ESCUELA POLITÉCNICA SUPERIOR

UNIVERSIDAD CARLOS III DE MADRID

PROYECTO FIN DE CARRERA

INSTALACIÓN Y CONFIGURACIÓN DE UN CLUSTER DE ALTO RENDIMIENTO

INGENIERÍA INFORMÁTICA

Área de Arquitectura y Tecnología de Computadores

Autor: Ignacio Verona Ríos

Tutor: Alejandro Calderón Mateos

Octubre 2010

A mi familia por su paciencia.

A Laura porque "*El que algo quiere algo le cuesta*"

;-)

TABLA DE CONTENIDO

Tabla de contenido 5

Índice de figuras 9

1 Introducción 11

1.1 Motivación 11

1.2 Objetivos 12

1.3 Planificación 13

1.4 Resto del documento 13

2 Tecnologías a usar 15

2.1 Introducción a los cluster 15

2.1.1 ¿Qué es un cluster? 15

2.1.2 Caracterísiticas de un cluster 16

2.1.3 Clusters de Alto Rendimiento (High Performance Clusters) 18

2.1.4 Clusters de Alta Disponibilidad (High Availability Clusters) 20

2.1.5 Tecnologías y Arquitectura Grid 21

2.2 Soluciones de clustering: Software 24

2.2.1 A nivel de aplicación (MPI, PVM) 24

2.2.2 A nivel de núcleo del Sistema Operativo (OpenMosix) 26

2.2.3 Sistemas Operativos completos (OSCAR, Rocks) 26

2.3 Soluciones de clustering: Comunicaciones 27

2.3.1 Ethernet 27

2.3.2 Myrinet 28

2.3.3 Infiniband 29

2.3.4 Hypertransport 30

2.4 Panorama actual de Clusters 30

2.4.1 Introducción 30

2.4.2 Análisis del Top500 31
2.4.3 BlueGene/L 33
2.4.4 MareNostrum 38
2.5 Clusters e iniciativas Grid en España 41
2.5.1 Altamira 41
2.5.2 CesViMa / Magerit 42
2.5.3 GridMadrid 44
2.5.4 Red Española de Supercomputación (RES) 44
3 Análisis 47
3.1 Equipos disponibles 47
3.2 Aplicaciones a ejecutar 48
3.3 Servicios requeridos 49
3.4 Seguridad 50
3.5 Escalabilidad 50
4 Diseño 53
4.1.1 Arquitectura Física 54
4.1.2 Arquitectura Red 54
5 Implantación 57
5.1 Instalación y configuración del frontend 57
5.2 Instalación de los Nodos de Cómputo 61
5.3 Tareas básicas de Gestión y Configuración 63
5.3.1 ¿Cómo añadir nuevos nodos al cluster? 64
5.3.2 ¿Cómo borrar nodos del cluster? 64
5.3.3 ¿Cómo crear nuevos usuarios en el cluster? 64
6 Pruebas de Rendimiento 67
6.1 Pruebas de red 67
6.1.1 iperf 67
6.1.2 Intel MPI Benchmark (IMB) 68
6.2 Pruebas de sistema de ficheros y disco 69

6.2.1 hdparm 69
6.2.2 Bonnie++ 69
6.3 Pruebas de CPU 69
6.3.1 stress 69
6.4 Rendimiento en conjunto 70
6.4.1 High Performance Linpack (HPL) 70
7 Ejecución y monitorización de trabajos 71
7.1 Configuración del entorno y mpirun 71
7.2 Cluster-fork 72
7.3 Cola de procesos SGE 73
7.4 Interfaz Web 75
8 Ampliaciones y actualizaciones 79
9 Conclusiones y Trabajos Futuros 81
9.1 Conclusiones 81
9.2 Trabajos futuros 81
10 Presupuesto 83
Apéndice A Trabajo previo a la instalación de Rocks 87
A.1 Instalación y pruebas individuales de cada nodo 87
A.2 Problemas con la clonación de los discos 88
Apéndice B Configuración del test High-Performance Linpack (HPL) 89
Apéndice C Virtualizando Rocks 91
C.1 Virtualización 91
C.1.1 Introducción 91
C.1.2 ¿Por qué virtualizar? 92
C.1.3 Diferentes tipos de virtualización 97
C.2 Alternativas de virtualización 108
C.2.1 VMware 108
C.2.2 BOCHS 109
C.2.3 Xen 110

C.2.4 KVM 111
C.3 Rendimiento de los sistemas de virtualización 112
C.4 Instalación de Rocks en entorno virtualizado 115
C.4.1 Posibles configuraciones virtuales con Xen Roll 116
C.4.2 Creando un cluster virtual en un solo nodo 118
11 Bibliografía 123

ÍNDICE DE FIGURAS

Ilustración 1 Grid computing 23
Ilustración 2 Top 500, Noviembre de 2006 32
Ilustración 3 estructura asic 36
Ilustración 4 arquitectura bluegene 37
Ilustración 5 red toroidal tridimensional 38
Ilustración 6 arquitectura de marenostrum 39
Ilustración 7 topología de red marenostrum 40
Ilustración 8 interconexiones marenostrum 43
Ilustración 9 arquitectura rocks 55
Ilustración 10 pantalla de arranque de rocks 57
Ilustración 11 selección de roll cd 58
Ilustración 12 datos de contacto 59
Ilustración 13 configuración de red 60
Ilustración 14 selección del tipo de nodo 62
Ilustración 15 detección de un nuevo nodo en el frontend 63
Ilustración 16 Jperf 67
Ilustración 17 portada de la interfaz web 76
Ilustración 18 estadísticas con ganglia 77
Ilustración 19 estadísticas por nodo 78
Ilustración 20 CONSOLIDACIón usando máquinas virtuales 94
Ilustración 21 pruebas usando máquinas virtuales 96
Ilustración 22 arquitectura con virtualización completa 100
Ilustración 23 arquitectura xen 104
Ilustración 24 anillos de privilegios 105
Ilustración 25 arquitectura de anillos xen 111
Ilustración 26 algoritmos de compresión 113
Ilustración 27 lecturas y escrituras en ram 114
Ilustración 28 netperf. io ethernet 115
Ilustración 29 frontend físico y nodos virtuales 117
Ilustración 30 frontend virtual y nodos virtuales 118
Ilustración 31 virt-manager 121

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

La computación paralela es, desde hace unos años, uno de los grandes exponentes en el ámbito investigador en la informática. Por supuesto, las actividades del grupo de Arquitectura y Tecnología de Computadores de la Universidad Carlos III incluyen éste tipo de desarrollos e investigaciones.

Cuando se inició el proyecto se contaba en el departamento con un pequeño cluster (Conocido como los "Patos") formado por equipos antiguos, y ya desfasados. Existía, por tanto, la necesidad de actualizar el equipamiento para ser capaces de desarrollar nuevas investigaciones en el ámbito de la computación paralela, con todo lo que ella implica (sistemas de ficheros, alto rendimiento, alta disponibilidad, *clustering*, *Grid*, etc.).

Buscando soluciones para esta carencia de equipación se pensó en diversas alternativas, tales como:

- Usar varios PC antiguos
- Usar video consolas, tipo Play Station 3[1].
- Usar equipos dedicados, comprados para tal efecto.

Evidentemente, cada alternativa tenía sus ventajas e inconvenientes. Había que pensar no sólo a corto sino también a largo plazo, y en que la inversión realizada mereciera la pena.

[1] PlayStation 3 FAT tuvo hasta la revisión 3.21 del firmware (1 de abril de 2010) la funcionalidad de ejecutar sistemas GNU/Linux. En revisiones siguientes esa posibilidad fue eliminada por Sony.

Este proyecto nació después de la compra, por parte del grupo de Arquitectura y Tecnología de Computadores, de un armario rack equipado con diez máquinas listas para usar, a las que era necesario configurar e instalar correctamente.

Después de algunas investigaciones previas, y de una larga búsqueda de alternativas y soluciones posibles, se procedió al desarrollo de éste proyecto[2].

1.2 OBJETIVOS

Los objetivos que se pretenden cumplir con el desarrollo de éste proyecto son muy diversos.

Evidentemente, se pretende obtener un rendimiento óptimo para el desembolso realizado en la compra de máquinas. Puesto que el grupo realiza multitud de trabajos en el área de la programación paralela, se pretende probar esos desarrollos en el cluster adquirido.

También es interesante la posibilidad de dotar a los miembros del grupo de una arquitectura sobre la que realizar nuevas aplicaciones, pues los investigadores del grupo trabajan en proyectos tales como:

- Paralelización de aplicaciones
- Tecnologías Grid
- Sistemas de ficheros paralelos
- Sistemas operativos paralelos

[2] En la fecha de presentación del PFC el cluster descrito en la memoria ya no está en uso pues la tecnología ha avanzado considerablemente; aún así, los conceptos tanto teóricos como prácticos descritos siguen vigentes.

Es, por lo tanto, importantísimo disponer de una plataforma adecuada sobre la que realizar todas las investigaciones, y que posibilite la creación de nuevos desarrollos.

De ahí surge la idea del proyecto, y por tanto, son los objetivos que se deben cumplir durante el desarrollo del mismo.

1.3 PLANIFICACIÓN

La planificación se puede dividir en cuatro fases bien diferenciadas, según las actividades que se realizan en cada una de ellas:

1. Trabajo previo, o "estado del arte": Previamente (o, en algunos casos, en paralelo) al desarrollo o análisis del proyecto realicé investigaciones sobre sistemas paralelos, orientados a dar servicios al departamento y las posibilidades que éstos ofrecen. Se investigó acerca de virtualización y migración de máquinas virtuales, y su aplicación en sistemas críticos.

2. Análisis: El siguiente paso comienza una vez fijados los objetivos del proyecto entorno a *clusters* de alto rendimiento. Se buscan las posibles soluciones y alternativas existentes para la instalación del cluster, aprovechando al máximo el hardware disponible, y facilitando alcanzar los objetivos prefijados.

3. Desarrollo: Una vez finalizado en análisis, el desarrollo consiste en llevar a cabo la solución elegida, y llegar a un entorno plenamente funcional, pasando por etapas de pruebas. Durante esta fase, se instala el cluster y se realizan pruebas de funcionamiento.

4. Explotación: Finalizadas las pruebas, el cluster pasa a considerarse funcional. Los usuarios pueden trabajar con él y realizar nuevos desarrollos y pruebas.

1.4 RESTO DEL DOCUMENTO

El resto del documento está organizado del una forma secuencial, siguiendo los pasos realizados en cada una de las fases explicadas anteriormente.

El Apartado 2 corresponde con la fase de Trabajo Previo, en la que se realiza una introducción a los términos y tecnologías usadas en el ámbito de *clusters* y computación distribuida o paralela.

Los apartados 3, Análisis y 4, Diseño corresponden a la fase de análisis, en la que se evalúan las posibles alternativas existentes, ofreciendo ventajas e inconvenientes, para finalmente decantarse por una de ellas.

El apartado 5, Implantación se relaciona con la fase de desarrollo, en la que se realiza la instalación de todo el hardware y software elegido en los apartados de Análisis y Diseño del proyecto.

El apartado 6, Ejecución y Monitorización de Trabajos detalla la fase de explotación, y la forma en la que los usuarios pueden trabajar con el cluster y las aplicaciones que se encuentran instaladas en el cluster.

Por último, el apartado de Conclusiones y Trabajos futuros discurre sobre las posibles ampliaciones que se pueden realizar sobre el proyecto, y las conclusiones que se pueden obtener una vez concluido el mismo.

Los tres apéndices incluidos permiten ampliar información sobre los trabajos que se realizaron antes de instalar Rocks (Apéndice A, Trabajos previos a la instalación de Rocks), una pequeña guía para configurar el test HPL, para obtener el máximo rendimiento y puntuación posibles (Apéndice B, Configuración del test *High Performance Linpack*) y en el Apéndice C se discurre sobre la utilidad de virtualizar un cluster usando Rocks como base.

2 TECNOLOGÍAS A USAR

2.1 INTRODUCCIÓN A LOS CLUSTER

2.1.1 ¿QUÉ ES UN CLUSTER?

El concepto de cluster tiene muchas definiciones posibles. De forma sencilla, se podría decir que un cluster es "un conjunto de máquinas unidas mediante una red de comunicaciones y, generalmente, trabajando por un objetivo común".

Esta descripción puede dar lugar a errores de concepto. Por ejemplo: Inicialmente, se planteó la posibilidad de desarrollar este proyecto usando como nodos de cómputo consolas de videojuegos. Conectadas mediante una red, y usando Linux como Sistema Operativo, formarían indiscutiblemente un cluster. Pero ¿qué ocurre si esas mismas consolas se unen en red para jugar? Entraría dentro de la definición, pero no sería un cluster como se entiende habitualmente.

Por eso, no es sencillo dar una definición inequívoca del término cluster. Para ello, podemos recurrir a expertos reconocidos. Del libro "Scallable Parallel Computing" de Kai Hwang y Khiwei Xu extraemos:

"Un clúster es la variación de bajo precio de un multiprocesador masivamente paralelo (miles de procesadores, memoria distribuida, red de baja latencia), con las siguientes diferencias:

- Cada nodo es una máquina que, posiblemente, no cuenta con todo el hardware.
- El nodo puede ser SMP o PC.
- Nodos conectados por una red de bajo precio como Ethernet o ATM.
- La interfaz de red no está muy acoplada al sistema de E/S.

- Todos los nodos tienen disco local.

Cada nodo tiene un sistema operativo propio, con una capa de software para soportar todas las características del clúster".

Esta definición acota mucho mejor las características de arquitectura de lo que se entiende actualmente por cluster.

Hay que destacar que las máquinas SMP no se consideran cluster, pues el medio de comunicación que utilizan los diferentes procesadores es un bus local, y no una red de comunicaciones.

2.1.2 CARACTERÍSITICAS DE UN CLUSTER

¿Qué caracteriza a un cluster? Podemos diferenciar varios elementos que necesariamente ha de tener, y otros que son opcionales. Diferenciamos:

1. Características Necesarias

Según las definiciones del apartado anterior, un cluster necesita que sus nodos estén interconectados de alguna manera. Por lo tanto, un modo de conexión es un requisito indispensable para cualquier cluster. Como veremos más adelante, se pueden usar multitud de arquitecturas de red, pero ésta siempre ha de estar presente.

Generalmente, también ha de existir una capa software que gestione el conjunto heterogéneo de elementos de proceso (nodos) del cluster. No es posible hacer esa gestión por hardware (como en los SMP) pues no se conoce a priori el tipo de elementos con los que se contará en el cluster.

2. Acoplamiento

Puesto que un cluster es un conjunto heterogéneo de recursos de cómputo, ha de existir un acoplamiento entre todos esos elementos. Esta integración se hace por software, y el hardware de red es el que produce la mayor parte de esta integración.

Si el acoplamiento es muy bajo, nos tendremos que basar en librerías de programación que nos permitan distribuir el trabajo entre los distintos nodos. Por ejemplo, podremos usar PVM o MPI para escribir los programas.

Incrementando un poco más el nivel de acoplamiento, obtenemos otros beneficios pero también problemas. Por ejemplo, si quisiéramos usar MOSIX para migrar hilos entre nodos, necesitaremos que todos los nodos usen el mismo sistema operativo, y compartan una versión común de las rutinas del núcleo que permiten esa migración.

Si el acoplamiento es muy fuerte, podemos llegar a un punto en el que no hablemos de clusters sino de sistemas operativos distribuidos, o a clusters basados en SSI (Single System Image).

3. Control

El método de control es otro punto importante del sistema. Mediante él se gestiona el funcionamiento de cluster, y de la capa software que se usa para integración/acoplamiento.

Mediante la capa de control se envían trabajos al cluster, se gestionan los usuarios, colas, carga del sistema, etc. Según cómo se lleve a cabo este control podemos hablar de:

- Control centralizado: Existe un nodo del cluster que se dedica sólo a las tareas de control. Se suele conocer a éste nodo como , y tiene el inconveniente de ser un punto único de fallo. Sin él, no se puede usar el cluster. En cambio, facilita mucho las tareas de gestión del cluster. Proporcionara servicios a los nodos de cómputo, y facilidades de administración.

- Control descentralizado: En este caso, cada nodo del cluster se gestionará por separado, haciendo muy complicadas las tareas de administración.

4. Escalabilidad

Según la Wikipedia, "la escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para o bien manejar el crecimiento continuo de trabajo de manera fluida o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

En general, también se podría definir como la capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes."

Ésta es una de las ventajas principal es de los cluster. En cualquier momento, y sin un coste muy elevado, podemos añadir o quitar nodos del mismo según las necesidades de potencia de cada momento.

Dependiendo del tipo de cluster, añadir un nuevo nodo no supondrá más que conectarlo a la red, a la alimentación eléctrica y o bien instalar el sistema operativo y la capa de control o incluso dejar que el propio sistema de control prepare el nodo para ser usado y lo incluya en la lista de nodos disponibles.

2.1.3 CLUSTERS DE ALTO RENDIMIENTO (HIGH PERFORMANCE CLUSTERS)

Éste tipo de clusters están destinados a obtener la máxima potencia de computo posible. Son el paso lógico siguiente a los grandes sistemas multiprocesador, muy costosos

y poco ampliables. Actualmente, tanto el sector de I+D como el empresarial tienden cada vez más a investigar e implementar estas soluciones, por su buena relación precio/potencia.

Como ya veremos en el apartado acerca de las soluciones de *clustering* disponibles, podemos dividir en dos tipos las implementaciones disponibles, dependiendo de a qué nivel se realicen:

- Nivel de aplicación o memoria distribuida, consiste en programar utilizando unas librerías especiales, que añaden una capa intermedia a la ejecución del programa para realizar la orquestación. Los programas han de estar específicamente diseñados para aprovechar estas soluciones.

A nivel de aplicación encontramos soluciones como PVM, MPI o incluso *clusters* Beowulf.

- Nivel de núcleo o memoria compartida, donde las características de alto rendimiento se incluyen dentro del núcleo del sistema operativo que es, por ejemplo, capaz de migrar hilos de un nodo a otro.

Una de las ventajas de este sistema es que no es necesario realizar modificaciones en los programas ya escritos, o éstas serán mínimas. En cambio, es posible que se aproveche peor la potencia del cluster, y además, el núcleo del sistema operativo se vuelve más complejo, siendo más propenso a fallos.

A este nivel, la solución más extendida es OpenMosix.

Evidentemente, los cluster de alto rendimiento se analizarán como posibilidad allí donde sea necesaria una gran potencia de cálculo, por cualquier razón. Algunos ejemplos de aplicación de estos cluster pueden ser:

- Cálculos científicos. Por ejemplo, astronómicos, o de complicadas fórmulas matemáticas.
- Simulaciones. Reacciones nucleares, dinámica de fluidos, sector automovilístico (túneles de viento) etc., son grandes clientes para este tipo de arquitecturas.
- Medicina. Síntesis de proteínas, nuevos fármacos.
- Investigación. Nuevos desarrollos sobre *clusters*, medidas de rendimiento, sistemas paralelos y distribuidos, sistemas de ficheros distribuidos, etc. Hay innumerables proyectos de investigación en torno a este campo.

2.1.4 CLUSTERS DE ALTA DISPONIBILIDAD (HIGH AVAILABILITY CLUSTERS)

Éste tipo de clusters son muy demandados desde hace, aproximadamente, una década. Se trata de arquitecturas especiales diseñadas para estar ofrecer una disponibilidad absoluta de todo tipo de servicios (lo que se conoce como disponibilidad 24/7, 24 horas al día 7 días a la semana).

Su objetivo, y por lo tanto, para lo que son diseñados, es mantener siempre activos una serie de servicios que por alguna razón resultan críticos para la empresa. Por ejemplo, contratación de billetes, correo electrónico, servicios web, etc., en los que una eventual caída del sistema podría suponer grandes pérdidas económicas.

Éstos clusters son el sustituto lógico para los caros sistemas redundantes, que en un solo sistema cuentan con hardware tolerante a fallos. Es más sencillo y económico para una empresa adquirir varios pequeños servidores e integrarlos en forma de HP cluster que comprar un solo sistema redundante. Además, como ya hemos comentado antes, las posibilidades de ampliación de un cluster son enormes.

Los usos mas comunes de los clusters de alta disponibilidad se producen en dos tipos de situaciones. Primero, las previstas. Por ejemplo, porque sea necesario actualizar algún servicio (nuevas versiones de software, parches, etc.) pero no sea posible dejar de ejecutar el mismo ni un instante. En ese caso, se puede actualizar primero un pool de servidores, dejando los redundantes dando servicio, y una vez realizada y probada la actualización, pasar a renovar esos equipos de reserva.

El segundo tipo, y también muy ventajoso, es ante fallos no esperados. Si la empresa cuenta con un único servidor destinado, por ejemplo, al correo electrónico y falla, se pierde el servicio. El tiempo de caída del servicio será la suma del tiempo que se tarda en descubrirlo mas el tiempo de reparación. Demasiado para un servicio crítico.

Si la empresa cuenta con servidores redundantes en cluster, y un servidor falla, no hay problema. El resto de equipos del cluster asumirán su carga mientras el equipo que falló es reparado. El servicio no sufrirá tiempo de caída.

Una de las soluciones libre más usadas para crear clusters Ha es Heartbeat: Es una solución incluida en las grandes distribuciones de Linux actuales, tales como Debian, SuSe. También se incluye en otros Unix como Solaris o FreeBSD. Es uno de los componentes básicos de Linux-HA (High-Availability Linux).

Mediante el envío de un pequeño paquete periódicamente a los servidores, puede detectar cual está funcionando o cual no, y realizar las medidas que tenga programadas para atajar el error en cuanto sea posible.

2.1.5 TECNOLOGÍAS Y ARQUITECTURA GRID

La arquitectura Grid es el siguiente paso en la computación distribuida. El concepto de cluster implica que las máquinas o nodos que forman el mismo estén en la misma ubicación (o muy cerca) geográficamente hablando.

Esto impide, por ejemplo, que un grupo de trabajo del Reino Unido pueda utilizar los recursos de cálculo del cluster instalado en la Carlos III. O lo que es más importante aún: Impide que se puedan sumar las potencias de cálculo de varios clusters separados por miles de kilómetros, para así conseguir uno de mayor potencia, y trabajar más rápido en los problemas a resolver.

Siguiendo con el caso anterior. Imaginemos que en la Universidad Carlos III se ha desarrollado un software muy avanzado para realizar cálculos matemáticos complejos en un entorno distribuido, y que este software se encuentra altamente optimizado para la arquitectura software y hardware del cluster que allí se tiene instalado. ¿No sería lógico que otros pudieran disponer de esa aplicación (junto con todo el hardware de cómputo y comunicaciones) sin necesidad de perder tiempo en instalarlo en su cluster local?

La tecnología o arquitectura Grid trata de solventar esos problemas. Pretende permitir que se puedan utilizar de forma coordinada recursos de muy diferentes tipos, tales como unidades de almacenamiento masivo, capacidad de cálculo , aplicaciones, etc. De esta forma, se podrían poner al servicio de otros grupos de trabajo aplicaciones o soluciones completas, permitiendo que el usuario esté a cualquier distancia de la ubicación física de los nodos de cómputo.

Además, se podría desear unir el cluster de otras ubicaciones al de la Universidad Carlos III, para así poder disponer de su potencia de cálculo. Imaginemos que varios grupos de trabajo se ponen de acuerdo para “unir” sus clusters. Tendríamos algo como un cluster de clusters. Resumiendo: Un Grid.

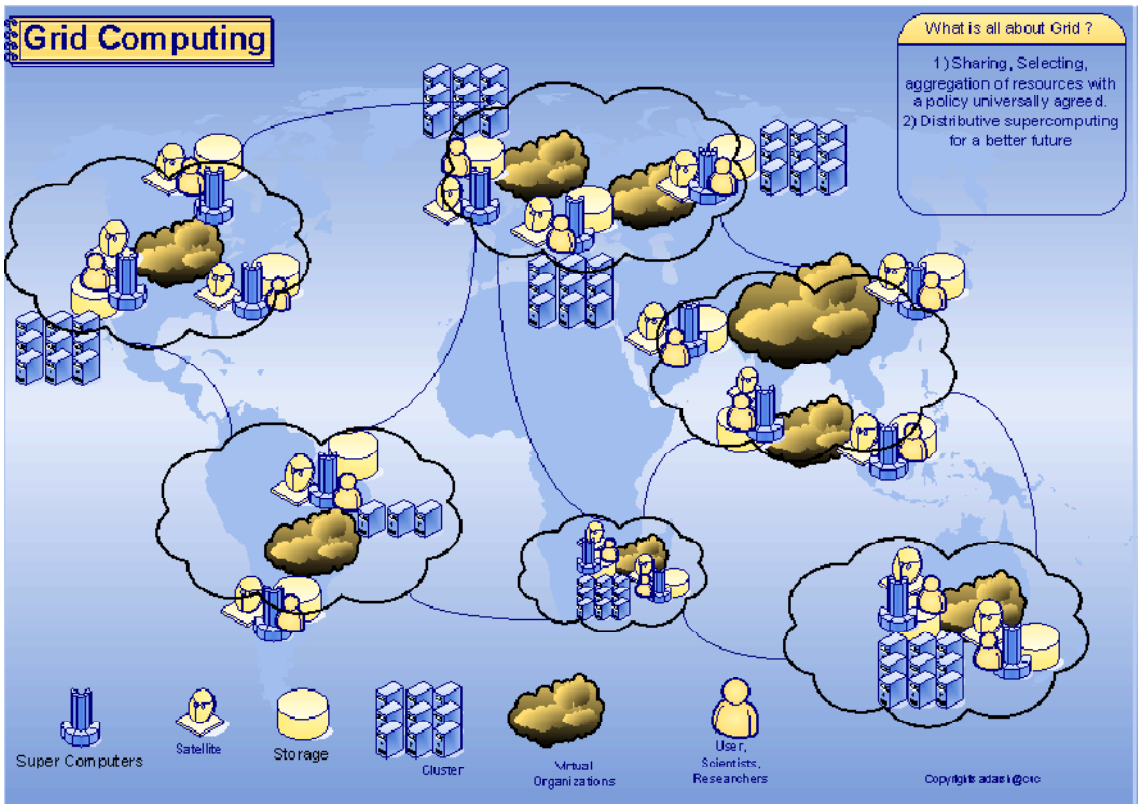

ILUSTRACIÓN 1 GRID COMPUTING

En la imagen se puede ver lo que podría ser un Grid a nivel mundial, en el que grupos de diferentes países aportan su capacidad de proceso, almacenamiento o comunicaciones a un objetivo común.

La arquitectura Grid requiere de una capa software intermedia que controle la forma en que se distribuyen y gestionan los diferentes procesos dentro del Grid. En este aspecto, la herramienta que se considera el estándar de-facto es Globus, desarrollado por IBM.

Globus permite gestionar los recursos de un Grid , publicar y descubrir servicios y gestionar el tráfico de datos y su almacenamiento. Principalmente Globus permite ejecutar trabajos en un Grid sin tener que preocuparse de qué máquinas, clusters o nodos de almacenamiento lo forman. Se podría considerar el equivalente al frontend de un cluster.

Otra ventaja de esta capa intermedia que proporciona Globus es la posibilidad de crear entornos heterogéneos, pudiendo usar los integrantes del Grid cualquier sistema operativo soportado: Apple OS X, RedHat, Fedora Core, Debian, Mandrake, Gentoo, SuSE, HP-UX, AIX, FreeBSD o Solaris. De esta forma, se amplían aún más las posibilidades.

La arquitectura Grid será el próximo paso en la evolución de los sistemas distribuidos. Se espera que con la implantación masiva de Ipv6 en Internet se desplieguen rápidamente, pues Ipv6 permite ciertos niveles de servicio que no están por ahora disponibles con Ipv4.

Algunos proyectos como seti@home, para la búsqueda de civilizaciones extraterrestres, o World Community Grid se pueden considerar parte de la tecnología grid, pues utilizan recursos geográficamente dispersos para unir potencia de cálculo con el fin de resolver problemas complejos en el menor tiempo posible.

2.2 SOLUCIONES DE CLUSTERING: SOFTWARE

Existen diferentes maneras de afrontar la construcción de un cluster desde el punto de vista del software. Dejando a un lado del hardware y la arquitectura física del sistema, las aplicaciones necesitan un método que les permita ejecutarse de forma paralela, aprovechando la potencia de cada nodo del cluster.

A continuación se detallan las diferentes aproximaciones que podemos encontrar para éste problema.

2.2.1 A NIVEL DE APLICACIÓN (MPI, PVM)

Como ya se ha dicho antes, las aplicaciones necesitan algún método para poder aprovechar la potencia disponible en el cluster. De otro modo, éstas se ejecutarían secuencialmente en la máquina desde la que se lanzan, y no obtendríamos ningún beneficio en términos de velocidad.

Cuando se habla de librerías paralelas, es decir, capas software que permiten realizar aplicaciones que se ejecuten en paralelo, MPI y PVM son los grandes conocidos. Los dos se basan en el Paso de Mensajes, el estándar de facto para la programación paralela en la actualidad.

Mediante el paso de mensajes, los diferentes nodos del cluster se comunican entre sí intercambiando datos, señales, etc. El paradigma de paso de mensajes no se usa sólo en programación paralela, sino también en otros sistemas como RMI o CORBA.

MPI es un conjunto de librerías que permiten al programador crear programas paralelos abstrayendo toda la capa de comunicaciones, y dejando que el diseño se centre en la distribución de la carga entre los nodos, en vez de tener que implementar a mano los sockets de comunicaciones.

Las librerías MPI están disponibles para C, C++, Ada y Fortran, lo que da un amplio espectro de posibilidades a los desarrolladores. Además, MPI tiene una ventaja muy importante, y es que ha sido portada y optimizada en casi todas las arquitecturas actuales, por lo que un programa MPI es completamente portable entre diferentes plataformas.

Actualmente, MPI se trata del modelo más utilizado en la programación de sistemas paralelos, muy por encima de PVM que está prácticamente en desuso.

PVM (o Parallel Virtual Machine) también es muy conocido. Aunque actualmente se está dejando de usar, a favor de MPI, ha sido la única opción durante muchos años. La primera versión fue lanzada en 1989, y desde entonces se ha usado para multitud de programas paralelos.

Al igual que MPI, se basa en el paradigma del intercambio de mensajes, distribuyendo la carga del programa entre los diferentes nodos de la red. Los programas

necesitan ser reescritos para funcionar bajo PVM, pues hay que usar si API para aprovechar la funcionalidad que ofrece.

2.2.2 A NIVEL DE NÚCLEO DEL SISTEMA OPERATIVO (OPENMOSIX)

Las soluciones a nivel del Sistema Operativo necesitan modificaciones en el núcleo del sistema. Además, por eso, se pierde algo de escalabilidad, porque todos los nodos deben tener versiones compatibles del kernel.

El objetivo de OpenMosix es conseguir un cluster usando la técnica de migrar hilos (threads) de ejecución entre nodos de forma dinámica. Mediante unos algoritmos implementados a nivel de núcleo se consigue migrar esos hilos, consiguiendo más potencia y balance de carga entre los nodos.

De cara al programador, esa migración es transparente, por lo que no es necesario hacer cambios en la aplicación siempre que ésta sea compatible con threads POSIX.

2.2.3 SISTEMAS OPERATIVOS COMPLETOS (OSCAR, ROCKS)

Recientemente se han creado distribuciones de Linux especialmente diseñadas para entornos cluster. La ventaja de éstas distribuciones es que contienen todas las herramientas necesarias para gestionar el cluster, y además, dan facilidades para instalaciones masivas, como por ejemplo la posibilidad de realizarlas desatendidas y automáticamente.

OSCAR fue el primer proyecto de éste tipo realizado por la comunidad Open Source . Es un entorno completamente integrado para la instalación sencilla de clusters de alto rendimiento.

Todo lo que se necesita para instalar, construir, mantener o programar un cluster se incluye en la distribución. OSCAR se instala sobre cualquier distribución de Linux soportada y construye una infraestructura para añadir nuevos nodos, gestionarlos, etc. Además, incluye una interfaz web de monitorización.

Rocks es otra de las distribuciones más conocidas. En este caso, no se trata de una serie de programas que se instalen sobre otra distribución Linux , sino una distribución completa.

Mediante Rocks podemos instalar un cluster partiendo de cero, configurando tanto el nodo principal (frontend o frontal) como los nodos de cómputo. Existen gran cantidad de ampliaciones fácilmente instalables, como sistemas de ficheros paralelos, o aplicaciones específicas para algunos campos.

Además, Rocks cuenta con una gran comunidad que le da soporte, con foros y listas de correo. Actualmente, algunos de los clusters incluidos en el Top 500 funcionan utilizando éste sistema.

2.3 SOLUCIONES DE CLUSTERING: COMUNICACIONES

2.3.1 ETHERNET

Ethernet y Gigabit Ethernet son los estándares más usados actualmente en redes domésticas y empresariales. Está definido por el estándar IEEE 802.3. El protocolo ethernet es muy usado por la sencillez de su topología, que actualmente siempre es en estrella, aunque anteriormente se usaba en bus.

El medio físico usado puede ser cable coaxial, cable de par trenzado o incluso fibra óptica, aunque éste ultimo medio no está muy extendido en ethernet.

Su capacidad ha variado durante los años, hasta alcanzar el máximo actual de 1 gigabit por segundo. El principal problema de ethernet para las comunicaciones entre nodos de un cluster es la latencia. Ethernet utiliza un medio compartido, y el acceso al mismo es controlado por contienda CSMA/CD, esto es: el nodo de la red que necesita enviar datos comprueba si el medio está libre, y en caso de que no lo esté (porque otro nodo ya esté enviando datos) se puede actuar de varios modos:

- Esperar un tiempo aleatorio e intentar emitir de nuevo
- Escuchar continuamente hasta que el canal este libre, y emitir en cuando se detecte.
- Escuchar continuamente hasta que el canal esté libre, y una vez detectado, ejecutar un algoritmo para decidir si emitir o no. Es una variación a lo anterior para reducir el número de colisiones.

En las dos últimas opciones se puede producir lo que se conoce como una "colisión", puesto que dos nodos pueden estar esperando a la vez a que el medio se quede libre y posteriormente emitir a la vez.

Para reducir los problemas causados por la latencia hay que segmentar adecuadamente la red, lo que reduce el número de nodos ethernet en una misma red y por ende, las colisiones que se producen al intentar transmitir todos a la vez.

2.3.2 MYRINET

Myrinet es un protocolo de red de área local diseñado por Myricom para interconectar nodos de un cluster. Gracias a que la sobrecarga debida al protocolo es menor que en otros estándares como ethernet, consigue mayores velocidades y menor latencia.

Aunque se puede utilizar como un protocolo tradicional, lo más habitual es que el software que utiliza comunicaciones basadas en Myrinet esté diseñado específicamente para usarse con éste tipo de red, por lo que está más optimizado y evita llamadas al sistema operativo para las comunicaciones.

La velocidad de transferencia puede llegar a 1.98 Gigabits por segundo de tasa mantenida, lo que supera con creces a ethernet. Pese a que la velocidad es mayor, en entornos de clusters y supercomputación el factor diferenciador de Myrinet es la latencia, mucho más baja que en ethernet como se ha comentado anteriormente.

El medio físico consiste en dos cables de fibra óptica, conectados mediante un único conector hacia un switch o un router. Myrinet incluye características avanzadas como control de flujo, control de errores o heartbeats para monitorizar continuamente el enlace.

2.3.3 INFINIBAND

Infiniband es un enlace de comunicaciones basado en el arquitectura switched fabric, diseñado principalmente para usarse en clusters y redes de almacenamiento (Storage Area Network).

Las velocidades que se alcanzan con Infiniband están muy lejos del rendimiento de otras redes como ethernet o myrinet, pudiendo alcanzar los 10, 20 o 40 Gigabits por segundo, con una latencia que se situa entorno a los 200 nanosegundos. Es capaz de alcanzar velocidades tan altas usando varios enlaces a la vez, que pueden ser, a su vez, simples, dobles o quádruples.

La topología usada en las redes Myrinet puede considerarse a la vez su punto débil y su gran ventaja. Al usar la topología switched fabric , los nodos se conectan a la vez a varios switches.

Ésta topología ofrece grandes ventajas, como la fiabilidad (en caso de que un enlace caiga el nodo sigue siendo accesible por lo demás), o la escalabilidad (una red con esta topología tiene un máximo teórico de nodos que alcanza los 16 millones).

Al igual que ofrece grandes ventajas, la arquitectura utilizada por Infiniband tiene un gran problema: el coste. Al mantener varias conexiones desde cada nodo, y tener replicados los elementos de red, el coste es exponencial.

Es por esto que éste tipo de conexiones sólo se utilizan en grandes clusters o en lugares donde las necesidades de transferencia de datos son muy elevadas, como en redes de almacenamiento (SAN). Actualmente, Infiniband es el estándar de-facto para computación de alto rendimiento, y muchos de los clusters listados en el Top500 lo utilizan.

2.3.4 HYPERTRANSPORT

HyperTransport (HT), también conocido como *Lightning Data Transport* (LDT) es una tecnología de comunicaciones bidireccional, que funciona tanto en serie como en paralelo, y que ofrece un gran ancho de banda en conexiones punto a punto de baja latencia.

La velocidad de transmisión ha ido evolucionando desde los 12.8 Gigabits por segundo iniciales hasta los 51.2 GB/s.

Habitualmente, Hypertransport se utiliza para comunicaciones entre los chips de un mismo circuito integrado, pero algunos clusters del Top500 como el Jaguar Cray utiliza ésta tecnología para interconectar los nodos de cómputo.

2.4 PANORAMA ACTUAL DE CLUSTERS

2.4.1 INTRODUCCIÓN

El mundo de los clusters y la supercomputación está en su momento de apogeo. El parqué de sistemas de éste tipo crece continuamente en el mundo científico, así como las nuevas necesidades y aplicaciones para ellos.

Cada vez son mayores las necesidades de cómputo, para tareas muy diversas, relacionadas con campos como:

- Investigaciones médicas. Síntesis de proteínas o pruebas de nuevos medicamentos.

- Investigaciones científicas. Simulaciones por ordenador, reacciones nucleares, etc.
- Aerodinámica. Simulación de túneles de viento y comportamiento de los distintos elementos.

Éstas son sólo algunas de las aplicaciones que se pueden ver funcionando actualmente en clusters. Cada vez son más las grandes empresas que demandan este tipo de sistemas, por lo que todo lo relacionado con ellos está en primera línea de investigación.

Para comprobar cuál es el estado actual de la supercomputación en el mundo, se puede acudir a la página Top500 [http://www.top500.org/] en la que se encuentran los 500 ordenadores más potentes del mundo. Es importante darse cuenta de que no todos los computadores listados en el Top500 son "clusters" tal y como los hemos definido en este documento.

Vamos a ver, por encima, qué ordenadores encabezan esa lista, para posteriormente, tratar en detalle algunos de ellos: Los que se basan en arquitecturas tipo cluster.

2.4.2 ANÁLISIS DEL TOP500

Aquí tenemos la lista de los 10 superordenadores más potentes del mundo, obtenido en Noviembre de 2006. En ella podemos observar los siguientes datos para cada instalación:

- Rango: La posición que ocupa en la lista
- Sitio: La localización física de las instalaciones.
- Computador: El nombre que recibe.
- Fabricante: Quién ha realizado y diseñado las instalaciones.

- Número de procesadores: Cuantos procesadores forman el sistema.
- Año de construcción.
- Rmax: El máximo rendimiento obtenido en el test LINPACK.
- Rpeak: Máximo rendimiento teórico del sistema.

Rank	Computer	Cores	R_{max}	R_{peak}
1	BlueGene/L - eServer Blue Gene Solution IBM	131072	280.60	367.00
2	Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core Cray Inc.	26544	101.40	127.41
3	BGW - eServer Blue Gene Solution IBM	40960	91.29	114.69
4	ASC Purple - eServer pSeries p5 575 1.9 GHz IBM	12208	75.76	92.78
5	MareNostrum - BladeCenter JS21 Cluster, PPC 970, 2.3 GHz, Myrinet IBM	10240	62.63	94.21
6	Thunderbird - PowerEdge 1850, 3.6 GHz, Infiniband Dell	9024	53.00	64.97
7	Tera-10 - NovaScale 5160, Itanium2 1.6 GHz, Quadrics Bull SA	9968	52.84	63.80
8	Columbia - SGI Altix 1.5 GHz, Voltaire Infiniband SGI	10160	51.87	60.96
9	TSUBAME Grid Cluster - Sun Fire x4600 Cluster, Opteron 2.4/2.6 GHz and ClearSpeed Accelerator, Infiniband NEC/Sun	11088	47.38	82.12
10	Jaguar - Cray XT3, 2.6 GHz dual Core Cray Inc.	10424	43.48	54.20

ILUSTRACIÓN 2 TOP 500, NOVIEMBRE DE 2006

Podemos observar varios detalles en esta lista. Primero, resalta el dominio de IBM en el campo de la supercomputación. Las 5 primeras posiciones de la lista las ocupan sistemas desarrollados por ellos.

Segundo, la diferencia entre el primer y segundo puesto. La capacidad teórica de cómputo del primer puesto es el triple que la del segundo. En el caso de la capacidad máxima obtenida en las pruebas, los resultados se acercan al triple, aunque no llegan a alcanzar esa medida teórica.

Éstas diferencias entre las medidas teóricas y reales se deben a los problemas inherentes de la programación paralela, que pese a ser ya una técnica madura y depurada, aún hace muy complejo aprovechar el 100% del rendimiento disponible en un sistema.

También es destacable el hecho de que el quinto puesto esté ocupado por un supercomputador instalado en nuestro país. Se trata del Mare Nostrum instalado en Barcelona por el BSC (Centro de Supercomputación de Barcelona, Barcelona Supercomputing Center). Está desarrollado por IBM sobre un cluster formado por 2560 blades, con un total de 10240 CPUs, 20TB de memoria y más de 300TB de almacenamiento en disco. En posteriores apartados se detallará la arquitectura de éste cluster.

2.4.3 BLUEGENE/L

BlueGene/L forma parte del grupo de supercomputadores BlueGene, diseñados por IBM. Blue Gene es una arquitectura completa, diseñada para producir una serie de supercomputadores de "nueva generación", diseñados para alcanzar potencias de cálculo en el rango de los petaflops, y que actualmente, consigue sin problemas llegar a los 360 teraflops.

Se trata de una alianza entre IBM, el Lawrence Livermore National Laboratory y el Departamento de Energía de los Estados Unidos.

Actualmente, existen cuatro supercomputadores Blue Gene en desarrollo:

- BlueGene/L
- BlueGene/C

- BlueGene/P
- BlueGene/Q

En este apartado vamos a tratar sólo el BlueGene/L, por el interés en conocer el supercomputador que ocupa el primer puesto en el Top500.

BlueGene/L, desarrollado junto con el LLNL, tiene una potencia teórica de 360TFLOPS, y obtiene una puntuación máxima sostenida de más de 280TFLOPS en el test Linpack.

La arquitectura BlueGene/L nace como fruto de la investigación que IBM inició en 1999, con el propósito de conseguir un supercomputador masivamente paralelo (massively parallel computer) para estudiar fenómenos biomoleculares, como la síntesis de algunas proteínas.

El proyecto tenía dos objetivos: Avanzar en el campo de los fenómenos biomoleculares y explorar nuevas formas de diseño y programación de sistemas paralelos. En éste último objetivo, podemos diferenciar varias sub-metas:

- Cómo aplicar la supercomputación a la consecución de avances científicos.
- Cómo conseguir aumentos de rendimiento.
- Cómo hacer más usables y accesibles los supercomputadores.
- Estudiar los costes de la supercomputación.

Finalmente, en Septiembre de 2004 el proyecto Blue Gene comenzó a dar señales de éxito. IBM anunció que un prototipo de BlueGene/L había sobrepasado el record anterior de rendimiento, ostentado hasta entonces por el Earth Simulator, de NEC.

Desde ese momento, hasta la actualidad, BlueGene/L ha ido escalándose hasta las características que tiene ahora: 280.6TFLOPS obtenidos mediante 65.536 nodos de cómputo, además de los 1024 nodos destinados a I/O.

Algunos de los objetivos obtenidos de la investigación llevada a cabo en el proyecto BlueGene/L son:

- Mantener el consumo energético dentro de límites aceptables.
- Arquitectura específica, system-on-a-chip.
- Conseguir un gran número de nodos.
- Sistema operativo ligero instalado en cada nodo

La arquitectura de BlueGene/L se basa en ASIC (Application-specific integrated circuit). Un ASIC es un circuito integrado especializado en una función particular, en contraposición a los chips multi-propósito.

Un ASIC moderno incluye procesador completo de 32 bits, ROM, RAM, almacenamiento Flash o EEPROM, etc. También se conocen, como antes vimos, como SoC o system-on-a-chip.

Continuando con BlueGene/L, cada nodo de cómputo o de I/O es un ASIC con chips de memoria DRAM asociados. El ASIC integra dos PowerPC 440 de 700Mhz, con dos unidades FPU (Floating Point Unit), una caché con controlador DRAM incluido, y la lógica necesaria para soportar varios subsistemas de comunicaciones. En la imagen inferior se puede ver la estructura de uno de los ASIC:

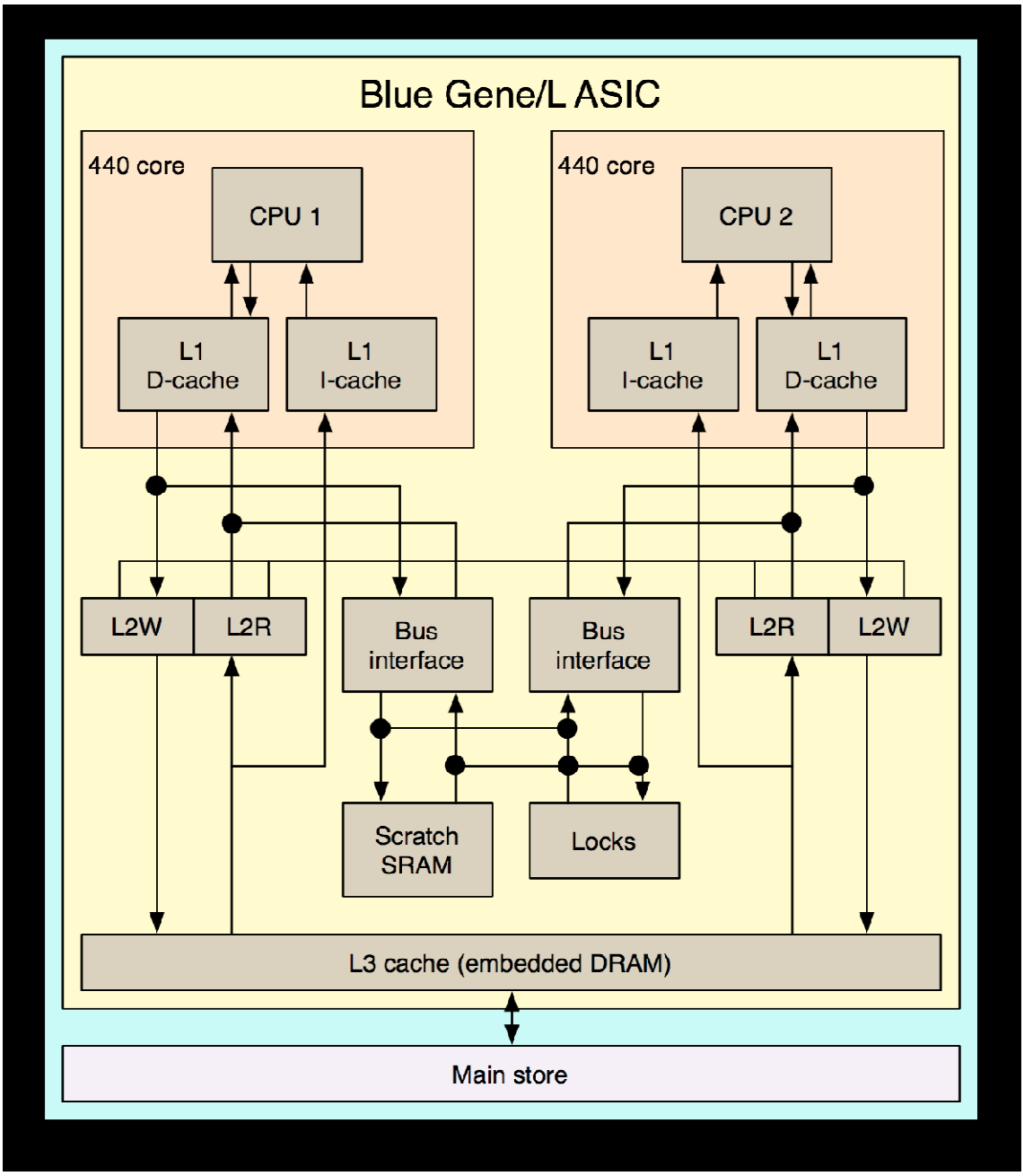

ILUSTRACIÓN 3 ESTRUCTURA ASIC

La doble FPU dota a cada nodo de BlueGene/L de una potencia teórica de cómputo de 5.6GFLOPS. Además, las CPUs de cada nodo no implementan ningún protocolo de coherencia de caché con los demás.

Gracias a las posibilidades que brindan los ASIC, integrando todos los subsistemas en un solo chip, cada nodo disipa poca potencia (por lo tanto, calor): Unos 17 watios,

incluyendo los chips DRAM. Eso permite una muy alta densidad de CPUs, permitiendo almacenar hasta 1024 nodos de computo más sus respectivos nodos de I/O en un armario rack de 19", sin exigir grandes cantidades de energía eléctrica o requisitos de ventilación. En la imagen inferior se puede ver esquemáticamente toda la arquitectura que se ha comentado en los párrafos anteriores:

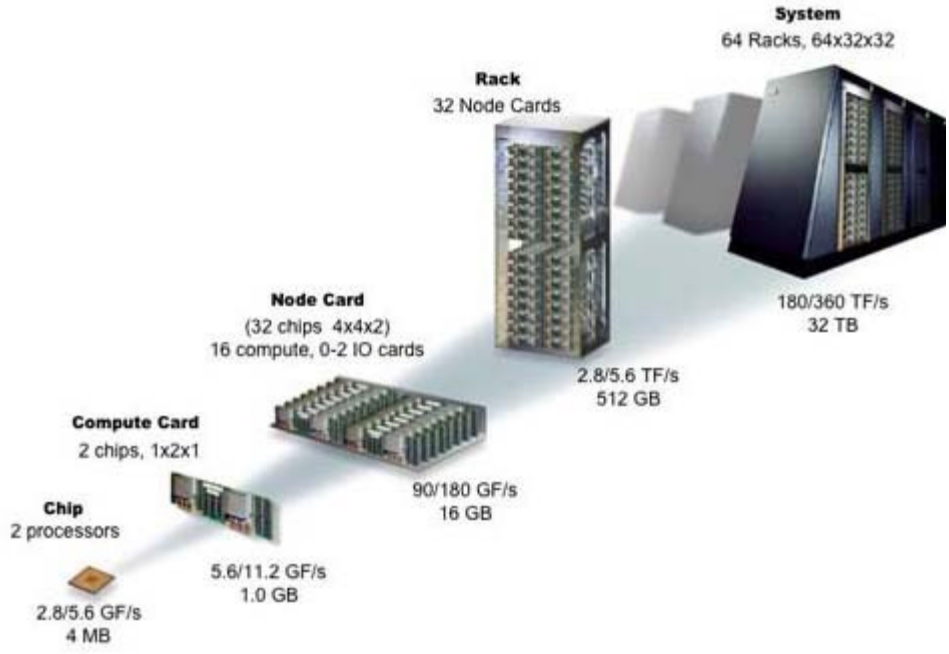

ILUSTRACIÓN 4 ARQUITECTURA BLUEGENE

En cuanto a las comunicaciones, cada nodo de BlueGene/L está conectado a tres redes de comunicaciones paralelas: Una red toroidal en tres dimensiones, para permitir comunicaciones punto a punto entre nodos; una red para comunicaciones colectivas; y una red global para permitir barreras rápidas.

La interconexión entre los nodos se realiza, como ya se ha mencionado, mediante una red toroidal tridimensional. Así, cada nodo se encuentra conectado a sus seis vecinos más cercanos. Los extremos de la matriz se conectan de nuevo al primer nodo de la fila o columna, eliminando el problema de programar una matriz tridimensional sin bordes. Esto también soluciona el problema de la conectividad de los nodos "esquina": Si esto no se hiciera, los nodos situados en los extremos no podrían estar conectados a otros 6 nodos.

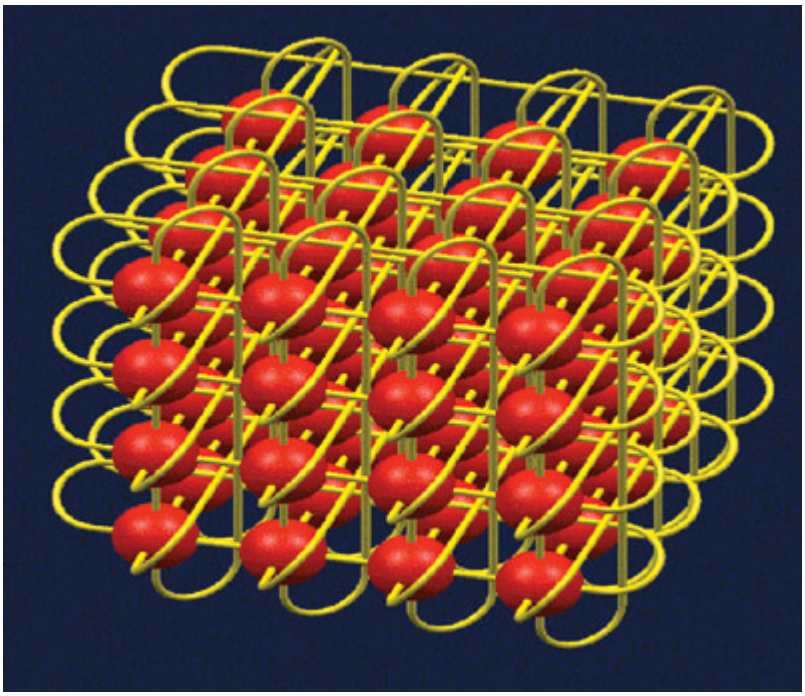

ILUSTRACIÓN 5 RED TOROIDAL TRIDIMENSIONAL

Además, los nodos de I/O, que utilizan Linux, se conectan al "mundo exterior" mediante red ethernet. También cuentan con una red ethernet distinta para permitir el acceso de configuración, arranque y diagnóstico.

Finalmente, otras dos redes forman la amalgama de comunicaciones de BlueGene/L: Una red Gigabit Ethernet, que conecta los nodos de cómputo y los nodos de I/O, y una red JTAG con propósito de permitir arrancar, controlar y monitorizar.

Los nodos de cómputo utilizan un sistema operativo muy ligero, que soporta sólo un programa de usuario simultáneamente. Para permitir ejecutar varios programas a la vez, un sistema BlueGene/L puede "dividirse" en grupos de nodos separados entre sí. Para ejecutar un programa, primero hay que reservar un grupo de nodos, y ejecutarlo en ellos.

2.4.4 MARENOSTRUM

MareNostrum, de nuevo, surge como resultado de la colaboración entre IBM y el gobierno Español en Marzo de 2004, que decidieron construir uno de los supercomputadores más rápidos de Europa.

Actualmente, MareNostrum ocupa el quinto puesto en el Top500, y a grandes rasgos, cuenta con las siguientes características:

- Rendimiento máximo de 94,2TFLOPS.
- 10240 CPUs PowerPC 970MP, funcionando a 2.3Ghz.
- 2560 blade, 44 racks y 120m2 de espacio
- 20TB de memoria principal
- Más de 300TB de almacenamiento.

La arquitectura de MareNostrum es un cluster, tal y como lo hemos definido en este documento: Se trata de varios nodos de propósito general unidos entre sí mediante algún tipo de canal de comunicación.

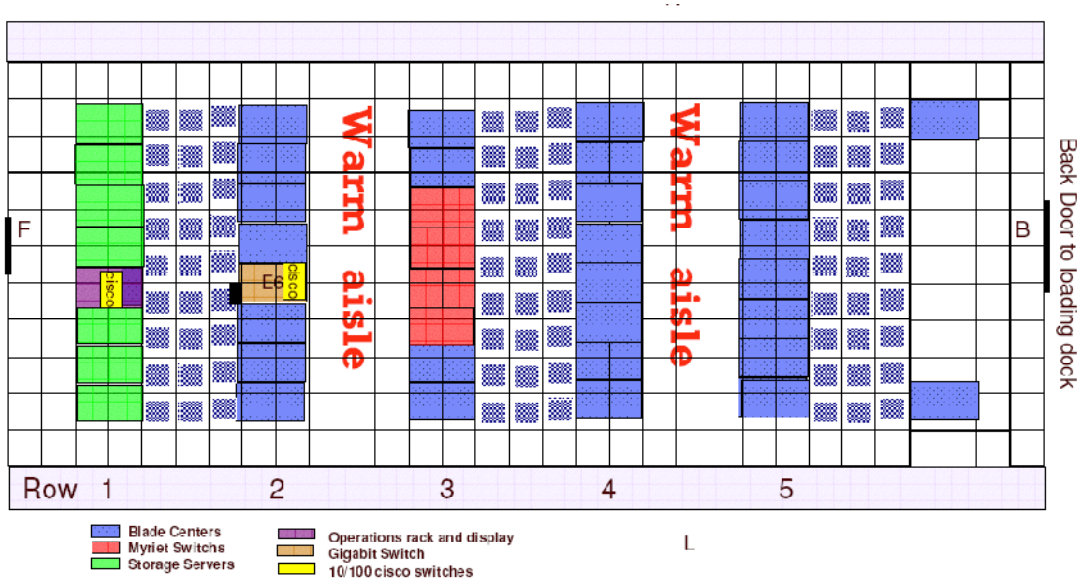

ILUSTRACIÓN 6 ARQUITECTURA DE MARENOSTRUM

Los nodos de cómputo están agrupados en blade centers. Cada uno de ellos, cuenta con 14 blades. Estas blades con computadores autónomos, pero agrupados con algunas funcionalidades en común, como la fuente de alimentación. Cada nodo cuenta con dos PowerPC 970MP a 2.3Ghz, 8Gb de memoria compartida entre las dos CPUs y un disco local de 40Gb.

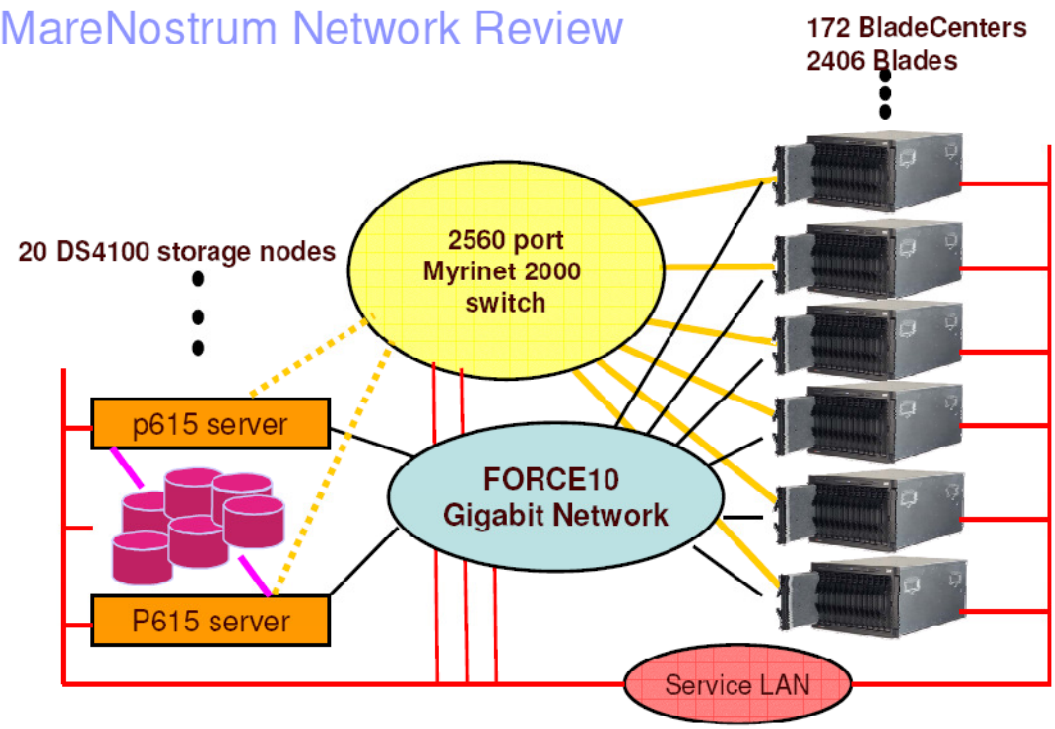

ILUSTRACIÓN 7 TOPOLOGÍA DE RED MARENOSTRUM

La red se construye a través de los adaptadores Myrinet con los que cuenta cada nodo, y se utiliza para las comunicaciones que requieren más velocidad. Además, cuenta con dos conexiones Gigabit ethernet, que se utilizan para el arranque, configuración y monitorización.

Aunque cada nodo cuenta, como ya dijimos, con un disco duro local, el sistema operativo se carga al arrancar através de la red Gigabit, minimizando los costes de mantenimiento. Más adelante se amplía la información sobre este aspecto. Además, MareNostrum cuenta con veinte servidores de almacenamiento, distribuidos en siete racks.

Gracias a ellos, se consigue 280TB, que funcionan bajo Global Parallel File System (GPFS), que combina una visión global del sistema de ficheros con un acceso paralelo al mismo. Los nodos acceden estos servidores de almacenamiento a través de la red Gigabit ethernet.

Otra de las características destacables de Mare Nostrum es la Diskless Image Management (DIM), una utilidad para permitir que la distribución Linux usada en los nodos de cómputo resida íntegramente en los servidores de almacenamiento, de forma que el nodo no tenga que gestionar el sistema de ficheros raíz.

De este modo, todos los ficheros necesarios para la operación de los nodos se obtienen a través de la red. Gracias a esto, los nodos pueden ponerse a funcionar nada más ser añadidos, sin necesitar un trabajo extra de instalación y configuración. Aún así, los nodos tienen un disco duro que se usa como almacenamiento temporal, y está preparado para ser usado por el sistema, si en el futuro se requiere.

La arquitectura Dim es la siguiente: Por cada 4 BladeCenters, hay un servidor NFS que aloja las imágenes de disco. Éste servidor, contiene 56 imágenes en modo lectura-escritura (una para cada nodo) y 1 imagen de sólo lectura, que comparten todos los nodos asociados a él. Éstas imágenes se montan vía loopback, y se exportan por NFS a cada blade.

Una gran ventaja de DIM frente a otros sistemas, es la posibilidad de cambiar las imágenes de los nodos en caliente, si necesidad de reiniciarlos.

2.5 CLUSTERS E INICIATIVAS GRID EN ESPAÑA

2.5.1 ALTAMIRA

Altamira es otra de las iniciativas de supercomputación más famosas dentro del España. Se construyó e instalo en colaboración con Instituto de Física de Cantabria (IFCA), centro mixto del Consejo Superior de Investigaciones Científicas (CSIC) y la Universidad de Cantabria.

A grandes rasgos, su arquitectura de computación , comunicaciones y almacenamiento es la siguiente:

El sistema consta de 3 bastidores (racks), cada uno con 6 blade centers, a su vez con 14 nodos con dos procesadores PowerPC 970FX. En total, cada bastidor dispone de un

total de 168 procesadores y 336 Gb de memoria principal, y una potencia pico teórica aproximada de 1,4 Tflops.

Cuenta con dis sistemas de almacenamiento, cada uno con dos nodos p615 responsables de servir las peticiones a disco, una controladora tipo FAStT100 y una unidad de expansión EXP100 (14+14=28 discos de 250 GB). La capacidad total de almacenamiento es de unos 14Tbytes.

La conexión de los nodos de cómputo entre sí y con los servidores de almacenamiento se realiza mediante una conexión Myrinet, y otra a la red Gigabit. Al igual que MareNostrum, utiliza el sistema distribuido de ficheros GPFS.

2.5.2 CESVIMA / MAGERIT

A finales del año 2004, la Universidad Politécnica de Madrid (UPM) y el Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT) decidieron aunar esfuerzos para crear el CeSViMa (Centro de Supercomputación y Visualización de Madrid). El centro, situado en el Parque Tecnológico de Montegancedo, se centra en el almacenamiento masivo de información, computación de altas prestaciones y la visualización interactiva avanzada.

Los objetivos de este centro son tres:

1. Poner a disposición de los usuarios equipos de supercomputación y visualización interactiva; ofreciendo servicios de valor añadido para facilitar la incorporación de estas tecnologías en sus campos de actividad.

2. Promocionar la utilización de la computación de altas prestaciones y de técnicas de visualización avanzadas en todos los ámbitos científicos, empresariales y de la administración.

3. Realizar actividades de investigación relacionadas con la generación de software especializado para la explotación de la supercomputación y visualización en diversos dominios de la ingeniería, la energía y el medioambiente.

Magerit es el supercomputador que se instaló a raíz de estos acuerdos. Es un clúster compuesto por 1200 nodos completamente independientes pero con la misma configuración hardware y software. Su potencia teórica alcanza los 14 Tflops.

Debido a sus dimensiones, el sistema está configurado para procesar paquetes de trabajos en modo batch. Para ello se utiliza un gestor de colas que planifica los distintos trabajos con el doble objetivo de maximizar el uso de la potencia del computador y procesar los trabajos de los distintos usuarios de la forma más rápida posible.

Magerit está compuesto por 1200 nodos eServer BladeCenter JS20 cada uno de los cuales dispone de 2 procesadores PPC de 2'2 GHz con 4 Gb de RAM. Para su interconexión se utiliza una red Myrinet de fibra óptica de altas prestaciones junto con redes auxiliares Gigabit para su control y gestión.

El sistema dispone de una capacidad de almacenamiento local de 65 Tb, proporcionada por 266 discos de 250Gb, que utiliza un sistema distribuido y tolerante a fallos.

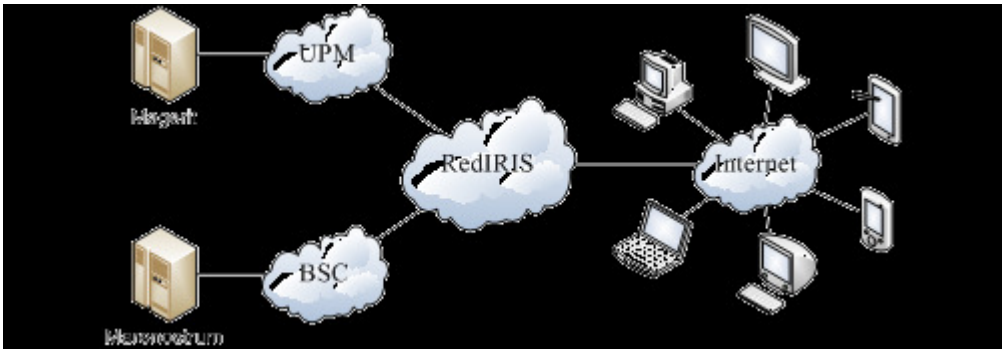

ILUSTRACIÓN 8 INTERCONEXIONES MARENOSTRUM

La conexión exterior se realiza a través de RedIRIS mediante enlaces de 1Gb y 10 Gb. Está prevista la actualización de esta conexión según lo aconseje la demanda.

Magerit es el segundo supercomputador más potente de España, sólo por detrás de MareNostrum, y se encuentra en el puesto 61 en el Top500.

2.5.3 GRIDMADRID

GridMadrid es una iniciativa que nace con dos objetivos primordiales:

- Establecer una infraestructura Grid de investigación en el ámbito de la Comunidad de Madrid, que a su vez proporcione tránsito hacia otras infraestructuras Grid nacionales e internacionales.

- Fomentar la colaboración entre las instituciones, los proyectos y las redes temáticas relacionadas con la investigación en tecnología Grid y migración de aplicaciones.

Mediante la adhesión de multitud de centros a esta iniciativa, se está creando una infraestructura muy valiosa para el estudio de las tecnologías Grid. Algunas universidades de la comunidad, como la Complutense, Politécnica, Autónoma, Rey Juan Carlos y la propia Carlos III ya se han unido al proyecto.

Está basada en Globus Toolkit versión 4, y WSRF (WebServices-Resource Framework), que forma parte de Globus.

2.5.4 RED ESPAÑOLA DE SUPERCOMPUTACIÓN (RES)

Las nuevas necesidades en la investigación y el desarrollo de las nuevas tecnologías están impulsando el desarrollo de nuevas estructuras más ágiles, descentralizadas y potentes. Ante esta situación, el Ministerio de Educación y Ciencia ha creado la Red Española de Supercomputación (RES). Se trata de una estructura de seis superordenadores distribuidos por la geografía nacional, conectados con redes de alta velocidad para dar soporte a las necesidades de supercomputación de los diferentes grupos de investigación españoles.

El núcleo de la RES se sitúa en el BSC (Barcelona Supercomputer Center), donde se encuentra instalado el superordenador Mare Nostrum, actualizado recientemente, lo que lo sitúa entre los ordenadores más potentes del mundo. Los demás nodos que forman parte de la RES son:

- Universidad Politécnica de Madrid. Rendimiento (TFlops): 21.190. Posición TOP500: 34
- Instituto de Astrofísica de Canarias. Rendimiento (TFlops): 4.506. Posición TOP500: 412
- Universidad Cantabria. Rendimiento (TFlops): 4.506. Posición TOP500: 413
- Universidad Málaga. Rendimiento (TFlops): 4.506. Posición TOP500: 415
- Universidad Valencia. Rendimiento (TFlops): 4.506. Posición TOP500: 416
- Universidad Zaragoza. Rendimiento (TFlops): 4.506. Posición TOP500: 417

En los ordenadores de la RES se ejecutaran los programas más complejos y ambiciosos de los investigadores españoles. Algunas de las líneas de investigación son:

- Estudio de proteínas y diseño de fármacos
- Genómica y ADN
- Cambio Climático y Calidad del Aire
- Simulación de barco español en la Copa América
- Formación y Evolución del Universo
- Simulación de Plasma en reactores de fusión

La mayoría de estos problemas serían inabordables sin un gran ordenador como el que ahora se constituye. El estar distribuido no supone problema para la realización de los cálculos, pues la red a alta velocidad que los une, permite intercambiar recursos e información a gran velocidad.

Se estima que la RES tendrá una potencia de cálculo de casi 150 Teraflops, aproximadamente la mitad de la que tiene el sistema que se sitúa en el número 1 del Top500 mundial.

3 ANÁLISIS

El objetivo de éste proyecto es instalar y configurar en forma de cluster los equipos adquiridos por el Grupo de Arquitectura y Tecnología de Computadores de la Universidad Carlos III de Madrid, de forma que se puedan realizar labores de investigación en el área de clusters, Grid y programación paralela.

3.1 EQUIPOS DISPONIBLES

Para llevar a cabo el desarrollo del proyecto se cuenta con los equipos adquiridos por el Departamento. Contamos con 11 equipos, de los cuales 10 serán nodos de cómputo y el restante se destina a la instalación del frontend.

Además, se cuenta con un switch Gigabit ethernet dedicado y un armario Rack para almacenar los nodos. El cableado necesario también está disponible (cableado de alimentación y cable UTP para comunicaciones).

En más detalle podemos comentar que los equipos destinados a los nodos de cómputo cuentan con las siguientes características:

- Procesadores Duales de 3.1Ghz
- 2Gb de Memoria RAM
- Disco duro de 500Gb
- Doble interfaz de red: Gigabit Ethernet + Ethernet 100Mbps

En cuanto al equipo que se instalará como frontend del cluster, sus características son las siguientes:

- Procesador Dual 3.1Ghz
- 2Gb de memoria RAM

- Disco duro de 450Gb
- Doble interfaz de red, 2xEthernet 100Mbps

Éstos son los equipos que formarán la red, correctamente instalados en el armario Rack y con sus conexiones de red/alimentación numeradas y organizadas.

Adicionalmente, en un futuro se espera contar con algunas de las máquinas disponibles en las aulas de informática pertenecientes al departamento. De ésta forma, se podría crear un grid para experimentar con ésta arquitectura. Por ahora, el proyecto se limita a las máquinas instaladas en el rack del departamento.

3.2 APLICACIONES A EJECUTAR

A la hora de abordar el diseño del cluster es importante tener en cuenta el tipo de aplicaciones que se desean ejecutar, pues la solución que se plantee dependerá de algún modo de ellas.

En principio, el cluster se usará para ejecutar aplicaciones desarrolladas por los integrantes del departamento, pues una de las labores que se realiza es la investigación en todo tipo de arquitecturas y sistemas de programación paralelos. El uso de librerías como MPI es por tanto primordial, y por supuesto se debe dotar al cluster de esa funcionalidad.

Además, otros proyectos se realizarán sobre el cluster, tales como investigación de sistemas de archivos paralelos (PVFS) y desarrollo de pruebas sobre entornos Grid, con Globus y planificadores CONDOR.

Una de las ventajas de Rocks es que es fácilmente ampliable, y permite de una manera sencilla distribuir nuevas aplicaciones y funcionalidades entre los nodos de cómputo.

Por lo tanto, los requisitos de aplicación para el funcionamiento óptimo del cluster en el departamento son:

- Disponibilidad de las librerías MPI/LAM y MPICH.
- Facilidad de ampliación e implantación de nuevas aplicaciones.

3.3 SERVICIOS REQUERIDOS

Un entorno de cluster es, por naturaleza, multiusuario. Por lo tanto, se deben ofrecer una serie de servicios que permitan la correcta convivencia de varios usuarios en el sistema, que se cumplan unos requisitos.

Por ejemplo, se debe seguir un orden en la ejecución de los trabajos de los usuarios, y además, permitir manejar las prioridades y las necesidades de cada uno. También debe existir un espacio de almacenamiento común para que todos los nodos puedan tener acceso a datos comunes.

Los servicios que se han propuesto como necesarios en la fase de análisis del cluster son:

- Sistema de Colas. Se debe implantar algún método que permita gestionar colas de trabajos en el cluster. De esta forma, los usuarios enviaran sus trabajos a la cola y se ejecutarán según el orden establecido.

Éstas colas pueden contar con ordenación dinámica, teniendo en cuenta las prioridades y necesidad de cómputo de cada trabajo, para permitir obtener el máximo rendimiento.

- Sistema de Monitorización: Debe existir un método de monitorizar el estado de cluster. Se podrán consultar parámetros como la carga del sistema en un momento dado, número de nodos en funcionamiento, nodos caídos etc.

Además, se guardan históricos de carga, para poder analizarlos posteriormente. Además, se puede visualizar el estado de las colas de trabajos y procesos asignados por nodo.

- Gestión de Cuentas: Como se ha apuntado antes, el cluster es un entorno multiusuario, que debe ser capaz de dar servicio de forma sencilla. Por lo tanto, debe existir un sistema para gestionar las cuentas de usuario, los permisos de cada uno y otros parámetros por usuario.

3.4 SEGURIDAD

Como en cualquier entorno multiusuario, la seguridad es un aspecto importante que debe ser tenido en cuenta durante el diseño.

Se debe garantizar que sólo los usuarios con permiso tengan acceso al cluster, protegiendo aquellas zonas sensibles. Esto se tendrá en cuenta a la hora de diseñar tanto la instalación del hardware como a nivel de software.

Por ello, la red estará diseñada de forma que el frontend y los nodos de cómputo estén protegidos del exterior, y el software sólo permita el acceso a los usuarios legítimos.

3.5 ESCALABILIDAD

La escalabilidad es otro de los aspectos fundamentales de los cluster, y además uno de sus principales beneficios. El término “escalabilidad” se refiere a que un sistema sea capaz de adecuarse a las necesidades de crecimiento de la demanda.

Por ejemplo, en los antiguos sistemas mainframe, utilizados durante años por grandes empresas, la única forma de ampliar el sistema era comprando caras actualizaciones de hardware al fabricante, o incluso renovando el sistema completo por otro nuevo más potente.

En cambio, mediante la arquitectura de cluster, añadir más potencia al mismo es tan sencillo como añadir nuevos nodos y permitir que el frontend los utilice, dividiendo así los trabajos de los usuarios entre ellos. Esto da mucha flexibilidad a las instalaciones informáticas.

Por supuesto, esta escalabilidad no es ilimitada. Si añadiéramos muchos nodos a un cluster podríamos tener problemas de saturación en la red, o con los algoritmos de planificación de trabajo. Pero los cluster también disponen de soluciones para esto. Se pueden crear "clusters de clusters" para llegar a arquitecturas de tipo Grid

De este modo, uno o varios *frontends* "maestros" dividen la carga entre los diferentes *clusters* que dependen de ellos, permitiendo aprovechar al máximo las capacidades de proceso.

4 DISEÑO

Un cluster, por definición, tiene una arquitectura preestablecida, en la que existe una máquina que actúa como *frontend*, o "controladora" por así decirlo, de las demás, que se conocen como nodos de cómputo, o simplemente, nodos.

La función del mencionado *frontend* es organizar y distribuir el trabajo a los nodos de cómputo, para cada uno realice pequeñas cantidades del trabajo global. De esta forma, se aprovecha al máximo la capacidad de procesamiento de cada nodo individual.

Los nodos de cómputo, por otra parte, reciben trabajo y datos de para realizar un procesamiento sobre ellos. Finalmente, los resultados se envían a un nodo indicando en el programa, para que junte o procese los resultados producidos por otros nodos.

Para evitar pérdidas de rendimiento, el *frontend* no debe formar parte de los nodos de cómputo, pues podría bloquearse e impedir el acceso a otros usuarios que deseen enviar trabajos a la cola de proceso.

Un punto muy importante en el diseño del cluster es el tipo de red elegida. Es fundamental que los canales de comunicación sean de gran capacidad, para permitir el envío masivo de datos entre nodos con la máxima rapidez, y así evitar cuellos de botella en las transmisiones.

Como ya se ha discutido en otros apartados, el material con el que contamos para la instalación del cluster es, a grandes rasgos, el siguiente:

- 10 nodos de cómputo
- *Frontend*

- Armario rack
- Switch Ethernet Gigabit dedicado
- Cableado UTP CAT/5
- Conexión a la red de la Universidad
- Ratón, teclado, monitor, switch KVM, cableado eléctrico, etc.

4.1.1 ARQUITECTURA FÍSICA

Los 10 nodos se encuentran fijados al rack mediante los tornillos correspondientes, pero con posibilidad de ser retirados fácilmente si fuera necesario dar servicio a alguno de ellos.

Además, los cables de alimentación se encuentran correctamente conectados y ordenados, fijando su recorrido, dentro de las posibilidades. Del mismo modo, los cables de red están organizados y con una etiqueta que identifica a que nodo pertenecen, para poder reconocer a simple vista si la red o el cable están funcionando correctamente.

4.1.2 ARQUITECTURA RED

Como se ha comentado antes, la red es uno de los aspectos más importantes a la hora de diseñar la instalación del cluster.

Por motivos fundamentalmente económicos, las interfaces de comunicaciones se basan en Ethernet. Éste no es el mejor tipo de red para un cluster, debido a parámetros como la alta latencia, pero si uno de los más usuales por su bajo coste, y velocidad de transferencia adecuada.

Habitualmente, el *frontend* cuenta con dos interfaces de red, una para conectar a la red externa/internet y otra para comunicarse con los nodos. De esta forma, los nodos están interconectados mediante una subred dedicada.

El esquema de red completo, mostrando los nodos, puntos de conexión y *frontend* es el siguiente:

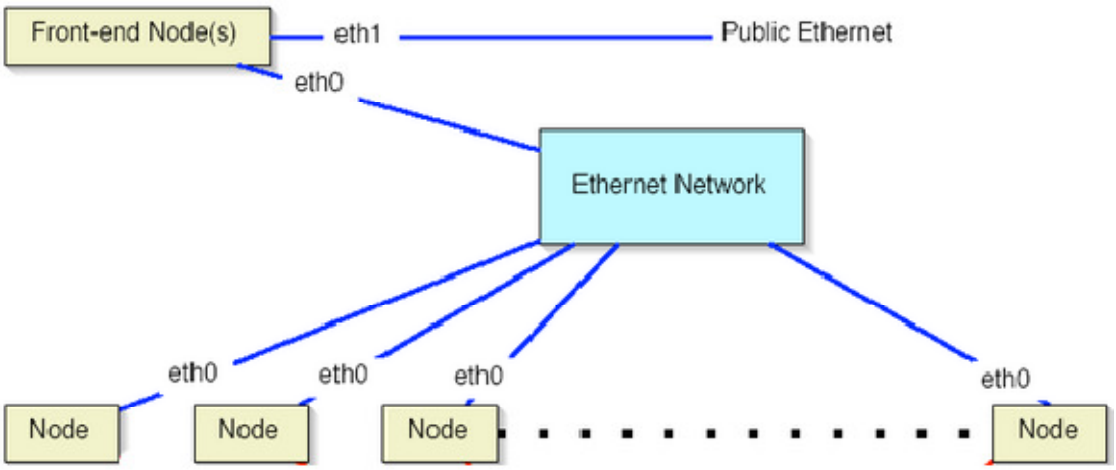

ILUSTRACIÓN 9 ARQUITECTURA ROCKS

En él podemos ver que el *frontend* cuenta con dos interfaces de red:

- eth1 que conecta a le red local de la Universidad
- eth0 que se conecta al *switch* Gigabit dedicado a los nodos.

Cada nodo cuenta con dos interfaces de red, aunque sólo uno se usa. En caso de ser necesario, se podría utilizar un interfaz para la transmisión de datos y otro para la administración, por ejemplo.

Otra posibilidad, tratada en algunos artículos de investigación es utilizar ambas interfaces simultáneamente, dividiendo la carga entre ellas a modo de balanceador, y aumentando la velocidad efectiva de transmisión.

En caso de que alguna aplicación así lo necesitara, es posible hacer que el *frontend* encamine la subred de los nodos hacia la ethernet pública de la Universidad, y de esa

forma se conseguiría por ejemplo que éstos tuvieran acceso a internet. Ésta característica se encuentra deshabilitada por defecto.

Actualmente, sólo es posible acceder a un nodo individual mediante una conexión previa el *frontend*, y realizando desde ahí accesos a los nodos. Esto se hace así por seguridad, y para evitar que el tráfico normal de la red pública ralentice la transferencia de datos entre los nodos, disminuyendo así su potencia de procesamiento.

5 IMPLANTACIÓN

5.1 INSTALACIÓN Y CONFIGURACIÓN DEL FRONTEND

Ésta sección describe cómo instalar y configurar el *frontend* de un cluster basado en Rocks. Los requisitos mínimos para la instalación son: Kernel Roll CD, Base Roll CD, HPC Roll CD y el OS Roll CD. Estos *Roll* se pueden instalar por separado o todos juntos utilizando el "Jumbo", el proceso de instalación es el mismo en los casos.

El primer paso es arrancar la máquina con el Kernel Roll CD o con el Jumbo (Base + HPC + Kernel Roll CD). Aparecerá la siguiente imagen:

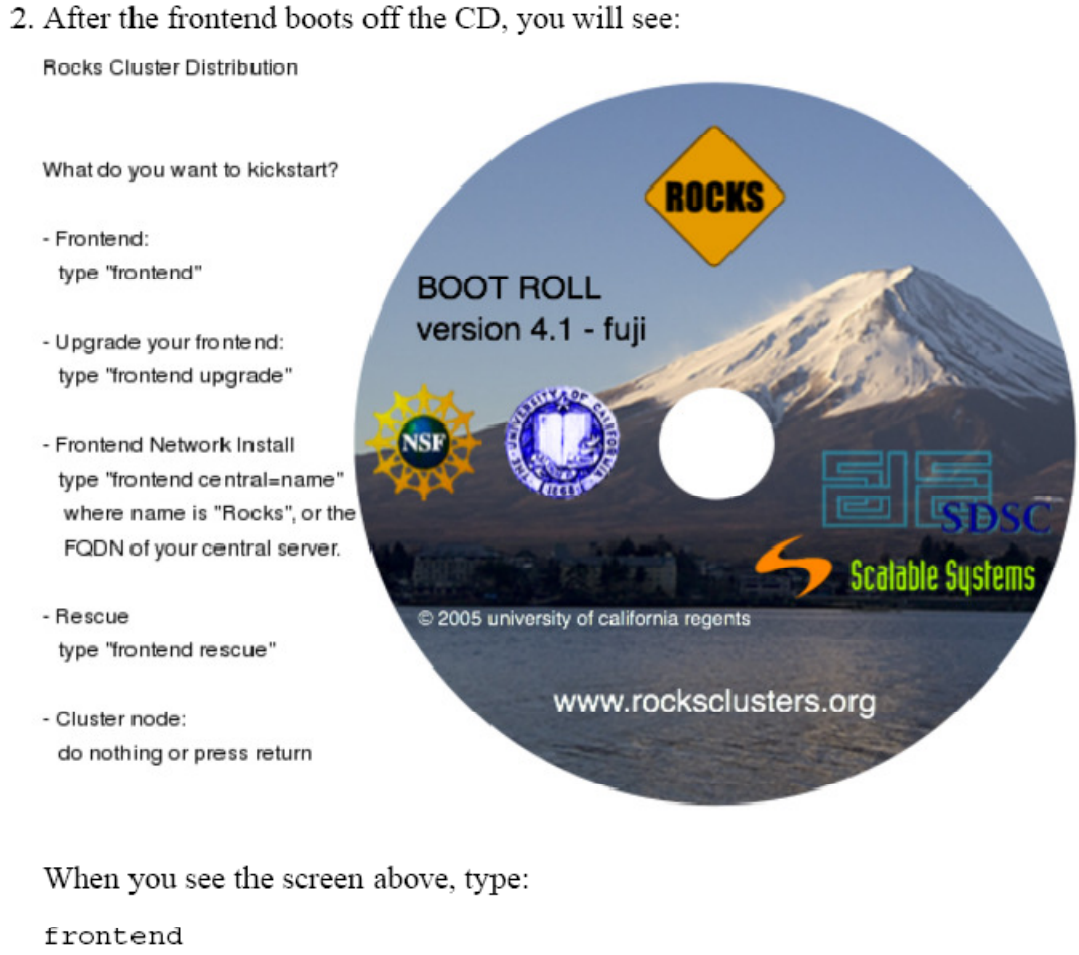

ILUSTRACIÓN 10 PANTALLA DE ARRANQUE DE ROCKS

Para instalar el *frontend*, tal y como dicen las instrucciones debemos escribir:

```
frontend
```

Es importante hacer esto rápidamente, pues sino el CD asumirá que estamos instalando un nodo del cluster y auto arrancará con ésa opción . Una vez que el sistema arranque la instalación, el siguiente paso es añadir nuevos Roll CD al *frontend*. En nuestro caso, queremos añadir el Roll para SGE. Veremos una pantalla parecida a esta:

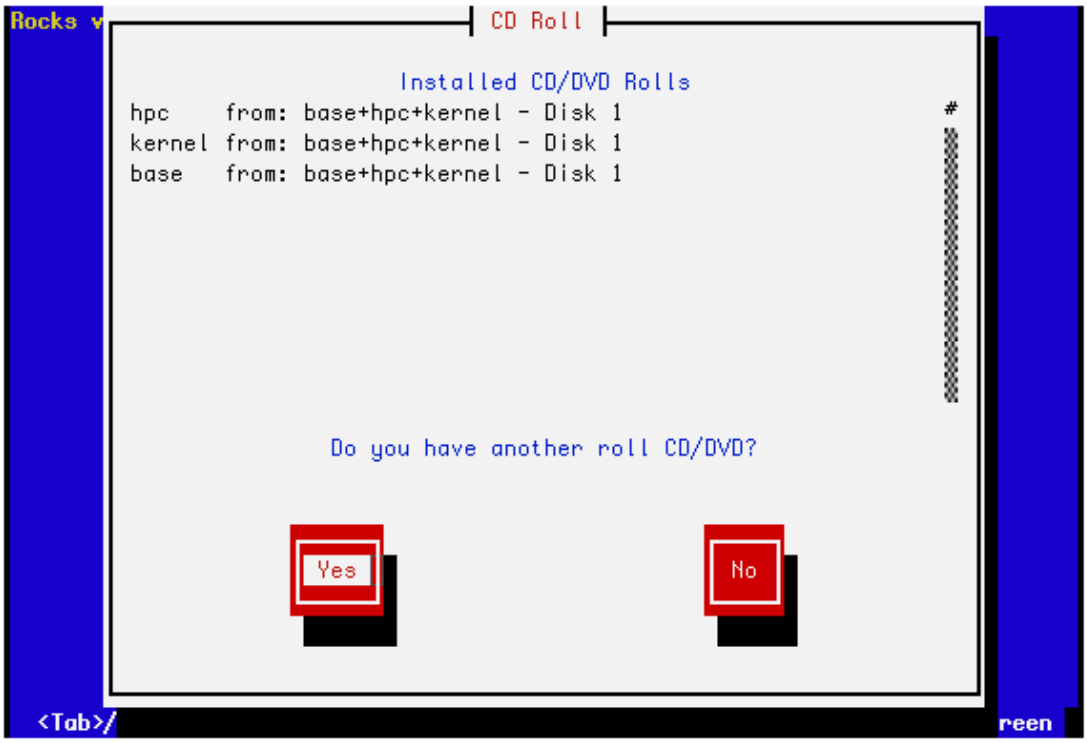

ILUSTRACIÓN 11 SELECCIÓN DE ROLL CD

Indicando los Roll CD que ya están instalados. Puesto que estamos usando el CD Jumbo, aparecerán como instalados:

area51 base ganglia hpc java kernel os sge viz web-server

Añadiremos o quitaremos Rolls según las necesidades del cluster que instalemos. Nosotros dejaremos todos menos area51 y viz, que no son necesarios para el uso que le daremos al cluster.

El siguiente paso es introducir la información para el *frontend*. Se nos pedirá un FQHN (nombre de la máquina incluyendo dominio), el nombre del cluster, y algunos datos sobre la organización y la ubicación del cluster:

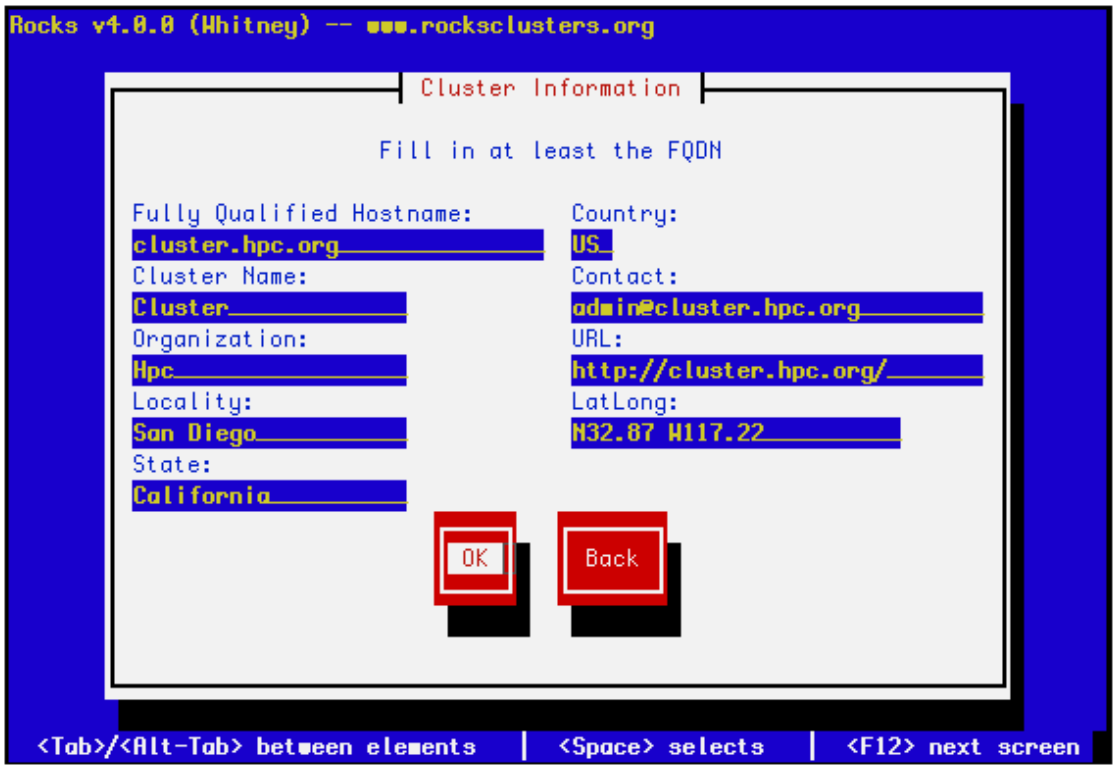

ILUSTRACIÓN 12 DATOS DE CONTACTO

Es importante rellenar correctamente el FQHN, *Cluster Name* y la URL, pues el resto de los datos no son tan importantes. Después, pasaremos a la pantalla en la que se realiza el particionamiento del disco. Seleccionaremos Autopartition, para que Rocks particione la unidad cómo él decida. Puesto que el *frontend* será un sistema dedicado, no hay problema de interferencias con otros sistemas operativos. Por defecto las particiones serán:

Partición	Tamaño
/	6Gb
Swap	1Gb
/export (enlace a /state/partition1)	Espacio libre restante

Ahora hay que configurar la red. Como ya vimos en la sección de arquitectura, Rocks utiliza dos interfaces de red, una para la parte pública, y otra, privada, para la comunicación entre los nodos.

Primero configuramos la red privada:

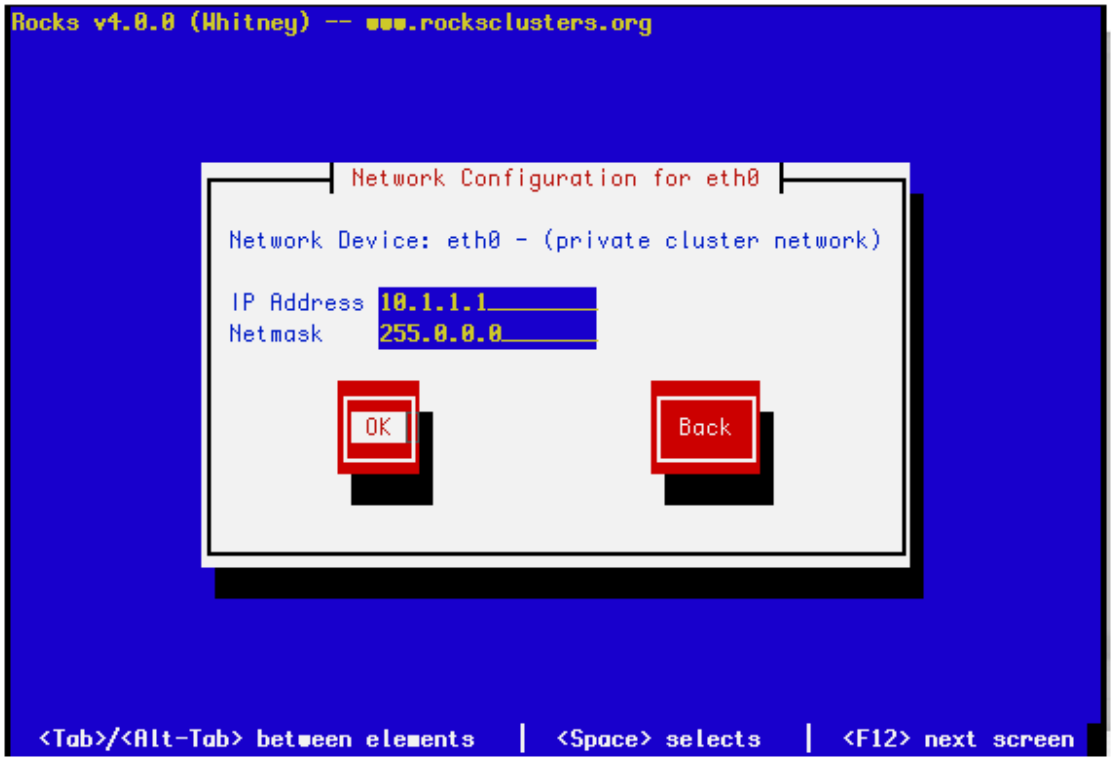

ILUSTRACIÓN 13 CONFIGURACIÓN DE RED

Podemos dejarlo por defecto tal y como viene, puesto que ésta red no interferirá con ninguna otra. Rocks actúa de cortafuegos entre la parte pública y la privada.

Después, configuraremos la red pública, con la IP y máscara que se utilicen en nuestra red local. El siguiente paso es la configuración de los parámetros de red, tales como el *gateway* o los servidores DNS.

El siguiente paso es configurar el tiempo, que nos permite sincronizar mediante NTP contra algún servidor, y la contraseña de root.

Una vez hecho esto, Rocks irá pidiendo los Roll CD que añadimos antes (si añadimos alguno) y una vez copiados todos los paquetes comenzará con la instalación de los mismos.

Después, el último paso será configurar el cargador de arranque. y algunos scripts de post-configuración se ejecutarán. Cuando la instalación termine, reiniciará la máquina.

Con éstos sencillos pasos ya tenemos el *frontend* de Rocks 4.1 instalado. Ahora sólo queda añadir nodos, y configurar algunas aplicaciones que por defecto Rocks no incluye, o necesitemos cambiar según nuestras necesidades (PVFS2, Globus, distintos planificadores, etc.).

5.2 INSTALACIÓN DE LOS NODOS DE CÓMPUTO

La instalación de los nodos es muy sencilla gracias a la infraestructura que brinda Rocks para hacerlo. La idea es que los nodos se instalan descargando toda la información necesaria desde el *frontend* (arranque 100% por red), o bien mediante el mismo CD con el que se instaló el *frontend* es posible instalar los nodos.

Para instalarlos nodos tenemos que preparar el *frontend* para que sea capaz de servir los ficheros de instalación. Para eso, tenemos que ejecutar el comando

```
inster-ethers
```

Éste comando lo que hace es capturar las peticiones DHCP hechas por los nodos al arrancar, e introducirlos en la base de datos de nodos que mantiene Rocks. En nuestro caso, tenemos que seleccionar la opción Compute como Appliance Type:

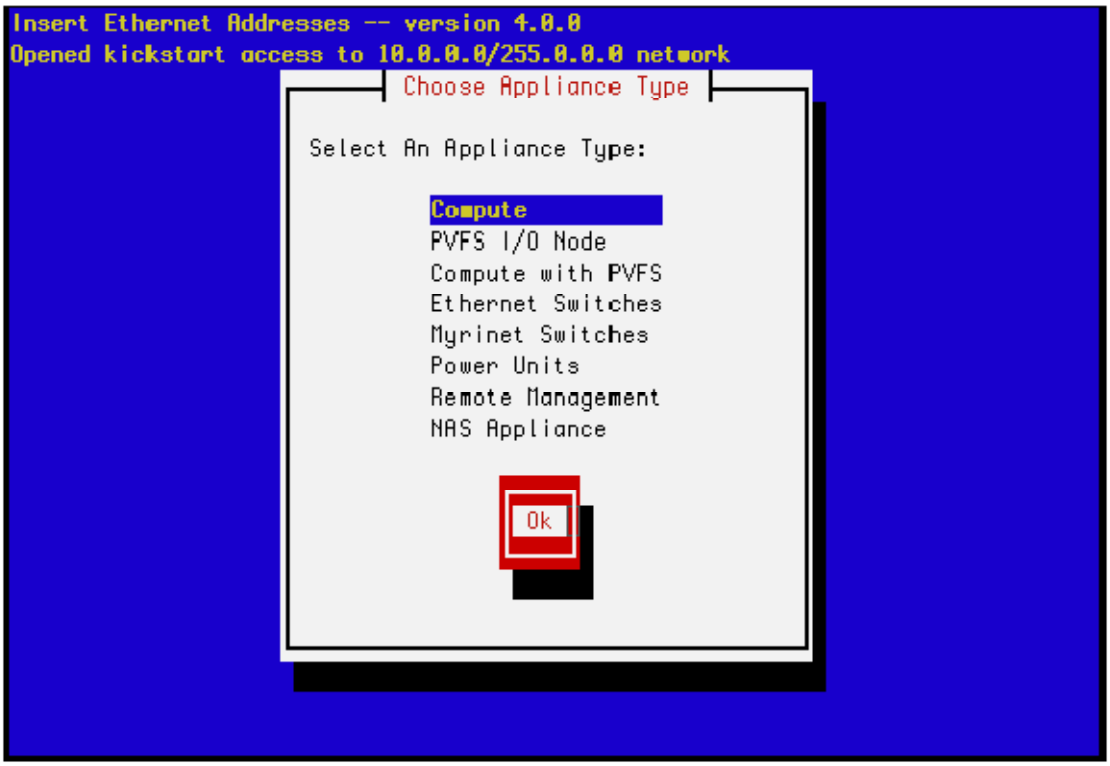

ILUSTRACIÓN 14 SELECCIÓN DEL TIPO DE NODO

En el siguiente paso, tenemos que introducir el CD que usamos para instalar el *frontend* en la unidad de los nodos, e ir encendiéndolos uno por uno, para que arranquen desde él. Como ya comentamos antes, también es posible hacer la instalación arrancando por red, usando PXE, o incluso un disquete si los nodos no tienen ninguna de las anteriores posibilidades.

Con el CD dentro, arrancamos el nodo, y cuando la máquina haya arrancado veremos que es detectada en el *Frontend*:

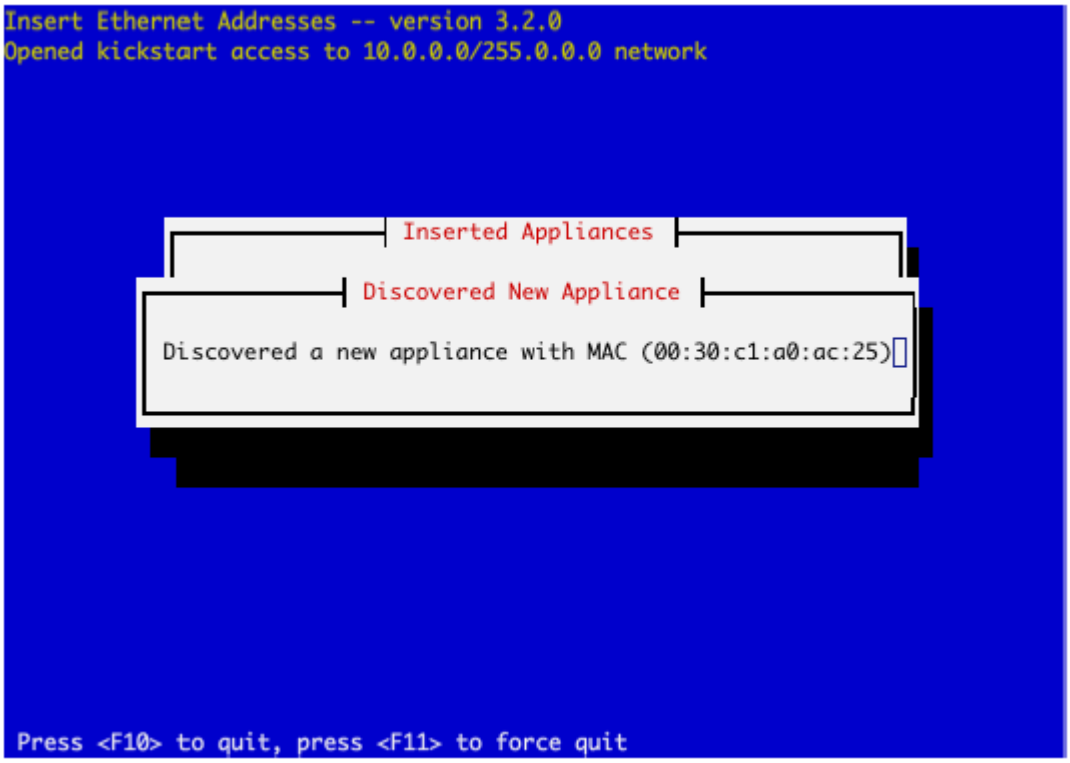

ILUSTRACIÓN 15 DETECCIÓN DE UN NUEVO NODO EN EL FRONTEND

Esto es todo lo necesario para instalar un nodo. A partir de aquí, la instalación será automática, y una vez finalizada el nodo se reiniciará y ya estará disponible para ejecutar procesos. Si en algún momento quisiéramos monitorizar el estado de la instalación podemos hacerlo por ssh, puesto que conocemos el nombre que se le ha asignado al nodo (aparecerá en el *frontend*, en el recuadro "inserted appliances". Para ello, basta con ejecutar:

```
ssh compute-x-x -p 2200
```

Después de haber instalado todos los nodos podemos salir de la aplicación pulsando F10 en el *frontend*.

5.3 TAREAS BÁSICAS DE GESTIÓN Y CONFIGURACIÓN

5.3.1 ¿CÓMO AÑADIR NUEVOS NODOS AL CLUSTER?

Una de las ventajas de los clusters en general, y de Rocks en particular, es la facilidad que ofrece para escalar la cantidad de potencia y recursos disponibles.

En cualquier momento, es posible añadir nuevos nodos, que instantáneamente pasarían a formar parte de la potencia de cómputo total disponible en el cluster.

El proceso para añadir un nodo adicional al cluster es el mismo que el explicado en el apartado 1.20. En resumen, los pasos que hay que realizar son los siguientes:

- Acceder como root al *frontend*, y ejecutar el comando insert-ethers desde la línea de comandos.
- Introducir el CD de instalación de Rocks en el/los nodos que se quieren instalar.
- Una vez finalizado el proceso, los nodos se reiniciarán, y ya formarán parte del cluster.
- Se puede salir de la aplicación insert-ethers en el *frontend* mediante la tecla F10.

5.3.2 ¿CÓMO BORRAR NODOS DEL CLUSTER?

Podemos borrar nodos de una forma muy sencilla, usando el comando insertethers -remove. Por ejemplo, para borrar el nodo compute-0-0, ejecutaríamos en la consola del *frontend*:

```
insert-ethers -remove= "compute-0-0"
```

5.3.3 ¿CÓMO CREAR NUEVOS USUARIOS EN EL CLUSTER?

Rocks utiliza un script propio para crear nuevos usuarios y propagarlos a cada nodo. También es importante destacar que los /home de cada usuario son automáticamente exportados por NFS a los nodos.

Para añadir un nodo, basta con ejecutar en el *frontend* el comando:

```
useradd nombre_usuario
```

Y automáticamente el usuario será creado y propagado a los nodos. Es importante darte cuenta de la facilidad con la que se realiza el proceso, y de que todo es gracias al sistema de gestión que integra Rocks.

Mediante él, es posible crear los usuarios en el *frontend*, y automáticamente se propagan al resto de nodos, y sus /home son compartidos a través de la red. También se habilita el acceso transparente entre nodos mediante SSH y se crea el par de claves privada/publica.

Si éste sistema de gestión no existiera, sería necesario hacer todos esos pasos a mano, con el consiguiente trabajo que conlleva. De igual modo, sería necesario eliminar el usuario de cada nodo cuando quisiera borrarse del cluster.

6 PRUEBAS DE RENDIMIENTO

En esta sección se muestran distintos métodos de realizar pruebas de rendimiento sobre clusters de alto rendimiento.

6.1 PRUEBAS DE RED

6.1.1 IPERF

Iperf se trata de una herramienta de prueba para generar tráfico TCP y UDP con el objetivo de medir el ancho de banda disponible en la red que transporta el tráfico. Consta de una aplicación servidora y otra cliente, y hace mediciones entre los dos extremos de la red donde se ejecutan dichas aplicaciones tanto en una dirección como en ambas.

Existe una interfaz gráfica para iperf, llamada Jperf que simplifica la parametrización de las pruebas [16]. A continuación una captura de pantalla de dicha herramienta:

ILUSTRACIÓN 16 JPERF

6.1.2 INTEL MPI BENCHMARK (IMB)

La suite de pruebas de rendimiento Intel® MPI Benchmark ofrece un set de pruebas orientadas a medir objetivamente el rendimiento de la mayor parte de funciones ofrecidas por MPI.

IMB no ofrece una interpretación o clasificación de los datos, se limita a ofrecer resultados referentes a los tiempos de ejecución de las pruebas. Una característica interesante de IMB es que permite ejecutar las pruebas en más de un grupo de procesos. Por ejemplo, si ejecutamos un test usando 4 procesos simultáneos, el usuario puede elegir ejecutar 2 grupos de 2 procesos, ejecutando de manera simultánea la misma prueba. Las pruebas disponibles en IMB son:

- PingPong
- PingPing
- Sendrecv
- Exchange
- Bcast
- Allgather
- Allgatherv
- Alltoall
- Alltoallv
- Reduce
- Reduce_scatter
- Allreduce
- Barrier

Para conocer más detalles sobre cada uno de los tests, así como acerca de cómo instalarlo en Rocks se pueden consultar los documentos [12] y [13]

6.2 PRUEBAS DE SISTEMA DE FICHEROS Y DISCO

6.2.1 HDPARM

hdparm es una utilidad para ver y ajustar los parámetros del hardware de los discos IDE y SATA. Puede ajustar parámetros como los caches del disco, el modo de descanso, el control de energía, la gestión acústica y los ajustes DMA.

Cambiar los parámetros de los valores por defecto a los ajustes óptimos puede aumentar el rendimiento sustancialmente. Por ejemplo, activar el modo DMA puede hacer que en ocasiones se doble o se triplique la velocidad de transferencia de datos.

6.2.2 BONNIE++

Bonnie++ es una suite de pruebas de orientada a cuantificar el rendimiento de los discos duros y sistemas de ficheros de un sistema. El test consiste en crear, acceder, modificar y borrar pequeños ficheros, con el objetivo de simular el comportamiento de aplicaciones como Squid proxy, servidores de correo etc.

Es interesante realizar esta prueba en los nodos del cluster para comprobar el rendimiento de los discos duros, pues los ficheros temporales que crean las aplicaciones se almacenan, por lo general, en el disco local mientras se está procesado un subconjunto de datos, para después enviar los resultados al nodo maestro del cluster.

6.3 PRUEBAS DE CPU

6.3.1 STRESS

Es importante conocer cómo se comporta la CPU en situaciones de carga muy alta. Stress es una aplicación que permite alcanzar cargas del 100% en la CPU, así como realizar

escrituras y lecturas en RAM. De este modo, se puede comprobar tanto la estabilidad de la CPU como el comportamiento del sistema de refrigeración.

Stress permite ejecutar varios hilos simultáneamente, pues en el caso de tener un sistema multiprocesador es necesario ejecutar tantos hilos como CPUs (lógicas o físicas) tenga el sistema. De este modo, se obtiene una carga máxima y se pueden realizar mediciones de temperatura y estabilidad a largo plazo.

6.4 RENDIMIENTO EN CONJUNTO

6.4.1 HIGH PERFORMANCE LINPACK (HPL)

HPL es una suite de pruebas de rendimiento que resuelve una serie de sistemas lineales de alta densidad, con elementos de doble precisión (64 bits) sobre un sistema distribuido. Se trata, por lo tanto, de una forma portable y libre de medir el rendimiento de un cluster.

El algoritmo que usa HPL se puede resumir en: Una distribución de datos de dos dimensiones, que se factoriza mediante una variante del algoritmo LU,

La suite HPL proporciona un programa para comprobar y medir la precisión de los resultados obtenidos, así como el tiempo invertido en producirlos. El mejor rendimiento que se puede conseguir depende de varios factores íntimamente relacionados con el sistema que se está probando, tales como el número de nodos, el tipo de interfaz de comunicaciones presente y la memoria total del sistema. Para saber cómo obtener la mejor configuración posible, consultar el Apéndice B.

7 EJECUCIÓN Y MONITORIZACIÓN DE TRABAJOS

En esta sección se explica cómo ejecutar trabajos en el cluster, usando las opciones disponibles:

- Directamente con mpirun
- Usando SGE para mandar trabajos a la cola

También se explican algunos comandos útiles para la ejecución de aplicaciones en el cluster. Todas estas utilidades asumen que hemos creado usuarios en el cluster, pues nunca se deben ejecutar como root (norma general en cualquier sistema Unix/Linux, por los riesgos de seguridad e integridad para el sistema que conlleva la ejecución como root).

7.1 CONFIGURACIÓN DEL ENTORNO Y MPIRUN

Antes de ejecutar cualquier aplicación en el cluster, es necesario configurar el entorno. Para ello, tenemos que definir la shell ssh que usaremos, y utilizar una herramienta de Rocks para propagar nuestra clave ssh al resto de nodos.

De esta forma podremos acceder mediante ssh a cualquier nodo sin que se nos pida la contraseña. También crearemos el fichero que le indica a MPI en qué máquinas debe ejecutar los procesos.

Para configurar nuestro agente ssh, tenemos que ejecutar lo siguiente (siempre como nuestro usuario, nunca como usuario root):

```
ssh-agent $SHELL
```

Una vez hecho eso, debemos propagar nuestra clave ssh a todos los nodos. Para ello utilizaremos la utilidad de rocks

```
ssh-add
```

Con estos dos sencillos pasos ya tendremos el entorno configurado. Para poder ejecutar aplicaciones usando mpirun debemos definir en un fichero qué nodos queremos usar.

Crearemos en nuestro directorio home el fichero machines cuyo contenido será el nombre de los nodos que queremos usar, por ejemplo:

compute-0-0

compute-0-1

compute-0-2

compute-0-3

Para usar los cuatro primeros nodos del cluster. Una vez guardado el fichero, ya estamos en condiciones de ejecutar cualquier programa con el comando mpirun.

Por ejemplo, podríamos ejecutar el test Linpack (incluido en Rocks, ver apéndice B para configurarlo) mediante el comando:

```
/opt/mpich/gnu/bin/mpirun –nolocal –np 2 –machinefile
machines /opt/hpl/gnu/bin/xhpl
```

7.2 CLUSTER-FORK

Cluster-fork es una utilidad proporcionada por Rocks que permite ejecutar el mismo comando en todos los nodos del cluster. Es muy útil para realizar labores de saneamiento del entorno MPI, tareas de administración, etc.

7.3 COLA DE PROCESOS SGE

SGE (Sun Grid Engine) es un planificador de procesos muy utilizado en clusters. Se utiliza para obtener el máximo rendimiento, manteniendo el cluster utilizado al 100% tanto como sea posible.

Para ello, cada proceso indica el número de CPUs que quiere usar, y SGE trata de acoplarlos entre ellos obteniendo la máxima ocupación.

Los trabajos SGE se basan en scripts, que contienen los parámetros necesarios (número de CPUs, directorio de ejecución, redirecciones de las salidas, etc.). para la ejecución del trabajo, y normalmente, tienen una llamada al mandato mpirun para ejecutar el trabajo MPI.

Un ejemplo básico de script SGE es el siguiente:

```
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#
date
sleep 10
date
```

Éste es un trabajo muy simple que ejecuta el comando sleep en un solo nodo

(no es un trabajo paralelo). Los modificadores indican que se ejecute desde el directorio actual (-cwd), que se una la salida estándar y la de error en un solo fichero (-j y) y el intérprete con el que se quiere ejecutar el script (-S /bin/bash).

El trabajo se puede enviar a la cola usando el comando qsub:

```
qsub sleep.sh
your job 16 ("sleep.sh") has been submitted
```

Un ejemplo de script para lanzar un trabajo paralelo, es el que incluye Rocks para ejecutar el test Linpack. Ahora incluimos una nueva variable, llamada $NSLOTS que contendrá el número de CPUs que queremos usar. Además, ahora el trabajo se lanza mediante mpirun para que sea el entorno MPI quien tome el control.

El script es el siguiente:

```
#!/bin/bash
#
#$ -cwd
#$ -j y
11
#$ -S /bin/bash
#
MPI_DIR=/opt/mpich/gnu
$MPI_DIR/bin/mpirun -np $NSLOTS -machinefile
$TMP/machines
\
```

```
/opt/hpl/gnu/bin/xhpl
```

El significado de las primeras líneas ya lo conocemos, por lo que pasamos a las siguientes. Lo que hace éste script es definir una variable, MPI_DIR, que contiene el directorio de la instalación de mpich, y luego llama al comando mpirun con el trabajo que queremos ejecutar. debemos llamar a mpirun pasándole $NSLOTS, y el fichero machines.

La forma de enviar este trabajo a la cola SGE es mediante el comando:

```
qsub –pe mpich 2 linpack.sh
```

Utilizando éste comando, enviamos el trabajo a SGE, indicándole que queremos utilizar mpich como entorno de ejecución y 2 procesadores (-pe mpich 2). Podemos monitorizar el estado de la cola utilizando el comando qstat, que nos mostrará una lista de procesos. Mediante el comando qdel podemos eliminar un trabajo de la cola, pasándole el job-ID tal como lo muestra qstat.

Mediante "qstat –f" podemos ver en más detalle cómo se están ejecutando los procesos, viendo a que CPU se han asignado cada uno, y la cantidad de trabajos que hay en la cola.

Toda esta información se puede obtener también a través de la interfaz web, explicada en el apartado 7.

7.4 INTERFAZ WEB

Uno de los puntos fuertes de Rocks es la interfaz Web que proporciona, que permite visualizar el estado de cada nodo, del *frontend*, de la cola de trabajos, etc. Está construida utilizando Ganglio, además de algunos otros módulos para monitorizar el estado y crear gráficos y estadísticas.

La página principal se puede acceder a través de http://tucan.arcos.inf.uc3m.es, y de ella podemos obtener también la documentación de Rocks y la sus Roll CD , así como acceder a la interfaz Ganglia:

ILUSTRACIÓN 17 PORTADA DE LA INTERFAZ WEB

Pulsando sobre "Cluster Status" accederemos al estado del cluster y sus nodos, y pulsando sobre "Users Guide" o "Roll Docs" podremos visualizar la guía del usuario y la documentación de los Roll CD respectivamente.

En la siguiente imagen podemos ver el aspecto que presenta la página principal de la interfaz Ganglia:

ILUSTRACIÓN 18 ESTADÍSTICAS CON GANGLIA

En ella se puede ver el estado en el que se encuentra el cluster, destacando:

- El número total de nodos, así como los que se encuentran activos o caídos en ese momento.
- Gráficos de carga, tanto de CPU, como de tráfico en la red.
- Gráficas de carga por cada nodo, identificadas por colores según el grado de uso en el que se encuentren.

También se puede acceder a un gráfico que muestra el esquema físico de la instalación. De él se pueden extraer también los datos de carga, número de CPUs en cada nodo, etc. :

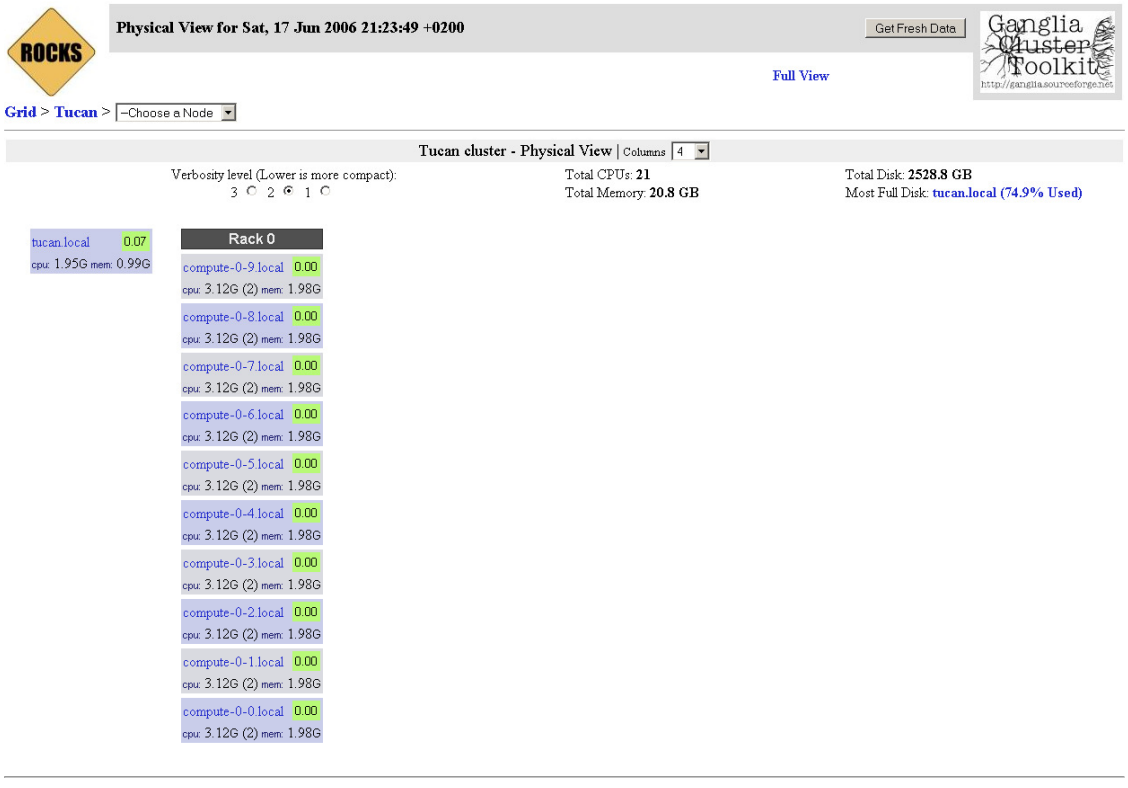

ILUSTRACIÓN 19 ESTADÍSTICAS POR NODO

8 AMPLIACIONES Y ACTUALIZACIONES

Desde el momento en el que se terminó la primera instalación del cluster hasta la fecha de entrega de éste proyecto, la distribución en la que se basa el cluster (Rocks) ha liberado varias actualizaciones.

La versión inicial que se utilizó fue Rocks 4.1 Fuji, liberada en Octubre de 2005. Esta versión fue plenamente funcional durante todo el tiempo de instalación y puesta en marcha del cluster, es decir, entre Septiembre de 2005 y Julio de 2006.

En Agosto de 2006, Rocks Clusters liberó la nueva versión 4.2 de su software para creación de clusters, Rocks, con nombre clave Hallasan, y unos meses después esa versión fue instalada por el personal del grupo ARCOS del Departamento de Informática de la Universidad Carlos III. Entre otras, las mejoras que ofrece se pueden destacar las siguientes:

- Instalación gráfica. Hasta la versión anterior de Rocks, la 4.1 Fuji, la instalación se realizaba, tanto para el *frontend* como para los nodos, en modo texto desde la línea de comandos, utilizando el instalador proporcionado por la distribución. Desde la liberación de la versión 4.2 está disponible el instalador gráfico de Red Hat, Anaconda. En cuanto a los nodos de cómputo, se puede comprobar el estado de la instalación en modo gráfico, utilizando la herramienta rocks-console que, mediante el servidor VNC de cada nodo, permite ver el progreso de la instalación.

- Restore Roll. Este Roll permite actualizar o reinstalar muy fácilmente el cluster Rocks. Antes de que esta herramienta existiera, al pasar de una versión a otra del software del cluster, o al tener que reinstalar por cualquier causa, había que crear una copia de seguridad de las cuentas de usuario, así como de otra información de configuración del cluster. La herramienta Restore Roll crea una imagen ISO que permite volver a cargar esa información en el cluster recién instalado o actualizado.

- Actualización de DHCP. Un problema recurrente, que se daba durante cualquier instalación de Rocks era el siguiente: Si un nodo de cómputo estaba conectad

simultáneamente a dos redes, mediante sus dos interfaces ethernet, en el momento de hacer la petición DHCP para obtener una dirección IP (supuestamente, la del *frontend*) atendía a cualquier respuesta, por lo que podía considerar a un servidor externo como *frontend*. Con esta nueva actualización, los nodos sólo atienden a las respuestas DHCP de un *frontend* Rocks válido.

Las anteriormente citadas son las dos mejoras más importantes de la nueva versión 4.2, pero además, se incluyen las siguientes actualizaciones y correcciones de errores:

- Sistema Operativo actualizado a CentOS 4/update 3, con todas las actualizaciones disponibles a 6 de Agosto de 2006.
- SGE actualizado a SGE 6 update 8.
- Globus actualizado a Globus Toolkit 4.0.2.
- Tentakel se ha añadido como alternativa a cluster-fork (ver [14])
- Open MPI se ha actualizado a la versión 1.1, y se instala automáticamente con el HPC Roll.
- El entorno de ejecución y desarrollo de Java se ha actualizado a la versión 1.5.0_07.

A fecha de Agosto de 2009, Rocks 5.2 está disponible, con algunas mejoras muy interesantes como el Xen Roll, que permite la creación de clusters virtuales basados en Xen de manera muy sencilla.

9 CONCLUSIONES Y TRABAJOS FUTUROS

Una vez completado el desarrollo del proyecto, es el momento de resumir las conclusiones y los datos obtenidos durante el tiempo invertido en él. Además, se pueden plantear algunas ampliaciones y mejoras que, evidentemente, se pueden realizar sobre el proyecto.

9.1 CONCLUSIONES

El ámbito en el que se ha desarrollado este proyecto, los clusters, es un tema actual, y sobre el que aún se están desarrollando multitud de investigaciones. El proyecto desarrollado a lo largo de éste documento pretende servir de punto de partida para otros proyectos. Es decir: Se pretende ofrecer una base sólida para el desarrollo de otros proyectos basados en clusters y sistemas distribuidos.

Durante el análisis y la implantación del cluster aquí descrito, se ha intentando hacer hincapié en los aspectos fundamentales de ésta tecnología, básicamente la potencia y las posibilidades de escalabilidad que ofrece.

A nivel personal, he disfrutado mucho con el proyecto. Se trata de algo que no implica sólo instalación de software, sino de hardware, y el mantenimiento del mismo. Ha habido que instalar los nodos en el rack, hacer la instalación física de los elementos de red, cableado, etc.

9.2 TRABAJOS FUTUROS

En cuanto a las posibles ampliaciones, se pueden realizar en varios aspectos:

- Hardware: Es posible, en un futuro, mejorar la infraestructura de comunicaciones, añadiendo por ejemplo redes Myrinet o Infiniband al cluster, lo que mejoraría enormemente la tasa de transferencia de datos. Evidentemente, también es posible añadir nuevos nodos al cluster, pero se discutirá sobre esto más adelante.

- Software: Puesto que el cluster es, en realidad, una simple plataforma sobre la que ejecutar aplicaciones, las ampliaciones en el campo del software son innumerables.

Ya se están realizando dentro del Grupo de Arquitectura y Tecnología de Computadores varios proyectos en base a él, como sistemas de ficheros distribuidos y paralelos (PVFS) o investigaciones sobre Grid.

Además, se pueden buscar nuevos algoritmos de planificación, para sustituir o complementar al actual Sun Grid Engine o Condor. De ésta forma se ganaría rendimiento y un mejor aprovechamiento de los recursos.

Son, como se puede ver, innumerables las aplicaciones que se pueden desarrollar sobre la base creada. Centrándonos en las mejoras sobre el proyecto realizado, una de las que se ha pensando en llevar a cabo es la siguiente:

Puesto que el departamento dispone de varias salas de PCs para los alumnos, se podría usar la potencia que éstos ordenadores desperdician cuando no están en uso, por ejemplo, por la noche o en fines de semana. De esta forma, instalando una versión del software que se ha usado en el proyecto (La distribución Rocks) se podría permitir que los PCs arrancaran con éste SO, y de ésta forma pasaran a formar parte del cluster.

Así, además, se podría contar con un pequeño Grid, o "cluster de clusters". Es decir, varios clusters que, mediante un planificador común basado en Globus, permitieran usarlo como uno sólo, enviando trabajos indistintamente a uno u otro.

10 PRESUPUESTO

Para la elaboración del presupuesto del proyecto se han tenido en cuenta precios del equipamiento a la fecha de presentación del mismo, Q1 2010, así como buscar equipos actuales equivalentes a los que se implantaron durante la instalación del proyecto en 2006/2007.

Los trabajos de instalación y configuración se han dividido en la parte de instalación física, que incluye:

- Montaje del rack.
- Ensamblado de servidores en el rack.
- Montaje del *switch* en el *rack*.
- Levantado del suelo técnico para preparar las conexiones eléctricas y de red adecuadas.
- Instalación y etiquetado del cableado de red.
- Instalación y etiquetado del cableado de alimentación eléctrica.

Del mismo modo, la parte de configuración software incluye los siguientes elementos:

- Pruebas básicas (encendido, conectividad entre servidores, pruebas de estabilidad).
- Instalación del software de gestión del cluster.
- Pruebas de funcionamiento y rendimiento.
- Tareas de documentación.

Componente	Cantidad	Coste/Unidad	Precio
Chasis para rack de 24U Dell PowerEdge 2420	1	1.930 €	1.930 €
Servidor Dell PowerEdge R410	10	1.235 €	12.350 €
Switch Ethernet Dell PowerConnect 2816	1	249 €	249 €
Instalación y configuración hardware	16	35 €	560 €
Instalación y configuración software	40	35 €	1.400 €
Cableado	1	100€	100€
		Base	**16.589 €**
		IVA 16%	**2.654,24 €**
		Total	**19.243,24 €**

APÉNDICE A TRABAJO PREVIO A LA INSTALACIÓN DE ROCKS

A.1 INSTALACIÓN Y PRUEBAS INDIVIDUALES DE CADA NODO

Para comprobar el correcto funcionamiento, tanto en estado normal como bajo carga extrema, se le realizó una instalación a cada nodo, utilizando una distribución de Linux .

La forma de realizarlo fue instalando un nodo, y clonando posteriormente esa instalación al resto de nodos, usando la utilidad dd, para simplemente copiar la imagen del disco duro a los otros. Éste es un proceso lento pero necesario para comprobar inicialmente que cada nodo será capaz de soportar sin problemas la carga a la que previsiblemente será sometido una vez forme parte del cluster.

Como ya se ha comentado, la instalación en cada nodo se realizo clonando los discos duros por completo, por lo que se tardó bastante tiempo, aunque el proceso era exponencial (primero utilizando un sólo nodo, luego de dos en dos, y finalmente de cuatro en cuatro). Una vez instalados los nodos, se les sometió a dos pruebas utilizando la utilidad cpuburn:

- La primera prueba, consistió en ejecutar dos veces seguidas la utilidad cpuburn durante 30 minutos, para comprobar la estabilidad a medio plazo con alta carga.
- La segunda prueba, con el objetivo de comprobar al 100% los nodos consistía en ejecutar cuatro veces seguidas cpuburn, con un tiempo de ejecución de cada una de 60minutos. El objetivo es comprobar si el calentamiento o la carga excesiva podría afectar a los nodos.

Todos los nodos superaron satisfactoriamente las dos pruebas, por lo que se pueden considerar completamente funcionales y aptos para pasar a formar parte del cluster en producción.

Igualmente, éstas pruebas se hicieron con el *frontend*, para comprobar su estabilidad, aunque en principio la carga que soportará no será tan alta como la de los nodos de cómputo, puesto que su función es sólo planificar y enviar los trabajos a cada nodo, así como hacer disponible a los investigadores la interfaz web que muestra el estado general del cluster.

A.2 PROBLEMAS CON LA CLONACIÓN DE LOS DISCOS

Como paso previo a la instalación de Rocks, se intento utilizar SuSe, para instalarla en cada nodo y gestionar la ejecución de los trabajos a través del *frontend* usando MPI. Se realizó la instalación en uno de los nodos sin problemas, y luego, igual que en las pruebas anteriores, se intentó clonar esa instalación a los otros nodos.

El problema surgió al descubrir que SuSe enlaza la configuración de la tarjeta de red a una MAC específica, por lo que al clonar la instalación e intentar arrancar el nodo, daba problemas y no se podía usar.

Para solventar este problema se intentaron utilizar algunos métodos definidos en el Cluster how-to (), pero no se consiguió . También se recurrió a la lista de correo de SuSe sin éxito en la respuesta, por lo que se siguió por otro camino en la instalación, utilizando la distribución Rocks para la instalación y configuración del cluster.

APÉNDICE B CONFIGURACIÓN DEL TEST HIGH-PERFORMANCE LINPACK (HPL)

El test HPL es uno de los más utilizados para medir el rendimiento objetivo de instalaciones cluster. Su funcionamiento es muy sencillo pero necesita ser parametrizado adecuadamente según las condiciones del cluster.

El fichero de configuración HPL.dat, tiene normalmente el siguiente contenido:

```
HPL.out output _le name (if any)
6 device out (6=stdout,7=stderr,_le)
1 # of problems sizes (N)
1000 Ns
1 # of NBs
64 NBs
1 # of process grids (P x Q)
1 Ps
2 Qs
16.0 threshold
3 # of panel fact
0 1 2 PFACTs (0=left, 1=Crout, 2=Right)
1 # of recursive stopping criterium
8 NBMINs (>= 1)
1 # of panels in recursion
2 NDIVs
1 # of recursive panel fact.
2 RFACTs (0=left, 1=Crout, 2=Right)
1 # of broadcast
1 BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1 # of lookahead depth
```

```
1 DEPTHs (>=0)
2 SWAP (0=bin-exch,1=long,2=mix)
80 swapping threshold
0 L1 in (0=transposed,1=no-transposed) form
0 U in (0=transposed,1=no-transposed) form
1 Equilibration (0=no,1=yes)
8 memory alignment in double (>0)
```

Los valores más importantes a la hora de adaptar el test a una instalación particular son:

- N, o el tamaño del trabajo que realizaremos. Hay que darle un valor de acuerdo al número de nodos que usamos y la cantidad de memoria que tengamos disponible en total. Una forma sencilla de calcular el valor es mediante la fórmula: .
- Ps y Qs indican el número de _las y columnas que tendrá el problema. Para obtener los mismos resultados hay que intentar mantener los dos lo más cerca posible (matriz cuadrada). PxQ indica el número total de procesadores (no nodos) que se usarán en el problema.
- NBs indica el tamaño de los bloques. Las pruebas realizadas indican que 160 es el óptimo para una red GbE.

En todas las pruebas realizadas, la mejor marca obtenida en el test es:

- 3.168e+01 Gflops con los valores siguientes: NBs = 160, P = 7, Q =

3, N=42279

APÉNDICE C VIRTUALIZANDO ROCKS

C.1 VIRTUALIZACIÓN

Si se acude a la Real Academia de la Lengua, virtual es aquello "Que tiene virtud para producir un efecto, aunque no lo produce de presente, frecuentemente en oposición a efectivo o real.". Por lo tanto, se refiere a algo que no existe, algo que se simula. En informática, virtualización es un término amplio que se refiere a la abstracción de los recursos de una computadora.

El tema en común de todas las tecnologías de virtualización es la de ocultar los detalles técnicos a través de la encapsulación. La virtualización crea un interfaz externo que esconde una implementación subyacente mediante la combinación de recursos en locaciones físicas diferentes, o mediante la simplificación del sistema de control.

Se explorarán en este capítulo las posibilidades que ofrecen las técnicas de virtualización en general para las Tecnologías de la Información y, más específicamente, la manera de aplicarlas al desarrollo de este proyecto.

C.1.1 INTRODUCCIÓN

En el campo de la informática se entiende por Virtualización la posibilidad de tener varios sistemas lógicos dentro de un mismo sistema físico. De esta manera, por ejemplo, una sola máquina servidora se puede utilizar para ejecutar varios servicios de manera independiente, como si se trataran de máquinas físicamente diferentes (en algunos casos, como veremos más adelante, la virtualización puede ser total o parcial).

Dentro de la virtualización hay otra alternativa, que es muy usada aunque en campos distintos al que plantea este proyecto. Se trata de la Emulación, muy usada en el desarrollo de aplicaciones para CPUs que aún no se producen en serie, o para el diseño de sistemas operativos para arquitecturas distintas al entorno de diseño y programación.

En los últimos años, y en gran parte gracias al exponencial incremento de potencia hardware, las tecnologías de virtualización han experimentado un incremento impresionante y, por fin, han empezado a poderse explotar en los departamentos de Tecnologías de la Información (TI).

Es por eso, que en este apartado se pretende proporcionar una introducción a las diferentes formas de virtualizar sistemas que existen en la actualidad, razonar el por qué de la virtualización y, finalmente, aplicar esos conceptos a la instalación de un Cluster Virtual usando Rocks como base. Se explicaran las ventajas y, por supuesto, los inconvenientes que conlleva.

C.1.2 ¿POR QUÉ VIRTUALIZAR?

La respuesta a esta pregunta no es, en absoluto, trivial. Cualquier cambio en la arquitectura de TI implica gran cantidad de efectos secundarios. Por eso, los cambios que se introduzcan tienen que estar controlados, para evitar problemas posteriores.

¿Cuáles son las metas que se buscan al virtualizar sistemas? ¿Por qué incrementar (aparentemente) la complejidad de la arquitectura de TI mediante la virtualización? ¿Qué beneficios aportará a la organización? Son dudas que, cambiando el término virtualización por cualquier otra nueva tecnología, se hará cualquier encargado de TI de una empresa.

La respuesta a la mayoría de esas preguntas, volviendo al caso de la virtualización, se sencilla: Se trata de simplificar y aprovechar de manera más eficientes la arquitectura

de sistemas. Además, ésta tecnología es una valiosa ayuda a la hora de implantar nuevos servicios, pues permite hacer pruebas de una manera muy sencilla sin necesidad de adquirir equipos nuevos ni modificar la arquitectura física existente.

En el párrafo anterior, se han introducido los dos usos más comunes que se da actualmente a la virtualización de sistemas en entornos empresariales:

- Optimizar recursos, mediante la consolidación de sistemas
- Simulación de arquitecturas complejas

Los siguientes apartados, se extienden sobre cada una de las alternativas de uso que se describieron anteriormente, pero también es importante tener en cuenta que la virtualización no sólo se utiliza en entornos de servidor. También comienza a usarse ampliamente en entornos de escritorio, para desarrollar y probar aplicaciones antes de trasladarlas a entornos reales.

Siguiendo con el comentario del párrafo anterior es posible, por ejemplo, desarrollar bajo un entorno basado en Windows e instalar una máquina virtual con Linux, Apache, PHP y MySql para probar aplicaciones sin necesidad de subirlas a un entorno real. Esto es lo que haremos con Rocks: Instalarlo en un entorno virtual, para permitir que los desarrolladores de aplicaciones distribuidas puedan comprobar el funcionamiento de sus aplicaciones antes de ejecutarlas en el cluster, sobre el que seguramente tengan unos límites de tiempo de uso y carga, haciendo así más eficiente su uso.

OPTIMIZACIÓN DE RECURSOS – CONSOLIDACIÓN

La virtualización aplicada a la optimización de recursos es una de las modalidades más usadas en la actualidad gracias al considerable ahorro que supone tanto en costes monetarios como en costes de mantenimiento y horas de trabajo.

La idea detrás de la virtualización cuando se aplica para la consolidación de sistemas es eliminar varias máquinas que tienen muy poca carga habitualmente para, una vez virtualizadas, ejecutarlas dentro de un solo sistema físico con el consiguiente ahorro de espacio y energía consumida. En la imagen inferior se representa esto gráficamente. Supongamos el siguiente caso:

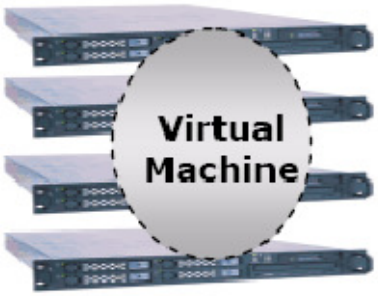

ILUSTRACIÓN 20 CONSOLIDACIÓN USANDO MÁQUINAS VIRTUALES

Una empresa cuenta con varios servicios. Por ejemplo, dando servicio FTP, Web y de resolución de dominios (DNS). Cada uno de ellos tiene poca carga, pero por motivos de seguridad se instalaron en máquinas separadas. Sería posible instalar una sola máquina, ejecutando herramientas de virtualización (bien sean hardware o software), y manteniendo los diferentes servicios aislados de forma lógica, como si siguieran ejecutándose en diferentes máquinas.

Otro caso que puede darse, y donde la virtualización es de gran ayuda son los sistemas comúnmente llamados legacy. Se entiende por legacy aplicaciones de cierta antigüedad, que funcionan sobre hardware antiguo, y que probablemente no puedan hacerse funcionar en una máquina actual, bien porque el sistema operativo no tiene soporte o la aplicación es dependiente del hardware.

En ese caso, si se quisiera actualizar el hardware, se puede hacer que el software legacy se ejecute en su propia máquina virtual, con los parámetros correctos para que todo el entorno sea similar al del antiguo hardware en el que se ejecutaba, pero con todas las

ventajas del nuevo equipamiento más las que la ejecución dentro de una máquina virtual puede aportar.

La consolidación y virtualización de la arquitectura hardware, repercute en una mejor utilización de los recursos disponibles, al evitar tener máquinas dedicadas que funcionan continuamente por debajo de un 10% de carga consumiendo energía y espacio.

Una gran ventaja de la virtualización es la completa independencia de la capa hardware de cara a ejecutar aplicaciones. Por ejemplo, en caso de que un servidor de virtualización falle, simplemente se pueden mover las máquinas virtuales que se ejecutaban en él hacia otro servidor que, independientemente de la plataforma hardware sobre la que se ejecute el entorno de virtualización. La migración en caliente de máquinas virtuales es una de las características más utilizas para sistemas de vital importancia, en los que no se puede tolerar una caída.

De esta forma, se pueden construir arquitecturas realmente complejas dentro de una misma máquina física, con todas las ventajas que ello conlleva. Además y, aunque esto se explica en mayor detalle en el siguiente apartado, la virtualización permite simular entornos complejos antes de pasarlos a producción, permitiendo detectar los posibles problemas sin poner en riesgo el correcto funcionamiento de otros sistemas ya en producción.

SIMULACIÓN DE ARQUITECTURAS COMPLEJAS

La simulación de arquitecturas complejas es otro de los puntos fuertes de la virtualización. Al igual que en el punto anterior (dedicado a la consolidación de servidores) el objetivo es conseguir tener varios sistemas lógicos ejecutándose sobre un único sistema físico. Pero, ahora, el propósito es muy diferente. No se busca la optimización, ni obtener mejores rendimientos o reducir espacio: Se pretende crear un entorno de pruebas para aplicaciones o arquitecturas de red, por ejemplo.

Imaginemos la siguiente situación, estrechamente relacionada con la temática de este proyecto: Un programador de arquitecturas paralelas está diseñando un programa que, una vez finalizado, se ejecutará en un gran cluster de ordenadores. ¿Cómo realiza las pruebas? ¿Cómo se asegura de que, una vez ejecutándose en el cluster, obtendrá los resultados que buscaba? Los clusters son muy caros, y su utilización suele estar restringida a pequeñas fracciones de tiempo para cada aplicación.

Al programador del ejemplo que acabamos de describir no le interesa el rendimiento, sino ser capaz de comprobar el funcionamiento lógico de su aplicación: Distribución de trabajo entre nodos, comunicación entre procesos, etc. Sería de una gran ayuda si de alguna manera pudiera "simular" un cluster en su propio ordenador, en el que está desarrollando la aplicación. Ese es el punto en el que la virtualización puede ayudar, preparando un escenario como el mostrado en la imagen inferior, donde en un solo PC se instalan varias máquinas virtuales, ejecutándose simultáneamente.

ILUSTRACIÓN 21 PRUEBAS USANDO MÁQUINAS VIRTUALES

Mediante la tecnología de virtualización, se podría instalar un pequeño cluster de 2 o 4 nodos en la misma máquina que se utiliza para desarrollar o, por ejemplo, tener una máquina dedicada a simular el cluster. De esta forma, se podría comprobar si el protocolo de comunicaciones es correcto, si la aplicación distribuye la carga de manera adecuada, o si el software tiene algún fallo que pueda bloquear el cluster o sobrecargarlo.

Obviamente, el rendimiento no será bueno, de hecho, será muchísimo más bajo que si se ejecutara el programa distribuido en una sola máquina pues la capa de virtualización

consume recursos. Pero aún así, resulta de gran ayuda poder probar el diseño del software antes de ejecutarlo sobre el cluster.

Y, precisamente, éste es el objetivo que se persigue en este apartado del proyecto: Instalar un cluster virtual que permita a los desarrolladores de programas distribuidos hacer pruebas de los mismos en su propia máquina.

C.1.3 DIFERENTES TIPOS DE VIRTUALIZACIÓN

En el campo de la informática, la virtualización es algo muy antiguo, y que se puede abordar desde distintos puntos de vista. La principal característica que tienen en común las diferentes tecnologías de virtualización es la de ocultar los detalles técnicos que subyacen. La virtualización crea una capa externa, un interfaz, que esconde los detalles de la implementación que hay por debajo de ella.

Atendiendo al objetivo de la virtualización, se pueden describir dos grandes grupos de tecnologías:

- Virtualización de plataforma, que trata de máquinas virtuales, y es en la que más se profundizará en este proyecto.
- Virtualización de recursos, que simula recursos.

Sobre ésta última no se prestará mucha más atención, pues no es la que interesa para el desarrollo de éste proyecto. Aún así, resulta interesante comentar que es la tecnología que está detrás de términos tan usados como RAID, las redes de almacenamiento (SANs) o redes privadas virtuales (VPN). También resulta interesante ver

que, desde este punto de vista, un cluster también se puede considerar una virtualización de recursos, pues realiza una "agregación" de muchos sistemas, para obtener un solo sistema de mayor potencia, y con una interfaz que "oculta" todos los detalles que hay detrás, tales como comunicaciones, distribución de trabajos, o la actualización de los nodos.

En cuanto al primer punto, la Virtualización de plataforma, es la técnica que realmente interesa para este proyecto, y de la que se viene hablando durante todo éste capítulo. El objetivo es: En una plataforma hardware se instala un software "host", también conocido como anfitrión, monitor o programa de control, que simula un entorno computacional, y permite instalar "guests" o sistemas huésped/invitados.

Hay muchas maneras de enfocar la virtualización de plataforma, aunque se pueden reducir a dos, que son las más comúnmente utilizados, y listadas en orden de mayor a menor abstracción:

- Paravirtualización
- Virtualización completa
- Emulación

Es posible añadir también un tercer tipo, la Virtualización ayudada por Hardware pero, en esencia, se trata de virtualización completa con ayudas por parte de la CPU que incluye extensiones especialmente diseñadas para ayudar en la tarea de virtualización. No se incluye como un apartado completo puesto que la Virtualización completa también es capaz de utilizar esas extensiones de la CPU para aumentar su rendimiento y, por tanto, se entra en detalles sobre ellas en el apartado de Virtualización Completa. Se podría decir que la Virtualización Completa y la Virtualización ayudada por hardware se solapan en cierto modo una con otra.

En los siguientes apartados se detalla cada una de las modalidades de virtualización de plataformas enumeradas en los párrafos anteriores, entrando en mayor detalle de su funcionamiento y de los diferentes productos que las implementan.

C.1.3.1 VIRTUALIZACIÓN COMPLETA

La virtualización completa o virtualización nativa es una técnica mediante la cual se utiliza una máquina virtual para simular un entorno hardware completo suficiente, de modo que se pueda instalar un sistema operativo diseñado para el mismo tipo de CPU sin ninguna modificación, y se ejecute de manera aislada en la máquina virtual.

El término "entorno hardware completo suficiente" se refiere a la cantidad de recursos del sistema que la capa de virtualización simula. En este caso, se recrea un PC completo basado en la misma CPU que el anfitrión. Esto quiere decir que se simula la BIOS y todo el hardware subyacente, creando un nuevo PC dentro del anfitrión y, a todos los efectos, eso es lo que el sistema operativo que se virtualiza ve: Un PC a su entera disposición.

En el siguiente gráfico se puede ver la arquitectura de un sistema que utiliza virtualización completa:

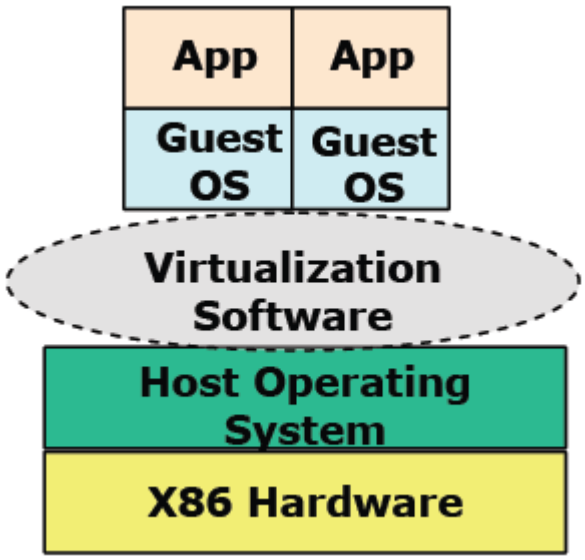

ILUSTRACIÓN 22 ARQUITECTURA CON VIRTUALIZACIÓN COMPLETA

Como se observa en el gráfico, los dos primeros niveles son los tradicionales de cualquier sistema:

- Un sistema hardware, en este caso basado en x86.
- Un sistema operativo, diseñado para funcionar sobre la plataforma hardware específica (de nuevo, x86 en este caso).

Todo hasta aquí resulta tradicional, pero es en la siguiente capa, denominada Capa de Virtualización donde se ven las diferencias. Esa capa es la que, como se vio anteriormente, permite simular un entorno hardware completo sobre el que se podrán instalar otros sistemas operativos. Sobre la Capa de Virtualización se ejecutan los Sistemas Operativos Invitados u anfitriones, y en ellos, las aplicaciones que se deseen.

Ésta técnica puede hacer uso de las capacidades de virtualización incluidas en los últimos microprocesadores de Intel y AMD. En el caso de Intel, las extensiones Intel VT; en el caso de AMD, las AMD-V. En determinadas circunstancias, estas extensiones permiten obtener un rendimiento cercano a la ejecución nativa del sistema operativo.

Antes de que existieran las extensiones Intel VT y AMD-V la arquitectura x86 no cumplía algunos de los requisitos básicos para ser virtualizada, por lo que hacía bastante complicado implementar la capa de abstracción sobre ella.

La Virtualización completa es relativamente reciente (la virtualización existe desde los años 70 y, como ejemplo, VMware se fundó en 1998). Se basa en técnicas avanzadas para detectar la ejecución de instrucciones peligrosas y reemplazarlas con llamadas a la capa de virtualización. Esto último es lo que se conoce como traducción binaria.

En computación se entiende por traducción binaria la emulación de un juego de instrucciones a través de otro, utilizando la traducción de código. Es decir: Una secuencia de instrucciones se traducen del juego de instrucciones origen al juego de instrucciones destino.

La traducción se puede hacer de manera estática o dinámica. El problema de hacerlo de forma estática es que hay partes de código que no se pueden traducir a priori, sin ejecutarlo como, por ejemplo, los saltos condicionales, que sólo se conocen en tiempo de ejecución. En cambio, la traducción dinámica, trata de traducir porciones de código pequeñas según las va descubriendo al ejecutar el código. Las traduce, y las almacena en un caché. De este modo, el código sólo se traduce en el momento que el traductor lo encuentra, permitiendo traducir correctamente las problemáticas ramas condicionales.

Es de este modo, mediante el uso de traducción dinámica, como el software de virtualización detecta las instrucciones que antes se denominaron peligrosas y las traduce a llamadas o traps a la capa de virtualización.

Entre las ventajas de la virtualización completa, la principal es la sencillez y la gran cantidad de posibilidades de uso que ofrece. Al no tener que modificar los sistemas operativos anfitriones se facilita la tarea de los administradores de sistemas, y se permite instalar cualquier sistema operativo, aunque no se tenga el código a disponible para modificarlo.

En cambio, como hay que simular gran cantidad de hardware, y además, hacerlo para cada máquina virtual que se ejecuta en el sistema, el rendimiento se reduce drásticamente. Pese a esto, las extensiones de virtualización por hardware han conseguido una notable mejora en el desempeño de las máquinas virtuales.

Otro problema que se puede achacar a la virtualización completa es común a gran parte de los sistemas de máquinas virtuales: Puede requerir la instalación de controladores específicos para conseguir un rendimiento alto, especialmente en operaciones de entrada/salida intensivas. Por ejemplo, se pueden necesitar controladores para las interfaces de red o discos duros, y esto puede conllevar problemas con algunos contratos de soporte técnico, que impiden la instalación de controladores no oficiales.

Las últimas versiones de los principales software de virtualización disponibles son capaces de aprovechar las extensiones hardware que proveen las CPUs de última generación.

Una vez descrita la tecnología de virtualización completa, tal y como se ha hecho en este apartado, se deduce que se trata de la solución óptima en los siguientes casos de uso:

- Entornos de desarrollo, donde se pueden instalar varias máquinas virtuales sobre el mismo PC usado para desarrollar aplicaciones. De esta forma, se puede probar el funcionamiento de arquitecturas cliente-servidor o sistemas distribuidos complejos sin necesidad de invertir en nuevo equipamiento, y con la comodidad que esto conlleva para el equipo de desarrollo.

- Cuando hay necesidad de utilizar sistemas operativos propietarios que no se pueden modificar, y el hardware no dispone de instrucciones específicas para virtualizar éste tipo de sistemas operativos.

- Entornos legacy, donde se debe contar con un hardware específico para que algunos sistemas operativos puedan ejecutarse.

C.1.3.2 PARAVIRTUALIZACIÓN

La Paravirtualización es una técnica de virtualización con un planteamiento totalmente distinto al visto anteriormente con la virtualización completa. En este caso no es necesario simular un hardware, si no proporcionar una interfaz de programación (API) que el sistema operativo debe usar para llamar al "hypervisor" o VMM (Virtual Machine Monitor).

En el párrafo anterior se da una descripción estrictamente técnica de la paravirtualización, pero también se introduce el concepto de hypervisor o monitor que, junto con su API, tienen consecuencias importantes en la manera de plantear la virtualización del sistema operativo.

En líneas generales, un hypervisor (a partir de ahora se utilizarán las palabras hypervisor o monitor indistintamente) es una tecnología que permite utilizar, al mismo tiempo, diferentes sistemas operativos en una misma máquina. Es decir, el concepto de hypervisor se aplica tanto en la virtualización completa vista anteriormente como en la paravirtualización. Pero el concepto es especialmente importante en ésta última, puesto que es el hypervisor quien proporciona la API que el sistema operativo tiene que usar para realizar llamadas al sistema.

En el caso de la paravirtualización se trata de un hypervisor de Tipo 1, es decir, que se ejecuta directamente sobre el hardware, en oposición al hypervisor de Tipo 2, que necesita ejecutarse sobre un sistema operativo.

Teniendo en cuenta lo dicho en los párrafos anteriores, la consecuencia más inmediata es que para poder ejecutar un sistema paravirtualizado se necesita modificar el código del sistema operativo, sustituyendo algunos accesos directos al hardware por llamadas al hypervisor. Esto supone el principal problema de la Paravirtualización: Es necesario modificar el sistema operativo anfitrión para hacerlo funcionar coordinado con el hypervisor.

En el siguiente gráfico podemos ver el esquema que sigue un sistema paravirtualizado, utilizando una de las aplicaciones más extendidas actualmente, y sobre la que se hablará más adelante: Xen.

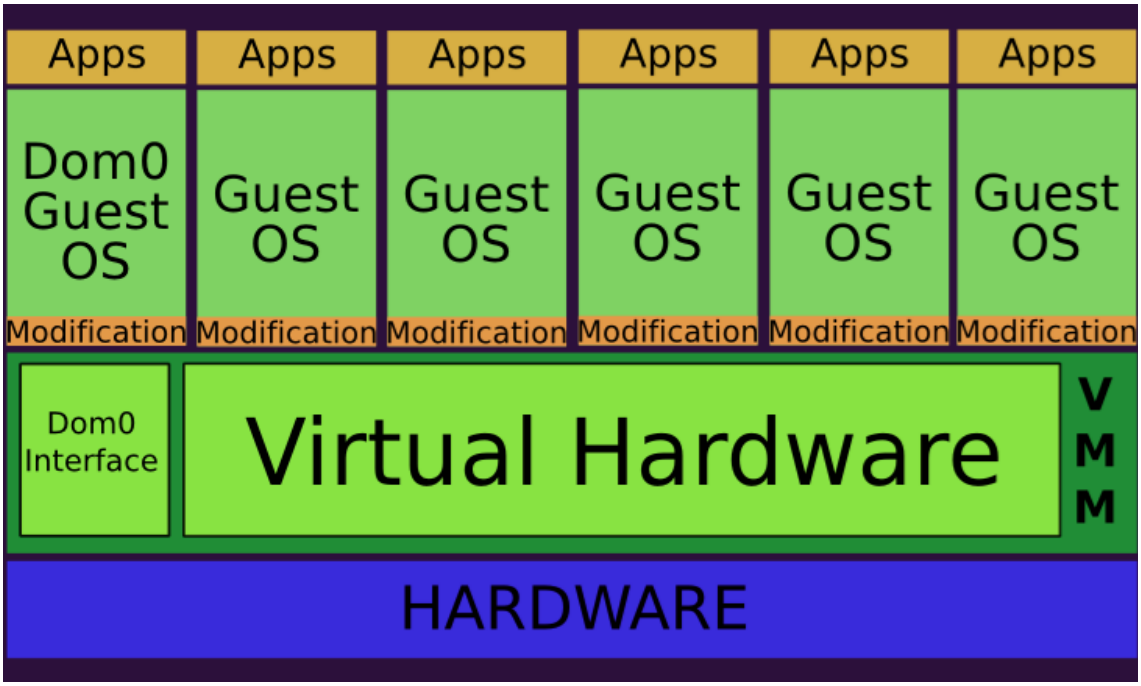

ILUSTRACIÓN 23 ARQUITECTURA XEN

Al mirar el gráfico, la primera diferencia que se puede observar es que no hay un sistema operativo entre el hardware y el hypervisor (VMM o Virtual Machina Monitor en este caso). Esto es, el hypervisor es en si mismo un sistema operativo, que ofrece "servicios" a los clientes, o sistemas anfitriones.

Para entrar en mayor nivel de detalle de la arquitectura paravirtualizada es necesario conocer el concepto de Anillo (o Ring, que es el término más usado) que

implementan algunas CPUs. En informática, los dominios de protección jerárquicos, o anillos de protección, son un mecanismo para proteger los datos y la funcionalidad de la CPU frente a fallos o comportamientos peligrosos.

Los sistemas operativos modernos ofrecen diferentes niveles de acceso a los recursos. Un anillo de protección consiste en uno o más niveles jerárquicos o capas de privilegios dentro de la arquitectura del sistema. Normalmente, el hardware de la CPU proporciona esta funcionalidad mediante diferentes modos de ejecución a nivel de firmware.

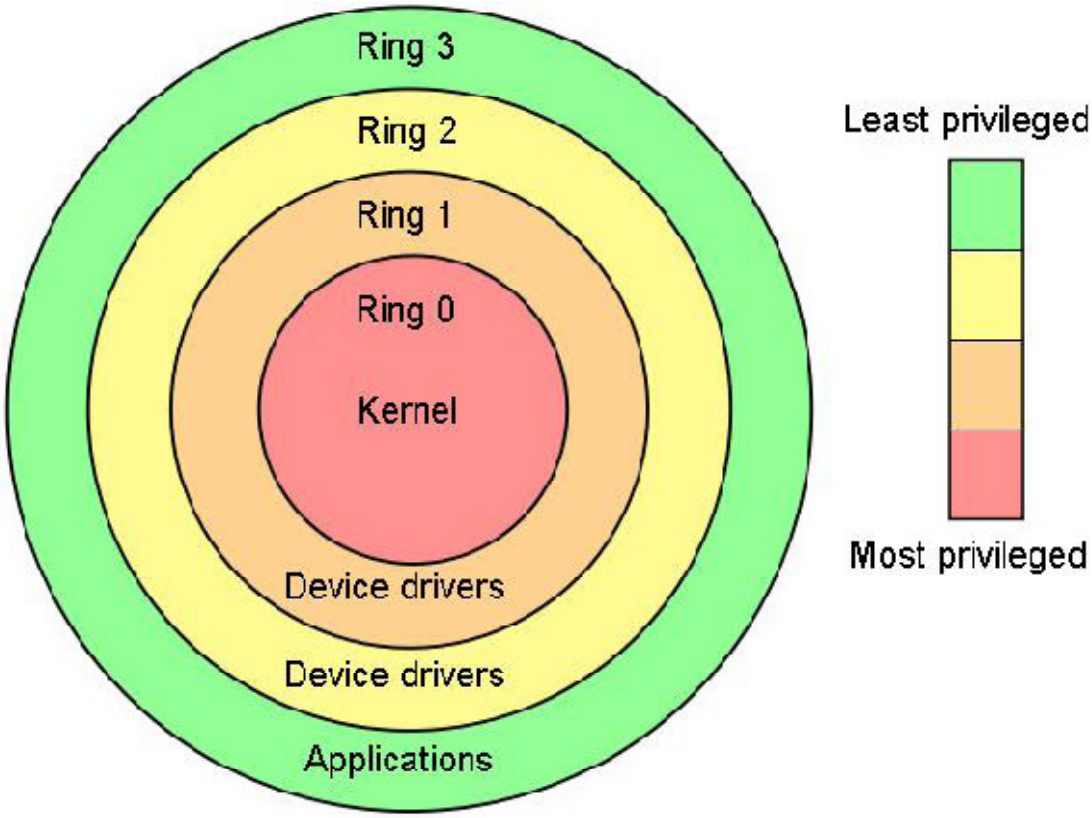

ILUSTRACIÓN 24 ANILLOS DE PRIVILEGIOS

Los anillos están ordenados jerárquicamente, desde el más privilegiado (normalmente el nivel 0) hasta los menos privilegiados, con el mayor numero de anillo. En los sistemas basados en CPUs x86 hay cuatro anillos: El ring 0 es donde se ejecuta el Sistema Operativo (con el mayor nivel de privilegio) y el nivel 3 donde se ejecutan las

aplicaciones. Los anillos 1 y 2 no se utilizan salvo raras excepciones (como el sistema OS/2 o el sistema de virtualización Xen como se verá más adelante.).

La idea detrás de la paravirtualización es aprovechar esta característica, los anillos jerárquicos, para ejecutar los sistemas operativos invitados en los niveles que habitualmente no se usan (1 y 2) y realizar llamadas al Monitor de Máquinas Virtuales, que se ejecuta en Ring 0, cuando sea necesario acceder al hardware.

La principal ventaja de los sistemas de virtualización que utilizan la técnica de la paravirtualización es el rendimiento, que es capaz de acercarse mucho al de un sistema no virtualizado. Otra ventaja es la relativa sencillez del Monitor de máquinas virtuales VMM. Obviamente, la desventaja más llamativa es la necesidad de modificar el sistema operativo para que llame al hypervisor cuando sea necesario.

En el capítulo dedicado a Xen se entrará más en profundidad en la arquitectura que éste sistema de virtualización utiliza.

C.1.3.3 EMULACIÓN

Un emulador es un software que duplica las funciones de un sistema utilizando otro diferente, de forma que el segundo sistema se comporta (y a todos los efectos, parece ser) el primer sistema. A diferencia de un simulador, que sólo trata de reproducir el comportamiento del programa, un emulador trata de modelar de forma precisa el dispositivo que se está emulando.

La mayoría de los emuladores solo emulan una determinada configuración arquitectura de hardware, y si el firmware (o sistema operativo, por ejemplo) también se requiere para emular cierto programa entonces ha de ser emulado también o cargado como un programa más dentro del emulador.

Aparte de la interpretación del lenguaje de la máquina emulada, es preciso emular el resto del equipo, como los dispositivos de entrada y salida, de forma virtual: si escribir en una región específica de la memoria debe influir en el contenido en pantalla, por ejemplo, esto también debe ser emulado.

Normalmente, un emulador se divide en módulos que corresponden de forma precisa a los subsistemas del equipo emulado. Lo más común, es que un emulador este compuesto por los siguientes módulos:

- Un emulador de CPU o un simulador de UCP (ambos términos son en la mayoría de los casos intercambiables).
- Un módulo para el subsistema de memoria.
- Varios emuladores para los dispositivos de entrada y salida.

Lo más común es que los buses no sean emulados, por razones de simplicidad y rendimiento, y para que los periférico virtuales se comuniquen directamente con la CPU y los subsistemas de memoria.

Evidentemente, todo el nivel de detalle al que llegan los simuladores habitualmente tiene su contrapartida: El rendimiento. Más adelante se hablará de uno de los simuladores más extendidos, BOCHS, pero basta decir que el rendimiento típico de éste es de -75 veces el rendimiento real. Es decir, la plataforma emulada es 75 veces más lenta que en la vida real, y todo esto en condiciones óptimas que raramente se alcanzan.

Pese a que un emulador podría bajar de nivel tanto como se quisiera (en teoría, hasta el nivel atómico), por las razones descritas en el párrafo anterior habitualmente se llega hasta el nivel hardware que se encuentra documentado en las especificaciones de la

plataforma, pero teniendo en cuenta que incluso los errores o bugs de la misma deben emularse para obtener un comportamiento completamente fidedigno.

Aún con los problemas de rendimiento que presentan, los emuladores son una ayuda de incalculable valor en el desarrollo de nuevas plataformas hardware o cuando se pretende probar aplicaciones en las que la velocidad no sea el objetivo principal de la prueba, si su comportamiento en la plataforma específica.

C.2 ALTERNATIVAS DE VIRTUALIZACIÓN

En este apartado se explorarán las diferentes aproximaciones a la virtualización de Rocks. Entre ellas, se encuentran productos comerciales y productos de código abierto, y que utilizan diferentes medios para conseguir el mismo objetivo: Virtualizar los recursos hardware para obtener un mejor aprovechamiento de los mismos, permitiendo alcanzar mayores cotas de uso de CPU y memoria.

Posteriormente, se detallará la instalación de la solución elegida y el proceso de instalación de Rocks sobre la plataforma de virtualización, puesto que se necesitan algunos ajustes para permitir un mejor aprovechamiento de los recursos.

C.2.1 VMWARE

VMware proporciona un entorno hardware completamente virtualizado al sistema operativo invitado. El software virtualiza el hardware para un adaptador de vídeo, tarjeta de red y disco duro. A su vez, provee mecanismos pass-through para puertos USB, serie y paralelo. De esta manera, las máquinas virtuales creadas con VMware son muy portables, por que cada host es idéntico de cara al sistema operativo huésped.

VMWare se presenta en varias versiones: WorkStation, Server y ESX. Todas las versiones ofrecen optimizaciones mayores que los emuladores (como BOCHS, discutido más adelante), como por ejemplo la simulación del set individual de instrucciones para cada CPU, o recompilación dinámica de bloques de código máquina la primera vez que son usadas, para almacenarlas y ejecutarlas más rápidamente la próxima vez que sean necesarias.

Éste último punto (la compilación dinámica de instrucciones) incrementa significativamente el rendimiento, pero también puede causar problemas a la hora de mover máquinas virtuales entre sistemas que utilicen un juego de instrucciones distinto, o entre hosts con un número distinto de CPUs.

VMware usa la CPU del sistema anfitrión (Host) para ejecutar el código directamente cuando es posible (por ejemplo, cuando la CPU del host se ejecuta en modo virtual 8086 en x86). Cuando éste tipo de operaciones no son posibles (por jeemplo, en código del kernel) VMware reescribe el código en tiempo real, utilizando la técnica de traducción binaria. Por eso, VMware es capaz de ejecutar un sistema invitado, en ocasiones, un 80% más rápido que un emulador como BOCHS. En general, VMware intruduce una sobre carga de entre un 3 y un 6% para aplicaciones que hacen uso intensivo de la CPU.

Además de todo esto, VMware ofrece características avanzadas, orientadas a los administradores de sistemas, como la posibilidad de mover una máquina virtual en ejecución, sin ningún tiempo de caída, de un host físico a otro. Ésta característica se conoce como VMotion.

Por supuesto, VMware ofrece servicios avanzados de backup de los sistemas huéspedes, con lo que la recuperación en caso de desastre se limita a mover las máquinas virtuales afectadas de un huésped a otro y arrancarlas de nuevo.

C.2.2 BOCHS

BOCHS es un emulador de código libre, capaz de emular una CPU con arquitectura Intel x86, dispositivos de Entrada/Salida así como la BIOS. Actualmente, Bochs puede ser compilado para emular un 386, 486 Pentium/Pentium II/Pentium III/ Pentium 4 o una CPU con arquitectura x86-64, incluyendo instrucciones adicionales como las MMX, SSEx y 3DNow!.

Además es capaz de correr una gran variedad de sistemas operativos dentro de su emulación , entre los cuales están Linux, DOS, Windows 95/98 and Windows NT/2000/XP y Windows Vista.

C.2.3 XEN

Xen es una máquina virtual de código abierto desarrollada por la Universidad de Cambridge.

La meta del diseño es poder ejecutar instancias de sistemas operativos con todas sus características, de forma completamente funcional en un equipo sencillo. Xen proporciona aislamiento seguro, control de recursos, garantías de calidad de servicio y migración de máquinas virtuales en caliente.

Los sistemas operativos pueden ser modificados explícitamente para correr Xen (aunque manteniendo la compatibilidad con aplicaciones de usuario). Esto permite a Xen alcanzar virtualización de alto rendimiento sin un soporte especial de hardware. Intel ha realizado diversas contribuciones a Xen que han permitido añadir soporte para sus extensiones de arquitectura VT-X Vanderpool.

Esta tecnología permite que sistemas operativos sin modificar actúen como hosts dentro de las máquinas virtuales de Xen, siempre y cuando el servidor físico soporte las extensiones VT de Intel o Pacifica de AMD.

En la arquitectura de Xen, el hypervisor se ejecuta en el Ring 0, mientras que los sistemas operativos invitados se ejecutan en nivel 1.

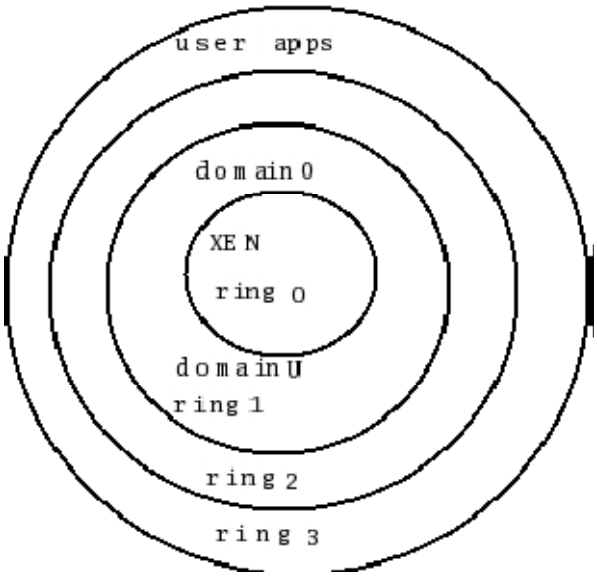

ILUSTRACIÓN 25 ARQUITECTURA DE ANILLOS XEN

C.2.4 KVM

Kernel-based Virtual Machine, o KVM, (en español Máquina virtual basada en el núcleo) es una solución para implementar virtualización completa con Linux sobre hardware x86. Está formada por un módulo del núcleo (con el nombre kvm.ko) y herramientas en el espacio de usuario, siendo en su totalidad software libre.

KVM permite ejecutar máquinas virtuales utilizando imágenes de disco que contienen sistemas operativos sin modificar. Cada máquina virtual tiene su propio hardware virtualizado: una tarjeta de red, discos duros, tarjeta gráfica, etc.

C.3 RENDIMIENTO DE LOS SISTEMAS DE VIRTUALIZACIÓN

Se muestran en esta sección los resultados de las pruebas de rendimiento sobre las dos plataformas de virtualización más usadas a nivel empresarial para virtualización y consolidación de servidores e, incluso, de equipos de escritorio para el usuario final.

Se han comparado éstos dos productos (VMware ESX y Xen) porque son los máximos representantes de los dos modos de virtualización por excelencia. Por una parte, VMware hace uso de la virtualización completa, que como ya se ha visto anteriormente, permite la ejecución de sistemas operativos sin modificar en un entorno virtualizado.

En cambio, Xen, hace uso de la paravirtualización, que permite eliminar el paso de la traducción binaria del juego de instrucciones pero, a cambio, requiere modificar el núcleo del sistema operativo. También es posible ejecutar sistemas operativos sin modificar a través de las instrucciones y hardware de virtualización presentes en las CPU de última generación.

La configuración del entorno de pruebas se basa en un Windows 2003 Server como máquina virtual, ejecutándose sobre VMware ESX y Xen respectivamente [8].

En la primera comparativa, que se muestra en la Ilustración 26 se puede ver el rendimiento ejecutando algoritmos de compresión. Como se puede comprobar, el rendimiento de ambos es muy similar estando en todos los casos por encima del 90% del equipo de referencia. La pérdida es mínima.

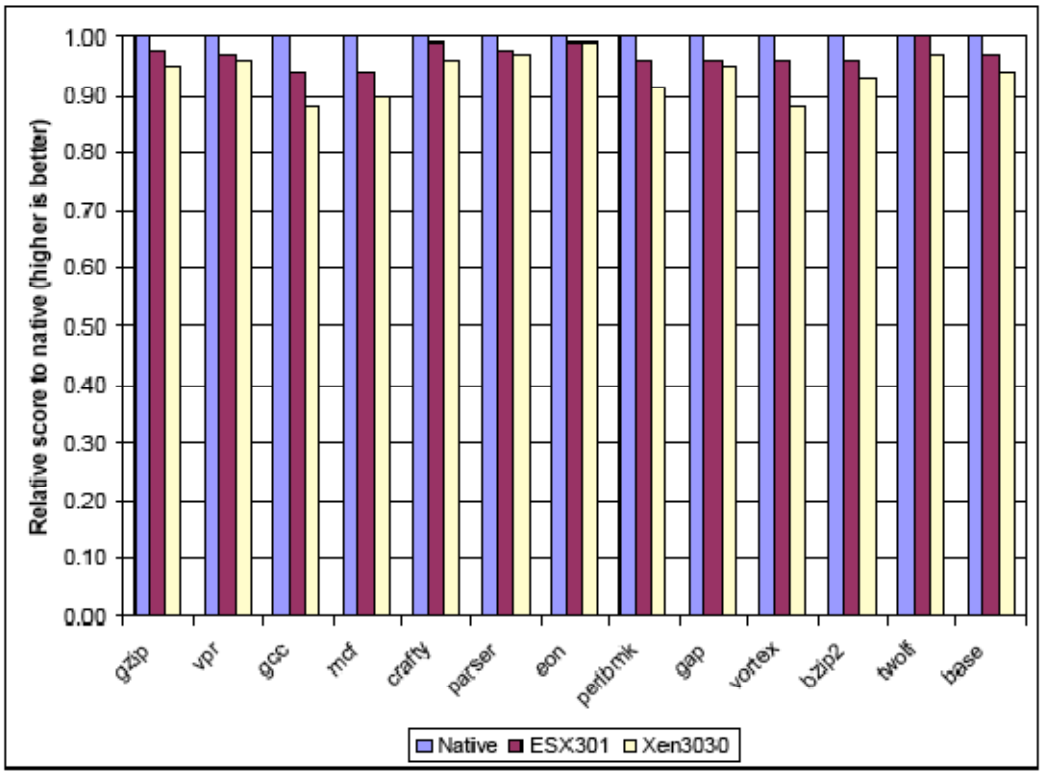

ILUSTRACIÓN 26 ALGORITMOS DE COMPRESIÓN

La Ilustración 27 muestra el rendimiento en lecturas y escrituras (con y sin caché) a memoria RAM. De nuevo, el rendimiento está en ambos casos muy cercano al del sistema real, y apenas de aprecia diferencia entre la virtualización completa y la paravirtualización.

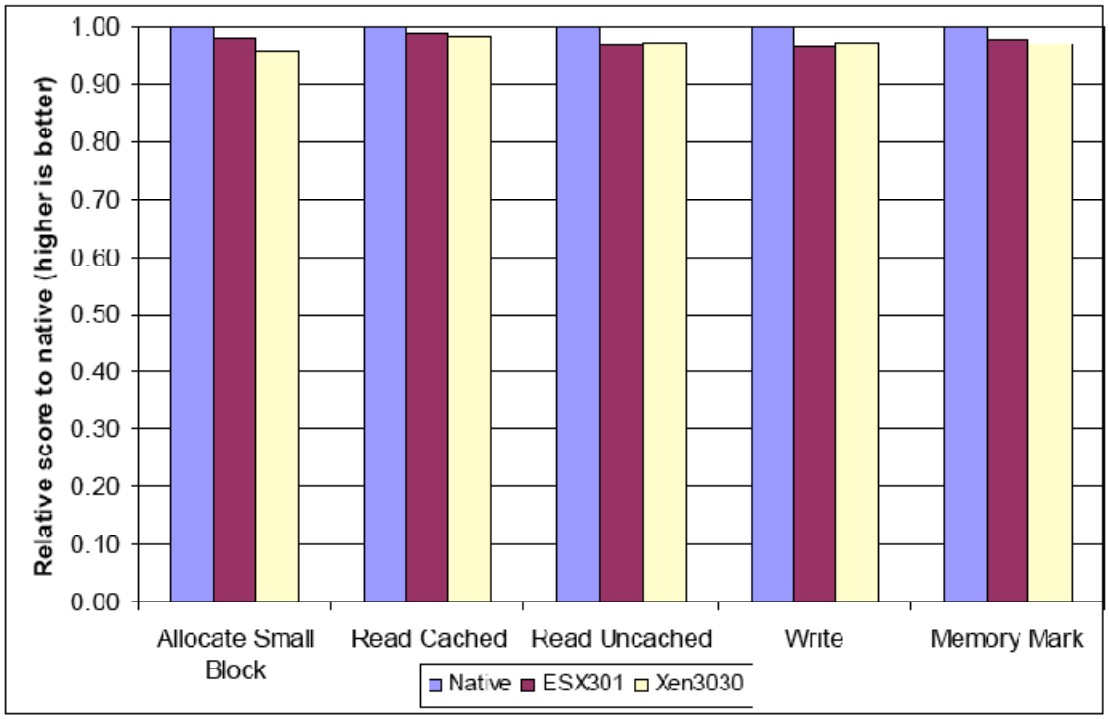

ILUSTRACIÓN 27 LECTURAS Y ESCRITURAS EN RAM

La siguiente prueba muestra, por primera vez en los test, claras diferencias entre ambos sistemas. Esto es debido a que entran en juego las operaciones de I/O, en las que Xen corriendo un kernel no modificado sale claramente perjudicado. El test está realizado con Netperf usando una controladora Ethernet dedicada y se muestra la tasa real de Mb/s obtenida (Ilustración 28):

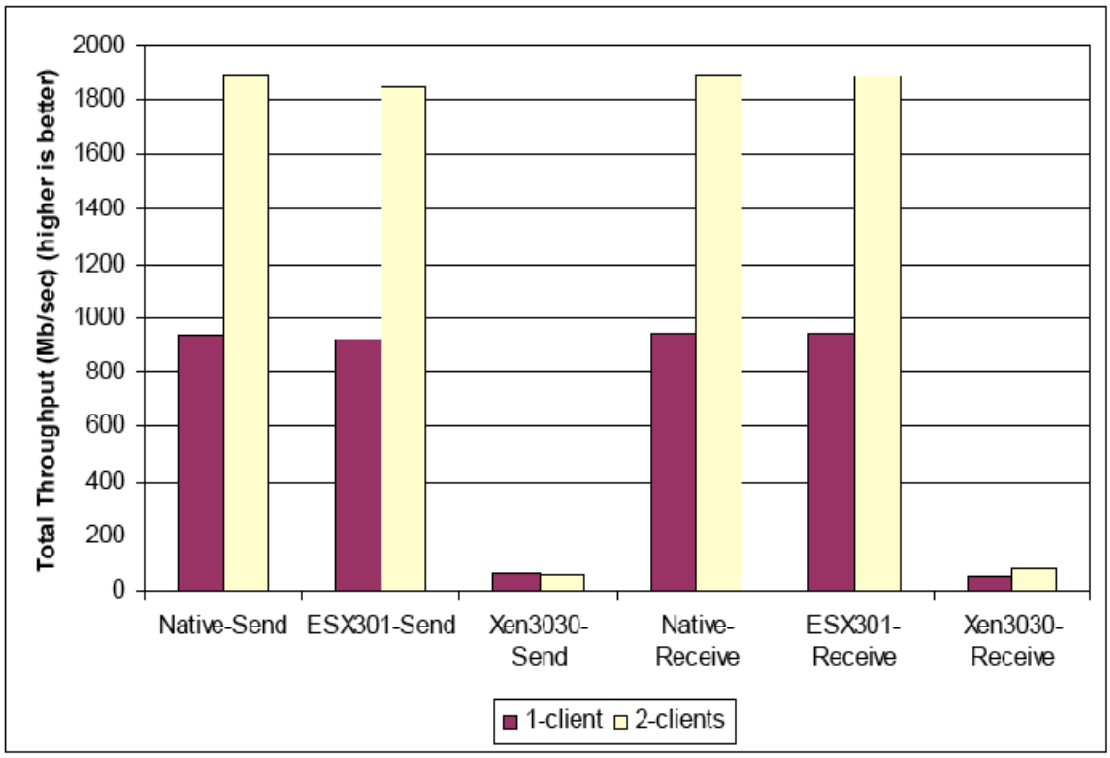

ILUSTRACIÓN 28 NETPERF. IO ETHERNET

C.4 INSTALACIÓN DE ROCKS EN ENTORNO VIRTUALIZADO

En el mundo empresarial actual, fuera de entornos "experimentales", la batalla por la virtualización la libran VMware y Xen. KVM está empezando a convertirse en un rival a tener en cuenta, pero aún no está tan maduro como los contrincantes.

Teóricamente, es posible instalar Rocks tanto sobre VMware como sobre Xen. La experiencia es que por los requisitos particulares de Rocks, es complejo instalarlo sobre VMware, y el rendimiento obtenido es menor.

Puesto que Rocks se basa en Linux, es muy sencillo ejecutar su kernel sobre Xen, y el rendimiento obtenido es aceptable para realizar pruebas de programación paralela y concurrente.

Por supuesto, no existe incremento de rendimiento al ejecutar las aplicaciones en un cluster virtual, pues realmente se trata de una sola CPU que se comparte entre frontal y nodos, pero es muy útil disponer de éste tipo de configuraciones para probar el software antes de ejecutarlo sobre un entorno cluster real.

En las últimas versiones de VMware se ha introducido el llamado Xen Roll, que permite instalar de forma sencilla un cluster virtual sobre un solo nodo o, si fuera necesario, ejecutar varios nodos virtuales sobre cada nodo físico del cluster.

C.4.1 POSIBLES CONFIGURACIONES VIRTUALES CON XEN ROLL

El cluster virtual con Rocks se puede configurar de varias maneras, dependiendo de los objetivos que se busquen cumplir en cada momento o para cada aplicación particular. Las dos configuraciones más comunes son las siguientes:

C.4.1.1 FRONTEND FÍSICO Y NODOS VIRTUALES

Se pueden añadir "contenedores" de máquinas virtuales a un cluster Rocks. Un contenedor es un tipo especial de nodo que es capaz de alojar máquinas virtuales xen, y mostrarlas al resto del cluster como si fueran nodos físicos independientes.

Ésta configuración es muy útil en caso de necesitar probar aplicaciones paralelas que requieran de un número muy alto de nodos. Evidentemente, como ya se ha discutido anteriormente, no se mostrará incremento de rendimiento, pero se pueden probar de manera muy sencilla los algoritmos paralelos, sin necesidad de despercidiar el costoso (y caro) tiempo de procesamiento en un cluster real con el número deseado de nodos.

En la Ilustración 29 se muestra esta configuración. Las máquinas expresso.rocksclusters.org, vm-container-0-0 y vm-container-0-1 son nodos físicos, mientras que compute-0-0-0 y compute-0-1-0 son nodos virtuales.

Por supuesto, es posible añadir tantos nodos virtuales como la memoria RAM del contenedor de máquinas virtuales permita, y Rocks se encargará de añadirlos para formar parte del cluster general.

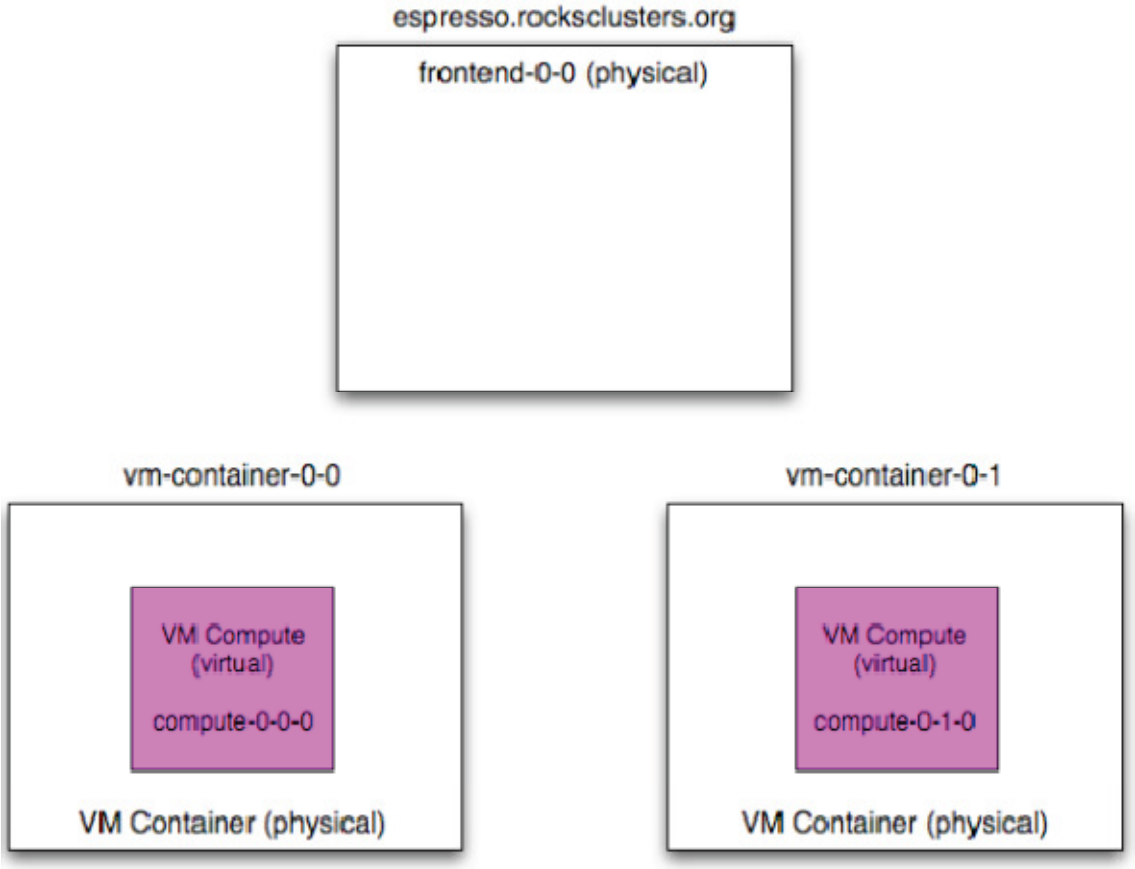

ILUSTRACIÓN 29 FRONTEND FÍSICO Y NODOS VIRTUALES

C.4.1.2 FRONTEND VIRTUAL Y NODOS VIRTUALES

Éste segundo tipo de configuración virtual con Rocks también es muy útil para las fases de diseño, pruebas y depuración de algoritmos paralelos. Es posible configurar en un solo nodo un cluster completo, con frontal más N nodos, siendo todos máquinas virtuales dentro de un anfitrión físico.

Se puede ver una representación de esta configuración en la Ilustración 30, donde tanto el frontend-0-0-0 como compute-0-0 y compute-0-1 son máquinas virtuales,

interconectadas entre si por una VLAN que Rocks crea automáticamente para que los nodos y el *frontend* formen parte de la misma red.

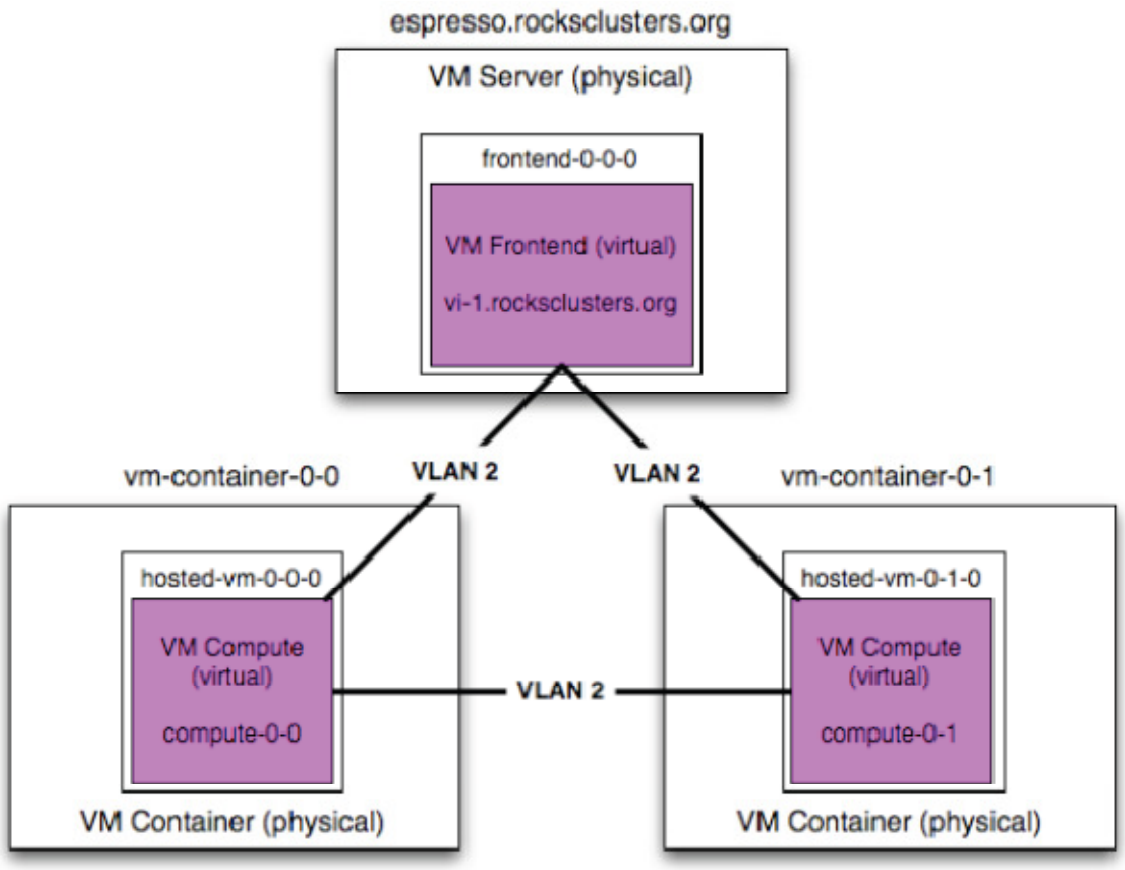

ILUSTRACIÓN 30 FRONTEND VIRTUAL Y NODOS VIRTUALES

Ésta topología virtual es la que se va a instalar en el apartado C.4.2, explicando los detalles de cómo configurarlo usando Rocks 5.2.

C.4.2 CREANDO UN CLUSTER VIRTUAL EN UN SOLO NODO

En este apartado se va a instalar una configuración completamente virtual, es decir, un cluster Rocks ejecutándose sobre una sola máquina física que, por ejemplo, se puede usar para realizar pruebas de algoritmos desarrollados en MPI antes de ejecutarlos en el cluster real.

El cluster estará formado por los siguientes elementos:

- Frontal o *frontend* (virtual)

- 2 nodos de computo (ambos virtuales)

El paso inicial para instalar un cluster de éste tipo basado en Rocks es instalar un frontal con el Xen Roll añadido. Para esto, se pueden seguir los pasos del apartado 1.19, añadiendo en el momento de la selección de Rolls el Xen Roll.

Si se está instalando el sistema desde un Jumbo DVD el Roll Xen está incluido en el DVD; en caso contrario, puede descargarse de la página de Rocks una imagen iso que contiene el Roll Xen, y que el instalador pedirá en el momento adecuado.

Una vez configurado un frontal con el Xen Roll realmente lo que se obtiene es un "contenedor" de máquinas virtuales, pues la máquina física no será un frontal en si misma hasta que configuremos uno. El primer paso para crear un cluster es añadirlo al sistema, mediante el comando "rocks add cluster". Los parámetros de éste comando son los siguientes:

- FQDN del *frontend*.
- IP del *frontend*
- Número de nodos de computo que formarán parte del cluster virtual (es posible añadir más una vez creado el cluster)

El comando "rocks add cluster" se encargará de crear una máquina virtual para alojar el frontal, de preparar las máquinas virtuales para los nodos de cómputo y de crear una VLAN única (como se vió en la Ilustración 30) para la red interna entre el cluster y los nodos. Además, realiza tareas internas en la base de datos de Rocks, tales como:

- Añadir una entrada a la tabla vm_nodes, para identificar qué host físico aloja cada máquina virtual.

- Añadir una entrada a la tabla vm_disks y reservar espacio en disco para las máquinas virtuales
- Generar direcciones MAC únicas para las interfaces ethernet de cada nodo.

Un ejemplo del comando "rocks add cluster" podría ser:

```
rocks add cluster virtual-tucan.arcos.inf.uc3m.es
137.110.119.118 2
```

Mediante el comando anterior, se crearía un cluster de dos nodos, con el FQDN y dirección IP mostradas en los parámetros. La respuesta de Rocks a éste comando será parecida a la siguiente:

```
created frontend VM named: frontend-0-0-0
created compute VM named: hosted-vm-0-0-0
created compute VM named: hosted-vm-0-1-0
```

Lo que indica que se ha creado un *frontend* virtual, de nombre frontend-0-0-0 y dos nodos virtuales, de nombres hosted-vm-0-0-0 y hosted-vm-0-1-0. El siguiente paso es instalar los nodos. Primero el *frontend*, mediante el comando:

```
rocks start host vm frontend-0-0-0
```

Para ver la consola de la máquina virtual, hay que usar el virt-manager. Pulsando sobre "localhost" y sobre la máquina frontend-0-0-0 se accede a la consola gráfica del *frontend*. El virt-manager se puede ver en la Ilustración 31:

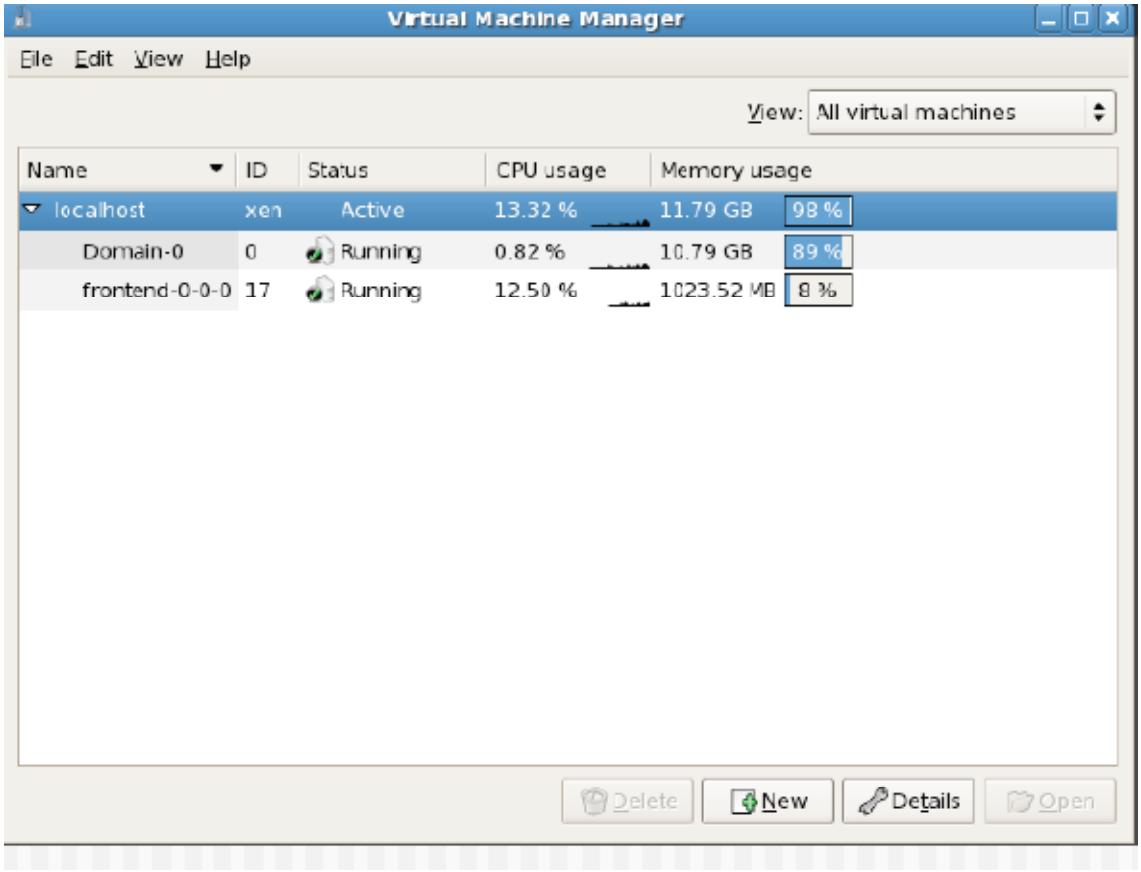

ILUSTRACIÓN 31 VIRT-MANAGER

Una vez instalado el *frontend*, se procederá de la misma manera que con un cluster físico. Hay que poner a Rocks en el modo "insert-ethers" y proceder a arrancar los nodos fisicos para que Rocks los identifique e instale como nodos del cluster. Para arrancar un nodo, se ejecuta el siguiente comando, por ejemplo, para arrancar el nodo virtual hosted-vm-0-0-0:

```
rocks start host vm hosted-vm-0-0-0
```

Como si se tratara de nodos físicos, Rocks comenzará a instalar el nodo virtual, y en unos minutos estará listo para usarlo en las aplicaciones paralelas.

11 BIBLIOGRAFÍA

[1] Top500 Supercomputing Sites [http://www.top500.org/]

[2] Diskless Image Management. IBM 2005. [http://www.bsc.es/media/511.pdf]

[3] MareNostrum Training. IMB 2004. [http://www.bsc.es/media/510.pdf]

[4] Myrinet racks. Barcelona Supercomputing Center 2009. [http://www.bsc.es/plantillaA.php?cat_id=207]

[5] Puesta en marcha del nodo de la Red Española de Supercomputación en la Universidad de Cantabria (R. Beivide, J. Marco) [http://grid.ifca.es/Presentaciones/2007/Puesta-en-marcha.ppt]

[6] Grid de Investigación en la Comunidad de Madrid. RedIRIS 2006. [http://www.gridimadrid.org/files/GRIDIMadrid-JT06.pdf]

[7] Cluster Howto. Ram Samudrala 2005. [http://www.tldp.org/HOWTO/Cluster-HOWTO.html]

[8] A Performance Comparison of Hypervisors. VMware. [http://www.vmware.com/pdf/hypervisor_performance.pdf]

[9] Xen VMs, Virtual Clusters and Programmatic Partitioning. Rocks UC Regents 2008. [http://www.rocksclusters.org/rocksapalooza/2008/rocks-2.pdf]

[10] Xen Virtualization and the Art of Virtual Clusters. Rocks UC Regents 2009. [http://www.rocksclusters.org/rocksapalooza/2009/xen.pdf]

[11] Bonie++ homepage [http://www.coker.com.au/bonnie++/]

[12] Intel® MPI Benchmarks, Users Guide and Methodology Description [http://www.uybhm.itu.edu.tr/documents/IMB_ug-3.0.pdf]

[13] Rocks Wiki, Intel MPI Benchmark [https://wiki.rocksclusters.org/wiki/index.php/Intel_MPI_Benchmark]

[14] Tentakel homepage [http://tentakel.biskalar.de/]

[15] Iperf homepage. [http://iperf.sourceforge.net/]

[16] Jperf homepage. [http://code.google.com/p/xjperf/]