimentar a los helpers que componen la secuencia. Podemos cambiar este comportamiento si especificamos los parámetros:

- `-r`: ejecutará el fichero de secuencia pasado por parámetro
- `-c`: ejecutará el fichero de secuencia pasado pero con el fichero de configuración indicado
- `-v`: mostrará los mensajes de debug
- `-n`: permitirá que se ejecute el mismo helper varias veces en la secuencia

D.3. Lgdeploy

D.3.1. Qué es lgdeploy

Es una herramienta que pretende facilitar la labor de automatización de tareas y procedimientos de administración de sistemas, ofreciendo un marco de trabajo común sobre el que desarrollar, ejecutar y controlar las más diversas tareas de administración.

D.3.2. Principios de diseño

- **Simple**: usar la menor complejidad para obtener la mayor funcionalidad posible. No rehacer servicios ni herramientas ya hechas
- **Modular**: las funcionalidades "no básicas" se incorporarán como módulos, incluyendo las menores dependencias posibles.
- **Escalable y flexible**: el diseño deberá permitir que los serivios estén distribuidos en distintos servidores.
- **Integrable**: se podrá integrar con las herramientas del sistema, cron, syslog, etc.
- **Ser plataforma**: cualquiera podrá escribir sus propios módulos o recetas para adaptarlo a sus necesidades.

D.3.3. Funcionalidades

- Arranques por red vía PXE
  - Arrancar con sistemas de rescate en modo mantenimiento.
  - Wakeonlan y arranques programados de diferentes sistemas operativos.
- Deployment de sistemas operativos
  - Usando imágenes de sistemas: desde un sistema de rescate con ayuda del Deployment Toolkit es posible tomar imágenes de sistemas operativos Linux y Windows y desplegarlas en múltiples máquinas al mismo tiempo (vía multicast).
  - Usando métodos de instalación desatendida implementados por el propio proveedor del sistema operativo: actualmente se encuentra implementado el soporte a preseed (Debian y Ubuntu) y kickstart (CentOS y VMWare).
- Ejecución centralizada de tareas en sistema de rescate
  - Backup y restauración de sistemas completos.
  - Chequeos y reparaciones de filesystems.
  - Reparaciones en sistemas de arranque.
  - Restauraciones de registro.
  - etc..
- Ejecución centralizada de tareas en sistema de producción
  - Modo pasivo: registro de agentes Cron y asignación de tareas.
  - Modo activo: envío de comandos vía ssh.
- Personalización y API para programación de tareas de alto nivel.

D.3.4. Tecnología

- **LAMP** GNU/Linux, Apache, MySQL, PHP 5.3, Zend Framework
- **Scripting** Bash, wsh
- **Booting** TFTP, FTS (FTSWebServer), NBD
- Files Repository Samba
- **Rescue System** SystemRescueCD
- **OS Deployment** Deployment ToolKit (librería bash propia basada en API de OpenGNSys), Preseed, Kickstart, Sysrep

D.3.5. Distribución de módulos

LGDeploy tiene un diseño modular. En la figura D.3 puede verse la distribución de los módulos.

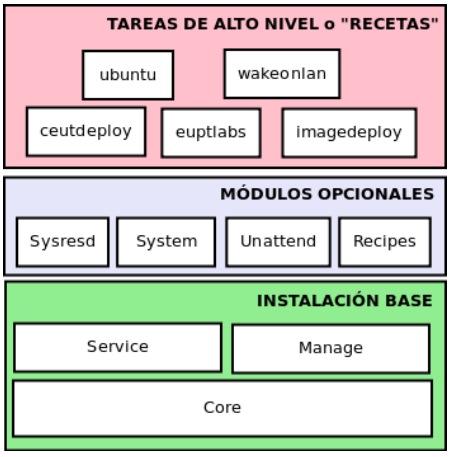


FIGURA D.3: Distribución en módulos de LGDeploy

D.3.6. Distribución de los componentes

LGDeploy es completamente distribuido. En la figura D.4 puede verse un escenario de ejemplo con la distribución de los componentes.

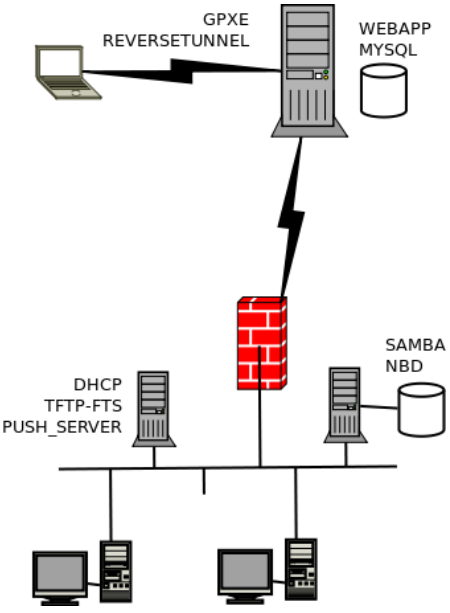


FIGURA D.4: Ejemplo de despliegue de componentes de LGDeploy

D.3.7. Estado actual

- **Rama 0.1** Versión funcional. Última versión 0.14. Es la versión presentada en este proyecto.
- **Rama 0.2** rediseño y refactorización en Ruby. Está en estado embrionario.
- **LGLabs** Versión funcional. Versión modificada por David Fuertes (Unizar) capaz de desplegar imágenes de sistemas Windows y Linux y realizar tareas de mantenimiento mediante sistema de rescate. Estuvo de 2011 a 2013 en aulas de informática de la EUPT.

D.3.7.1. Pantallazos LGLabs

Ya se vieron en la memoria algunos pantallazos de LGDeploy en acción en el ámbito del clúster. Sin embargo, como acabamos de ver, LGDeploy pretende ser una plataforma sobre la que desarrollar nuevas aplicaciones. Una de las aplicaciones que se desarrollaron sobre LGDeploy fue LGLabs. A continuación muestro algunos pantallazos de la misma.

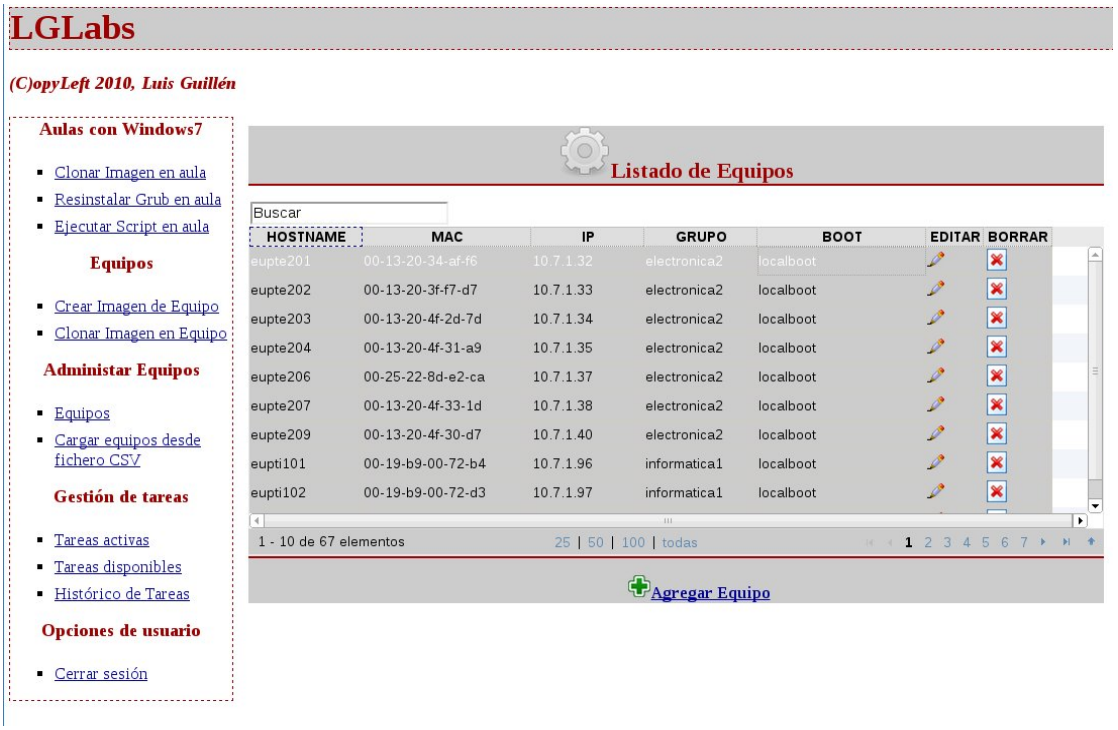


FIGURA D.5: Listado de equipos en LGLabs

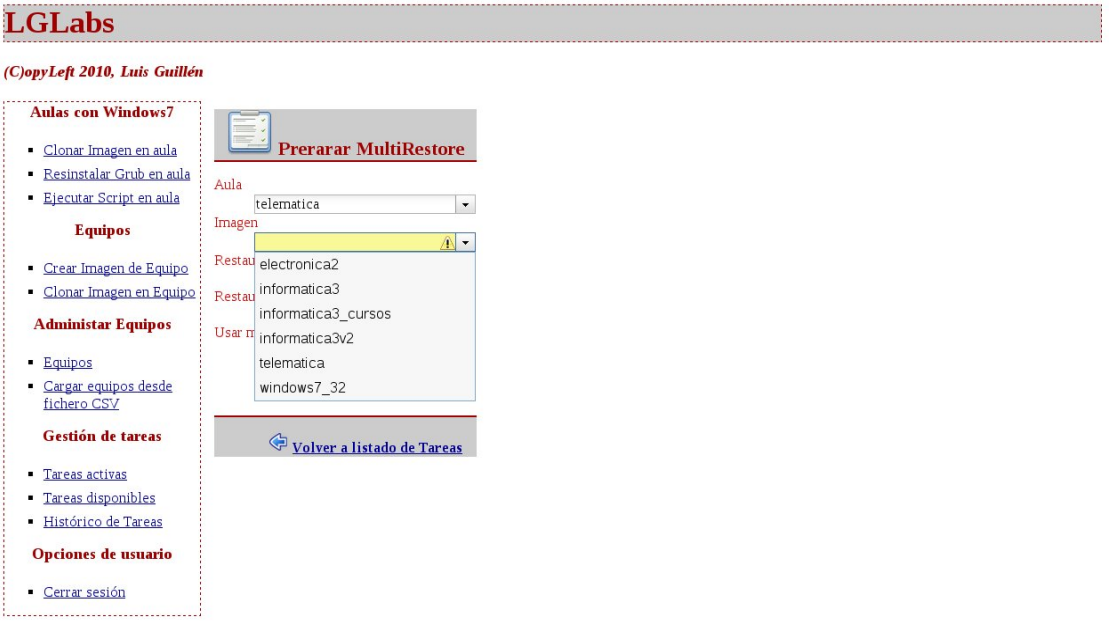


FIGURA D.6: Receta de clonación de un Aula de la EUPT con LGLabs

FIGURA D.7: LGLabs clonando un aula de informática en la EUPT

Anexo E

Escenario de pruebas desarrollado

E.1. Introducción

Como se comentó al inicio de la memoria, este proyecto ha sido desarrollado bajo un paradigma de *la infraestructura como código*. Esto quiere decir que todo el proceso de creación y puesta en marcha de la infraestructura necesaria se ha automatizado en software, permitiendo tener una infraestructura de pruebas prácticamente idéntica al escenario real.

La creación de máquinas virtuales de la infraestructura se ha realizado utilizando lgvbox y se adjunta el código fuente descriptivo que permite la generación de la misma. Por otro lado, el despliegue de la configuración se ha realizado utilizando lgsetup. Además se incluye un servidor virtual lgdeploy listo y adaptado a la infraestructura con el cual realizar las pruebas de automatización del despliegue de los sistemas operativos.

E.2. Escenario

Para el desarrollo del proyecto se ha implementado un escenario de redes virtuales similar al real. Para ello se ha definido una máquina (router) con varias interfaces de red, una de ellas conectada al host usando el modo NAT y las otras conectadas a las distintas redes internas. El despliegue de esa máquina hace un NAT en la salida al resto de redes y realiza un ruteo entre las redes internas. Además, para poder acceder de forma más cómoda a los recursos internos incluye un servidor vpn en modo bridge que da dirección en la red de usuarios.

El escenario puede verse en la figura Escenario pruebas.

E.2.1. Redes

Las redes existentes en el escenario son:

- **Red HOST - Virtual**: (NATed) es la red existente entre el router y el host. La gestiona directamente virtualbox y usa un direccionamiento interno.
- **Red de infraestructura**: (services_net, 192.168.84.0/24) es la red en la cual están los servidores de servicios.
- **Red de usuarios**: (machines_net, 192.168.81.0/24) es la red en la cual están las estaciones de trabajo de los usuarios. El router usa un segmento del rango para dar direcciones en una vpn bridged para poder acceder a los recursos internos de la infraestructura virtual. Se usa el software Openvpn.
- **Red general hpc**: (hpc_net, 192.168.85.0/24)
- **Red almacenamiento hpc**: (hpc_net_storage, 192.168.14.0/24)
- **Red cómputo hpc**: (hpc_net_mpi, 192.168.13.0/24)

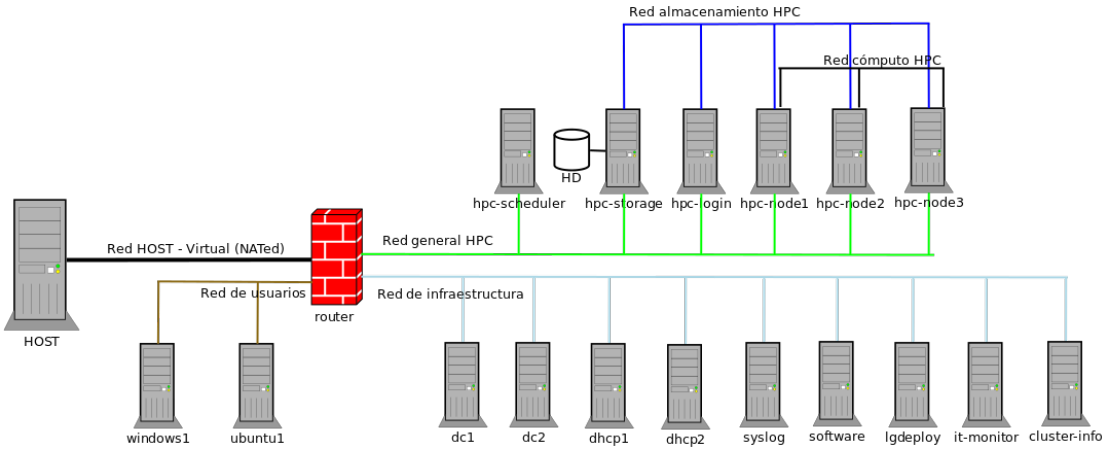


FIGURA E.1: Escenario de pruebas desarrollado para el PFC

E.2.2. Máquinas

Las máquinas existentes en el escenario son:

- **router**
  - **Rol**: Enrutador
  - **IP**: dinámica, 192.168.84.254, 192.168.81.254, 192.168.85.254

- **Descripción**:
  - Es la máquina que realiza el enrutamiento entre las redes virtuales y comunica con el mundo exterior realizando un NAT de salida en la interfaz también nateada por VirtualBox.
  - Lleva un servidor SSH que está mapeado al puerto 2222/TCP del host para poder hacer de máquina de salto.
  - Lleva un servidor OpenVPN en modo bridge (con vpn de nivel 2) mapeado al puerto del host 1194/UDP, que da dirección en la red de usuarios e inyecta las rutas a las redes virtuales.
  - Lleva un agente de relay dhcp para servir las ips en las redes general y usuarios desde los servidores dhcp.

- **dc1**
  - **Rol**: Controlador de dominio 1
  - **IP**: 192.168.84.1
  - **Descripción**:
    - Es un controlador de dominio de Active Directory implementado con Samba4. Lleva todos los roles Active Directory asignados (que se pueden migrar al otro controlador). El Active Directory se ha creado con un nivel funcional de 2008R2 y agregado esquema NIS. Replica la información de sysvol contra el segundo controlador usando rsync (es el maestro).
    - Además lleva servidor DNS con Bind9 y servidor NTP.

- **dc2**
  - **Rol**: Controlador de dominio 2
  - **IP**: 192.168.84.2
  - **Descripción**:
    - Es un controlador de dominio de Active Directory implementado con Samba4. El Active Directory se ha creado con un nivel funcional de 2008R2 y agregado esquema NIS. Replica la información de sysvol contra el segundo controlador usando rsync (es el esclavo).
    - Además lleva servidor DNS con Bind9 y servidor NTP.

- **dhcp1**
  - **Rol**: Servidor dhcp 1
  - **IP**: 192.168.84.3

- **Descripción**:
  - Es un servidor dhcp que comparte pools con el otro servidor para garantizar alta disponibilidad.
  - Replica las concesiones estáticas y parte de la configuración vía Unison.

- **dhcp2**
  - **Rol**: Servidor dhcp 2
  - **IP**: 192.168.84.4
  - **Descripción**:
    - Es un servidor dhcp que comparte pools con el otro servidor para garantizar alta disponibilidad.
    - Replica las concesiones estáticas y parte de la configuración vía Unison.

- **syslog**
  - **Rol**: Servidor syslog
  - **IP**: 192.168.84.5
  - **Descripción**:
    - Lleva un servidor syslog que escucha por tcp y vuelca el contenido sobre ficheros de log y base de datos MySQL.
    - Lleva instalado el software web Loganalyzer.

- **software**
  - **Rol**: Servidor distribución de software
  - **IP**: 192.168.84.6
  - **Descripción**:
    - Lleva un servidor apt-cacher-ng para el cacheo de paquetes.
    - Lleva instalado el software reprepro y con unos repositorios locales creados.

- **lgdeploy**
  - **Rol**: Servidor con lgdeploy
  - **IP**: 192.168.84.7
  - **Descripción**:
    - Lleva instalado el software lgdeploy con receta cefcahpc

- **it-monitor**

- **Rol**: Servidor de monitorización
- **IP**: 192.168.84.8
- **Descripción**:
  - Lleva instalado el software cacti y nagios

- **cluster-info**
  - **Rol**: Servidor de información hpc
  - **IP**: 192.168.84.9
  - **Descripción**:
    - Lleva instalado servidor web con mediawiki, autenticando contra el dominio
    - Lleva instalado los componentes gmetad y ganglia-webinterface

- **windows1**
  - **Rol**: Máquina windows 1
  - **IP**: 192.168.81.10
  - **Descripción**:
    - Máquina de pruebas windows
    - Lleva instaladas las RAST para gestionar el Active Directory

- **ubuntu1**
  - **Rol**: Máquina ubuntu 1
  - **IP**: 192.168.81.11
  - **Descripción**:
    - Máquina de pruebas ubuntu

- **hpc-storage**
  - **Rol**: Servidor con almacenamiento compartido HPC
  - **IP**: 192.168.85.1, 192.168.14.1
  - **Descripción**:
    - Lleva el servidor NFS y los recursos compartidos del HPC.

- **hpc-scheduler**
  - **Rol**: Servidor con componente Qmaster
  - **IP**: 192.168.85.2

- **Descripción**:
  - Lleva el servidor qmaster del gridscheduler.

- **hpc-login**
  - **Rol**: Servidor login hpc
  - **IP**: 192.168.85.3, 192.168.14.3
  - **Descripción**:
    - Lleva el servidor de login del hpc.

- **hpc-nodeX**
  - **Rol**: Servidores de cómputo hpc
  - **IP**: 192.168.85.10-20, 192.168.14.10-20, 192.168.13.10-20
  - **Descripción**:
    - Lleva los componentes de ejecución del gridscheduler.

E.3. Vídeo demostrativo

Realicé un vídeo que está congado en Youtube con el título *Demostración despliegue infraestructura virtual con LGVbox* en el cual hago una demostración de despliegue de la infraestructura. Puede verse en [1].

[1] http://www.youtube.com/watch?v=w32DHBDZVXA

Anexo F

Distribución UbuntuCefca14

F.1. Introducción

Con el motivo de homogeneizar las workstations Linux del personal científico, entre las distintas distribuciones disponibles se optó por la distribución Ubuntu por ser la más popular y la que mejor compatibilidad ofrecía con el hardware. Conforme se implantaba la homogeneización, se vio la necesidad de tener que instalar los mismos paquetes en cada equipo y agregar respositorios adicionales. Pero el mayor inconveniente era que la mayoría de los paquetes científicos que se usaban no existían en esta distribución. Lo que se hizo a continuación fue automatizar en forma de scripts bash todo el proceso de despliegue de la estación de trabajo de forma que se instalasen los paquetes de la distribución y se bajase el código fuente y compilase todo el software de manera automática en la workstation. Aquellos scripts cumplían su cometido para la instalación pero la actualización del software era un horror. Además, conforme los scripts evolucionaban tenías distintas versiones desplegadas por las máquinas, lo cual acababa por "deshomogenizarlas". Por ello se optó por crear una pequeña distribución que incluyese la mayoría del software científico que se usa en CEFCA y las configuraciones básicas (artwork, ligthdm, certificados, etc). Todo el software de la distribución se colgaría en un repositorio local en el que además se incluirían otros paquetes que no estuviesen en los repositorios oficiales.

F.2. Diseño

Aunque el objetivo principal eran las estaciones de trabajo y que por lo tanto era necesario entorno gráfico, se planteó la necesidad de que fuese también posible instalar la

distribución en servidores o nodos de computo de forma que se aprovechase el trabajo realizado. Por ello se realizó el diseño de metapaquetes que se muestra en la figura `diagrama ubuntucefca`.

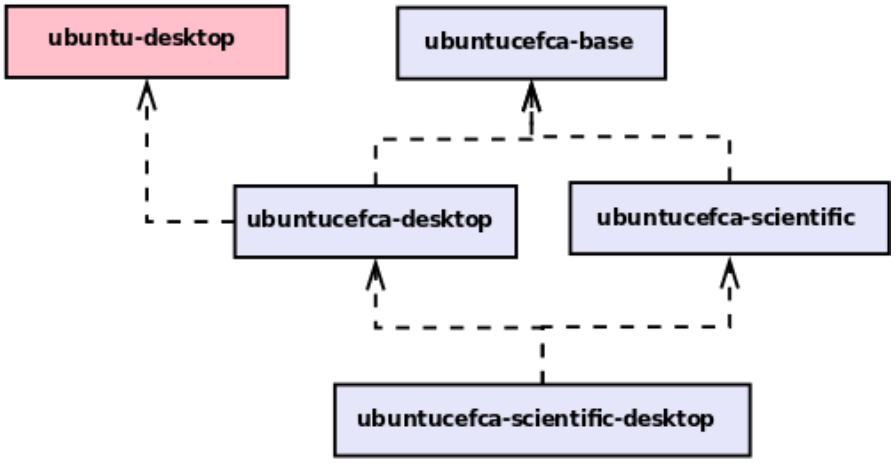


FIGURA F.1: Diseño de metapaquetes en UbuntuCefca14

Un metapaquete no es más que un paquete que tiene como dependencia a otros paquetes. Usaremos los metapaquetes para definir diferentes conjuntos o perfiles de software y nos aprovecharemos además del sistema de dependencias entre metapaquetes para nuevos perfiles.

Los metapaquetes que se definieron fueron:

- ubuntucefca-base: utilidades base sin interfaz gráfica
- ubuntucefca-desktop: utilidades base, artwork, settings y programas de escritorio de uso general
- ubuntucefca-scientific: utilidades de ciencia y desarrollo que no emplean interfaz gráfica
- ubuntucefca-scientific-desktop: además del escritorio y todas las utilidades que no emplean interfaz gráfica, añade aplicaciones de ciencia que emplean la interfaz gráfica

Puede comprobarse la distinción entre entorno gráfico y no entorno gráfico. Siguiendo este diseño, en una workstation instalaremos `ubuntu-scientific-desktop` y en un nodo de cómputo instalaremos `ubuntucefca-scientific`. Además el diseño permite en un futuro crear metapaquetes `ubuntu-engineering` y `ubuntu-engineering-desktop` que creen un entorno para el personal de ingeniería.

F.3. Implementación

Junto a los fuentes del proyecto pueden obtenerse todo el código fuente de los paquetes. Todos los paquetes se suben a un servidor con unos repositorios gestionados por el software *reprepro*. Los repositorios definidos son:

- ubuntucefca: repositorio denominado estable que se instala en todas las máquinas.
- ubuntucefca-testing: repositorio de pruebas y que es configurado únicamente en las máquinas de usuarios voluntarios beta-testers.

Cuando se hace algún cambio, el paquete se sube al repositorio de testing. Transcurrido algún tiempo y tras la opinión de los beta-testers se pasa al repositorio estable.

El listado de paquetes actualmente que incluye paquetes de terceros y desarrollados por mi es el siguiente:

- cefca-artwork_0.1-1_all.deb
- cefca-ca-certificates_1_all.deb
- eye_1.4.1-2_amd64.deb
- gridengine-common_6.2u5-7.3cefca1_all.deb
- gridengine-master_6.2u5-7.3cefca1_amd64.deb
- gridengine-qmon_6.2u5-7.3cefca1_amd64.deb
- hp-ams_2.0.0-1345.12_amd64.deb
- hpsa-dkms_3.4.8_all.deb
- iraf_2.16_amd64.deb
- missfits_2.8.0-2_amd64.deb
- oracle-java7-installer_7u60-0 webupd8 0_all.deb
- oracle-java7-set-default_7u60-0 webupd8 0_all.deb
- oracle-jdk7-installer_7u60-0 webupd8 0_all.deb
- plymouth-theme-cefca-logo_0.1-1_all.deb
- psfex_3.17.1-2_amd64.deb

- pyraf_2.1.6-1_amd64.deb
- python-ephem_3.7.5.3-1_amd64.deb
- python-stsci.tools_3.2.2 dev-1_all.deb
- scamp_2.0.4-2_amd64.deb
- sextractor_2.19.5-2_amd64.deb
- skymaker_3.10.5-2_amd64.deb
- stiff_2.4.0-2_amd64.deb
- stsci-iraf_3.16_amd64.deb
- stuff_1.26.0-2_amd64.deb
- swarp_2.38.0-2_amd64.deb
- ubuntucefca-base_0.1_amd64.deb
- ubuntucefca-defaults-settings_0.1_all.deb
- ubuntucefca-desktop_0.1_amd64.deb
- ubuntucefca-scientific_0.1_amd64.deb
- ubuntucefca-scientific-desktop_0.1_amd64.deb
- virtualbox-4.3_4.3.18-96516 Ubuntu raring_amd64.deb
- weightwatcher_1.12-2_amd64.deb

Anexo G

Despliegue de servicios de infraestructura

G.1. Active directory con Samba4

G.1.1. Despliegue del primer controlador de dominio

G.1.1.1. Configuración de red del sistema

Tras realizar una instalación de Ubuntu Server 14.04LTS básica realizaremos una configuración manual de la interfaz de red:

```
root@test-vir-001:~# cat /etc/network/interfaces

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.84.1
        netmask 255.255.255.0
        gateway 192.168.84.254
        dns-nameservers 8.8.8.8
        dns-search test-scenario.lan
```

NOTA: En ubuntu server, se hace uso de la utilidad resolvconf para gestionar la configuración del cliente dns. Esto hace que la configuración del cliente dns se realice vía /etc/network/interfaces.

Hay que hacer que el hostname esté correctamente configurado, de modo que la resolución apunte a la dirección ip de servicio y no a una loopback, para ello modificaremos los ficheros:

```
root@test-vir-001:~# cat /etc/hostname
test-vir-001
```

y

```
root@test-vir-001:~# cat /etc/hosts
127.0.0.1 localhost
192.168.84.1    test-vir-001.test-scenario.lan  test-vir-001


# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Realizaremos un par de pruebas para comprobar que está todo correcto:

```
root@test-vir-001:~# hostname --fqdn
test-vir-001.test-scenario.lan

root@test-vir-001:~# ping -c 1 test-vir-001.test-scenario.lan
PING test-vir-001.test-scenario.lan (192.168.84.1) 56(84) bytes of data.
64 bytes from test-vir-001.test-scenario.lan (192.168.84.1): icmp_seq=1 ttl=64
    time=0.025 ms

--- test-vir-001.test-scenario.lan ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.025/0.025/0.025/0.000 ms
```

G.1.1.2. Configuración inicial de servidor dns

A continuación pasaremos a instalar el servidor dns, para ello:

```
\#  apt-get install bind9 dnsutils
```

Procederemos a configurar el servidor para que sirva de servidor caché para nuestras redes y empleando servidores reenviadores.

```
root@test-vir-001:/etc/bind# cat /etc/bind/named.conf.options
acl "trusted" {
        127.0.0.1;
        192.168.0.0/16;
};

options {
        directory "/var/cache/bind";
```

```
        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        forward first;
        forwarders {
                8.8.8.8;
                8.8.4.4;
        };

        // valores de las directivas incluidas en la documentación de samba
        auth-nxdomain yes;
        notify no;
        empty-zones-enable no;
        allow-transfer { none; };
        tkey-gssapi-keytab "/var/lib/samba/private/dns.keytab";
        // fin de directivas especificadas en samba

        listen-on { any; };
        listen-on-v6 { none; };

        allow-recursion { trusted; };
        allow-query { trusted; };
};
```

NOTA: En la acl "trusted" incluiremos las subredes a las que permitiremos consultar a nuestro servidor dns. Nótese que dejamos ya configuradas algunas directivas necesarias para la posterior integración con samba4 como es la tkey-gssapi-keytab.

Una vez configurado, reiniciaremos el servidor de nombres:

```
root@test-vir-001:~\#  service bind9 restart
```

Y modificaremos nuestra configuración para que el servidor use una resolución local:

root@test-vir-001: # cat /etc/network/interfaces

```
# Fichero generado mediante config_net_ifaces.sh

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.84.1
        netmask 255.255.255.0
        gateway 192.168.84.254
        dns-nameservers 127.0.0.1
```

```
        dns-search test-scenario.lan
```

Reiniciaremos:

```
root@test-vir-001:~\#  ifdown eth0 ; ifup eth0
```

Y comprobaremos que resolvconf ha modificado bien la configuración del cliente dns:

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
search test-scenario.lan
```

Comprobaremos entonces el correcto funcionamiento:

```
root@test-vir-001:~# host www.google.com
www.google.com has address 173.194.67.106
www.google.com has address 173.194.67.147
www.google.com has address 173.194.67.105
www.google.com has address 173.194.67.103
www.google.com has address 173.194.67.99
www.google.com has address 173.194.67.104
www.google.com has IPv6 address 2a00:1450:4003:805::2004
```

G.1.1.3. Configuración del servidor ntp

Instalaremos el servidor ntp:

```
root@test-vir-001:~\#  apt-get install ntp
```

Configuraremos el servidor ntp:

```
root@test-vir-001:~# cat /etc/ntp.conf
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help

driftfile /var/lib/ntp/ntp.drift

# Enable this if you want statistics to be logged.
statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# You do need to talk to an NTP server or two (or three).
server hora.rediris.es
server hora.roa.es
server 127.127.1.0
fudge 127.127.1.0 stratum 13
```

```
# By default, exchange time with everybody, but don't allow configuration.
#restrict -4 default kod notrap nomodify nopeer noquery
#restrict -6 default kod notrap nomodify nopeer noquery
restrict -4 default kod notrap nomodify nopeer
restrict -6 default kod notrap nomodify nopeer

# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1
```

NOTA: Aquí configuraremos los servidores ntp de los que obtendremos el origen de tiempo y permitiremos por defecto la consulta.

Reiniciaremos el servicio:

```
root@test-vir-001:~\#  service ntp restart
```

Y comprobaremos tras esperar un tiempo la correcta sincronización:

```
root@test-vir-001:~# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*hora.rediris.es .GPS.            1 u  406 1024  377   13.965   -0.196   1.090
+hora.roa.es     .GPS.            1 u  714 1024  377   31.113   -0.573   2.778
 LOCAL(0)        .LOCL.          13 l   38   64  377    0.000    0.000   0.001
```

G.1.1.4. Instalación del primer DC

En primer lugar satisfaremos uno de los requisitos de samba4 y es disponer de soporte de acl y atributos extendidos en nuestro sistema de archivos (el que vaya a contener los datos de samba).

```
root@test-vir-001:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=237d0764-b677-4d40-bd60-81681479b7d4 /               ext4    errors=remount-
    ro,user_xattr,acl,barrier=1 0       1
# swap was on /dev/sda5 during installation
UUID=d7bde0ca-1be0-47c2-862c-9a1a1a224b3b none            swap    sw
     0       0
```

Como vemos en nuestro caso es el directorio raíz. Modificaremos las opciones y realizaremos el remontaje del sistema de archivos y comprobaremos.

```
root@test-vir-001:~# mount -o remount /
root@test-vir-001:~# mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro,user_xattr,acl,barrier=1)
   ....
```

A continuación instalaremos samba junto a algunas utilidades aceptando los valores predeterminados que nos da debconf:

apt-get install samba samba-client samba-common-bin winbind samba-testsuite cifs-utils acl dnsutils krb5-user ldb-tools ldap-utils

Detendremos todos los servicios asociados a samba que estén activos:

```
root@test-vir-001:~#  for service in smbd nmbd winbind samba-ad-dc; do service $service stop; done
```

Y eliminaremos el fichero de configuración:

```
root@test-vir-001:~\#  rm -f /etc/samba/smb.conf
```

Y Realizaremos la provisión del nuevo dominio:

```
root@test-vir-001:~# samba-tool domain provision --realm="TEST-SCENARIO.LAN" --
    domain="TEST-SCENARIO" --dns-backend="BIND9_DLZ" --server-role=dc --function-
    level=2008_R2 --use-xattr=yes --use-rfc2307
Administrator password will be set randomly!
Looking up IPv4 addresses
    ....
Server Role:           active directory domain controller
Hostname:              test-vir-001
NetBIOS Domain:        TEST-SCENARIO
DNS Domain:            test-scenario.lan
DOMAIN SID:            S-1-5-21-3078859445-3913205560-4029987288
```

Con esto se ha realizado la provisión del nuevo dominio y se ha asignado la password aleatoria impresa en la salida. Como es un dominio de pruebas, reduciremos la complejidad de las contraseñas en nuestro dominio:

```
root@test-vir-001:~# samba-tool domain passwordsettings set --complexity=off
Password complexity deactivated!
All changes applied successfully!
root@test-vir-001:~# samba-tool domain passwordsettings set --min-pwd-length=1
Minimum password length changed!
All changes applied successfully!
```

Y fijaremos la pasword para el administrador del dominio y haremos que no caduque:

```
root@test-vir-001:~# samba-tool user setpassword Administrator
New Password:
Changed password OK
root@test-vir-001:~# samba-tool user setexpiry Administrator --noexpiry
Expiry for user 'Administrator' disabled.
```

Integración con Bind9 A continuación configuraremos la integración de samba4 con bind9.

Para ello:

root@test-vir-001: # cat /etc/bind/named.conf

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
include "/var/lib/samba/private/named.conf";
```

Adecuaremos permisos:

```
root@test-vir-001:~# chown root:bind /var/lib/samba/private/named.conf
root@test-vir-001:~# chown root:bind /var/lib/samba/private/dns.keytab
root@test-vir-001:~# chmod 640 /var/lib/samba/private/dns.keytab
```

Comprobaremos la versión de named:

```
root@test-vir-001:~# named -v
BIND 9.9.5-3ubuntu0.1-Ubuntu (Extended Support Version)
```

Y modificaremos a tal efecto:

```
root@test-vir-001:~# cat /var/lib/samba/private/named.conf
# This DNS configuration is for BIND 9.8.0 or later with dlz_dlopen support.
#
# This file should be included in your main BIND configuration file
#
# For example with
# include "/var/lib/samba/private/named.conf";

#
# This configures dynamically loadable zones (DLZ) from AD schema
# Uncomment only single database line, depending on your BIND version
#
dlz "AD DNS Zone" {
        # For BIND 9.8.0
        # database "dlopen /usr/lib/x86_64-linux-gnu/samba/bind9/dlz_bind9.so";

        # For BIND 9.9.0
        database "dlopen /usr/lib/x86_64-linux-gnu/samba/bind9/dlz_bind9_9.so";
};
```

Haremos unos ajustes en apparmor:

```
root@test-vir-001:~# cat /etc/apparmor.d/local/usr.sbin.named
# Site-specific additions and overrides for usr.sbin.named.
# For more details, please see /etc/apparmor.d/local/README.
/var/lib/samba/** rm,
/var/lib/samba/private/dns.keytab rk,
/var/lib/samba/private/named.conf r,
/var/lib/samba/private/dns/** rwk,
/usr/lib/x86_64-linux-gnu/samba/** rm,
/usr/lib/x86_64-linux-gnu/ldb/** rm,
/var/tmp/** rwk,
```

Y reiniciaremos:

```
root@test-vir-001:~# service apparmor teardown
root@test-vir-001:~# service apparmor start
```

Para finalizar, reiniciaremos el servicio bind9:

```
root@test-vir-001:~\#  service bind9 restart
```

Configuración cliente kerberos Configuraremos el cliente local kerberos, para ello:

```
root@test-vir-001:~# cat /etc/krb5.conf
[libdefaults]
 dns_lookup_realm = true
 dns_lookup_kdc = true
 default_realm = TEST-SCENARIO.LAN
```

Consolidamos la configuración Por fin reiniciaremos los servicios de samba:

```
root@test-vir-001:~# for service in nmbd smbd samba-ad-dc; do service $service
    start; done
nmbd start/running
smbd start/running, process 13089
samba-ad-dc start/running, process 13109
```

Comprobaciones de configuración Realizaremos algunas comprobaciones:

- Testearemos la correcta resolución local del servidor dns:

```
root@test-vir-001:~# host test-vir-001.test-scenario.lan
test-vir-001.test-scenario.lan has address 192.168.84.1
```

- Comprobaremos algunos registros:

```
root@test-vir-001:~# host -t SRV _ldap._tcp.test-scenario.lan
_ldap._tcp.test-scenario.lan has SRV record 0 100 389 test-vir-001.test-scenario.
    lan.
root@test-vir-001:~# host -t SRV _kerberos._udp.test-scenario.lan
_kerberos._udp.test-scenario.lan has SRV record 0 100 88 test-vir-001.test-
    scenario.lan.
```

- Comprobaremos que las actualizaciones dinámicas funcionan:

```
root@test-vir-001:~# samba_dnsupdate --verbose
IPs: ['192.168.84.1']
Looking for DNS entry A test-scenario.lan 192.168.84.1 as test-scenario.lan.
    ....
Checking 0 100 3268 test-vir-001.test-scenario.lan. against SRV _gc._tcp.default-
    first-site-name._sites.test-scenario.lan test-vir-001.test-scenario.lan 3268
No DNS updates needed
```

- Testearemos el cliente kerberos:

```
root@test-vir-001:~# kinit administrator
Password for administrator@TEST-SCENARIO.LAN:
root@test-vir-001:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: administrator@TEST-SCENARIO.LAN

Valid starting     Expires            Service principal
12/01/15 13:26:22  12/01/15 23:26:22  krbtgt/TEST-SCENARIO.LAN@TEST-SCENARIO.LAN
        renew until 13/01/15 13:26:19
```

- Comprobaremos los servicios smb:

```
root@test-vir-001:~# smbclient -L test-vir-001 -U administrator -k
Domain=[TEST-SCENARIO] OS=[Unix] Server=[Samba 4.1.6-Ubuntu]

        Sharename       Type      Comment
        ---------       ----      -------
        netlogon        Disk
        sysvol          Disk
        IPC$            IPC       IPC Service (Samba 4.1.6-Ubuntu)
Domain=[TEST-SCENARIO] OS=[Unix] Server=[Samba 4.1.6-Ubuntu]

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            TEST-VIR-001
```

Configuraremos certificados ssl/tls del servicio ldap Para ello detendremos los servicios de samba y copiaremos los ficheros:

```
root@test-vir-001:~# for service in smbd nmbd samba-ad-dc; do service $service
    stop; done
root@test-vir-001:~# rm -f /var/lib/samba/private/tls/cert.pem
root@test-vir-001:~# rm -f /var/lib/samba/private/tls/key.pem
root@test-vir-001:~# rm -f /var/lib/samba/private/tls/ca.pem
root@test-vir-001:~# cp service_ldap.crt /var/lib/samba/private/tls/cert.pem
root@test-vir-001:~# cp service_ldap.key /var/lib/samba/private/tls/key.pem
root@test-vir-001:~# cp TEST_CA_Root.crt /var/lib/samba/private/tls/ca.pem
root@test-vir-001:~# chown root:root /var/lib/samba/private/tls/*
root@test-vir-001:~# chmod 600 /var/lib/samba/private/tls/key.pem
root@test-vir-001:~# for service in smbd nmbd samba-ad-dc; do service $service
    start; done
```

Configuraremos servidor DNS Agregaremos zonas dns inversas (usamos kerberos, es necesario que dispongamos del ticket con kinit administrator):

```
root@test-vir-001:~# samba-tool dns zonecreate test-vir-001 84.168.192.in-addr.
    arpa -k yes
Zone 84.168.192.in-addr.arpa created successfully
root@test-vir-001:~# samba-tool dns zonecreate test-vir-001 85.168.192.in-addr.
    arpa -k yes
Zone 85.168.192.in-addr.arpa created successfully
root@test-vir-001:~# samba-tool dns zonecreate test-vir-001 81.168.192.in-addr.
    arpa -k yes
Zone 81.168.192.in-addr.arpa created successfully
```

Agregaremos el registro ptr del servidor:

```
root@test-vir-001:~# samba-tool dns add test-vir-001 84.168.192.in-addr.arpa \
>                 1 PTR test-vir-001.test-scenario.lan -k yes
Record added successfully
```

Agregaremos diversos cnames:

```
root@test-vir-001:~# samba-tool dns add test-vir-001 test-scenario.lan \
>                 dc1 CNAME test-vir-001.test-scenario.lan -k yes
Record added successfully
root@test-vir-001:~#  samba-tool dns add test-vir-001 test-scenario.lan \
> ldap1 CNAME test-vir-001.test-scenario.lan -k yes
Record added successfully
root@test-vir-001:~#  samba-tool dns add test-vir-001 test-scenario.lan \
> kdc1 CNAME test-vir-001.test-scenario.lan -k yes
Record added successfully
root@test-vir-001:~#  samba-tool dns add test-vir-001 test-scenario.lan \
> ns1 CNAME test-vir-001.test-scenario.lan -k yes
Record added successfully
```

Si finalmente deseamos agregar un segundo controlador de dominio, daremos de alta el host en el dns:

```
root@test-vir-001:~# samba-tool dns add test-vir-001 test-scenario.lan \
>                 test-vir-002 A 192.168.84.2
Record added successfully
root@test-vir-001:~# samba-tool dns add test-vir-001 84.168.192.in-addr.arpa \
>         2 PTR test-vir-002.test-scenario.lan -k yes
Record added successfully
root@test-vir-001:~# samba-tool dns add test-vir-001 test-scenario.lan \
> .ldap2 CNAME test-vir-002.test-scenario.lan -k yes
Record added successfully
root@test-vir-001:~# samba-tool dns add test-vir-001 test-scenario.lan \
> .kdc2 CNAME test-vir-002.test-scenario.lan -k yes
Record added successfully
root@test-vir-001:~# samba-tool dns add test-vir-001 test-scenario.lan \
> .ns2 CNAME test-vir-002.test-scenario.lan -k yes
Record added successfully
```

G.1.2. Configuración de un segundo controlador de dominio

Realizaremos la instalación básica y procederemos a configurar la red, pero usando en esta ocasión como servidor DNS al controlador de dominio que hemos instalado:

```
root@test-vir-002:~# cat /etc/network/interfaces
# Fichero generado mediante config_net_ifaces.sh

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.84.2
        netmask 255.255.255.0
        gateway 192.168.84.254
        dns-nameservers 192.168.84.1
        dns-search test-scenario.lan

root@test-vir-002:~# cat /etc/hostname
test-vir-002
root@test-vir-002:~# cat /etc/hosts
127.0.0.1 localhost
192.168.84.2    test-vir-002.test-scenario.lan  test-vir-002
    ....
```

Probaremos la correcta resolución usando el registro inverso creado en el otro servidor dns:

```
root@test-vir-002:~#  host -r 192.168.84.2

2.84.168.192.in-addr.arpa domain name pointer test-vir-002.test-scenario.lan.
```

G.1.2.1. Configuración de DNS y NTP

Realizaremos los mismos pasos realizados para el primer controlador de dominio.

Configurando cliente kerberos Configuramos el cliente kerberos:

```
root@test-vir-002:~# cat /etc/krb5.conf
[libdefaults]
 dns_lookup_realm = true
 dns_lookup_kdc = true
 default_realm = TEST-SCENARIO.LAN
```

y obtenemos un ticket:

```
root@test-vir-002:~# kinit administrator
Password for administrator@TEST-SCENARIO.LAN:
root@test-vir-002:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: administrator@TEST-SCENARIO.LAN

Valid starting     Expires            Service principal
12/01/15 18:10:41  13/01/15 04:10:41  krbtgt/TEST-SCENARIO.LAN@TEST-SCENARIO.LAN
        renew until 13/01/15 18:10:38
```

G.1.2.2. Agregar el nuevo controlador al dominio

Antes de poder agregar, deberemos de configurar el resolvedor dns de modo que use el servidor DNS del otro controlador de dominio (ya que el servidor dns que hemos montando todavía no sabe de la existencia del dominio).

A continuación configuraremos los prerequisitos del almacenamiento vistos e instalaremos el mismo software que instalamos en el otro controlador de dominio.

Agregaremos la máquina al dominio:

```
root@test-vir-002:~# samba-tool domain join test-scenario.lan DC -Uadministrator
    --realm=TEST-SCENARIO.LAN --dns-backend=BIND9_DLZ
Finding a writeable DC for domain 'test-scenario.lan'
Found DC test-vir-001.test-scenario.lan
Password for [WORKGROUP\administrator]:
workgroup is TEST-SCENARIO
realm is test-scenario.lan
checking sAMAccountName
    ....
Setting up secrets database
Joined domain TEST-SCENARIO (SID S-1-5-21-3078859445-3913205560-4029987288) as a
    DC
```

Tras la provisión editaremos modificaremos el fichero smb.conf para agregar las opciones que no tiene el join y que son necesarias para nuestro despliegue (únicamente use rfc2307):

```
# Global parameters
[global]
        workgroup = TEST-SCENARIO
        realm = TEST-SCENARIO.LAN
        netbios name = TEST-VIR-002
        server role = active directory domain controller
        server services = s3fs, rpc, nbt, wrepl, ldap, cldap, kdc, drepl, winbind
    , ntp_signd, kcc, dnsupdate
        idmap_ldb:use rfc2307 = yes

[netlogon]
        path = /var/lib/samba/sysvol/test-scenario.lan/scripts
        read only = No

[sysvol]
        path = /var/lib/samba/sysvol
        read only = No
```

Agregando registros DNS necesarios Debido a este bug [1] será necesario crear una serie de registros necesarios. En primer lugar obtenemos el objectGUID del servidor:

```
root@test-vir-002:~#  ldbsearch -H /var/lib/samba/private/sam.ldb '(invocationId
    =*)' --cross-ncs objectguid
# record 1
dn: CN=NTDS Settings,CN=TEST-VIR-002,CN=Servers,CN=Default-First-Site-Name,CN=
    Sites,CN=Configuration,DC=test-scenario,DC=lan
objectGUID: d1073ae1-e25b-42ec-a60a-201dcebcc63e

# record 2
dn: CN=NTDS Settings,CN=TEST-VIR-001,CN=Servers,CN=Default-First-Site-Name,CN=
    Sites,CN=Configuration,DC=test-scenario,DC=lan
objectGUID: 91d99709-e232-416f-a91c-0996d323fd45

# returned 2 records
# 2 entries
# 0 referrals
```

Y agregaremos el cname necesario (el registro A de test-vir-002 ya lo agregamos anteriormente):

```
root@test-vir-002:~# samba-tool dns add test-vir-001 _msdcs.test-scenario.lan \
>  d1073ae1-e25b-42ec-a60a-201dcebcc63e CNAME test-vir-002.test-scenario.lan
Record added successfully
```

Y comprobamos que se encuentra el objeto en el DNS:

[1] https://bugzilla.samba.org/show_bug.cgi?id=10928

```
root@test-vir-002:~# host -t CNAME d1073ae1-e25b-42ec-a60a-201dcebcc63e._msdcs.
    test-scenario.lan
d1073ae1-e25b-42ec-a60a-201dcebcc63e._msdcs.test-scenario.lan is an alias for
    test-vir-002.test-scenario.lan.
```

Integración con Bind9 Realizaremos exactamente los mismos pasos vistos en el primer controlador de dominio.

Tras la configuración configuraremos el cliente DNS para que apunte a 127.0.0.1 como hicimos anteriormente.

Consolidamos la configuración Por fin reiniciaremos los servicios de samba:

```
root@test-vir-001:~# for service in nmbd smbd samba-ad-dc; do service $service
    start; done
nmbd start/running
smbd start/running, process 13089
samba-ad-dc start/running, process 13109
```

G.1.3. Configurando replicación de sysvol

Para el correcto funcionamiento de las directivas de grupo, es necesario tener replicado el volumen sysvol.

Para ello deberemos recurrir a una utilidad externa de réplica. Existen multitud de utilidades, pero vamos a emplear la utilidad rsync. Las modificaciones sobre el sysvol se realizan sobre el servidor que tiene el rol de emulador de PDC.

Sobre el controlador de dominio 1

Instalaremos rsync:

```
root@test-vir-001:~\#  apt-get install rsync
```

Configuraremos rsyncd:

El lanzamiento del demonio:

```
root@test-vir-001:~# cat /etc/default/rsync
RSYNC_ENABLE=true
RSYNC_CONFIG_FILE=/etc/rsyncd.conf
RSYNC_OPTS=''
RSYNC_NICE=''
```

La configuración:

```
root@test-vir-001:~# cat /etc/rsyncd.conf
uid = root
gid = root
use chroot = yes
max connections = 20
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid

[SysVol]
path = /var/lib/samba/sysvol
max connections = 20
read only = true
list = false
uid = root
gid = root
auth users = sysvol-replication
secrets file = /etc/samba/sysvol-replication.secrets
hosts allow = 192.168.84.2
```

En hosts allow definiremos la dirección ip de los controladores de dominio adicionales.

Crearemos el fichero con la password:

```
root@test-vir-001:~# echo -e "sysvol-replication:password_replicacion" > /etc/
    samba/sysvol-replication.secrets
root@test-vir-001:~# chown root:root /etc/samba/sysvol-replication.secrets
root@test-vir-001:~# chmod 400 /etc/samba/sysvol-replication.secrets
```

Reiniciaremos el servicio:

```
root@test-vir-001:~#  service rsync restart
```

Sobre el controlador de dominio 2:

Instalamos rsync

root@test-vir-002: # apt-get install rsync

Crearemos un fichero con las credenciales:

```
root@test-vir-002:~# echo -e "sysvol-replication:password_replicacion" > /etc/
    samba/sysvol-replication.secrets
root@test-vir-002:~# chown root:root /etc/samba/sysvol-replication.secrets
root@test-vir-002:~# chmod 400 /etc/samba/sysvol-replication.secrets
```

Crearemos un script cron que sincronize cada hora:

```
#!/bin/bash

rsync -XAavz --delete-after --password-file=/etc/samba/sysvol-replication.secrets
          rsync://sysvol-replication@192.168.84.1/SysVol/ /var/lib/samba/sysvol
```

G.1.4. Instalando un cliente windows 7 con RAST

Hay que descargar el pack de "Herramientas de administración remota del servidor para Windows 7 con Service Pack 1" para poder administrar el Active Directory [2].

Después de instalar nos iremos al Panel de Control, Programas, y "Activar o desactivar características de Windows" y bajo "Herramientas de administración remota del servidor", seleccionaremos las opciones señaladas en la captura de pantalla de la figura G.1 y aceptaremos.

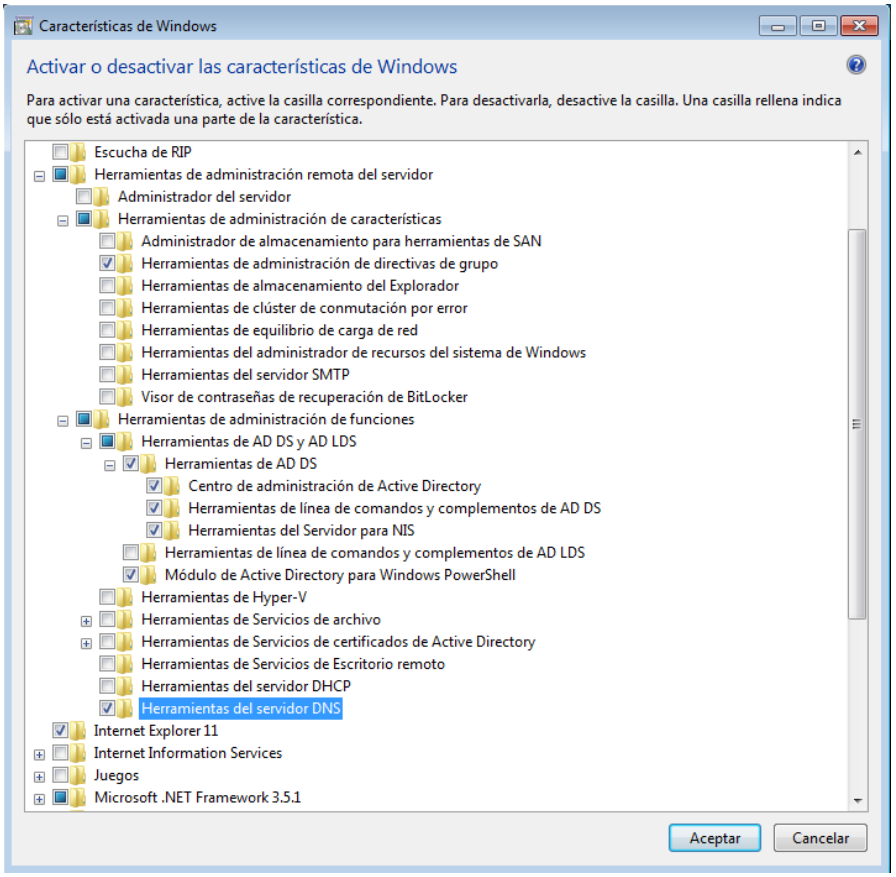


FIGURA G.1: Herramientas de administración remota para administrar Samba4

Agregaremos la máquina al dominio.

Como todavía no tenemos desplegados los DHCP, realizaremos una configuración estática. Es muy importante que fijemos las direcciones de los servidores DNS que acabamos de instalar.

Después en configuración avanzada, fijaremos el sufijo dns.

Aceptaremos y seleccionaremos que es una Red de trabajo.

[2] http://www.microsoft.com/es-es/download/details.aspx?id=7887

Después en Equipo -> Propiedades -> Cambiar configuración -> Cambiar

Agregaremos la máquina al dominio. Para ello seleccionaremos dominio y test-scenario.lan.

Proporcionaremos como usuario "administrator" y la contraseña proporcionada en el despliegue del controlador del dominio.

Reiniciaremos el equipo.

Iniciaremos sesión en el dominio, pulsando en "Cambiar de usuario", otro usuario e indicaremos el usuario administrator.

Sobre el menú de inicio introduciremos:

C:\ProgramData\Microsoft\Start Menu\Administrative Tools

Aquí obtendremos todas las herramientas administrativas. Ejecutaremos "Usuarios y equipos de Active Directory"para ver que está todo correctamente como en la figura G.2.

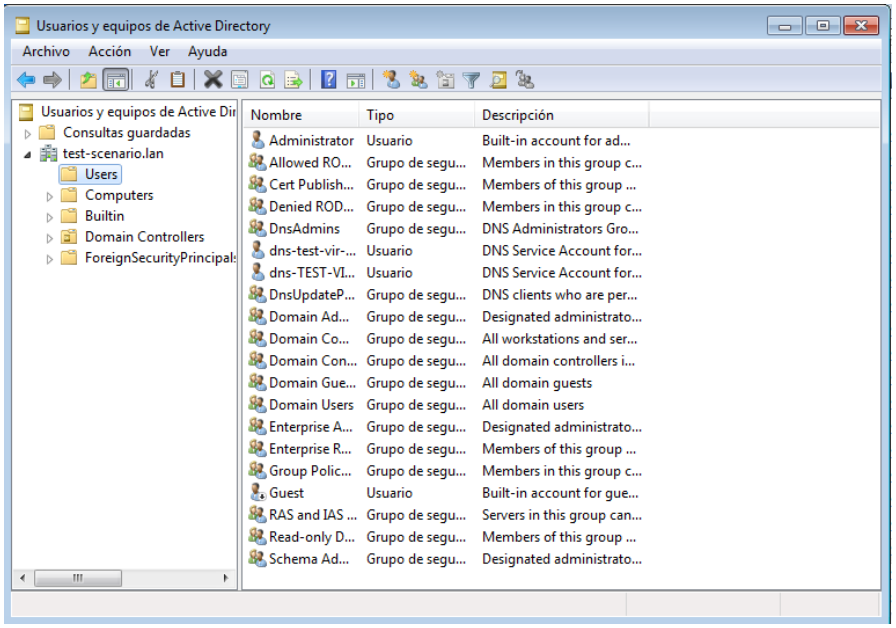


FIGURA G.2: Usuarios y equipos de active directory de nuestro dominio

También ejecutaremos "DNS" para ver que podemos administrar los servidores dns. Conectaremos con servidor dns test-vir-001. Observaremos en la figura G.3 la correcta configuración de la zona dns y las zonas de búsqueda inversa que hemos creado en el proceso de instalación.

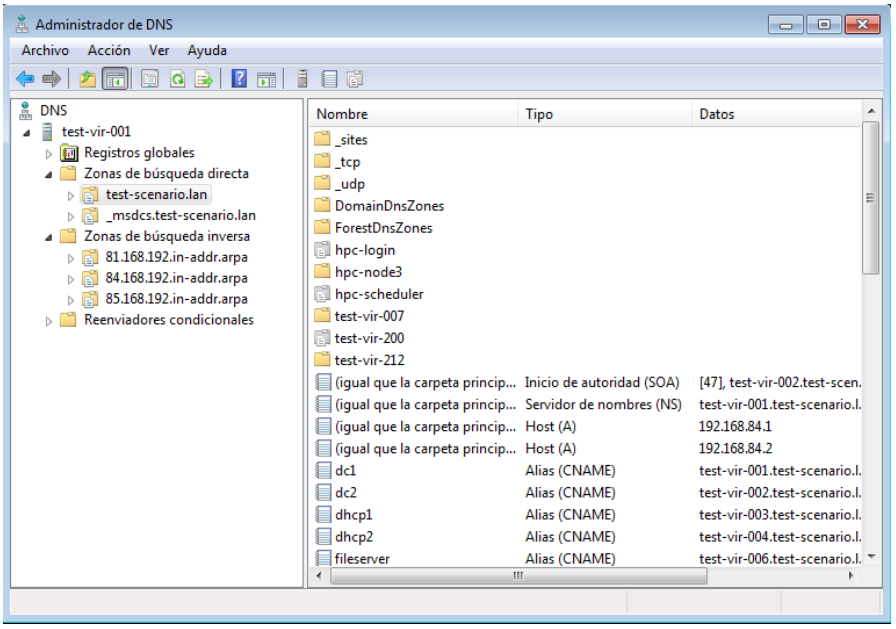


FIGURA G.3: Administración zonas DNS de nuestro dominio

G.2. Servidores dhcp

G.2.1. Configuración de red del sistema

Tras realizar una instalación de Ubuntu Server 14.04LTS básica realizaremos una configuración manual de la interfaz de red:

```
root@test-vir-003:~# cat /etc/network/interfaces
# Fichero generado mediante config_net_ifaces.sh

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.84.3
        netmask 255.255.255.0
        gateway 192.168.84.254
        dns-nameservers 192.168.84.1 192.168.84.2
        dns-search test-scenario.lan
```

Y

```
root@test-vir-004:~# cat /etc/network/interfaces
# Fichero generado mediante config_net_ifaces.sh

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
```

```
        address 192.168.84.4
        netmask 255.255.255.0
        gateway 192.168.84.254
        dns-nameservers 192.168.84.1 192.168.84.2
        dns-search test-scenario.lan
```

El resto de los pasos procederemos de igual modo al visto anteriormente.

G.2.2. Instalación y configuración de software

Instalaremos el software

```
root@test-vir-003:~\#  apt-get install isc-dhcp-server bind9utils
```

Crearemos una estructura de directorios para la configuración del servidor:

```
root@test-vir-003:~# mkdir -p /etc/dhcp/dhcpd/leases
root@test-vir-003:~# mkdir -p /etc/dhcp/dhcpd/omapi
root@test-vir-003:~# mkdir -p /etc/dhcp/dhcpd/peers
```

Configuraremos la configuración global del servidor dhcp:

```
root@test-vir-003:~# cat /etc/dhcp/dhcpd/common.conf
ddns-update-style none;
log-facility local7;
authoritative;

default-lease-time 3600;        # 1 hora
max-lease-time 7200;            # 2 horas



option domain-name "test-scenario.lan";
option domain-name-servers 192.168.84.1, 192.168.84.2;
option ntp-servers ntp1.test-scenario.lan, ntp2.test-scenario.lan;
```

Configuraremos OMAPI

Para ello crearemos una key omapi en cada servidor (las keys serán distintas).

```
root@test-vir-003:~# cd /etc/dhcp/dhcpd/omapi
root@test-vir-003:/etc/dhcp/dhcpd/omapi# dnssec-keygen -r /dev/urandom -a HMAC-
    MD5 -b 512 -n HOST omapi_key
Komapi_key.+157+10766
root@test-vir-003:/etc/dhcp/dhcpd/omapi# cat Komapi_key.+157+10766.private
Private-key-format: v1.3
Algorithm: 157 (HMAC_MD5)
Key: O1c/Z6nvasIlWAlXXXXUk8bidth8YzMiI2/trtOhfe4ZLbGXXXXXX6+PPR/N5oMB0bO4UdUJbGy7
    /ZdjFeyg+g==
Bits: AAA=
Created: 20150130164539
```

```
Publish: 20150130164539
Activate: 20150130164539
```

Y realizaremos la configuración en un fichero copiando la key que nos ha dado:

```
root@test-vir-003:/etc/dhcp/dhcpd/omapi# cat omapi.conf
key omapi_key {
        algorithm HMAC-MD5;
        secret "O1c/Z6nvasIlWAlNKUkXXXXXh8YzMiI2/trtOhfe4ZLXXX+PPR/
    N5oMBOb04UdUJbGy7/ZdjFeyg+g==";
}

omapi-port 7911;
omapi-key omapi_key;
```

Configurando el "peer", indicaremos la configuración del otro servidor dhcp:

```
root@test-vir-003:/etc/dhcp/dhcpd/peers# cat peer.conf
failover peer "test-scenario" {
        primary;
        address 192.168.84.3;
        port 1647;

        peer address 192.168.84.4;
        peer port 1647;
        max-response-delay 30;
        max-unacked-updates 10;
        load balance max seconds 3;
        mclt 300;
        split 128;
}
```

Y en el otro servidor:

```
root@test-vir-004:/etc/dhcp/dhcpd/peers# cat peer.conf
failover peer "test-scenario" {
        primary;
        address 192.168.84.4;
        port 1647;

        peer address 192.168.84.3;
        peer port 1647;
        max-response-delay 30;
        max-unacked-updates 10;
        load balance max seconds 3;
        mclt 300;
        split 128;
}
```

Será necesario indicar el nombre del peer, la dirección local en la que escucha...

Configurando las redes y concesiones. Las realizaremos en el directorio `/etc/dhcp/dhcpd/leases`.

Aunque no demos direcciones, deberemos definir siempre la subred en la que se encuentra nuestro servidor dhcp y realizar definiciones también de las subredes en las que hagamos definiciones estáticas:

```
root@test-vir-003:/etc/dhcp/dhcpd/leases# cat subnet-192.168.84.0.conf

subnet 192.168.84.0 netmask 255.255.255.0 {

}
root@test-vir-003:/etc/dhcp/dhcpd/leases# cat subnet-192.168.85.0.conf

subnet 192.168.85.0 netmask 255.255.255.0 {

}
```

Aquí por ejemplo definiremos un pool en el que servirán direcciones los dos servidores dhcp:

```
root@test-vir-003:/etc/dhcp/dhcpd/leases# cat subnet-192.168.81.0.conf

subnet 192.168.81.0 netmask 255.255.255.0 {
        option routers 192.168.81.254;

        pool {
                failover peer "test-scenario";
                deny dynamic bootp clients;
                range 192.168.81.150 192.168.81.200;
        }

}
```

Definiremos concesiones estáticas:

```
root@test-vir-003:/etc/dhcp/dhcpd/leases# cat group-hpc_servers.conf
group {

        host test-vir-200 {
                hardware ethernet 08:00:27:73:ac:45;
                fixed-address test-vir-200.test-scenario.lan;
        }
#       aquí irían muchas mas...

}
```

Finalmente generaremos el fichero dhcpd.conf que incluirá los ficheros de configuración creados:

```
root@test-vir-003:~# cat /etc/dhcp/dhcpd.conf
include "/etc/dhcp/dhcpd/common.conf";
include "/etc/dhcp/dhcpd/omapi/omapi.conf";
include "/etc/dhcp/dhcpd/peers/peer.conf";
include "/etc/dhcp/dhcpd/leases/group-hpc_servers.conf";
include "/etc/dhcp/dhcpd/leases/group-users_workstations.conf";
```

```
include "/etc/dhcp/dhcpd/leases/subnet-192.168.81.0.conf";
include "/etc/dhcp/dhcpd/leases/subnet-192.168.84.0.conf";
include "/etc/dhcp/dhcpd/leases/subnet-192.168.85.0.conf";
```

G.2.3. Sincronización de la configuración de las concesiones

Para configurar la sincronización instalaremos en ambos servidores unison y generaremos una key que compartiremos en ambos hosts:

```
root@test-vir-003:/root# apt-get install unison
root@test-vir-003:/root# ssh-keygen -t rsa
```

Copiaremos tanto id_rsa como id_rsa.pub al otro host y los llevaremos a `/root/.ssh` (deberemos crear el diretorio con permisos 700).

En ambos hosts agregaremos la key al authorized_keys

```
root@test-vir-003:/root/.ssh# cat id_rsa.pub >> authorized_keys
root@test-vir-003:/root/.ssh# chmod 600 authorized_keys
```

Crearemos un directorio de configuración de unison y sus configuraciones:

```
root@test-vir-003:/root# mkdir -p .unison
root@test-vir-003:/root# cat .unison/dhcpleases_auto.prf
auto = true
batch = true
root = /
root = ssh://dhcp2.test-scenario.lan//
path = etc/dhcp/dhcpd/leases
logfile = /var/log/sincronizacion_unison.log
log = true
times = true
prefer = newer
root@test-vir-003:/root# cat .unison/dhcpleases_manual.prf
auto = false
batch = false
root = /
root = ssh://dhcp2.test-scenario.lan//
path = etc/dhcp/dhcpd/leases
logfile = /var/log/sincronizacion_unison.log
log = true
times = true
```

Crearemos sencillos lanzadores de la sincronización:

```
root@test-vir-003:~# cat /root/bin/sync_leases_auto.sh
#!/bin/bash

export HOME=/root
unison dhcpleases_auto
```

```
Crearemos sencillos lanzadores de la sincronización:
root@test-vir-003:~# cat /root/bin/sync_leases_manual.sh
#!/bin/bash

export HOME=/root
unison dhcpleases_manual
```

Ahora simplemente podremos lanzar la sincronización con:

```
root@test-vir-003:~# /root/bin/sync_leases_auto.sh
Contacting server...
The authenticity of host 'dhcp2.test-scenario.lan (192.168.84.4)' can't be
    established.
ECDSA key fingerprint is c4:b1:c4:83:7c:93:46:3e:da:e4:57:e4:d4:af:44:6b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dhcp2.test-scenario.lan,192.168.84.4' (ECDSA) to the
    list of known hosts.
Connected [test-vir-003 -> test-vir-004]
Looking for changes
Warning: No archive files were found for these roots, whose canonical names are:
        test-vir-004
    ....
          Waiting for changes from server
        Reconciling changes
        props     <==== props       etc/dhcp/dhcpd/leases/group-hpc_servers.conf
        props     <==== props       etc/dhcp/dhcpd/leases/group-users_workstations.
    conf
        props     <==== props       etc/dhcp/dhcpd/leases/subnet-192.168.81.0.conf
        props     <==== props       etc/dhcp/dhcpd/leases/subnet-192.168.84.0.conf
        props     <==== props       etc/dhcp/dhcpd/leases/subnet-192.168.85.0.conf
        Propagating updates
        UNISON 2.40.102 started propagating changes at 11:31:54.43 on 02 Feb 2015
        [BGN] Copying properties for etc/dhcp/dhcpd/leases/group-hpc_servers.conf
     from test-vir-004 to /
    ....
        [END] Copying properties for etc/dhcp/dhcpd/leases/subnet-192.168.85.0.
    conf
        UNISON 2.40.102 finished propagating changes at 11:31:54.44 on 02 Feb
    2015
        Saving synchronizer state
        Synchronization complete at 11:31:54  (5 items transferred, 0 skipped, 0
    failed)
```

G.3. Servidor syslog

G.3.1. Configuración de red del sistema

Tras realizar una instalación de Ubuntu Server 14.04LTS básica realizaremos una configuración manual de la interfaz de red:

```
root@test-vir-005:~# cat /etc/network/interfaces
# Fichero generado mediante config_net_ifaces.sh

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.84.5
        netmask 255.255.255.0
        gateway 192.168.84.254
        dns-nameservers 192.168.84.1 192.168.84.2
        dns-search test-scenario.lan
```

G.3.2. Instalación de base de datos

```
root@test-vir-005:~#  apt-get install mysql-server
```

Estableceremos una contraseña en el proceso de instalación. `root@test-vir-005:~#  apt-get install rsyslog-mysql`

Nos preguntará si deseamos usar dbconfig-common para crear la base de datos, diremos que no. La crearemos nosotros a mano.

Borraremos el fichero creado:

```
root@test-vir-005:~#  rm /etc/rsyslog.d/mysql.conf
```

Crearemos la base de datos

```
root@test-vir-005:~#  mysqladmin -uroot -p create syslogdb
```

Importaremos la base de datos:

```
root@test-vir-005:~#  mysql -uroot -p syslogdb < /usr/share/dbconfig-common/data/rsyslog-mysql/install/mysql
```

Crearemos un usuario con privilegios en la base de datos:

```
root@test-vir-005:~\#  mysql\_setpermission root@test-vir-005:~#
    mysql_setpermission
Password for user  to connect to MySQL:
######################################################################
## Welcome to the permission setter 1.4 for MySQL.
## made by Luuk de Boer
######################################################################
What would you like to do:
  1. Set password for an existing user.
  2. Create a database + user privilege for that database
```

```
        and host combination (user can only do SELECT)
  3. Create/append user privilege for an existing database
        and host combination (user can only do SELECT)
  4. Create/append broader user privileges for an existing
        database and host combination
        (user can do SELECT,INSERT,UPDATE,DELETE)
  5. Create/append quite extended user privileges for an
        existing database and host combination (user can do
        SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,INDEX,
        LOCK TABLES,CREATE TEMPORARY TABLES)
  6. Create/append full privileges for an existing database
        and host combination (user has FULL privilege)
  7. Remove all privileges for for an existing database and
        host combination.
        (user will have all permission fields set to N)
  0. exit this program

Make your choice [1,2,3,4,5,6,7,0]: 4


Which database from existing databases would you like to select:
You can choose from:
  - information_schema
  - mysql
  - performance_schema
  - syslogdb
Which database will it be (case sensitive). Type * for any:
syslogdb
The database syslogdb will be used.

What username is to be created: syslogdb
Username = syslogdb
Would you like to set a password for syslogdb [y/n]: y
What password do you want to specify for syslogdb:
Type the password again:
We now need to know from what host(s) the user will connect.
Keep in mind that % means 'from any host' ...
The host please: localhost
Would you like to add another host [yes/no]: no
Okay we keep it with this ...
The following host(s) will be used: localhost.
########################################################################

That was it ... here is an overview of what you gave to me:
The database name       : syslogdb
The username            : syslogdb
The host(s)             : localhost
########################################################################

Are you pretty sure you would like to implement this [yes/no]: yes
Okay ... let's go then ...

Everything is inserted and mysql privileges have been reloaded.
```

G.3.3. Instalación y configuración de software

Configuramos syslogd para que almacene en mysql:

```
root@test-vir-005:~# cat /etc/rsyslog.d/71-mysql.conf
### Configuration file for rsyslog-mysql
### Changes are preserved

$ModLoad ommysql
*.* :ommysql:localhost,syslogdb,syslogdb,syslogdbpasswd
```

Reiniciamos syslog

```
root@test-vir-005:~# service rsyslog restart
rsyslog stop/waiting
rsyslog start/running, process 16441
```

Nos conectaremos a mysql y veremos que rsyslogd está almacenando datos:

```
root@test-vir-005:~# mysql -usyslogdb syslogdb -p
mysql> select count(id) from SystemEvents;
+------------+
| count(id) |
+------------+
|          4 |
+------------+
1 row in set (0.00 sec)
```

Habilitando syslog para que admita logs de los demás hosts:

```
## Habilitamos la recepción UDP y TCP

$AllowedSender UDP, 127.0.0.1, 192.168.0.0/16
$AllowedSender TCP, 127.0.0.1, 192.168.0.0/16

$ModLoad imudp
$UDPServerRun 514

$ModLoad imtcp
$InputTCPServerRun 514
```

Escuchamos tanto UDP como TCP.

Instalando la interfaz gráfica loganalyzer:

Instalaremos apache y php

```
root@test-vir-005:~#  apt-get install libapache2-mod-php5 php5-mysql php5-gd
```

Instalaremos el certificado de nuestra CA en el servidor:

```
root@test-vir-005:~# mkdir /usr/share/ca-certificates/test-scenario.lan/
root@test-vir-005:~# cp TEST_CA_Root.crt /usr/share/ca-certificates/test-scenario
    .lan/
root@test-vir-005:~# echo "test-scenario.lan/TEST_CA_Root.crt" >> /etc/ca-
    certificates.conf
root@test-vir-005:~# update-ca-certificates
```

Instalaremos los certificados de nuestro servidor:

```
root@test-vir-005:~# cp server_syslog_test.crt /etc/ssl/certs/
root@test-vir-005:~# cp server_syslog_test.key /etc/ssl/certs/
root@test-vir-005:~# chown root:ssl-cert /etc/ssl/private/server_syslog_test.key
root@test-vir-005:~# chmod 640 /etc/ssl/private/server_syslog_test.key
```

Habilitamos módulo apache ssl

```
root@test-vir-005:/etc/apache2# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create
    self-signed certificates.
To activate the new configuration, you need to run:
  service apache2 restart
```

Modificaremos `/etc/apache2/sites-available/default-ssl.conf`

```
ServerName syslog.test-scenario.lan
SSLCertificateFile      /etc/ssl/certs/server_syslog_test.crt
SSLCertificateKeyFile /etc/ssl/private/server_syslog_test.key
```

Habilitamos la configuración ssl:

```
root@test-vir-005:/etc/apache2/sites-enabled#  ln -s ../sites-available/default-ssl.conf 001-default
-ssl.conf
```

Reiniciamos apache:

```
root@test-vir-005:~#  service apache2 restart
```

Configurando autenticación de apache utilizando pam:

```
root@test-vir-005:~#  apt-get install libapache2-mod-authnz-external pwauth libapache2-mod-authz-unixgroup
```

Creamos unos ficheros

```
root@test-vir-005:~# mkdir -p /etc/apache2/includes

root@test-vir-005:~# cat /etc/apache2/includes/pwauth.conf
<IfModule mod_authnz_external.c>
                AddExternalAuth pwauth /usr/sbin/pwauth
                SetExternalAuthMethod pwauth pipe
</IfModule>

root@test-vir-005:~# cat /etc/apache2/includes/local_sudo_group.auth
AuthType Basic
AuthName "Restricted Admin staff Area"
AuthBasicProvider external
AuthExternal pwauth
Require unix-group sudo
```

Preparamos

```
root@test-vir-005:~# mkdir /etc/apache2/apps-available
root@test-vir-005:~# mkdir /etc/apache2/apps-enabled
root@test-vir-005:~# mkdir /etc/apache2/ssl-apps-enabled
```

Agregaremos la siguiente línea al fichero `/etc/apache2/sites-available/default-ssl.conf` dentro de la sección VirtualHost.

```
Include /etc/apache2/ssl-apps-enabled/
```

G.3.4. Instalación visor de logs

Bajaremos loganalyzer:

```
root@test-vir-005:~# wget http://download.adiscon.com/loganalyzer/loganalyzer
    -3.6.6.tar.gz
root@test-vir-005:~# tar zxf loganalyzer-3.6.6.tar.gz
root@test-vir-005:~# mv loganalyzer-3.6.6 /usr/local/loganalyzer
```

Creamos la configuración apache para la aplicación en apps-available

```
root@test-vir-005:~# cat /etc/apache2/apps-available/loganalyzer.conf
<IfModule mod_alias.c>
        Alias /loganalyzer /usr/local/loganalyzer/src
</IfModule>

Include /etc/apache2/includes/pwauth.conf

<Directory /usr/local/loganalyzer/src>
        AllowOverride all

        # Opciones de seguridad
        Include /etc/apache2/includes/local_sudo_group.auth
</Directory>
```

Habilitaremos la configuración:

```
root@test-vir-005:~# cd /etc/apache2/ssl-apps-enabled/
root@test-vir-005:/etc/apache2/ssl-apps-enabled# ln -s ../apps-available/
    loganalyzer.conf .
```

Reiniciaremos apache

```
root@test-vir-005:~\#  service apache2 restart
```

Pondremos la url https://syslog.test-scenario.lan/loganalyzer en un navegador nos pedirá usuario y contraseña.

Nos saldrá que debemos realizar la instalación.

En el paso 2 nos dirá que necesita escribir en el fichero `config.php`. Para ello:

```
root@test-vir-005:~# touch /usr/local/loganalyzer/src/config.php
root@test-vir-005:~# chgrp www-data /usr/local/loganalyzer/src/config.php
root@test-vir-005:~# chmod 660 /usr/local/loganalyzer/src/config.php
```

En la figura G.4 podemos ver una captura final con la aplicación instalada.

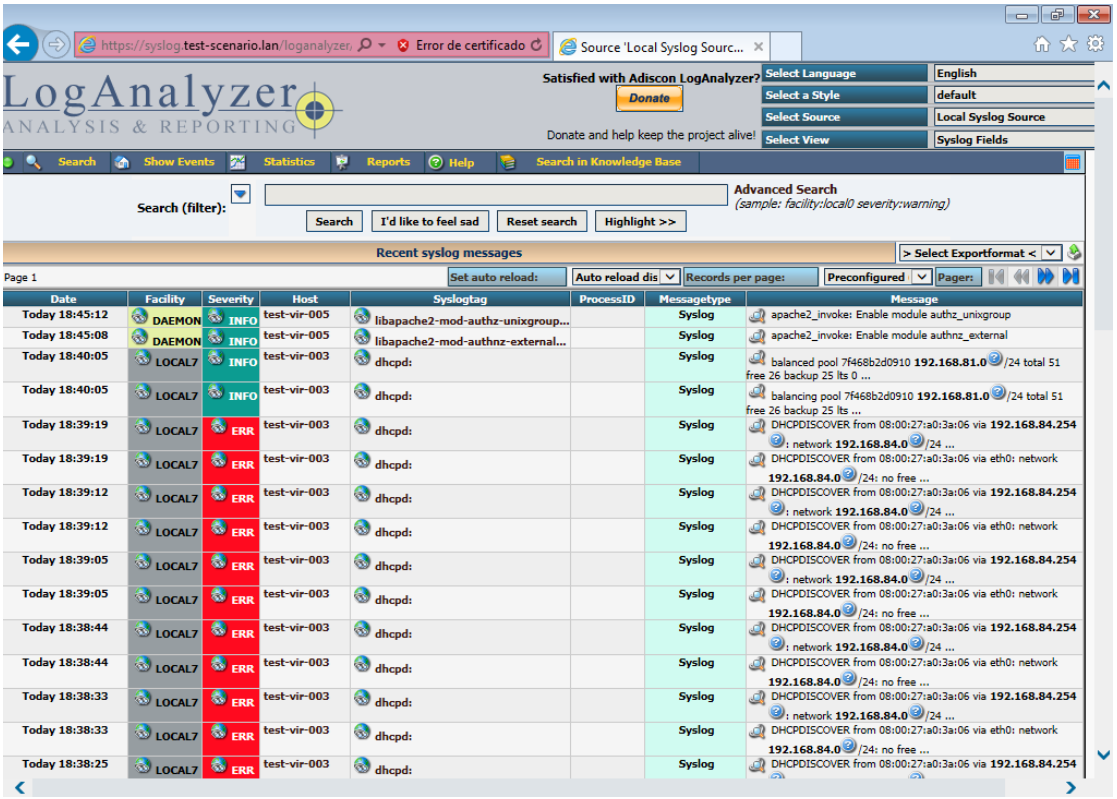


FIGURA G.4: Examinando los logs con Loganalyzer

Anexo H

Contenido del DVD

- **easybuildrepo**: repositorio easybuild desarrollado
- **escenario**: escenario de pruebas
- **lgbashlib**: código fuente librería bash
- **lgdeploy**: código fuente lgdeploy (incluye receta cefcahpc)
- **lgsetup**: código fuente lgsetup (incluye paquetes pfc-full y hpc-exec-node)
- **lgvbox**: código fuente lgvbox
- **mediciones**: resultados, fuents y gráficas mediciones realizadas
- **pdf**: documento pdf de la persente memoria
- **ubuntucefca14**: código fuente y algunos binarios de distribución ubuntucefca14
- **vídeos**: vídeos de demostración realizados

Bibliografía

[1] Charles Severance, Kevin Dowd: High Performance Computing, 2nd Edition. Editorial O'Reilly Media. 1998.

[2] Alexander Supalov , Andrey Semin , Michael Klemm , Christopher Dahnken: Optimizing HPC Applications with Intel Cluster Tools. Editorial Apress. 2014.

[3] Douglas Eadline: High Performance Computing for Dummies. Editorial Wiley Publishing. 2011.

[4] Brendan Gregg: Systems Performance: Enterprise and the Cloud. Editorial Prentice Hall. 2013.

[5] Eduardo Ciliendo, Takechika Kunimasa, Byron Braswell: Linux Performance and Tuning Guidelines. Editorial IBM. 2007.

[6] A Performance Guide For HPC Applications On the IBM System x iDataPlex. Editorial IBM. 2012.

[7] HPCBIOS. High Performance Computing for BIOinformatics Software (and beyond). Readthedocs.org.