Supercomputadoras basadas en “clusters” de PCs

Diego H. Milone
d.milone@ieee.org

Adrián A. Azar
adrian@fi.uner.edu.ar

Leonardo H. Rufiner
lrufiner@fi.uner.edu.ar

02 de julio de 2002

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

Resumen: un “cluster” es un grupo de computadoras interconectadas que trabajan conjuntamente en la solución de un problema. Estos sistemas constituyen una solución flexible, de bajo costo y gran escalabilidad para aplicaciones que requieren una elevada capacidad de cómputo y memoria. A diferencia de las supercomputadoras tradicionales, los sistemas en cluster requieren de una inversión inicial menor, ya que pueden construirse a partir de PCs de uso masivo. Más aún, y por esta misma razón, los costos de mantenimiento y actualización se reducen notablemente. Por otro lado, cuando cualquier nodo del cluster se daña, todos los otros siguen funcionando normalmente absorbiendo la carga total del sistema y de esta forma se obtiene una tolerancia a fallas superior. En cuanto a la escalabilidad, es muy simple incorporar nuevos nodos en un cluster y expandirlo progresivamente con una baja inversión. Este artículo resume las tareas de instalación, configuración y prueba de un cluster para el cómputo paralelo, realizadas en el Laboratorio de Cibernética de la Facultad de Ingeniería de la Universidad Nacional de Entre Ríos (FI-UNER). El Grupo de Reconocimiento Automático del Habla utiliza actualmente este cluster para la realización de sus experimentos numéricos.

Palabras clave: supercomputadoras, cluster, experimentación numérica, Linux.

Abstract: a “cluster” is a group of interconnected computers that work jointly in the solution of a problem. These systems constitute a flexible solution, of low cost and great scalability, for applications that require a high capacity of computation and memory. Unlike traditional supercomputers, cluster systems require a smaller initial investment, since they can be constructed from massive use PCs. For this same reason, maintenance and update costs are remarkably reduced. On the other hand a superior tolerance to faults can be obtained, because when any node of the cluster is damaged, all the others continue working normally absorbing the total load of the system. From scalability viewpoint, it is very simple to incorporate new nodes in cluster and expanding it progressively, with a low investment. This paper summarizes installation, configuration and test tasks of a cluster for parallel computing, made in Laboratorio de Cibernética, Facultad de Ingeniería, Universidad Nacional de Entre Ríos (FI-UNER). At this moment the Automatic Speech Recognition group uses this cluster for the accomplishment of its numerical experiments.

Key Words: supercomputers, cluster, numerical experimentation, Linux.

1 Introducción

Un cluster es una agrupación de computadoras que trabajan en un problema en común. En este artículo se presenta una breve descripción de los sistemas en cluster y un resumen de los trabajos llevados en la construcción de un cluster de PCs. Todo el cluster está basado en sistema operativo Linux, por lo que en muchos casos se hará referencia a herramientas y procedimientos de trabajo típicos en sistemas operativos Linux/UNIX. Sin embargo, no es el objetivo de este artículo profundizar en cuestiones relacionadas con estos sistemas, sino tocar los puntos centrales relacionados con la construcción del cluster. Para obtener mayor información acerca de los sistemas Linux/UNIX recomendamos consultar **Catalina Gallego** 1998. Por otro lado, gran parte del trabajo de instalación y configuración requiere de conocimientos previos acerca de la administración de redes de computadoras en Linux y su conectividad con sistemas Windows. Para obtener más información acerca de la administración de redes puede consultar **Kirch** y cols 2000. Un buen artículo introductorio al tema de los clusters se puede encontrar en **Hargrove** y cols 2001, y una revisión más extensa de las diversas arquitecturas en **Spector** 2000. Para profundizar en los aspectos relativos a la programación paralela se puede consultar **Gropp** y cols 1999.

En la primera parte de este artículo se realiza una breve revisión acerca del origen de los clusters, su motivación principal y las tecnologías involucradas. También se destacarán las aplicaciones y ventajas de los clusters y las principales tareas a llevar a cabo para su construcción. Se dedica una sección para introducir los componentes específicos para cómputo paralelo y balance automático de carga (sistemas MOSIX, MPI y PVM). A continuación, se describe la estructura hardware y software del cluster con todas las herramientas de administración necesarias, especificando su instalación y configuración. En el final se presenta el estado actual del desarrollo, las conclusiones del trabajo y se describe la forma propuesta para el crecimiento organizado del sistema, junto con los recursos mínimos necesarios.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

1.1 Breve reseña histórica

Las tendencias en computación requieren de un aumento constante de recursos. Cuando se tienen más recursos se resuelven los problemas planteados y entonces se piensa en resolver problemas más complejos y, nuevamente, se llega a la conclusión de que se necesitan más recursos. Este círculo vicioso ha guiado los avances del hardware y software, y la inserción de las computadoras en casi cualquier campo de aplicaciones. Se llega así a necesitar de “supercomputadoras” que, como bien se sabe, son de un altísimo costo, tanto en la inversión inicial como en su mantenimiento y actualización. Sin embargo, un tipo particular de supercomputadoras se logra mediante la agrupación de muchas computadoras en un cluster.

Las principales aplicaciones de las supercomputadores se han centrado en el diseño y realización de experimentos numéricos para la simulación de sistemas grandes y complejos como:

- Modelos de sistemas biológicos
- Inteligencia Artificial
- Reacciones químicas
- Física de partículas
- Estudios sociológicos
- Predicciones económicas
- Predicción meteorológica
- Diseño aerodinámico

Entre muchos otros, además se utilizan para la solución de grandes problemas de identificación y optimización en estos mismos campos. Las soluciones obtenidas suelen tener un gran impacto económico, lo que justificó los precios de las supercomputadoras y los programas asociados. Muchas herramientas de optimización fueron desarrolladas durante la guerra y utilizadas en supercomputadoras de institutos del gobierno para optimizar el uso de

los recursos disponibles. Los primeros usuarios privados de supercomputadoras fueron las grandes compañías como las petroleras y los bancos. En la década del '60, se lanza el Proyecto Apolo. La misión Hombre a la Luna fue el suceso Nº 1 en la historia de la simulación basada en computadoras. Se simuló el equipo físico y los procedimientos operacionales usados en la misión. En la década del '70 los esfuerzos se concentraron en el desarrollo de modelos de políticas gubernamentales. La motivación para este esfuerzo fue el deseo de encaminar proyectos de desarrollo económico, social y del medio ambiente. Se desarrollaron así, técnicas para la modelización de sistemas grandes y complejos.

A partir de 1994 un grupo de científicos de la NASA se embarcaron en el proyecto de construir una supercomputadora para cómputo en paralelo a partir de muchas computadoras que ya poseían. El principal fin perseguido era obtener un equipo de bajo costo aunque con una capacidad de cómputo semejante a la de una supercomputadora tradicional. Así se constituyó el primer cluster con consistía en 16 máquinas con procesadores Intel 486DX4, conectados mediante una red Ethernet de 10Mbit/seg. La capacidad conseguida fue de unos 70 MFLOPS, que prácticamente equivalía a una supercomputadora pequeña de la época, por una décima parte del costo. El software utilizado fue un sistema denominado Parallel Virtual Machine (PVM) previamente existente para máquinas con procesadores múltiples. Este primer sistema de cluster se denominó *Beowulf* y aún es muy utilizado en todo el mundo. Toda esta investigación se realizó en base a un sistema operativo experimental que se denominaba *Linux*. En aquellas épocas, Linux era un clon poco conocido de UNIX para procesadores Intel y otras arquitecturas. Sin embargo este sistema operativo proveyó a los investigadores un excelente desempeño y aprovechamiento del hardware, con código fuente abierto y a un precio inmejorable: *gratis*.

Actualmente Linux es un sistema operativo maduro y ampliamente difundido en todo el mundo, con decenas de millones de copias en uso sobre las más diversas arquitecturas de computadoras (PowerPC, Sun Sparc, Alpha, etc). Más recientemente se han desarrollado núcleos de Linux que ya incorporan las herramientas básicas para administrar clusters. Este es el caso del sistema MOSIX, desarrollado por Amnon Barak, de la Universidad Hebrea de Jerusalem (**Amir** y cols 2000, **Barak** y cols 1993). Este sistema provee un balance de carga adaptativo y un conjunto de algoritmos para mover regiones de memoria automáticamente, con una gran eficiencia y amplias ventajas en la escalabilidad. De acuerdo a la carga requerida en un nodo del sistema y la disponibilidad de recursos en cualquiera de los otros nodos (recursos que se monitorean permanentemente), se activa la migración de procesos de forma transparente para el usuario. Todas las ventajas relacionadas con la migración automática de procesos conviven con la posibilidad de asignar tareas con los métodos tradicionales disponibles en otros clusters. Dado que en el cluster implementado se ha utilizado un núcleo de este tipo, en la Sección 2.1 se tratará con mayor detalle el sistema MOSIX.

1.2 Cómputo paralelo y clusters

La idea básica de un cluster es simple: se trata de un conjunto de computadoras conectadas a través de una red, trabajando en un gran problema de cómputo que ha sido dividido en varios subproblemas pequeños. Existen diferentes configuraciones de hardware y mucho software disponible para trabajar en clusters, dividiendo los problemas y aprovechando al máximo los recursos disponibles. Todos estos paradigmas de resolución de problemas tienen sus bases en el "cómputo paralelo". El paralelismo consiste en poder dividir una tarea en partes que trabajen independientemente en lugar de poseer una única tarea en la que todos sus procesos se encadenan uno tras otro, necesitando de los resultados del anterior para poder comenzar. El paralelismo posee dos componentes bien diferenciados: el hardware y el software.

1.2.1 Hardware para paralelismo

El paralelismo a nivel hardware puede realizarse a nivel del mismo procesador y simultáneamente a nivel del sistema en su conjunto. El paralelismo a nivel del procesador consiste en la capacidad que tienen los nuevos procesadores[1] para realizar más de una cosa a

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

la vez. Estos procesadores están divididos internamente en múltiples unidades de ejecución e *intentan* realizar más de una operación en cada unidad de tiempo (del reloj que controla al procesador). Sin embargo, la mayoría de los programas son compilados como una secuencia muy lineal de operaciones y solamente un buen optimizador (específico para el procesador en cuestión) logra en algunos casos organizar el código binario para aprovechar estas diferentes unidades de ejecución.

El paralelismo a nivel del sistema es donde se pueden ubicar los clusters, junto a las computadoras con múltiples procesadores. En cualquiera de los casos tendremos múltiples procesadores y necesitaremos, como aspecto clave, una forma de compartir los recursos de memoria. En principio, todos los procesadores, ya sea en una misma máquina o en muchas diferentes, deberán tener alguna forma de acceso a la memoria RAM global del sistema y a los discos disponibles. Esto es implementado de muchas formas dependiendo, en primer lugar, de la arquitectura hardware que conecta a los diferentes procesadores. Este es uno de los aspectos que más influye, junto con la capacidad de los procesadores, en el desempeño de un sistema para cómputo paralelo. En general se tienen en cuenta parámetros como: la velocidad de acceso a la memoria, el tiempo de espera en la respuesta y la velocidad de transmisión del bus de acceso. En las computadoras con múltiples procesadores, la comunicación con las unidades de memoria suele ser muy eficiente. En los clusters esta capacidad de comunicación está directamente relacionada con el hardware y configuración de red. Es por esto que el sistema de red es un aspecto fundamental en la construcción de un cluster.

Suele hacerse una distinción entre los sistemas paralelos según: sistemas de “grano fino” y sistemas “grano grueso”. Entre los primeros sistemas se incluyen los paralelismos hardware a nivel de procesador y de computadoras con múltiples procesadores. Cuando se habla de “grano grueso” o distribuido, se hace alusión a sistemas compuestos por una colección de computadoras actuando conjuntamente. En esta última clasificación se encuentran los sistemas de cluster.

1.2.2 Software para paralelismo

El software es uno de los aspectos más complejos de resolver en el paralelismo. Dado un problema en particular, el objetivo es encontrar áreas bien definidas que pueden resolverse independientemente de las demás. Los beneficios que puedan obtenerse de un sistema para cómputo paralelo están de acuerdo con la capacidad de descomponer un programa en módulos que puedan resolverse de forma independiente. Muchos problemas tienen una formulación naturalmente paralela. Sin embargo, algunos otros problemas sólo pueden resolverse de forma secuencial y en estos casos pocos beneficios pueden obtenerse de un sistema de cómputo paralelo. En una posición media, existen problemas que, si bien no parten de una formulación típicamente paralela, pueden reformularse para aprovechar las ventajas de un sistema de este tipo. Otra circunstancia, bastante común, que permite aprovechar una arquitectura paralela es la corrida de varios experimentos similares con diferentes valores para algunos de sus parámetros (experimentos de “sintonización”). En estos casos cada experimento pueden correrse en forma simultánea siendo el propio sistema quien se encarga de asignar cada uno al mejor nodo disponible en el cluster.

1.3 Principales ventajas

¿Por qué construir clusters de computadoras para cómputo masivo cuando existen supercomputadoras de hardware específico que ya hacen esto? La respuesta es simple: porque un cluster es mucho más barato. Y no hablamos simplemente de la inversión inicial sino también de los altísimos costos de mantenimiento y actualización de las supercomputadoras. En general, la velocidad con que una supercomputadora se hace obsoleta no difiere sustancialmente de la velocidad con que lo hace una PC común. Los fabricantes de supercomputadoras nos ofrecen usualmente una única opción para la actualización: “compre el nuevo modelo de este año”. Además, ni pensemos lo que representa si algún componente de la supercomputadora se daña fuera de su período de garantía.

Existe también una ventaja implícita en el hecho de utilizar varias PCs iguales o similares

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

en paralelo, que es la tolerancia a fallas que tiene el conjunto. Es decir que si una o más PCs se dañan o quedan temporalmente fuera de servicio, el resto puede continuar haciendo su trabajo sin mayor problema. De hecho, en estos casos la carga de cómputo se redirecciona automáticamente al resto de los equipos. Esto también se traduce en facilidad de mantenimiento y flexibilidad para la actualización de los componentes individuales o de los equipos.

Otros indicadores destacan la importancia de los clusters en las aplicaciones que requieren grandes capacidades de cómputo. En las listas de supercomputadoras que se publican periódicamente era común ver los diferentes modelos de las ya conocidas marcas comerciales (Sun, IBM, Cray, etc.), pero a partir de 1998 comenzaron a estar presentes los "Clusters Linux".

Otra ventaja adicional es que, en el caso de los sistemas MOSIX, también es posible agregar equipos usados relativamente nuevos. Sin embargo, el rendimiento óptimo se logra con equipos iguales o similares de última generación. Una supercomputadora CRAY-2, introducida en 1985, tenía un desempeño máximo de 1.9 GFLOPS. En ese tiempo la CRAY-2 tenía la memoria central más grande del mundo con un total de 2048 MBytes. Su costo era del orden de los 10 millones de U$S. El modelo T94 fue introducido en 1995 y posee 4 procesadores vectoriales de 1.8 GFLOPS/CPU y 57.6 GFLOPS máximo. Su valor es de 2.5 millones de U$S. El cluster del Institute of Computer Science, Hebrew University, Israel cuenta con 55 nodos (cuadruple, doble, simple) PIII 1GHz - 550 MHz, 128-256MB RAM, conectado por Ether-Express Pro 100 NICs (**`www.mosix.org`**). Un cluster de este tipo puede tener un desempeño del orden de unos 50 GFLOPS por un precio de unos 50.000 U$S.

1.4 La construcción del cluster

En esta sección se presentan brevemente las tareas que se realizaron para la construcción del cluster. En las secciones siguientes de este informe se tratará estos puntos con mayor detalle. Las tareas realizadas fueron de diferente tipo y complejidad y pueden resumirse en las siguientes:

1. Revisión bibliográfica y estudio de las bases de funcionamiento de los clusters.
2. Selección del tipo adecuado para las necesidades del grupo (MOSIX).
3. Selección del equipo mínimo necesario para implementar el cluster elegido.
4. Adquisición y prueba del hardware seleccionado.
5. Instalación de placa de Red adicional en el nodo maestro.
6. Cableado general y puesta a punto de la conexión del servidor a la red local de la FI-UNER.
7. Instalación y conexión de los nodos mediante un Switch.
8. Instalación del sistema operativo Linux en ambas PCs
9. Instalación del núcleo del sistema MOSIX en ambas PCs
10. Configuración de todos los servicios y protocolos de red necesarios.
11. Configuración y puesta a punto del MOSIX
12. Instalación y puesta a punto del MPI
13. Instalación y puesta a punto del PVM
14. Instalación y puesta a punto del sistema de colas de procesos (PBS).
15. Instalación y puesta a punto del sistema de administración remota (Webmin).
16. Instalación de paquetes, lenguajes y herramientas específicas para cálculo numérico.
17. Prueba del sistema con diferentes aplicaciones (software de redes neuronales, programas de cálculo pesado, etc.)
18. Instalación de clientes X (entorno gráfico) y SSH para acceso desde las terminales Windows.
19. Instalación de los servicios para el acceso a través de Internet, desde fuera de la FI-UNER.
20. Documentación del proceso (versiones preliminares de este informe)

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

2 Balance de carga y cómputo paralelo

Como se mencionó antes, procesar un algoritmo o programa en paralelo significa que dicha aplicación se ejecuta en forma distribuida sobre un cierto número de nodos del cluster. Así, sobre cada nodo trabaja una parte del algoritmo completo y esto permite que diferentes procesos corran en forma simultánea en lugar de hacerlo secuencialmente (uno por vez) sobre un sólo procesador. Los procesos "distribuidos" o "paralelizados" de un algoritmo, en general, no son independientes entre sí; es decir que para completar ciertos procesos sobre un nodo del cluster se requiere información de otros procesos que corren sobre otros nodos. Esta información debe ser transmitida a través de la red que los conecta. Cuando un programador desarrolla una aplicación en paralelo, debe conocer la información que necesita cada proceso y además debe especificar cómo y cuándo se transmite entre los nodos que la requieran. Para facilitar la transmisión de la información entre los procesos, se han desarrollado herramientas especiales. Una de ellas es la "Interfaz de Paso de Mensajes" (MPI, del inglés *Message Passing Interface*) y la otra es conocida como Máquina Paralela Virtual (PVM, del inglés *Parallel Virtual Machine*).

Por otro lado, en muchos casos es útil que sea un sistema automático el que distribuya los procesos y aproveche al máximo las capacidades disponibles en el cluster de computadoras. Para esto se ha desarrollado el sistema MOSIX, que constituye una solución escalable para clusters basados en Linux. A partir de un conjunto de computadoras conectadas en red, los algoritmos de MOSIX comparten los recursos disponibles en forma adaptativa mediante un sistema de migración de procesos. Desde cualquier puesto de trabajo externo al cluster se puede ver una sola máquina, el nodo principal o maestro, y todos los procesos son ejecutados desde esta máquina. Sin embargo, internamente y de forma transparente para el usuario, los procesos serán migrados al nodo que más recursos disponibles posea. Dado que las demandas en cada nodo pueden variar en el tiempo, los procesos migran automáticamente de un nodo a otro para aprovechar el sistema al máximo. De forma alternativa, si fuera necesario, el usuario puede migrar "manualmente" sus procesos e incluso fijar la ejecución a un nodo en particular.

A continuación daremos algunos detalles más acerca de la instalación y configuración de los paquetes MOSIX, MPI y PVM, mencionando algunas pruebas realizadas y las ventajas de cada uno.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

2.1 Sistema MOSIX

El sistema MOSIX se ha desarrollado como una solución a nivel del mismo "núcleo" de Linux. De esta forma, cuando una máquina tiene instalado el sistema MOSIX es posible arrancar con el núcleo de Linux estándar[2] para no formar parte del cluster o con el núcleo MOSIX e integrarse al cluster. Cuando se arranca con el núcleo de MOSIX se preserva toda compatibilidad con Linux pero cada proceso puede migrarse automáticamente a cualquier parte del cluster, dependiendo de los recursos disponibles en cada nodo.

Se instaló la versión 0.97.10-2.2.18i586 del núcleo MOSIX, correspondiente a la versión 6.2 de Linux RedHat. Este sistema es de distribución gratuita, código fuente abierto y puede bajarse directamente desde Internet[3].

2.1.1 Tipos de cluster MOSIX

Existen dos modelos básicos para clusters MOSIX: el modelo multi-usuario con entorno de tiempo compartido y el modelo de partición dinámica del cluster en sub-clusters. Trataremos el primer caso que constituye nuestro principal interés ya que podemos encontrar las siguientes arquitecturas:

– *Server-pool*: en esta arquitectura todos los nodos son compartidos pero las estaciones de trabajo quedan fuera del cluster. Para configurar un server-pool es necesario incluir solamente las direcciones IP de los nodos en el archivo **`/etc/mosix.map`**. Debe realizarse una copia de este archivo en cada nodo. Mediante esta arquitectura los procesos nunca serán migrados a la máquina desde donde se realiza la conexión

(estación de trabajo). Esta es la arquitectura que se prefirió en nuestra implementación ya que se requieren muy pocos recursos en las estaciones de trabajo (se podría utilizar un 486 o Pentium). Todo el cómputo se realiza en el cluster y las estaciones de trabajo se utilizan para visualización de resultados y administración de procesos, es decir, se utilizan como simples terminales de ejecución y control. Desde fuera del cluster (desde las estaciones de trabajo) se puede acceder únicamente al nodo maestro (**shannon.fi.uner.edu.ar**). Ningún nodo interno al cluster puede ser accedido por los usuarios. Los procesos se ejecutan en el nodo maestro y MOSIX automáticamente los migra para aprovechar los recursos disponibles. Independientemente del nodo en que se esté corriendo, el control de proceso se realiza desde el nodo maestro y los resultados finales quedan nuevamente en el nodo maestro. De esta forma la existencia de un cluster es totalmente transparente al usuario. No es necesario saber cuantos nodos hay disponibles, cuales son sus características, donde están ubicados, ni muchos otros detalles que administra internamente el cluster.

- *Adaptive-pool*: en este caso los nodos son compartidos y las estaciones de trabajo pueden unirse o separarse del cluster según requiera el usuario o en forma automática, según horarios preestablecidos en la misma estación de trabajo. Aquí también se instala el mismo archivo **/etc/mosix.map** en todas las máquinas (incluyendo nodos y estaciones de trabajo[4]) y en este archivo se incluyen las direcciones IP de todas las máquinas. Esta arquitectura permite un mejor aprovechamiento de las estaciones de trabajo, en caso de que éstas posean relativamente buenas capacidades de cómputo. Para la configuración de incorporación automática al cluster se puede utilizar el servicio **crontab** y ejecutar los comandos **mosctl expel** y **mosctl expel** a las horas requeridas (por ejemplo, una estación de trabajo podría incorporarse al cluster de 22:00 a 08:00 hs. y durante el día servir para el uso particular). La principal razón que no nos permite implementar esta arquitectura actualmente es que la estructura de red hacia todas las estaciones de trabajo debe soportar, al menos, en ancho de banda de 100Mb/s sin pasar a través de hubs. Contando con una estructura de red más rápida, esta es una alternativa válida para implementar en un futuro.
- *Half-duplex pool*: en este caso los nodos son compartidos y las estaciones de trabajo pueden enviar procesos al cluster. En este caso la estación de trabajo es parte del cluster solamente para los procesos que se corren desde ésta y no recibe procesos de otras estaciones de trabajo. Para lograr esta configuración es necesario ejecutar el comando **mosctl expel** desde el script de inicio de MOSIX (**/etc/rc.d/init.d/mosix**) de la estación de trabajo.

2.1.2 Instalación y configuración

Para comenzar, todos los nodos del cluster deben estar conectados en una red de área local (LAN) que soporte el protocolo TCP/IP bajo Linux. La instalación se puede realizar directamente de los binarios comprimidos en un paquete RPM o descomprimir y recompilar el núcleo de MOSIX desde el archivo **tar.gz**. En nuestro caso, ya que trabajamos con un sistema RedHat, utilizamos los archivos:

kernel-mosix-0.97.10-2.2.18.i386.rpm
mosix-redhat-0.97.10-1.i386.rpm

y la utilidad **rpm** (opción **-i**), lo que simplificó la instalación. Sin embargo, esta instalación de los binarios precompilados no aprovecha todas las características de nuestro sistema (procesadores, memoria, tipo de red, etc.). En futuras reinstalaciones sería conveniente recompilar el núcleo MOSIX (a partir de los **tar.gz**) con las opciones más apropiadas a nuestro sistema.

La primera parte de la configuración consiste en definir la arquitectura del cluster. Esto se realiza editando el archivo **/etc/mosix.map**. En este archivo se especifican los nodos y sus direcciones IP. En caso de utilizar los nombres de cada nodo en lugar de su dirección IP, esta relación deberá definirse en **/etc/hosts**. Para esta primera configuración de mosix el archivo **/etc/mosix.map** quedó, tanto en el nodo principal como en el maestro, simplemente:

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

```
1 node1 2
```

lo que significa que a partir del nodo **1** (número de nodo en el cluster), que se encuentra en la dirección **node1** (la dirección IP está definido en **/etc/hosts**), hay **2** nodos en el rango de IP. Por ejemplo, si el primer nodo tienen la dirección IP 100.100.100.1, entonces buscará el segundo nodo en 100.100.100.2.

El demonio de MOSIX debe ser activado después que los demonios de red. De forma similar, MOSIX debe ser desactivado antes de desactivar los demonios de red. Para ver el estado actual de carga en cada nodo del cluster se puede ejecutar la utilidad **mon**. Esta utilidad muestra para cada nodo, entre otros, la carga de CPU y la memoria utilizada.

2.2 Bibliotecas MPI

En la actualidad MPI se está transformando rápidamente en un estándar para la comunicación de programas en paralelo. El paquete consiste de una serie de bibliotecas con funciones que el programador puede utilizar en el código de su aplicación y así transmitir la información entre las tareas distribuidas en el cluster. Además, provee el entorno y protocolos sobre el sistema operativo, para que el código paralelo se ejecute en forma consistente y ordenada (esto frecuentemente se conoce como *communicator* en la terminología MPI).

La complejidad subyacente en los algoritmos en paralelo determina que no siempre sea conveniente distribuir una tarea. Las aplicaciones de cálculo numérico (con cierta estructura algorítmica) donde se requiere cálculo intensivo pueden ser paralelizados en forma eficiente. Un ejemplo típico son los algoritmos de inversión de matrices o de solución de sistemas de ecuaciones algebraicos, presentes en una gran variedad de disciplinas. Este es el caso que compete aquí y por el cual fue necesario instalar y configurar el entorno MPI.

Existen muchas distribuciones e implementaciones de MPI. En nuestro caso se instaló MPICH (versión 1.2.1), desarrollada por un equipo del ANL. Este es un software de código abierto y distribución gratuita que está disponible en Internet[5]. Se decidió utilizarlo porque es totalmente compatible con el software PETSc, que se describirá más adelante.

Para la instalación de MPICH se procedió a bajar los archivos comprimidos desde la dirección de Internet detallada antes y luego de su descompresión, se ejecutó un archivo de comandos *configure* . Este archivo se genera automáticamente mediante la utilidad *autoconf* y genera los archivos de compilación (*makefile*) en función del compilador instalado (GNU C++ en este caso) y la arquitectura y procesador del nodo en cuestión. Luego se ejecuta la utilidad *make* que llama al compilador propiamente dicho y genera todos los archivos binarios de la biblioteca MPI. A continuación es necesario indicar a MPI la estructura del cluster sobre el que está corriendo. Esto se hacer modificando el archivo *util/machines/machines.LINUX* en el que se agregaron las siguientes líneas:

```
node1.cluster.fi.uner.edu.ar
node2.cluster.fi.uner.edu.ar
```

Finalmente se compilaron y probaron algunos programas en paralelo para verificar el correcto funcionamiento del paquete. En particular se compilaron y corrieron los ejemplos de **examples/basic** primero con 1 solo nodo y luego con los 2 del cluster a la vez.

Un requisito importante a tener en cuenta, es que el servicio NIS se encuentre instalado y funcionando correctamente. El sistema de paso de mensajes utiliza una conexión *rsh* para transmitir los datos a través de los nodos y la información de todos los usuarios debe estar disponible mediante este sistema de información en red tradicional en UNIX (puede ver más detalles de su instalación y configuración en la Sección 4.2.6).

La documentación incluida con MPICH es completa y clara, por lo que la instalación es relativamente sencilla. Esta documentación refiere solamente a comandos específicos y algunas herramientas propias de esta distribución. Para una introducción y comprensión más completa de los conceptos detrás de MPI, el lector debería referirse a **Gropp** y cols 1999.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

2.3 PVM

El sistema PVM permite que una colección heterogénea de computadoras puedan se utilizadas como un sistema coherente y flexible para el cómputo paralelo. Este sistema está basado en un conjunto de bibliotecas que deben ser encadenadas desde el código fuente (*libpvm3.a* y *libfpvm3.a*) y un demonio (*pvmd3*) que se encarga de la comunicación y control de los procesos entre los nodos.

En nuestro cluster se instaló la versión 3.4.3-4. Este paquete fue desarrollado por varias empresas y Universidades:

- University of Tennessee, Knoxville TN.
- Oak Ridge National Laboratory, Oak Ridge TN.
- Emory University, Atlanta GA.
- Carnegie Mellon University, Pittsburgh PA
- Pittsburgh Supercomputer Center, Pittsburgh PA

y puede ser descargado gratuitamente de Internet[6] o bien directamente desde el CD de instalacción de Linux-RedHat. Para la instalación se puede recurrir inicialmente a un paquete RPM o descompactar el archivo **`tar.gz`**. En cualquier caso, antes de la compilación de las bibliotecas, demonios y ejemplos es necesario configurar la variable que indica la posición del directorio raíz, en nuestro caso mediante C-Shell:

```
setenv PVM_ROOT /usr/share/pvm3
```

luego puede incluirse esta variable de entorno en la configuración de cada usuario. A continuación se utiliza la utilidad *aimk* para la configuración y compilación definitiva.

Para verificar la instalación se pueden utilizar los ejemplos del directorio **`$PVM_ROOT/gexamples`**. Estos ejemplos se compilan desde este directorio mediante **`../lib/aimk`**. Los ejecutables se ubican automáticamente en el directorio **`$PVM_ROOT/bin/LINUX`** y pueden probarse también desde este directorio. Previamente es necesario iniciar el demonio de PVM:

```
/etc/rc.d/init.d/pvmd start
```

Para agregar un nodo al sistema se puede utilizar el comando *add* desde dentro del entorno **`pvm`** o utilizar la intefaz gráfica **`xpvm`**.

3 Programas para aplicaciones específicas

En esta sección se describirán muy brevemente las aplicaciones instaladas actualmente y su propósito. Muchas de ellas son de utilización estándar en Linux/UNIX.

3.1 Lenguajes de programación

3.1.1 Perl

Perl es un lenguaje para escritura de scripts (guiones) optimizado para explorar archivos de texto arbitrarios, extraer información de esos archivos e imprimir reportes basados en esa información. De esta forma puede usarse como el “pegamento” que une programas y resultados obtenidos de distintas fuentes, y de ahí su utilidad para automatizar procesos y experimentos. Es también un buen lenguaje para varias tareas de administración de sistemas. Esta pensado para ser práctico (fácil de usar, eficiente, completo) en lugar de bonito (pequeño, elegante, mínimo). Perl combina algunas de las mejores características de C, sed, awk, y sh, de manera que las personas familiarizadas con estos lenguajes deberían tener pocas dificultades con él. La sintaxis de las expresiones corresponden muy cercanamente a la de las expresiones del C.

3.1.2 GNU C/C++ Fortran

Lenguaje de programación muy difundido en todos el mundo, con compiladores múltiples sistemas y plataformas. Hoy en día se lo puede tomar como un lenguaje estándar. El compilador GNU C++ es el más utilizado en el mundo. El C++, con Fortran, son los lenguajes

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

recomendados para el desarrollo de aplicaciones en paralelo.

3.1.3 Java jre

El Java Runtime Environment contiene la máquina virtual de Java, las bibliotecas para la ejecución y el programa que inicia las aplicaciones Java que son necesarios para ejecutar cualquier programa en Java.

Java es un lenguaje de programación que es independiente de la plataforma utilizada. Este paquete no tiene las herramientas para la compilación y la depuración.

3.2 Herramientas de programación

3.2.1 GNU AWK

Esta herramienta es una utilidad para el procesamiento de texto.

3.2.2 GNU Make

Herramienta para controlar la generación de ejecutables y otros archivos binarios de un programa desde los archivos fuentes del programador. Make también permite a los usuarios generar e instalar paquetes sin que tengan conocimiento acerca de los detalles del proceso de generación. La generación es realizada según las configuraciones del archivo Makefile.

3.2.3 GNU gdb y XXGdb

Gdb es un depurado, permite seguir la ejecución de los programas y examinar el estado interno en cualquier momento. Gdb trabaja con programas C y C++ compilados con GNU C. Xxgbd es el gdb con entorno gráfico.

3.2.4 GNU automake y autoconf

Automake es un generador de Makefile. Autoconf es una herramienta para la configuración de código fuente y Makefiles. Se lo utiliza para que los programadores puedan crear paquetes configurables y portables.

3.2.5 CVS

Herramienta para archivar y coordinar el desarrollo de proyectos (códigos fuentes).

3.3 Bibliotecas y otras herramientas

3.3.1 PETSc

PETSc (Portable and Extensible Toolkit for Scientist Calculations) comprende un conjunto de bibliotecas específicamente diseñadas para la asistencia en el desarrollo de programas en paralelo y está fuertemente orientado a la solución numérica de sistemas de ecuaciones diferenciales parciales, mediante su discretización. Para ello, incluye funciones que permiten:

- Crear vectores y matrices distribuidos (paralelos), con asignación dinámica de memoria.
- Realizar operaciones básicas predefinidas entre matrices y vectores.
- Ejecutar algoritmos para la solución de sistemas de ecuaciones algebraicas distribuidas.
- Utilizar otras herramientas específicas.

El aspecto más destacable de este software es que maneja en forma automática (y transparente para el usuario) la comunicación de paso de mensajes o MPI entre los procesos distribuidos. Por lo tanto, una vez que el programador toma algunas decisiones básicas con respecto a la parelización de su programa, puede concentrarse en aspectos realmente específicos de la aplicación, dejando a PETSc la administración de la compleja tarea de comunicación entre procesos. Así, incluso usuarios con poca o nula experiencia en

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

programación MPI pueden desarrollar en poco tiempo programas en paralelo que mejoran la eficiencia numérica de sus aplicaciones.

PETSc es también un software de código abierto y distribución gratuita, desarrollado por integrantes del "Argonne National Laboratory" (ANL), y se puede obtener de Internet[7]. No incluye internamente las funciones MPI, pero es completamente compatible con la distribución MPICH (ver Sección 2.2). El entorno MPI debe estar instalado antes de la puesta en marcha de PETSc. Cabe mencionar que la mayor parte de las operaciones de cálculo con matrices y vectores realizadas en PETSc, utilizan el muy optimizado paquete numérico de álgebra lineal LAPACK[8]. Este se puede obtener conjuntamente con PETSc y su instalación es muy sencilla.

Con MPICH y LAPACK instalados, se descomprime el paquete PETSc, y antes de comenzar la compilación se debe establecer el valor para algunas variable de entorno. En nuestro caso, basados en un C-Shell, configuramos[9]:

```
setenv PETSC_DIR /opt/petsc/petsc-2.1.0
setenv PETSC_ARCH linux
```

A continuación se editó el archivo:

```
${PETSC_DIR}/bmake/${PETSC_ARCH}/base.site
```

donde se configuran las ubicaciones de los paquetes MPI y LAPACK. Para esto se modificaron las siguientes variables[10]:

```
BLASLAPACK_LIB = -L/opt/lapack/fblaslapack -lflapack -lfblas
MPI_HOME       = /opt/mpi/mpich-1.2.1
```

Finalmente se realiza la compilación indicando las opciones de optimización, en nuestro caso utilizamos las de máxima optimización y sin información para depuración:

```
make BOPT=O all >& make_log
```

Las bibliotecas construidas durante la compilación contienen todas las funciones que ofrece PETSc y las mismas se pueden llamar desde otros programas de aplicación. Puesto que PETSc fue implementado en lenguaje C, se recomienda su utilización. Sin embargo, también es compatible con Fortran, otro popular lenguaje de programación para aplicaciones de cómputo intensivo.

Para verificar el normal funcionamiento de PETSc, se probaron algunos ejemplos y benchmarks desde:

```
src\benchmarks
src\sles\examples\test
```

mediante una directiva de compilación de la forma:

```
make BOPT=O nombre
```

3.3.2 MATLAB

MATLAB es un lenguaje para computación técnica de gran versatilidad y facilidad de uso. Integra cálculo, visualización y programación en una interfase amigable. Las aplicaciones típicas incluyen modelado, simulación, análisis de datos y generación de gráficos científicos. El lenguaje permite la manipulación a alto nivel de matrices y arreglos, junto con comandos de control de flujo, funciones, estructuras de datos, entrada/salida y características de programación orientada a objetos. Existen gran cantidad de colecciones de programas o funciones, bibliotecas o toolboxes que implementan algoritmos de procesamiento de señales, estadística, redes neuronales, etc.

3.3.3 GNU Plot

Gnuplot es un programa interactivo manejado por comandos para gráficos y dibujos técnicos y científicos. Permite cargar archivos de datos, crear ejes, etiquetas y graficar varias funciones juntas en una, dos o tres dimensiones. Incorpora funciones y operadores de C, y también algunos propios. Permite la definición de funciones propias del usuario.

3.3.4 HTK

HTK es un conjunto de herramientas (toolkit) para construir sistemas basados en Modelos Ocultos de Markov (Hidden Markov Models o HMMs). Los HMMs pueden usarse para modelar series temporales en general, por lo que HTK es en principio de propósito general. Sin

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

embargo, HTK está diseñado principalmente para construir sistemas de procesamiento del habla basados en HMM, en particular reconocedores. Por ello gran parte de la estructura de soporte en HTK está dedicada a esta tarea. Para esto se requieren dos etapas principales de procesamiento. Primero, las herramientas de entrenamiento de HTK se utilizan para estimar los parámetros de un conjunto de HMMs utilizando emisiones de entrenamiento y sus correspondientes transcripciones. Segundo, las emisiones desconocidas puede ser transcriptas utilizando las herramientas de reconocimiento de HTK. Para un mayor detalle consultar **Young** y cols 1999[11].

3.3.5 SNNS

Stutgart Neural Networks Simulator (SNNS) es un paquete muy completo p ara simulación de redes neuronales. Puede utilizarse desde línea de comando o mediante una interfase X-Windows. Posee las arquitecturas y algoritmos de aprendizaje más utilizados tantos para redes estáticas como dinámicas. Permite su utilización en modo paralelo[12].

3.4 Procesamiento y visualización de documentos

LaTeX es un lenguaje para escritura de documentos técnicos o científicos. El lenguaje esta descripto en detalle en el libro de su creador **Lamport** de 1994 . LaTeX es un paquete de macros para TeX, no una modificación del programa fuente de TeX. De esta forma posee también todas las capacidades de este lenguaje de preparación de documentos creado originalmente por Donald E. **Knuth** en 1984. LaTeX alienta a los escritores a pensar en el contenido y no en la forma de sus documentos. Esto se logra mediante una estructura jerárquica de comandos. También provee comandos para escritura de fórmulas y ecuaciones matemáticas, tablas, figuras, etc. Herramientas adicionales permiten el manejo de referencias y bibliografía, indexado, corrección ortográfica, etc. Permite salida en diferentes formatos estándar como Postcript o Acrobat PDF. El presente documento fue realizado con LaTeX.

3.4.1 Visualización PS y PDF

Para la visualización de archivos de documentos PS y PDF como los generados por LaTeX se instalaron los paquetes GhostScript y Xpdf.

3.4.2 Emacs

Emacs es un editor/interfaz de desarrollo muy potente y difundido entre programadores y usuarios Unix. Permite manipulación de múltiples documentos, un sistema de macros muy completo y posibilidad de integración con compiladores y depuradores.

3.4.3 Netscape

Navegador con capacidad de funcionar como agente de correo electrónico y servicios de news.

3.4.4 Open Office

Paquete de oficina compatible con la distribución Microsoft Office (versión 2000 y XP). Este paquete es de distribución gratuita y de código abierto. Contiene procesador de texto, planilla de cálculo, generador de presentaciones, editor de HTML, editor de fórmulas y editor de dibujos.

4 Configuración y administración

En esta sección se presentarán brevemente los pasos realizados para la configuración de todos los aspectos hardware y software del cluster (algunos de los cuales son propios de cualquier sistema Linux). También se describirán algunas herramientas para la administración del cluster. Se debe destacar que esta sección está dirigida a aquellos que quieran reproducir

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

nuestra experiencia de instalación y configuración. Existen muchas posibles configuraciones alternativas que no se han explorado, y por lo tanto no se describen en este documento. Sin embargo con este primer esquema cualquier cambio debería ser directo para aquellos usuarios con experiencia previa en manejo y administración de sistemas Linux/UNIX. Por lo anterior en esta sección se supone que el lector posee conocimientos previos relativos a este tema.

4.1 Configuración hardware actual

El hardware puede dividirse en las siguientes dos secciones.

4.1.1 Unidades de procesamiento (CPU)

Hasta la fecha se cuenta con 6 equipos similares con las siguientes características:

- Procesador Pentium III 700 Mhz.
- Memoria RAM de 256 MB.
- Disco rígido de 20 GB.
- Placa de Red de 10/100Mbits Full duplex (el nodo maestro tiene una placa de red adicional para conectarse con la red de la FI-UNER).
- Placa de Video AGP de 8MB.
- Monitor, CDROM, teclado, y ratón (solo necesario en el nodo maestro).

4.1.2 Arquitectura de red

Las primeras pruebas se realizaron con una configuración de dos máquinas conectadas a través de un cable de red par trenzado cruzado. Más recientemente, el laboratorio de Biomecánica Computacional adquirió a través del PID#6057 de la UNER, un swtich 10/100Mbit para conectar todos los nodos del cluster, como muestra la Figura 1[dd1].

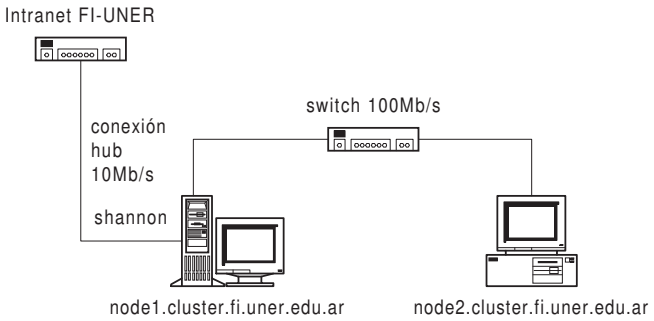

Figura 1: Configuración inicial del cluster, luego de incorporar el switch.

4.2 Configuración de servicios de red

4.2.1 Controladores de las placas de red

En **todos** los nodos del cluster se deben agregar al archivo **`/etc/hosts`** las direcciones IP y los nombres de todos los nodos del cluster. Por ejemplo, para tres nodos:

```
10.0.2.1 node1
10.0.2.2 node2
10.0.2.3 node3
```

A continuación se detallan las configuraciones particulares para el nodo maestro y los nodos secundarios.

4.2.1.A Nodo maestro

Es altamente recomendable que la placa mapeada a través de la interfaz **`eth0`** se utilice para la comunicación con los nodos del cluster y la **`eth1`** para la comunicación con el exterior.

Para la configuración de **`eth1`** hay que pedir la información al administrador de la red

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

local. Se necesita saber si se utilizará una asignación dinámica de dirección IP (DHCP) o una dirección IP asignada manualmente. En el primer caso se configura el archivo:

/etc/sysconfig/network-scripts/ifconfig-eth1

modificando la variable **BOOTPROTO="dhcp"**. Para el segundo caso se necesitan los datos de: dirección IP, máscara de red, gateway y servidores de DNS. Se puede utilizar LinuxConf (Sección 4.3.1), Webmin (Sección 4.3.2) o editar directamente el archivo antes mencionado.

A **eth0** se le puede asignar la primera IP disponible del rango de direcciones IP privadas elegidas. Luego debe incorporarse toda la información de configuración en el archivo:

/etc/sysconfig/network-scripts/ifconfig-eth0.

Como primera medida de seguridad no se debe activar el ruteo en este equipo, ya que no se desea que los usuarios puedan tener acceso.

4.2.1.B Nodos secundarios

Los nodos secundarios deben configurarse de modo que la dirección IP se obtenga automáticamente del nodo maestro (por DHCP). Se debe tomar nota de la dirección hardware de la placa de red. Esto se pude hacer a través del comando **ifconfig** (HWaddr). Esta dirección servirá luego para configurar el servidor de DHCP (ver Sección 4.2.4). En el archivo **/etc/sysconfig/network-scripts/ifconfig-eth0** queda la configuración para que la dirección IP y los restantes parámentros se configuren por DHCP al inicial el equipo.

4.2.2 Servicios de internet (INET)

El demonio **inetd** inicializa a otros servicios de red. No se detallará su instalación y configuración dada la extensión prevista para este artículo. A partir de este demonio se activan los servicios detallados en el archivo **/etc/inetd.conf**. Obsérvese que el demonio de ejecución se debe iniciar a través del comando **mosrun** de MOSIX. Se deben todos el acceso completo para la red interna en el archivo **/etc/hosts.allow**. Para que usuarios de otras redes no puedan acceder a los servicios del nodo maestro del cluster se debe agregar el comando **ALL : ALL** en el archivo **/etc/hosts.deny**.

4.2.3 Transferencia de archivos (FTP)

Este servicio da soporte al intercambio de archivos entre dos nodos dentro del cluster, por medio del protocolo FTP. Este servicio no está habilitado hacia afuera del cluster ya que es un método poco seguro para la comunicación. Para la transferencia de archivos desde la red de la FI-UNER se puede utilizar el servidor Samba (ver Sección 4.2.8) y para transferencia en Internet el protocolo FTP seguro (ver Sección 4.4.1).

4.2.4 Servidior dinámico de IP (DHCP)

El demonio **dhcpd**, establece automáticamente la configuración de red de los nodos secundarios, a partir de la información que hay en el nodo maestro.

El valor de *hardware ethernet* se obtiene cuando se instala la placa de red o utilizando el comando **ifconfig**. Se le ha asignado una dirección IP estática a cada nodo a través de la configuración *fixed-address*. La dirección se resuelve a través del servidor de nombres (ver Sección 4.2.7). El archivo de configuración es **/etc/dhcpd.conf**.

4.2.5 Sistema de archivos en red (NFS)

Este servicio provee acceso al sistema de archivos de Linux/UNIX a través de la red. Para la configuración se puede utilizar Webmin para configurarlo o directamente editar el archivo **/etc/exports**.

4.2.6 Sistema de información de red (NIS)

Este servicio proporciona toda la información necesaria (el nombre de login, contraseñas, directorios de la casa (home), grupos, etc.) para que los usuarios puedan conectarse en cualquiera de los nodos de la red sin tener una copia de los archivos en cada nodo. Mediante el

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

NIS se puede habilitar la conexión de usuarios al en cualquier máquina en la red, siempre y que tenga el programa cliente y la contraseña del usuario esté grabada en la base de datos de claves del NIS.

Es importante aclarar que, por razones de seguridad, el servicio de NIS está habilitado únicamente sobre la red interna al cluster. Este servidor requiere del servicio llamadas a procedimientos remotos (RPC, del inglés *Remote Procedure Calls*), que se configura automáticamente durante la instalación de linux.

Para la configuración del servidor NIS se edita **`/etc/ypserv.conf`** y para el cliente **`/etc/yp.conf`**.

4.2.7 Servidor de nombres de dominio (DNS)

El DNS es un sistema jerárquico de denominaciones basadas en textos separados por puntos (por ejemplo **`master.cluster.fi.uner.edu.ar`**). Mediante estos textos se crea un nombre que puede ser traducido en una dirección de IP. La configuración se realiza a través de los archivos **`/etc/named.conf`** y **`/var/named/cluster.db`**. También es posible utilizar el programa Webmin para configurar este servicio.

4.2.8 Servidor de archivos para Windows (Samba)

El sistema Samba provee una emulación de un servidor Windows NT, “Lan Manager” o SMB, que permite tener acceso a los archivos del nodo maestro directamente desde clientes tales como Windows 95, Windows 98, Windows Nt, Windows 2000, Windows Me, etc. Mediante este servidor es posible ver los directorios de Linux como simples carpetas en Windows. Para la configuración es posible utilizar la interfaz Webmin o también se puede realizar editando el archivo **`/etc/smb.conf`**. Cabe destacar que se colocó al servidor de la FI-UNER para validar los usuarios de este servicio.

4.2.9 Sincronización de relojes (NTP)

El demonio **`ntpd`** se encarga de sincronizar la hora entre todos los nodos del cluster. La configuración es sencilla y no se darán más detalles en esta sección.

4.2.10 Servidor Web (Apache)

El Apache Web Server se encarga de proveer el acceso a páginas web que se alojen dentro del cluster, a través del demonio **`httpd`**. Actualmente no se ha activado este servicio. En un futuro está planeado incluir este artículo y otras informaciones y utilidades en una página web alojada en el nodo maestro del cluster.

4.3 Administración del sistema

En esta sección se tratan brevemente tres herramientás básicas para la administración del sistema. Estas herramientas no son particulares de un sistema en cluster sino que son de uso general en sistemas Linux. Por esta razón no se darán mayores detalles acerca de su instalación y configuración.

4.3.1 Configuración avanzada de Linux (LinuxConf)

Esta herramienta es ampliamente utilizada para la configuración y administración del sistema Linux. No se trata de una herramienta común a los sistemas UNIX y en particular ha sido desarrollada para las distribuciones Linux RedHat. Linuxconf funciona tanto en las terminales de texto como en un entorno de ventanas X. Mediante una estructura en árbol y una interfaz relativamente sencilla, Linuxconf provee acceso a prácticamente todos los archivos de configuración y permite activar o desactivar los diferentes demonios del sistema.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

4.3.2 Administración a través de la web (Webmin)

Webmin es una poderosa interfaz Web de administración para sistemas UNIX. Con esta herramienta se pueden configurar muchos servicios y recursos, como por ejemplo los servidores de DNS, Samba y NFS, los sistema de archivos locales o remotos, el manejo de usuarios y grupos, etc.

Para poder tener acceso *seguro* a Webmin se deben instalar previamente los paquetes de *Secure Sockets Layer*, *Transport Layer Security* y el módulo *Perl Net::SSLeay* (ver Sección 4.4.1). El sistema Webmin queda accesible mediante HTTP *seguro* (https) solamente desde la intranet de la FI-UNER. Esto es configurable a través del mismo Webmin. Este servicio es independiente del servidor de páginas web.

4.3.3 Sistema de colas de trabajo (OpenPBS)

OpenPBS (del inglés *Portable Batch System*) es una versión para múltiples plataformas del conocido sistema para la administración de colas de trabajo *NQS*. Este sistema es de distribución gratuita, con código abierto y puede bajarse de Internet[13].

Los sistemas de administración de colas de trabajo permiten enviar trabajos al cluster y que estos se ejecuten de acuerdo a los recursos asignados a cada usuario o grupo de usuarios. Es posible de esta forma asignar a cada grupo de usuarios una cantidad de recursos y así administrar la ejecución de los trabajos de muchos usuarios. Además, el sistema PBS puede recuperar trabajos interrumpidos por fallas de hardware o cortes en la alimentación. Para cada grupo de usuarios o para cada usuario individual se pueden definir, entre otros: la lista de computadoras desde donde puede enviar trabajos, los nodos a los que puede asignar trabajos, la máxima cantidad de horas de CPU, la máxima cantidad de memoria (física o virtual, por proceso o en total), el máximo tamaño por archivo creado.

4.4 Acceso e interfaces de usuario

4.4.1 Servidor de acceso seguro (SSH)

El Secure Shell Server ofrece varias herramientas para el acceso remoto mediante comunicaciones encriptadas. Previo a la instalación de SSH propiamente dicho se instalaron los siguientes paquetes:

- OpenSSL: se traata de una biblioteca OpenSource que implementa los protocolos Secure Sockets Layer (SSL v2/v3) y Transport Layer Security (TLS v1) para criptografía de propósito general.
- SSLeay: este módulo en lenguaje Perl (Net::SSLeay) permite utilizar las bibliotecas de criptografía OpenSSL.
- Bibliotecas de compresión y descompresión (zlib-1.1.3-6)

Luego de instalar estos paquetes se procedió a instalar el SSH propiamente dicho. Para que MOSIX distribuya los procesos entre los nodos cuando los usuarios se conecten en forma remota hay que *forzar* a que el demonio **`sshd`** se ejecute en cualquiera de los nodos mediante **`mosrun`**. Esto se hace en el script de inicio de **`sshd`** en **`/etc/rc.d/init.d/sshd`** y modificando la línea que contiene **`sshd $OPTIONS`** por **`/bin/mosrun -l sshd`**. Hay que tener en cuenta que si **`$OPTIONS`** no es nulo, se lo puede cambiar por: **`/bin/mosrun -l -z sshd $OPTIONS`**

4.4.2 Clientes para el acceso remoto seguro

- PuTTY: este es un muy buen cliente de Telnet y SSH para Windows. Consta de un archivo ejecutable de sólo 220 kB y no requiere instalación. Es de distribución gratuita, de código abierto y se puede bajar directamente de Internet[14].
- OpenSSH: otro cliente más sofisticado que incorpora también una interfaz visual para transferencia de archivos (sFTP). Este producto comercial puede utilizarse libremente en instituciones académicas y se distribuye para una gran variedad de sistemas operativos[15].

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

4.4.3 Interfaces de texto (shells)

Cada usuario puede elegir diferentes "shells" de trabajo. Sin bien existe un conjunto común de comandos del sistema (como *cp*, para copiar), cada una de estas interfaces ofrece diferentes alternativas y comodidades de uso. Luego de que el usuario se conecte mediante SSH, una de estas interfaces se ejecuta según se haya configurado en el archivo **`/etc/password`**.

4.4.4 Interfaces gráficas (X-Windows)

El sistema de ventanas X provee la tecnología base para el desarrollo de interfaces gráficas. X-Windows soporta aplicaciones remotas, o sea aplicaciones X ejecutándose en otras computadoras mientras se visualiza en su máquina. Esto se puede lograr mediante una filosofía cliente-servidor, donde el cliente puede ser local o remoto. A continuación se describen algunos servidores y clientes de X-Windows:

1. Servidor XFree86: este es el servidor estándar que se distribuye gratuitamente con Linux. La configuración, común a cualquier Linux RedHat, se puede realizar directamente en el archivo **`/usr/X11R6/lib/X11/XF86Config`** o utilizar el programa **`xf86config`**.
2. Servidor para acceso remoto X-VNC: este servidor se basa en un Servidor X y en un entorno de ventanas predefinido por cada usuario. Luego, mediante un cliente VNC se puede acceder desde una máquina remota y ver en la pantalla lo mismo que si se estuviese sentado en el servidor. Este es un programa gratuito (tanto el servidor como el cliente) y puede bajarse de Internet[16].
3. Gestores de ventanas X: los gestores de ventanas X proveen una interfaz visual más depurada y con un aspecto bien terminado. Para Linux, los gestores más conocidos son KDE y Gnome. La instalación se realiza durante la misma instalación de Linux y requieren que se haya configurado correctamente el Servidor X.
4. Clientes X: existen muchos clientes X para diversos sistemas operativos. En el caso del cluster será muy frecuente que los clientes X realicen un acceso remoto al nodo maestro y, quizás también, desde un sistema operativo Windows. Es por esto que a continuación destacamos algunos clientes X para acceso remoto desde Windows:
   - VNC: el cliente VNC tiene la gran ventaja de ser muy pequeña su implementación (200kB).
   - X-Win32: este es un programa comercial que puede utilizarse gratuitamente con la restricción de que todas las ventanas se cierran cada 2 hs.
   - Exceed: provee una buena integración entre Windows y el gestor de ventanas (KDE o Gnome). Este programa comercial provee también un amplio conjunto de herramientas para la comunicación remota con sistemas Linux/UNIX.

5 Conclusiones y trabajos futuros

En este artículo describimos nuestra experiencia en la implementación de un cluster para cómputo paralelo. Actualmente se encuentran finalizadas la etapas de construcción del prototipo y realización de pruebas piloto, así como la utilización del cluster en numerosos experimentos reales. El prototipo, como se describió antes, consiste en seis Pentium III conectados a través de un switch, con sistema operativo Linux y núcleo MOSIX (ver Figura 2). A continuación presentamos un resumen del estado actual:

- Se completó toda la configuración necesaria para el cómputo paralelo y el balance automático de carga.
- Se realizó la configuración inicial de la red interna, los usuarios y grupos de trabajo.
- Se instalaron los paquetes de utilidad general.
- Se probaron ejemplos simples para cada paquete instalado.
- Se realizaron numerosos pruebas con software de cómputo masivo.
- Se ajustó y corrigió el funcionamiento general.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

Este desarrollo constituye un punto de partida para construir una herramienta de uso general, que puede extenderse fuera del contexto del Laboratorio de Cibernética.

5.1 Experiencias como usuarios del cluster

5.1.1 Grupo de Reconocimiento Automático del Habla

Se ha hecho un uso intensivo del cluster para el entrenamiento y prueba de sistemas de reconocimiento automático del habla y los beneficios fueron claros (aún con solamente dos nodos). Se llevaron a cabo del orden de 50 experimentos completos, con un total de unas 1400 horas de cómputo. Por otro lado se volvieron a utilizar máquinas más viejas que ahora sirven como terminales, dejando las máquinas más potentes en el cluster.

Inicialmente se había considerado la compra de una placa de procesamiento digital de señales (DSP), una computadora con múltiples procesadores o con tecnología RISC. A partir de nuestra experiencia y las de otros grupos conocidos, concluimos que la relación costo-beneficio favorece ampliamente a los clusters de PCs.

En el grupo de usuarios que trabajó en el cluster se observó cierto rechazo inicial a la metodología de trabajo, ya que no está tan orientada a las interfaces visuales como el sistema operativo Windows. Esto se debió pricipalmente a que muchos de ellos no habían trabajado en sistemas Linux. Sin embargo, venciendo esta resistencia inicial, los beneficios de nueva metodología de trabajo fueron notables. Las herramientas desarrolladas por investigadores y para investigadores siempre se basaron en sistemas UNIX. Desde Linux se aprovechan directamente todas estas herramientas y se agiliza notablemente el trabajo. Se debe destacar también que existen excelentes interfaces visuales para Linux que pueden accederse a través de cualquier cliente X instalado sobre una máquina con Windows, o directamente desde terminales Linux.

En el contexto del proyecto “Sistema de reconocimiento automático del habla” se pudieron comprobar las ventajas que provee el sistema en cluster. Resumiendo, podemos decir que estas ventajas se tradujeron principalmente en:

- un mejor aprovechamiento de los recursos del laboratorio,
- mayor capacidad de cómputo,
- facilidades para el seguimiento y documentación de los experimentos,
- seguridad y accesibilidad a los datos,
- seguimiento de los procesos por internet desde máquinas externas a la Facultad.

5.1.2 Grupo de Biomecánica Computacional

Actualmente, el Grupo Biomecánica Computacional de la FI-UNER está trabajando con códigos Fortran que implementan el Método de Elementos Finitos (MEF). Este método numérico requiere la construcción, ensamble y solución de sistemas matriciales no-lineales de ecuaciones, miles de veces durante la ejecución del programa. Por lo expresado anteriormente, estos programas son muy apropiados para su paralelización. Integrantes del grupo de Biomecánica Computacional están modificando sus códigos numéricos para que utilicen las funciones provistas en el paquete PETSc. Las pruebas iniciales han sido muy satisfactorias y se espera lograr la paralelización de los programas en un corto plazo.

5.2 Instalación de nuevos paquetes y actualizaciones

Como trabajos futuros quedan pendientes las siguientes tareas de instalación y actualización (entre otras):

- Instalar la nueva versión de Linux RedHat y el correspondiente núcleo de MOSIX.
- Instalar utilidades específicas para la administración del cluster. Por ejemplo: mosixview 0.8 y qps 1.9.7-5.
- Completar la configuración de paquetes de utilidad general y programación: StarOffice, Lyx, Emacs, Numerical Recipes in C, Evolutionary Object, ToFy, etc.

- Implementar un sistema de instalación automática de nuevos nodos.
- Configurar un sistema de backups automáticos en cinta.
- Investigar la utilización de nodos sin disco rígido (diskless).
- Investigar la configuración de un nodo master "espejo" para ser utilizado como backup en caso de fallas de hardware.
- Investigar la utilización de paquetes para memoria global compartida.
- Investigar la utilización de paquetes para almacenamiento global compartido (superdiscos) y/o la configuración de un servidor de archivos.
- Generar listas de correo electrónico para los diferentes grupos de usuarios y administradores.
- Ajustar toda la configuración de seguridad del sistema para minimizar la posibilidad de recibir accesos no permitidos y acciones maliciosas. En particular el sistema Linux es muy seguro, ya desde su configuración de instalación estándar.
- Instalar un nodo pequeño como servidor de impresión.
- Adecuar un lugar físico para instalar toda la estructura del cluster.
- Desarrollar una interfaz web (más amigable) para ver el estado de carga del cluster, enviar trabajos y administrar las colas de trabajo.
- Desarrollo de una página web para el cluster.
- Investigar la posibilidad de mejorar la estructura de red de la FI-UNER para que permita implementar la arquitectura *adaptive-pool* (ver Sección 2.1).

5.3 Estructura prevista para el crecimiento ordenado

Para una estructura con pocos nodos se puede realizar una interconexión "todos con todos", en donde cada nodo está conectado directamente a todos los demás. Sin embargo, este tipo de conexionado requiere que cada máquina tenga un número de placas de red igual a la cantidad de máquinas a la que se deba conectar. En un caso de 4 nodos, cada uno necesita 3 placas de red. En nuestras pruebas iniciales se utilizó esta arquitectura con 2 nodos. Como se puede ver fácilmente, si bien es una alternativa barata para pocos, esta estructura no permite aumentar mucho más la cantidad de nodos en el cluster.

La siguiente alternativa (ver Figura 2) es la que se tiene implementada en estos momentos y consta de una conexión a través de switch. Además, en el nodo maestro hay dos placas de red de forma de poder separar la red de la FI-UNER de la red interna – también conocida como red privada – del cluster. Así los usuarios del cluster ven desde afuera una única computadora (el nodo maestro) y la estructura interna del cluster queda oculta. En esta estructura es importante destacar el sistema de backup (por ejemplo en cintas magnéticas) y el de almacenamiento de información.

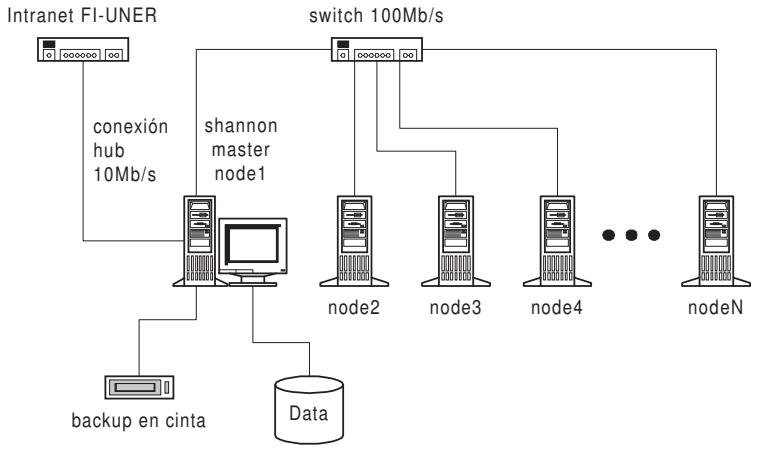

Figura 2: Configuración actual. Se incluye también un sistema de backup en el nodo maestro.

La velocidad de acceso a través de la red sigue siendo el punto más crítico de la arquitectura de cluster y de la optimización de la transmisión de datos depende en gran medida el desempeño del sistema en su conjunto. En la Figura 3 se muestra una arquitectura con un servidor de archivos dedicado, dos switches y dos placas por cada nodo. Con esta configuración es posible centralizar la mayor parte de los datos (e incluso trabajar con nodos diskless), lo que reduce los costos totales y aumenta la velocidad de acceso.

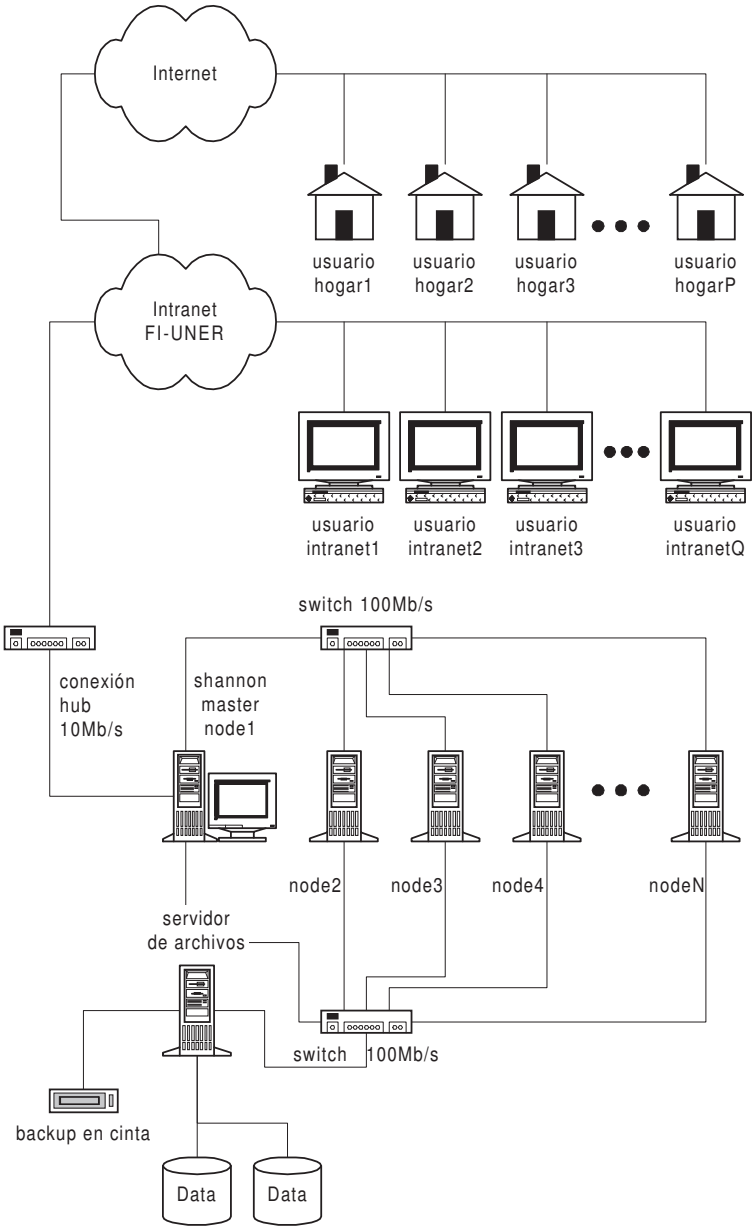

Figura 3: Configuración con sistema de backup y servidor de archivos dedicado.

5.4 Resumen de los recursos necesarios para crecer

A continuación se describen los requerimientos para que el sistema de cluster crezca y pueda brindar servicios a un número mayor de usuarios o proyectos de investigación. Esta lista no es exhaustiva y pretende dar una idea de magnitud acerca de los recursos necesarios para la construcción de un cluster de mayores dimensiones.

- *Nodos PC*: los beneficios del cluster serán proporcionales a la cantidad y capacidad de cómputo de los nodos que se incorporen. Con el switch que ya poseemos se pueden incorporar hasta 15 nodos utilizando la arquitectura de la Figura 2. En esta estructura

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.

podrían utilizarse nodos PC con procesadores Pentium III o superior. Inicialmente se pueden utilizar nodos menos potentes pero el rendimiento en ciertas aplicaciones se puede ver perjudicado por la utilización de un conjunto muy heterogéneo.

- *Nodo maestro*: el nodo maestro, además de ser un nodo más disponible para el cómputo, debe administrar todo el sistema de usuarios, conexiones, distribución de la carga, etc. Por estas razones siempre es conveniente dotar al nodo maestro de mayores capacidades de procesamiento y memoria RAM.
- *Lugar físico*: se debe contar con el espacio suficiente para alojar, al menos, 15 nodos PC (sin monitor), 1 nodo maestro y un servidor de archivos y backup. También es necesario contar con algún mueble, gabinete o dispositivo que permita ubicar todo el conjunto de manera ordenada y de fácil acceso (lo ideal consistiría en un "rack").
- *Acondicionador de aire*: el ambiente debe estar a temperatura constante no superior a 20 grados centígrados.
- *Sistema de alimentación ininterrumpida*: para salvaguardar los trabajos de todos los usuarios y proteger el hardware es necesario que el cluster esté alimentado mediante una UPS acorde a su consumo (proporcional al número de nodos en cuestión). Este sistema debería garantizar alimentación ininterrumpida durante 24 hs.
- *Conexión a los laboratorios*: es necesario poseer un buen ancho de banda para la conexión entre el cluster y los laboratorios que lo utilicen. Este no es un recurso indispensable para las conexiones basadas en texto pero si se quiere trabajar con interfaces visuales se necesitaría, mínimamente, una conexión directa desde cluster a los laboratorios que lo utilicen.
- *Servidor de archivos*: el servidor de archivos actualmente lo constituye el mismo nodo maestro pero esto, cuando se aumente la cantidad de nodos en el cluster, resultará altamente ineficiente. Se requiere una computadora con características similares a las del nodo maestro pero dedicada exclusivamente a servir archivos al resto del sistema en cluster.
- *Grabadora de backups*: dada la gran importancia la grabación periódica de backups es necesario contar con un sistema de grabación DAT (Digital Audio Tape) y una grabadora de CDs.
- *Conexionado interno*: para el conexionado interno del cluster es necesario contar con 2 switchs (10/100 Mb/s) con la cantidad de bocas necesarias, y conectividad de categoría 5: cables, fichas de conexión, rosetas, cable-canal, patcheras, etc.
- *Recursos humanos*: para brindar un servicio a muchos usuarios es necesario contar con una persona dedicada a la administración, actualización y mantenimiento del cluster.

5.5 Agradecimientos

La secciones relativas a MPI y PETSc se realizaron con la colaboración del Bioing. Diego Campana, del Grupo de Biomecánica Computacional.

Los integrantes del Grupo de Reconocimiento Automático del Habla quieren agradecer especialmente la participación desinteresada de Adrián Azar en todas las etapas de este desarrollo. De no haber contado con sus conocimientos y experiencia en la administración de redes y sistemas Linux, seguramente no se habrían podido lograr los objetivos planteados en el tiempo previsto.

Notas y referencias

[1]Por ejemplo: Sun UltraSPARC, DEC/Compaq Alpha o incluso los Intel Pentium II/III.

[2]También es posible arrancar con otros sistemas operativos que estén instalados, como OS/2 o Windows.

[3]Dirección: **`http://www.mosix.org/txt_distribution.html`**

[4]Deben tener instalado el núcleo MOSIX.

[5]Dirección: **www.mcs.anl.gov/mpi/mpich/download.html**

[6]Por ejemplo desde: **http://www.netlib.org/pvm3/index.html**

[7]Dirección: **www-fp.mcs.anl.gov/petsc/**

[8]Dirección: **www.netlib.org/lapack/**

[9]Estas variables de entorno son luego configuradas automáticamente para cada usuario que utilice el paquete.

[10]Todo lo relativo al entorno X fue suprimido para estas pruebas.

[11]Dirección: **htk.eng.cam.ac.uk**

[12]Dirección: **www-ra.informatik.uni-tuebingen.de/SNNS**

[13]Dirección: **www.openpbs.org**

[14]Dirección: **www.chiark.greenend.org.uk/sgtatham/putty/**

[15]Dirección: **www.ssh.com**

[16]Dirección: **http://www.uk.research.att.com/vnc/**

Bibliografía

AMIR, Y. ; AWERBUCH, B. ; BARAK, A. ; BORGSTROM, R.S. y KEREN, A. “An opportunity cost approach for job assignment in a scalable computing cluster”. En: IEEE Tran. Parallel and Distributed Systems **11** (2000), no. 7, 760–768.

BARAK, A. ; GUDAY, S. y WHEELER, R. “The MOSIX distributed operating system. Load balancing for unix”. En: Lecture notes in computer science, vol. 672, Springer-Verlag, 1993.

CATALINA GALLEGO, MIGUEL. UNIX/Linux. Iniciación y referencia, 1era ed., McGraw-Hill, Noviembre 1998, ISBN: 8448121015.

GROPP, WILLIAM; LUSK, EWING y SKJELLUM, ANTHONY. Using MPI: Portable parallel programming with the message passing interface, 2nd ed., MIT. Press, Cambridge, MA, 1999.

HARGROVE, WILLIAM W.; HOFFMAN, FORREST M. y STERLING, THOMAS. “The Do-It-Yourself Supercomputer”. En: Scientific American **256** (2001), no. 2, 62–69.

KIRCH, OLAF y DAWSON, TERRY. Linux network administrator’s guide, 2nd ed., O’Reilly, June 2000, ISBN: 1-56592-400-2.

KNUTH, DONALD E. The TeXbook, Addison Wesley, Massachusetts, 1984, ISBN 0-201-13448-9.

LAMPORT, LESLIE. LaTeX: A document preparation system*, 2/e*, Addison Wesley, Digital Equipment Corporation, 1994, ISBN: 0-201-52983-1.

SPECTOR, DAVID H.M. Building Linux clusters, 1st ed., O’Reilly, July 2000.

YOUNG, STEVE; KERSHAW, DAN; ODELL, JULIAN; OLLASON, DAVE; VALTCHEV, VALTCHO y WOODLAND, PHIL. The HTK book, Entropic Ltd., 1999.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, A. Azar & H. L. Rufiner; "Supercomputadoras basadas en clusters de PCs"
Revista Ciencia, Docencia y Tecnología, Vol. XIII, No. 25, pp. 173-208, nov, 2002.