Estrategias de instalación y gestión de clusters con software libre

Francisco Javier Ridruejo, Jon Agirre y José Miguel-Alonso

Resumen—**Aunque existen soluciones comerciales que incorporan técnicas de agrupación de máquinas, aquellas que gozan de una mayor aceptación en la comunidad científica siguen siendo las que proponen la utilización de software libre o abierto. En este documento se analiza la evolución de esta clase de software frente al discurrir de la relación calidad/precio de los componentes de consumo, aportando como documento práctico las experiencias obtenidas en la construcción de clusters mediante las técnicas que se describen.**

Palabras clave—**Clusters, software libre, componentes de consumo.**

I. Introducción

ES una realidad que el software libre está cambiando la forma de concebir las cosas. Y, aplicado en lo que a este artículo compete, está posibilitando el desarrollo de sistemas de alto rendimiento, fiabilidad y productividad a un coste muy bajo. Ha llegado el momento de extraer una visión general de cómo el software libre ayuda a la creación de un entorno HPC a muy bajo coste, así como de apuntar las líneas de desarrollo que marcarán el futuro del *cómputo de grandes números*.

II. Tipos de clusters

La evolución de los clusters ha venido siempre marcada por los pasos que se han seguido a la hora de desarrollar software para los mismos. Para comprender dicha evolución, es necesario entender cuál es el principio, y cuál será presumiblemente el final. Partimos de un grupo de ordenadores personales construidos con componentes de consumo, cuyo coste es cada día más bajo. Con dicho material, se quiere llegar a tener un sistema en el que poder aplicar todos los paradigmas de programación que serían aplicables en una máquina multiprocesador: memoria compartida, paso de mensajes, balanceo de carga, etc.

Con este objetivo se ha ido produciendo software a lo largo de los años, y la evolución del mismo ha generado tres tipos diferenciados de clusters:

- **Escasamente acoplados** — Una agrupación de computadores está escasamente acoplada si, aún siendo capaz de realizar procesamiento paralelo mediante librerías de paso de mensajes o de memoria compartida, no posee un sistema de instalación y gestión integrado que posibilite una recuperación rápida ante fallos y una gestión centralizada que ahorre tiempo al administrador.
- **Medianamente acoplados** — Podemos mejorar el modelo anterior mediante la incorporación de un sistema centralizado de instalación y gestión del cluster. Esto posibilitaría una rápida intervención del administrador en caso de error en alguno de los nodos, debido además a que la instalación del sistema operativo de cada uno de los nodos de cómputo es considerada *temporal* y nunca se almacenan datos relevantes para la estabilidad o coherencia del sistema. Esta nueva aproximación representa un paso interesante en la evolución de los clusters, debido a que permite una gran escalabilidad del sistema y posibilita la inclusión de diferentes paquetes de software dentro del propio sistema de gestión e instalación para su instalación automática.
- **Altamente acoplados** — El paso natural a dar una vez se ha conseguido una instalación medianamente acoplada es el de la incorporación de un sistema de equilibrado de carga de procesos. El ideal a conseguir es el de *replicar* el esquema de procesamiento de una máquina SMP. Por lo tanto, ¿por qué no eliminar las unidades de disco de los nodos de cómputo y así utilizar únicamente el procesador y la memoria?

III. Software y clusters: instalaciones

Una vez descritos los diferentes tipos de clusters, pasaremos a analizar las experiencias obtenidas en la aplicación del patrón impuesto por la sección anterior.

A. Beowulf, o el moderno prometeo

La condensación de hardware de diferentes fabricantes (no siempre el mejor) en un mismo multicomputador junto al sistema operativo Linux recibe el nombre de *Beowulf* [7]. Es el mejor ejemplo de sistema escasamente acoplado, ya que su concepción más básica solamente exige la interconexión de equipos y un mínimo de software para hacerlos trabajar de forma coordinada.

Para realizar el análisis de un sistema escasamente acoplado, construimos un cluster *Beowulf* basado en componentes de una calidad ligeramente superior a los que en el momento se consideraban estándar para los

Departamento de Arquitectura y Tecnología de Computadores, Facultad de Informática, Universidad del País Vasco/Euskal Herriko Unibertsitatea. Paseo Manuel de Lardizabal, 1. 20028 Donostia/San Sebastián. Gipuzkoa (España). Direcciones de correo electrónico: {miguel,acbripef}@si.ehu.es y agirre@sc.ehu.es.
Trabajo realizado con financiación del MCyT (TIC2001-0591-C02-02)

equipos domésticos de sobremesa. A continuación se enumeran sus características:

- Un nodo servidor biprocesador Athlon MP 2000+ (256 KB cache), 1 GB RAM, 80 GB disco duro UW-3 SCSI, adaptador Gigabit Ethernet Intel PRO/1000 MT Desktop, adaptador Fast Ethernet Intel PRO/100 M Desktop, unidad de disquete y CD-ROM.
- 10 nodos con similares características, salvo que los discos duros son 40 GB IDE y no cuentan con el adaptador Fast Ethernet.
- 2 conmutadores Gigabit Ethernet de 8 puertos, conectados en cascada (lo que posibilita la conexión de hasta 14 equipos).
- 1 conmutador KVM[1] de 16 puertos.
- 1 teclado, 1 monitor, 1 ratón.

Las máquinas fueron montadas en un armario *rack* y se conectaron al conmutador KVM para mayor comodidad a la hora de la instalación y gestión. Las conexiones gigabit son totalmente internas: sólo en nodo servidor se conecta al resto de la red de la UPV/EHU vía su adaptador Fast Ethernet. Por lo tanto, los nodos pueden utilizar direcciones IP privadas, y el servidor tiene una dirección IP privada y otra pública. No realiza ninguna función de encaminamiento[2].

Para realizar una instalación rápida de los nodos, y dado que el sistema era completamente homogéneo, se produjo un archivo de *kickstart* para su uso con la distribución de Red Hat 7.3 [10]. Este método facilita al proceso instalador de dicho sistema operativo los datos de configuración para eliminar la mayoría de las decisiones que, normalmente, se toman de forma interactiva. Dado que no se usa DHCP [11], la única tarea necesaria tras la instalación es la configuración de las direcciones IP (privadas).

En cuanto a la configuración de software, se instalaron sistemas de desarrollo para programación paralela por paso de mensajes (LAM con XMPI [5]), además del compilador gcc. Posteriormente, se le añadiría otra implementación de MPI, MPICH [8]. La Figura 1 muestra la ejecución, en este cluster, de una aplicación MPI en el entorno LAM-XMPI.

Aparte de software para el desarrollo de aplicaciones paralelas de alto rendimiento, se instaló Condor [14] para la realización de tareas que requieren de alta productividad.

Condor puede verse como un sistema avanzado de gestión de colas batch, desarrollado en la universidad de Wisconsin (Madison). Gestiona trabajos en entornos no dedicados, tales como salas de usuarios o equipos de sobremesa destinados a tareas ofimáticas. No obstante, es un magnífico gestor de colas para un cluster, además de una conexión inmejorable con software específico de Grid Computing como el Globus Toolkit [15].

Para aportar seguridad al sistema, además de mejorar el rendimiento global, se eliminaron aquellos servicios que se consideran innecesarios en un cluster: kudzu, sendmail y rhnsd.

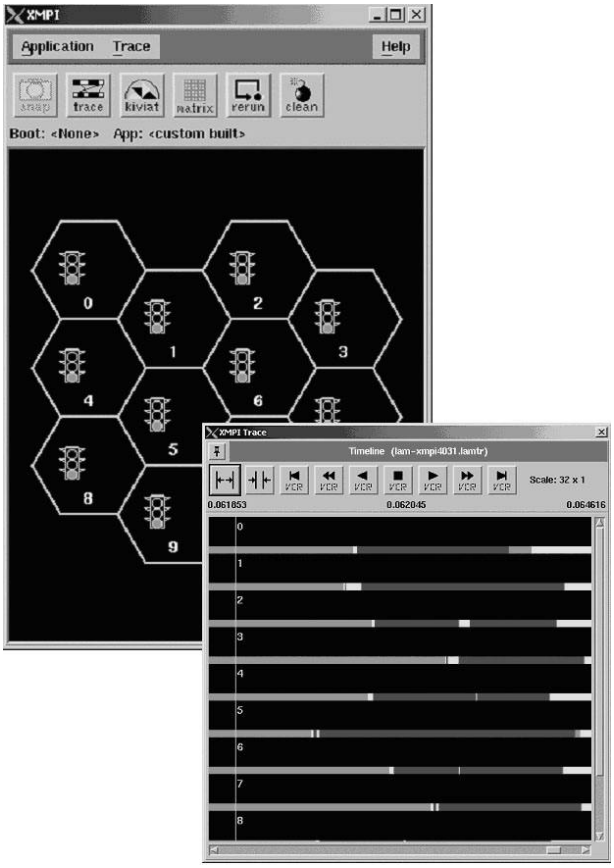

Figura 1: Ejecución de una aplicación MPI en un entorno LAM-XMPI.

Se exportó mediante NFS el directorio /home con los datos de los usuarios para conseguir SSI[3] a nivel de almacenamiento. Se habilitó el servicio SSH para conseguir canales seguros de comunicación, y se eliminaron comandos tradicionalmente inseguros como rsh, rcp, telnet, ftp, etc. Las cuentas de los usuarios se exportaron mediante NIS.

Es interesante remarcar que se ha utilizado NTP para la sincronización, ya que para medir ciertos parámetros de rendimiento es necesario afinar bien la precisión en el tiempo, además de para poder establecer relaciones de causalidad en los eventos reflejados en los archivos de log.

Posteriormente a la instalación, se desarrollaron comandos de administración usando el shell de Linux y con la ayuda de SSH y la ejecución remota de comandos que proporciona.

Añadir un nuevo nodo es sencillo: se procede a su instalación con *kickstart*, se configuran las direcciones IP y se añaden los datos del nuevo nodo a los ficheros de configuración /etc/hosts del resto del cluster, así como a los ficheros equivalentes de LAM y MPICH.

[1] Acrónimo de Keyboard-Video-Mouse. Se trata de dispositivo que conmuta dichos periféricos a un número determinado de computadores.

[2] Aunque se podría incorporar esta función, añadiendo servicios de encaminamiento y NAT.

[3] Single System Image, o imagen única de sistema. Se trata de ver el cluster como una única máquina. Se puede conseguir a diferentes niveles, como almacenamiento global, memoria compartida, espacio único de procesos, etc.

B. SCore, o la sencillez de lo oriental

Para poner en práctica la construcción de un cluster de máquinas medianamente acopladas, se escogió un software desarrollado por el PC Cluster Consortium (Japón) denominado SCore Cluster System [9]. La razón de esta decisión fue la siguiente: se poseían 5 máquinas homogéneas interconectadas por una red Myrinet, y si por algo se caracteriza SCore es por poseer una librería de comunicación de bajo nivel (PM) que proporciona un alto rendimiento para Myrinet. También posee una versión para Ethernet y otros dispositivos de red.

Además de MPICH para las aplicaciones de paso de mensajes, SCore ofrece una implementación de OpenMP sobre un sistema de memoria compartida distribuida basado en páginas denominado SCASH. La ventaja obvia de este software es la de poder utilizar en un cluster programas que contienen pragmas OpenMP, y que en un principio fueron pensados para funcionar en multiprocesadores.

La instalación de SCore exige la selección de un nodo servidor que contendrá una instalación completa[4] de Red Hat (distribución 7.3), en el cual quedará instalado el control del cluster. En nuestro caso, se seleccionó un equipo de diferente configuración que el resto. El hardware utilizado fue:

- 1 nodo servidor AMD K6-II, 333 MHz (64 KB cache), 196 MB RAM, dos discos duros SCSI de 3 y 4 GB, dos adaptadores Fast Ethernet. Unidad de disquete, lectora CD-ROM.
- 5 nodos, biprocesador Pentium II a 350 Mhz. (512 KB de cache), 128 MB RAM, 4 GB disco DURO IDE. Adaptador Fast Ethernet, adaptador Myrinet (SAN). Unidad de disquete.
- 1 conmutador Fast Ethernet (3Com) de 16 puertos.
- 1 conmutador Myrinet SAN 2x8 puertos.
- 1 conmutador KVM, teclado, monitor, ratón.

Durante la instalación, unos sencillos menús guían al usuario por la configuración del cluster, en los que se le preguntan cosas como el espectro de direcciones IP a utilizar o los dispositivos de red a los cuales se quiere aplicar la librería de comunicación PM. Una vez concluido el proceso, se insta al usuario a crear unos disquetes de instalación para los nodos del cluster[5], cuya única función es cargar un núcleo reducido de Linux y comunicarlos con el nodo servidor para que éste les pase por la red la imagen del disco duro que van a usar[6]. Este proceso es necesario sólo en la instalación, debido a que en las sucesivas ocasiones los nodos arrancan desde el disco duro propio. Esto deja absolutamente claro que los nodos de cómputo no pueden contener nada relevante en los discos duros, ya que su contenido es perfectamente *desechable.*

SCore proporciona un shell paralelo en el que se pueden ejecutar comandos en todos los nodos del cluster a la vez, y mediante el cual se pueden realizar comprobaciones rutinarias del estado del cluster, como verificar la hora, buscar archivos, o ver el estado de los procesos lanzados en cada nodo.

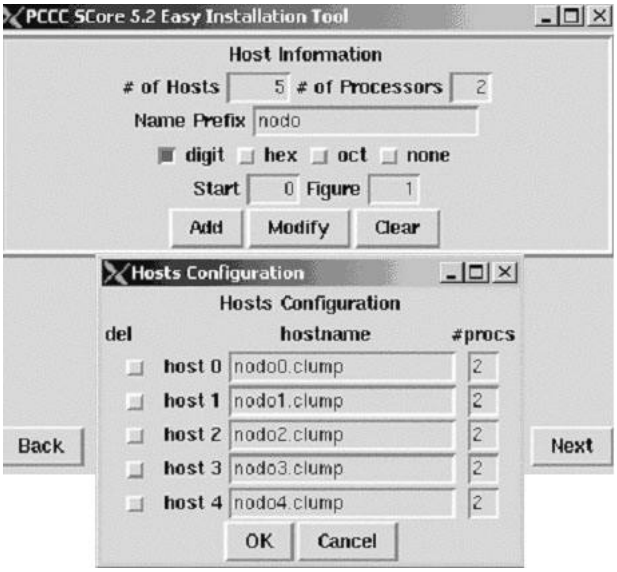

Figura 2: Detalle de la instalación de Score.

Además de lo antes descrito, SCore también instala el sistema de colas Portable Batch System. Por nuestra parte, añadimos una instalación de Condor como sistema de colas estandarizado en nuestro grupo.

Añadir un nodo nuevo supone ejecutar de nuevo el software de control del cluster, para incorporar los datos de ese nodo, y arrancar dicho nodo con el disquete preparado al efecto. El servidor servirá una copia de la distribución de Linux a instalar en ese nodo, que quedará ya incorporado al cluster.

La experiencia demostró que, superada la etapa inicial de acostumbrarse al sistema, y adaptarlo a nuestra configuración, la instalación de los equipos es muy rápida y sencilla.

C. OpenMosix, LTSP y sin discos, el triple salto mortal—fallido

La creación de un cluster altamente acoplado implica la disposición de un sistema centralizado de instalación y gestión, un sistema de distribución de carga entre los nodos del cluster, y que el conjunto resultante presente una imagen única desde el exterior (SSI). Además es interesante que el mantenimiento sea sencillo y esté centralizado, para ahorrar tiempo y problemas al administrador del sistema.

Para la implementación de un cluster de este tipo se ha elegido openMosix [16], derivación de código abierto del proyecto Mosix, una extensión del núcleo que proporciona a un grupo de máquinas interconectadas una imagen única, donde los procesos lanzados en un nodo pueden migrar de uno a otro, para ejecutarse en el nodo más adecuado según la carga del sistema y las necesidades de computación del propio proceso.

[4] Una decisión que se toma por sencillez en el proceso ya que, al cubrir esta distribución necesidades dispares, gran parte de los paquetes de una instalación completa nunca se llegarán a usar.

[5] Estos disquetes no serían necesarios si los adaptadores Fast Ethernet incluyesen PXE (descrito más adelante).

[6] Aquí podemos indicar que los disquetes creados por SCore no funcionaban correctamente con nuestros equipos, porque el núcleo básico que incorporan no incluye soporte para nuestros adaptadores Fast Ethernet. Fue necesario modificar el contenido de dicho disquete para añadir el soporte requerido en forma de módulo.

OpenMosix proporciona una distribución de carga transparente desde el punto de vista del programador y del usuario, que no necesitan modificar en absoluto los programas ni la forma de ejecutarlos para que aprovechen dicho equilibrado de carga distribuyéndose los procesos entre los nodos del sistema.

Los sistemas de este tipo son fácilmente mantenibles ya que si dejara de funcionar un nodo en cualquier momento, el sistema lo detectaría al instante y dejaría de enviarle procesos. Si por el contrario se añadiese un nuevo nodo al sistema, openMosix contiene un modo de detección y reconfiguración automática del cluster, con lo cual no haría falta la intervención del administrador de sistemas, mas que para instalar en el nuevo nodo el sistema operativo con todos los servicios utilizados, el núcleo openMosix correspondiente al nuevo nodo y configurar los dispositivos necesarios para el funcionamiento del sistema.

Para esta prueba se decidió hacer un cluster sin *diskless* (ni disco duro, ni unidad de disquete, ni unidad de CD-ROM). Por cluster *diskless* entendemos un conjunto de máquinas interconectadas, con una imagen única desde el exterior y en las que hay un único nodo servidor encargado de la comunicación con el exterior y que es el único que tiene discos, ya que los demás nodos carecen de ellos.

Este sistema se propuso por varias razones. A pesar de que la instalación inicial es mucho más complicada que las otras dos alternativas explicadas en este artículo, requiere un mantenimiento mucho menor ante cualquier fallo de un nodo. También el tiempo medio entre fallos es mayor debido a que el número de componentes que lo forman es más reducido que en las otras configuraciones de cluster explicadas. Otra razón de peso es el presupuesto, que en el caso de un sistema cuyos nodos clientes carecen de disco duro, CD-ROM y unidad de disquete, se abarata bastante.

Se decidió crear un cluster SSI formado por cinco nodos: uno servidor y cuatro de cómputo, todo ello con un exiguo presupuesto de 1900 euros. El hardware utilizado fue:

- Un servidor con estas características: procesador AMD Athlon XP 1800+ (256 KB cache). 1 GB RAM DDR a 266MHz. Tarjeta de red Fast Ethernet (VIA Rhine II, con soporte PXE). Tarjeta de red Fast Ethernet adicional para comunicación con el exterior. Disco duro de 40GB a 7200 RPM. Unidades de disquete y CD-ROM.
- Cuatro nodos de cómputo con las mismas características, salvo que cuentan sólo con un adaptador de red (con PXE) y, por supuesto, sin discos.

En lo referente al software, inicialmente, se optó por utilizar el sistema propuesto por el proyecto LTSP [6] para arrancar los nodos clientes. Este proyecto trata de convertir ordenadores viejos y con pocas prestaciones en terminales gráficos que reciben todo desde un servidor, al estilo de los antiguos terminales conectados a un *mainframe*. Tiene contribuciones que hacen fácilmente arrancable un cliente a través de la red mediante PXE[7] [4] y Etherboot [3]. Un nodo arrancado de esta manera incluye una distribución mínima de Linux con soporte XFree86 que, por defecto, hace que el sistema se comporte como un simple X-Terminal conectado al servidor y no ejecuta ninguna aplicación en modo local. Opcionalmente, se puede permitir la ejecución de ciertas aplicaciones locales, pero sólo de aquellas parcheadas para trabajar bajo LTSP. Los ficheros se comparten vía NFS; todos los nodos importan los mismos directorios desde el servidor.

Una vez hecha la instalación, comprobamos que adolecía de serias limitaciones. En primer lugar, la instalación de Linux de los nodos es muy reducida, lo que impide la ejecución de la mayoría de las aplicaciones de nuestro interés. Aunque se puede integrar fácilmente openMosix con LTSP, lo cierto es que el cluster resultante queda con unos nodos demasiado ligeros, poco funcionales, por diferentes problemas de compatibilidad de aplicaciones y librerías. Sin embargo, esta instalación preliminar sirvió como experiencia para arrancar los nodos por red usando PXE y EtherBoot.

D. OpenMosix, ClusterNFS y sin discos, el triple salto mortal—con éxito

El proyecto ClusterNFS [2] tiene como objetivo, precisamente, dar soporte a clusters *diskless*. Nuestra tarea ha sido integrarlo con openMosix y el arranque por red mediante EtherBoot y PXE. En particular, ClusterNFS trata de que los nodos monten todo su sistema vía NFS.

Existen varias opciones para poner en marcha el sistema. La primera puede ser tratar de montar, en los nodos, la raíz del sistema de ficheros del servidor, pero esto presenta varios problemas, ya que el sistema de ficheros no puede ser exactamente el mismo en todos los nodos (incluyendo el servidor), porque cambian las configuraciones y los directorios de dispositivos "/dev" y procesos "/proc". Para solventar ese problema se pueden hacer dos cosas: (1) replicar el sistema de ficheros para cada cliente, cada uno con su configuración, o bien (2) replicar sólo lo estrictamente necesario y montarlo correctamente durante el arranque. ClusterNFS, como se verá más adelante, opta por esta última opción. A continuación describimos los pasos dados en la instalación.

En el servidor se instaló la distribución de Linux Red Hat 9, con soporte para desarrollo del núcleo, DHCP, desarrollo para la interfaz gráfica XFree86, herramientas de desarrollo, y librerías de paso de mensajes MPI (LAM). No se incluye el soporte para NFS porque se añade más tarde al incorporar el paquete ClusterNFS.

Aparte de las opciones de instalación arriba comentadas, propias de la distribución de Linux elegida, se instalaron los siguientes paquetes: el último núcleo de openMosix para Athlon, las utilidades de usuario de dicho núcleo, el código fuente del núcleo para

[7] PXE o Pre-eXecution Environment es una tecnología desarrollada por Intel que permite cargar y ejecutar un programa dearranque por red desde un servidor de la red antes de arrancar mediante el sistema operativo del disco duro.

openMosix versión Athlon, las utilidades "mknbi"[8] del proyecto Eherboot, un servidor de TFTP [12], ClusterNFS, openMosixview (para el que hace falta instalar aparte las librerías qt-devel y glut) y una Etherboot ROM (imagen de arranque del proyecto Etherboot) para la tarjeta de red VIA Rhine II de los clientes. En un futuro se instalará, además, Condor.

Es necesario, además, compilar un núcleo específico para los clientes, con las opciones de openMosix, autoconfiguración IP por DHCP, soporte de NFS y root sobre NFS, además de incluir el driver de red del cliente en el propio núcleo. Dicho núcleo debe ser etiquetado con la herramienta mknbi-linux para que pueda ser transmitido por la red a un cliente.

El proceso de arranque de los clientes se hace a través de la red mediante PXE. Al arrancar un cliente, éste interroga vía multicast al servidor DHCP de la red para que le otorgue una IP y le dé los demás parámetros de configuración de red, como son puerta de enlace, máscara, servidor de DNS y quién es el servidor de TFTP. Una vez configurada la interfaz ethernet, el nodo, siguiendo la indicación del servidor DHCP, se descarga del servidor TFTP un programa cargador (obtenido de rom-o-matic [13]) creado para su tarjeta de red, que incluye un núcleo mínimo. Dicho núcleo se encarga de hacer una pequeña inicialización de parámetros del hardware, como habilitar la memoria RAM extendida (hasta ese momento sólo se disponía de la memoria base 640KB, insuficiente para cargar un núcleo completo en memoria.

A continuación, el cargador vuelve a contactar con el servidor DHCP para que éste le indique cuál es nombre del fichero con el núcleo completo a descargar para arrancar la máquina. El servidor de DHCP recibe una nueva pregunta (que ya no está etiquetada como "PXEClient", sino como "Etherboot") y responde con el nombre del servidor TFTP y del fichero que contiene el núcleo solicitado. El nodo descarga el núcleo, lo carga en memoria y comienza a arrancar. En este punto montará la raíz del sistema de ficheros mediante NFS, tal y como se le ha indicado en el proceso de compilación del núcleo.

El servidor de NFS no es el normal de una distribución linux, sino que es ClusterNFS. Cuando un cliente NFS realiza una petición a ClusterNFS, éste primero comprueba varias reglas antes de servir el fichero. Entre estas reglas se comprueba que exista el fichero en cuestión, pero con el sufijo $$IP$$, donde IP es la dirección IP del cliente que realizó la petición, o el sufijo $$CLIENT$$, que se servirá si la petición la realiza cualquier nodo. De esta manera, se pueden crear diferentes directorios y ficheros específicos para cada uno de los nodos, ya que hay muchos de ellos que deben ser diferentes, como son "/etc", "/var", "/dev" y "/proc". Por ejemplo, se creará un directorio "/etc$$10.0.1.1$$" que será el directorio "/etc" del nodo de dirección 10.0.1.1.

Un vez puesto en marcha el sistema ClusterNFS se adaptó un script que crea los directorios y configura los parámetros de cada uno de los clientes, creando para cada uno su propio árbol de directorios específicos, posibilitando una configuración adecuada acorde a cada uno de los clientes. De esta manera se consigue que arranquen todos los clientes y monten su árbol de directorios mediante NFS, compartiendo todo excepto lo mínimo: configuración, dispositivos, y servicios. Cada cliente tendrá disponible una distribución Linux completa con todos sus aplicaciones, servicios y librerías.

Este sistema es muy versátil en cuanto a mantenimiento se refiere. Cuando se instala un nuevo programa y se quiere que esté en todos los nodos, basta con instalarlo en el servidor y automáticamente estará disponible para todos, siempre que no se instale en un directorio que esté personalizado pare un nodo dado. También es sencillo añadir nuevos nodos al cluster, ya que basta con ejecutar el script del servidor indicando el índice de un nuevo nodo y se creará toda la estructura de directorios y la configuración específica para el mismo. Sólo habría que añadir una nueva entrada en el fichero de configuración del servidor DHCP, indicando la dirección MAC del equipo. Si el nuevo fuese distinto a los ya existentes, por ejemplo con distinta tarjeta de red, sería necesario además recompilar un núcleo apropiado para él.

Si un nodo degradase su configuración, bastaría con ejecutar de nuevo el script para volver de nuevo a tenerlo totalmente funcional

Una vez montado y configurado el sistema se instaló openMosixview, un conjunto de utilidades para monitorizar y gestionar de una forma gráfica muy atractiva el estado de un cluster openMosix. Proporcionan visión y control de los procesos del cluster en tiempo real, además de medidas de rendimiento, estadísticas e históricos (ver Figura 3).

IV. Software adicional

El uso diario de los clusters descritos en las secciones anteriores nos ha permitido detectar algunas carencias en lo que se refiere a la capacidad de gestión y administración. Estamos trabajando en cubrirlas usando software libre, o software desarrollado por nosotros.

Una de ellas es la de encender y apagar un cluster como si fuese un equipo único (SSI a nivel de encendido y apagado). Para ello, en primer lugar, hemos desarrollado un pequeño script que, a petición del administrador y desde el servidor, apaga todos los nodos y luego apaga el servidor. En cuanto al arranque, una vez que el servidor está listo y tiene en marcha todos los servicios necesarios, otro script usa la capacidad WOL [1] (Wake On Lan) de los adaptadores Ethernet de los nodos para que estos se enciendan (todos a la vez, o en secuencia) sin intervención de un operador. Se usa para ello un programa que envía una trama especial denominada "paquete mágico" a la dirección MAC que se indica, con el resultado de que dicha máquina inicia su encendido[9]. La secuencialización tiene dos ventajas: por un lado, no arrancan los nodos hasta que el servidor está totalmente listo. Por otro lado, en clusters con muchos nodos, evita la aparición de fuertes picos de consumo eléctrico.

[8] mknbi es una utilidad que adapta imágenes de sistemas operativos para que sean arrancables mediante Etherboot.

[9] Para ello, es necesario que el equipo esté apagado, pero en un estado especial que, entre otras cosas, mantiene activa la tarjeta Ethernet.

La segunda tarea a mejorar es la monitorización de los clusters (uno a uno o, mejor aún, en conjunto). Estamos evaluando la posibilidad de instalar y adaptar paquetes libres, o desarrollar nuestro propio monitor.

V. Conclusiones

Aunque existen muchas y diversas iniciativas para la instalación automatizada de clusters, no hemos encontrado ninguna que, de forma inmediata y directa, se adapte al hardware disponible, o a las aplicaciones a ejecutar. Es habitual encontrarse con falta de soporte para determinados dispositivos, o entornos de ejecución que imponen restricciones excesivas. A pesar de contar con manuales de instalación detallados, las instalaciones aquí descritas no han sido, en ningún caso, sencillas, ni inmediatas.

La primera solución descrita resulta muy útil para las tareas antes descritas: un sistema de colas batch y un buen entorno de ejecución de aplicaciones MPI. El mayor inconveniente ha sido su elevado coste, al replicar muchos elementos que apenas se usan. La última exprime al máximo el euro, llegando a una solución muy flexible—claro que de escalabilidad limitada, porque el nodo servidor acabaría convirtiéndose en un cuello de botella. La solución SCore parece la ideal: instalación casi automática, gestión de colas, ejecución de programas MPI y sistema de memoria compartida distribuida. Se acercaría a la perfección si incorporase el sistema de equilibrado de carga de openMosix y la posibilidad de trabajar sin discos en los nodos.

La conclusión final es que montar clusters con software libre es viable, pero no está exento de complicaciones. Aún queda mucho trabajo por hacer.

VI. Referencias

[1] Becker, D. Using Wake-On-Lan with Linux. http://www.scyld.com/expert/wake-on-lan.html
[2] ClusterNFS Home Page http://clusternfs.sourceforge.net
[3] Etherboot Project http://www.etherboot.org
[4] Intel Wired for Management (WfM) www.intel.com/ial/wfm
[5] LAM / MPI Parallel Computing http://www.lam-mpi.org/
[6] Linux Terminal Server Project http://www.ltsp.org/
[7] Merkey, Phil “Beowulf History”. Disponible en http://www.beowulf.org/beowulf/history.html
[8] MPICH-A Portable Implementation of MPI http://www-unix.mcs.anl.gov/mpi/mpich/indexold.html
[9] PC Cluster Consortium. Score Cluster System. http://pdswww.rwcp.or.jp/
[10] Red Hat Inc. http://www.redhat.com/
[11] Resources for DHCP. http://www.dhcp.org/
[12] RFC 1350. The TFTP Protocol (Revision 2). http://www.faqs.org/rfcs/rfc1350.html
[13] ROM-o-matic.net Home Page http://rom-o-matic.net/
[14] The Condor Project Homepage http://www.cs.wisc.edu/condor/
[15] The Globus Project http://www.globus.org/
[16] The openMosix project. http://openmosix.sourceforge.net/

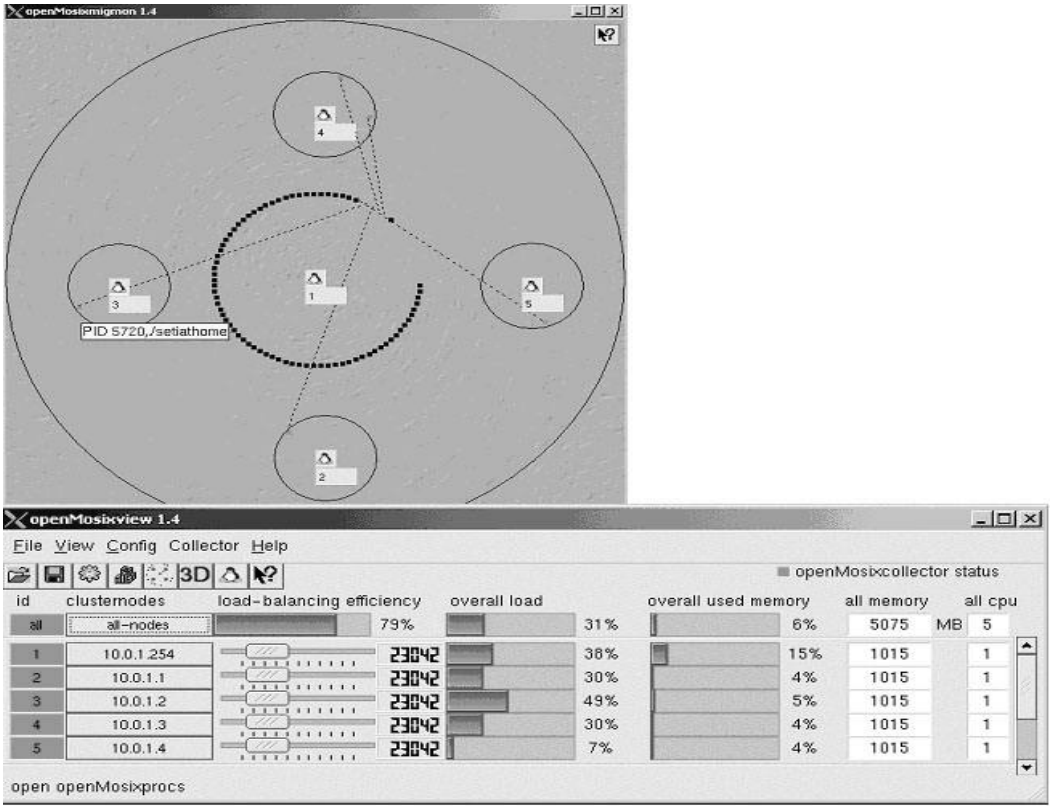

Figura 3: Monitorización de un entorno openMosix, en el que se están ejecutando varias aplicaciones, repartidas entre los nodos del cluster.