Computación de Alto Rendimiento con *Clusters* de PCs

Bernal C. Iván, Mejía N. David y Fernández A. Diego

imbernal@mailfie.epn.edu.ec, {r_mejia, dafernandezr}@fie201.epn.edu.ec
Escuela Politécnica Nacional
Quito-Ecuador

Abstract

En la actualidad, es factible disponer de alta capacidad computacional, incluso equivalente a la encontrada en las poderosas y costosas supercomputadoras clásicas, mediante *clusters* (conglomerados) de computadoras personales (PCs) independientes, de bajo costo, interconectadas con tecnologías de red de alta velocidad, y empleando software de libre distribución. El conglomerado de computadoras puede trabajar de forma coordinada para dar la ilusión de un único sistema. Este artículo presenta las ideas básicas involucradas en el diseño, construcción y operación de *clusters,* presentando aspectos relacionados tanto al software como al hardware. Se presentan los diferentes tipos de *clusters*, su arquitectura, algunas consideraciones de diseño, y se mencionan ejemplos concretos del hardware para los nodos individuales y para los elementos de interconexión de alta velocidad, así como ejemplos concretos de los sistemas de software para el desarrollo de aplicaciones y administración de los *clusters*.

1. Introducción

Un *cluster* es una solución computacional estructurada a partir de un conjunto de sistemas computacionales muy similares entre si (grupo de computadoras), interconectados mediante alguna tecnología de red de alta velocidad, configurados de forma coordinada para dar la ilusión de un único recurso; cada uno de estos sistemas estará proveyendo un mismo servicio o ejecutando una (o parte de una) misma aplicación paralela. Un *cluster* debe tener como característica inherente la compartición de recursos: ciclos de CPU *(Central Processing Unit)*, memoria, datos y servicios.

Los sistemas computacionales (nodos) que conforman el *cluster* podrían ser computadoras de uno o varios procesadores; estos sistemas podrían estar montados en un *rack*, ubicados en un espacio dedicado exclusivamente a almacenar computadoras, o en el cubículo de un empleado; lo que cuenta es como están relacionados, como son accesados, y que tipo de aplicación están ejecutando.

La idea de los *clusters* tomo impulso en los 90s, cuando se dispuso de microprocesadores de alto rendimiento, redes de alta velocidad, y herramientas estándar para computación distribuida (*Message Passing Interface*, MPI, *Parallel Virtual Machine*, PVM (Quinn, 2003; Pacheco, 1997)) y a costos razonables. Pero también el desarrollo de los *clusters* fue impulsado por deficiencias de los Sistemas Multiprocesador Simétricos (*Symmetric MultiProcessors*, SMP*s* (Culler y Singh, 1999)). Las grandes máquinas SMP son costosas, propietarias, tienen un único punto de falla, no están ampliamente disponibles, y sufren de problemas de escalabilidad, en términos de número de procesadores y capacidad de memoria. Según Lucke (2005), los sistemas SMP más grandes conocidos, escalan hasta un número de alrededor de 128 CPU*s*.

En 1994, T. Sterling y D. Becker, trabajando en CESDIS (*Center of Excellence in Space Data and Information Sciences*) bajo el patrocinio del Proyecto de la Tierra y Ciencias del Espacio (ESS), construyeron un *cluster* de computadoras que consistía de 16 procesadores 486DX4, usando una red Ethernet a 10Mbps, con un costo de $40,000. Ellos llamaron a su sistema *Beowulf*, un éxito inmediato, y su idea de proporcionar sistemas en base a COTS (*Components Of The Shelve*) para satisfacer requisitos de cómputo específicos, se propagó rápidamente a través de la NASA y en las comunidades académicas y de investigación. En la actualidad, muchos *clusters* todavía son diseñados, ensamblados y configurados por sus propios operadores; sin embargo, existe la opción de adquirir *clusters* prefabricados.

El problema que se intenta resolver con estos *clusters* es el de disponer de capacidad computacional equivalente al encontrado en poderosas y costosas supercomputadoras paralelas tradicionales (*Cray/SGI T3E*) (Gordon y Gray, 2001), pero empleando componentes de bajo costo y ampliamente disponibles (*commodities*). Los altos requerimientos computacionales a los que se hace mención, son típicos en aplicaciones como algoritmos genéticos, simulación de líneas de fabricación, aplicaciones militares, bases de datos, síntesis de imágenes, recuperación de imágenes por contenido, simulación de modelos para clima, análisis de sismos, algoritmos para solución a problemas de electromagnetismo, dinámica de fluidos, química cuántica, biomedicina, etc. (Buyya, 1999, Vol. II).

Las grandes supercomputadoras tradicionales, propietarias y costosas, están siendo reemplazadas por *clusters* a una fracción del costo. Esto permite a pequeñas organizaciones de investigación, departamentos de IT (*Information Technology)*, y grupos de ingeniería, poseer sus propias supercomputadoras, a una fracción del costo previamente requerido para obtener el equivalente computacional. Otros aspectos económicos a considerarse son costos explícitos, necesarios para utilizar y mantener un centro especializado con supercomputadoras: espacio, aire acondicionado, consumo de potencia, personal para administración y consultas, etc. Cuando se posee un *cluster* estos costos están implícitos ya que se operan centros propios y no se los debe sustentar como usuarios de un centro especializado (Gordon y Gray, 2001).

En consecuencia, una de las ventajas de una solución con *clusters* es construir una plataforma que se ajuste a un presupuesto determinado y que sea adecuada para un grupo considerable de aplicaciones. Además, un *cluster* puede ser expandido con relativa facilidad, incrementando el número de nodos o la capacidad de los nodos individuales existentes, añadiendo memoria y/o procesadores.

El número de procesadores en un *cluster* se ha incrementado notablemente, se habla de cientos de procesadores. De acuerdo al reporte de http://www.top500.org, de Noviembre de 2004, entre las diez supercomputadoras mas rápidas del mundo, figuran cinco basadas en *clusters*, empleando un número de nodos en el orden de 4000. El *cluster* más rápido se denomina *MareNostrum*, un eServer BladeCenter JS20, formado por 4536 procesadores PowerPC970 de 2.2 GHz e interconectados con una red Myrinet, es considerada la cuarta computadora más veloz y fue desarrollada por el *Barcelona Supercomputer Center* de España. La quinta computadora más veloz se denomina *Thunder*, formada por 4096 procesadores Intel Itanium 2 de 1.4 GHz e interconectados con una red Quadrics, fue desarrollado por *Lawrence Livermore National Laboratory* de los Estados Unidos. La sexta computadora se denomina *ASCI Q* y está conformada por 8192 procesadores AlphaServer SC45 de 1.25 GHz, fue desarrollada por *Los Alamos National Laboratory* de los Estados Unidos. La séptima computadora se denomina *System X* y está conformada por 2200 procesadores 1100 Dual Apple XServe de 2.3 GHz y posee dos redes una Cisco GigE y otra Mellanox Infiniband, fue desarrollada por *Virginia Tech* de los Estados Unidos. La décima computadora también es un cluster, se denomina *Tungsten* y está formada por 1450 procesadores P4 Xeon de 3.06 GHz y utiliza una red Myrinet para la interconexión de los nodos, fue desarrollada por NCSA de los Estados Unidos.

En la actualidad, se pretende que este tipo de solución se inserte entre las tendencias ampliamente utilizadas en el mundo de IT, que no sean únicamente un instrumento en universidades e institutos de investigación. Más aún, como Gordon y Gray (2001) señalan, se prevé una convergencia de *clusters* con tópicos como computación *P2P* (*peer-to-peer*) y computación *Grid (Grid Computing)*.

Por lo mencionado, existe la necesidad de un adecuado entendimiento de lo que la computación con *clusters* puede ofrecer, cómo los *clusters* de computadoras pueden construirse, y cuál su impacto en aplicaciones identificadas como adecuadas para un ambiente basado en *clusters*.

2. Clasificación

El término *cluster* tiene diferentes connotaciones para diferentes grupos de personas. Los tipos de *clusters*, establecidos en base al uso que se de a los *clusters* y los servicios que ofrecen, determinan el significado del término para el grupo que lo utiliza. Así, un grupo involucrado en computación científica (que requiere alto rendimiento y muchos recursos, *High Performance*) tendrá una imagen diferente de lo que es un *cluster*, respecto a la imagen que tendría un grupo involucrado en sistemas de alta disponibilidad (por ejemplo: Servicios Web disponibles incluso frente a fallas, *High Availability*). Un tercer punto de vista corresponde a los grupos cuyo interés está en los *clusters* que permiten ejecutar un gran número de tareas independientes en paralelo (*High Throughput*).

2.1 *High Performance*

- Para tareas que requieren gran poder computacional, grandes cantidades de memoria, o ambos a la vez.
- Las tareas podrían comprometer los recursos por largos periodos de tiempo.

2.2 *High Availability*

- Máxima disponibilidad de servicios.
- Rendimiento sostenido.

2.3 *High Throughput*

- Independencia de datos entre las tareas individuales.
- El retardo entre los nodos del *cluster* no es considerado un gran problema.
- La meta es el completar el mayor número de tareas en el tiempo mas corto posible.

Los *clusters* se los puede también clasificar como *Clusters* de IT Comerciales (*High Availability, High Throughput*) y *Clusters* Científicos (*High Performance*) (Lucke, 2005). A pesar de las discrepancias a nivel de requerimientos de las aplicaciones, muchas de las características de las arquitecturas de hardware y software, que están por debajo de las aplicaciones en todos estos *clusters*, son las mismas. Más aun, un *cluster* de determinado tipo, puede también presentar características de los otros.

3. Aspectos a considerarse de la tecnología de *Clusters*

Para diseñar, implementar, probar y mantener un *cluster* se requiere un entendimiento básico pero claro de hardware de computadoras, de redes de computadoras y de sistemas operativos, y la habilidad para investigar algunos tópicos especializados, como dispositivos de interconexión de alta velocidad, talvez reintroducirse a lenguajes de programación como FORTRAN, y librerías para el desarrollo de aplicaciones como MPI. Una vez escogido un sistema operativo, dígase Linux, se requiere algo de experiencia en la administración de sistemas Linux y en la forma de realizar conexiones de red. En términos generales, es necesario tener un adecuado entendimiento referente a varios aspectos de los *clusters*:

- Diseño, implementación, configuración, realización de pruebas de los *clusters*.
- Identificación de las aplicaciones que pueden beneficiarse de la tecnología de *clusters*.
- Desarrollo de las aplicaciones a ejecutarse en los *clusters*.
- Conocimiento de las limitaciones del rendimiento de los *clusters*.
- Formas de realizar una administración efectiva de los *clusters*.

4. Diseño, Instalación, Pruebas y Administración de los *Clusters*

4.1 Diseño

Se debe entender el propósito del *cluster*: es para correr una aplicación (solucionar un problema) en particular o es de propósito general, para ejecutar múltiples aplicaciones paralelas. Definiendo estos aspectos y requerimientos de rendimiento, puede determinarse el número total de nodos participantes, el número y tipo de CPUs en cada nodo, el tipo de HSI (*High Speed Interconnects*), los requerimientos de los *switches*, estimados de costos y tiempo para construcción, el sistema operativo, *middleware* (Buyya, 1999, Vol. I), y librerías de desarrollo (MPI, PVM).

En *clusters* de gran tamaño, puede ser necesario realizar *benchmarking* en un nodo para determinar requerimientos de velocidad de la memoria, características de I/O, rendimiento del CPU, etc. Pruebas del rendimiento de las tarjetas de red y *switches*, en conjunto con el software que se utilizará en el *cluster* pueden ser también necesarias (Lucke, 2005).

Finalmente, las condiciones del lugar en donde se instalará el *cluster* deben considerarse; restricciones de espacio, necesidades de alimentación de energía, aire acondicionado, espacio para el cableado, *racks* necesarios, etc.

4.2 Instalación

Una vez que el diseño del *cluster* esté concluido, se procede al ensamblaje de todo el sistema. Se debe preparar el sitio de instalación, se debe ensamblar y verificar el hardware, y se debe instalar y configurar el software. El tamaño del *cluster*, y quien realice las tareas mencionadas, determinarán el tiempo necesario.

Respecto al software, según Lucke (2005), para muchos es la parte mas compleja e invisible del *cluster*, y la consideran la mas difícil de tener corriendo de forma consistente. Algunos de los principales componentes del software son:

- El sistema operativo y *drivers* para los dispositivos.
- Compiladores y librerías de desarrollo de aplicaciones.
- Librerías especiales para HSI.
- Planificación de tareas y balanceo de carga.
- Servicios de autenticación y autorización.
- Sistema de archivos para el *cluster*.
- Herramientas de administración del sistema.
- Aplicaciones.

4.3 Pruebas

En esta etapa se verifica la operación del *cluster* como un único recurso (Pfister, 1995) y se busca cumplir las metas de rendimiento y estabilidad. Aquí se pueden encontrar cuellos de botella debidos a las interconexiones de red, o debidos a la configuración del software, o a problemas con las políticas de seguridad, planificación de tareas o balanceo de carga (Buyya, 1999, Vol. I). Se desarrolla en realidad un proceso de prueba, corrección y nuevas pruebas.

4.4 Administración

Existen diversas maneras de coordinar las actividades de las tareas esclavas, y en general, de llegar a la respuesta. Algunas estrategias hacen mejor uso de los recursos que otras, algunas minimizan el tiempo empleado, algunas son mas fáciles de implementar, y otras son mas resistentes a fallas. El desarrollo de aplicaciones paralelas es todavía una tarea algo complicada; “toma una mente especial para visualizar y desarrollar algoritmos paralelos, particionar los datos, y asociar una solución con una configuración de hardware paralelo en particular” (Lucke, 2005).

5. Arquitecturas y Elementos de los *Clusters*

Un *cluster* puede tener una gama de categorías de componentes para su operación, típicamente la razón para separar la funcionalidad del *cluster* en estas categorías es evitar interferencia entre operaciones de cálculo u operaciones de I/O, con la comunicación usando el HSI (Lucke, 2005). Otra razón es la de proveer mayores niveles de disponibilidad y seguridad a ciertos componentes u operaciones. En un *cluster* podrían encontrarse:

- **Nodo Maestro** ("*head node*"). Utilizado para proveer al usuario con el acceso a los recursos de cómputo, planificación de tareas o espacio para almacenamiento. Esconde los recursos, dando al mundo externo la visión de un único recurso.
- **Nodos de Cómputo**. Realizan las porciones asignadas de los cálculos o cómputos de la aplicación paralela, o una unidad de un servicio escalable (si se habla de disponibilidad, por ejemplo).
- **Nodo Administrativo.** Provee servicios administrativos como monitoreo del rendimiento y generación de eventos para los administradores del *cluster*.
- **Nodo de Infraestructura.** Provee servicios esenciales para el *cluster*, tales como servicios de licenciamiento, servicios de autenticación, planificación de tareas y balanceo de carga.
- **Nodo de I/O o Servidor de Archivos.** Provee acceso a los recursos de almacenamiento del *cluster* para los usuarios y las aplicaciones.
- **Varias Redes:** de administración del *cluster*, de acceso a datos, HSI, de consolas de administración de los nodos de cómputo (serialmente, por ejemplo).

La figura 1 presenta un diagrama combinado, tanto físico como lógico de un posible *cluster*. En el ejemplo se ha dividido los nodos de cómputo en varios grupos. Esto permite obtener varias ventajas: segregar las diferentes redes para preservar seguridad entre las redes de datos y administración, de ser necesario; la reducción de cableado entre los *racks*; disponer de mayor ancho de banda entre cada *rack* y el *switch* principal (*core switch*) que permite la interconexión de las diferentes redes. El nodo maestro pertenece a todas las redes del *cluster*, pueden existir varios y deben ofrecer alta disponibilidad, por lo que redundancia puede ser necesaria.

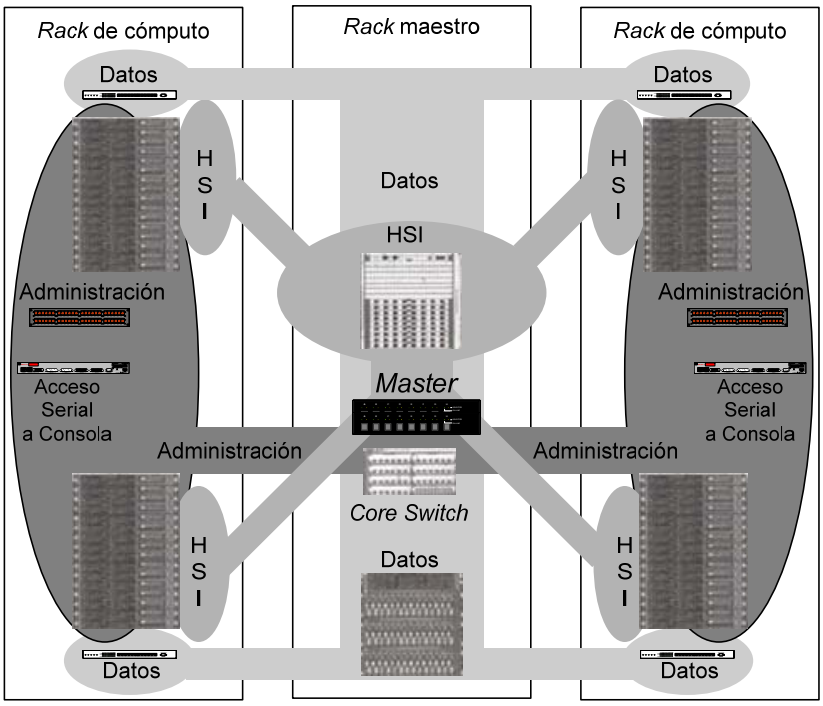

Figura 1 Diagrama Combinado (físico y lógico) de un *cluster*

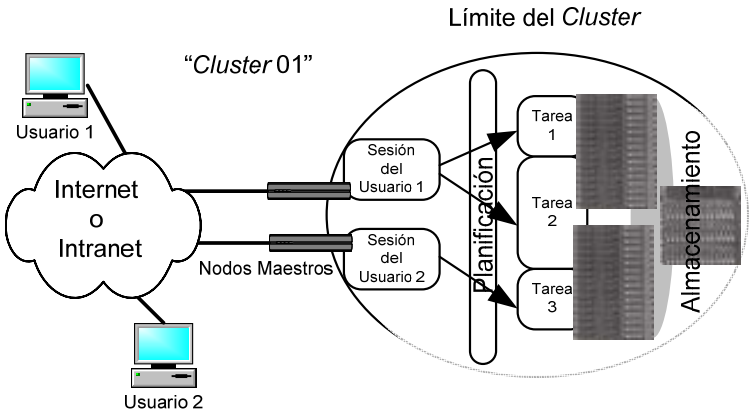

Figura 2 Ocultamiento de los recursos utilizados de un *cluster* a los usuarios

Las redes y recursos internos del *cluster* son normalmente privados, y por lo tanto invisibles, excepto por los puntos de acceso: el nodo maestro y el de administración. Un usuario puede ingresar al sistema y entregar sus tareas; los usuarios no conocen que nodos y recursos están siendo utilizados, simplemente conocen que los recursos requeridos serán satisfechos y la tarea será planificada y ejecutada. La figura 2 presenta una posible configuración con dos usuarios accediendo a los recursos del *cluster*. Configuraciones similares pueden utilizarse para Servidores de Bases de Datos, Servidores Web y otras configuraciones comerciales de *clusters*; en ciertos casos, podría ser que los nodos maestros estén ausentes (Sistema de Bases de Datos Paralela), o reemplazados con un director para balanceo de carga (Servidor Web Paralelo).

5.1 Configuraciones para *clusters* de *High Throughput*

5.1.1 *Clusters* tipo "Alfombra" (*Carpet Clusters*)

Hacen referencia a los *cluster* que generalmente se arman en un cubículo u oficina, sobre la alfombra del piso. Suelen utilizar computadoras que debido a su rendimiento a en ese momento se consideran obsoletas, y que probablemente serán desechadas. Por lo general son útiles para un número pequeño de personas, no requieren gran esfuerzo ni inversión para su instalación y mantenimiento.

Instalar *clusters* utilizando *racks*, es nada más una forma de controlar espacio requerido, acceso, calor, potencia y ruido para un gran número de nodos; en estos casos cada nodo no dispone de elementos como monitor, teclado y *mouse*. Sin embargo, no es raro que un *cluster* se construya en base a PCs o estaciones de trabajo individuales completas.

La figura 3 presenta un ejemplo de este tipo de *cluster*, en el cual una única red (probablemente Ethernet) conecta todos los componentes del *cluster*, para datos, administración y HSI. Este tipo de *cluster* se utiliza, por ejemplo, para compilaciones de código fuente de un grupo pequeño de desarrolladores.

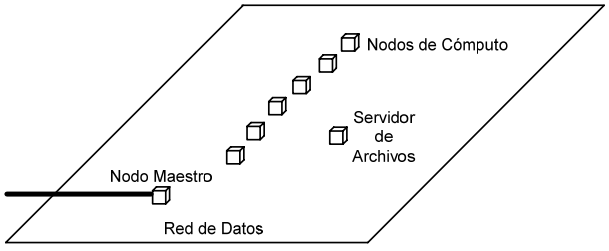

Figura 3 Arquitectura de un *cluster* de "alfombra"

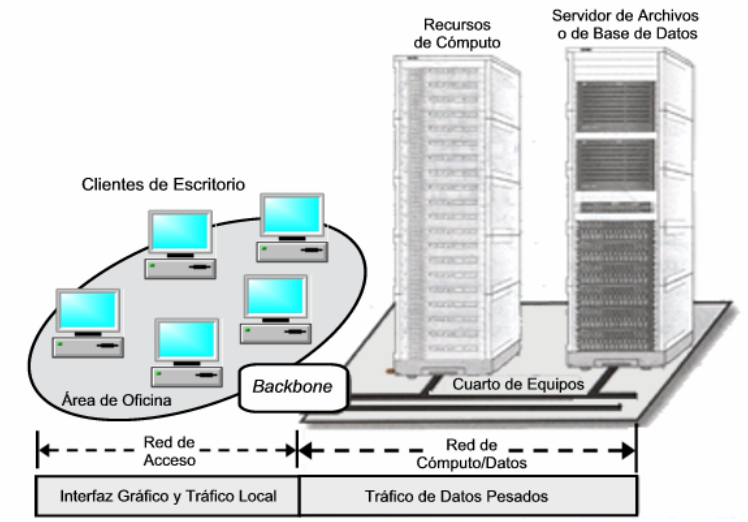

Figura 4 Estructura funcional de un *cluster* "granja" o "rancho"

5.1.2 *Clusters* de cómputo tipo "Granjas y Ranchos" (*Farms and Ranches*)

La diferencia entre estos tipos no es muy clara, y se suele hacer mención a que radica en su tamaño. La idea es que si el número de tareas excede lo que puede manejar un solo sistema, se proveen nodos extra para planificar muchas tareas independientes al mismo tiempo. Se suelen mirar como una formalización de los *cluster* tipo "alfombra", con los nodos montados en un *rack*. Se utilizan para compilaciones grandes y complejas, y para verificación de circuitos, como parte de EDA *(Electronic Design Automation*).

Es posible dividir las redes, en una para el interfaz gráfico (*Graphics User Interface*, GUI) en el escritorio de los usuarios, y otra para los datos y las aplicaciones; se limita así el tráfico pesado de datos al cuarto que contiene los *racks*, en donde está el HSI. Para programas que requieren visualización de datos, manejar el GUI con programas que manejan escritorios remotos (como VNC, *Virtual Network Computing, http://www.realvnc.com/*), evita trasladar grandes cantidades de datos. La figura 4 presenta un ejemplo de esta alternativa.

En estos *clusters*, los pedidos de los usuarios, típicamente, ya son manejados por un servicio de balanceo de carga; si las demandas de los usuarios exceden los recursos disponibles se genera una cola de espera para las tareas, la que va acompañada de software para la planificación de tareas. Se dispone también de nodos administrativos, que pueden ser manejados en una red diferente para mejorar la seguridad y evitar interferencias con la red de datos. Servidores de archivos pueden proveer los datos a partir de una red SAN (*Storage Area Network*) (Clark, 2003). La figura 5 presenta un esquema con los componentes mencionados.

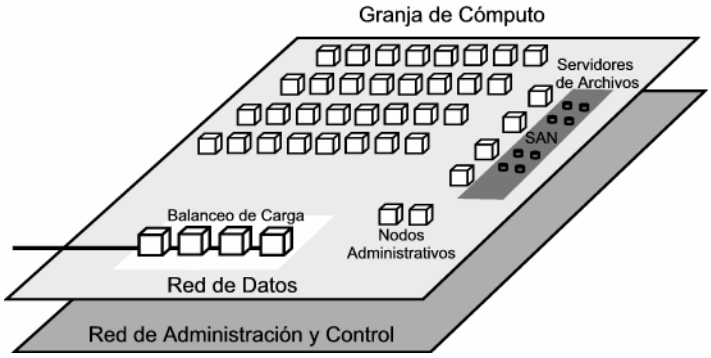

Figura 5 Arquitectura de un *cluster* "granja" o "rancho"

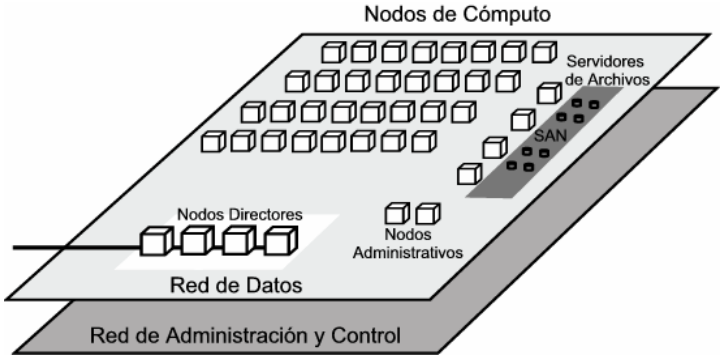

Figura 6 Arquitectura de un *cluster* para un Servidor Web

Un *cluster* de este tipo puede tener una mezcla de diferente hardware y sistemas operativos, para su empleo en diferentes tipos de aplicación. El planificador de tareas es capaz de escoger la ubicación mas adecuada para ejecutar una tarea específica. La flexibilidad par esconder detalles de los recursos de los usuarios es una gran ventaja de esta alternativa.

5.2 Configuraciones para *clusters* de *High Availability*

Aunque todos los *clusters* requieren alguna medida de confiabilidad y disponibilidad, esto puede no ser su meta principal de diseño. Alta disponibilidad se ofrece aprovechando las características de escalabilidad de un *cluster*, combinadas con la característica de que los pedidos de servicio son independientes entre si. Pueden utilizarse en Servicios Web, Servidores de bases de datos, y otros tipos de servicios tradicionales. Gran parte de la confiabilidad en este tipo de *cluster* se obtiene con software que detecta fallas del hardware y de los servicios, lo que activa recursos alternos de respaldo, evitando que exista un único punto de falla.

La figura 6 presenta un *cluster* operando como un Servidor Web simple. Los clientes hacen pedidos utilizando una única dirección de red, asignada a un director que realiza el balanceo de carga. El director redirecciona un pedido a uno de los servidores virtuales que finalmente realiza el trabajo. El cliente no tiene conocimiento de este redireccionamiento, ya que los servidores virtuales permanecen escondidos tras el director. El director tiene un reemplazo en *standby* para garantizar disponibilidad. Los datos son compartidos a través de un sistema de archivos de alta disponibilidad y se presenta una única vista de dichos datos a los clientes.

5.3 Configuraciones para *clusters* de *High Performance*

Debido a problemas de confiabilidad que se tiene en este tipo de *clusters*, se suelen proveer facilidades para almacenar el estado de la aplicación paralela en algún punto intermedio durante su ejecución, para luego poder reiniciarla desde ese punto (*checkpoint and restart*), lo cual es útil frente a fallas.

Puede requerirse teraflops de recursos computacionales y terabytes de RAM para obtener resultados para una precisión deseada, y el tiempo empleado podría estar en el orden de semanas. Aplicaciones típicas son visualización 3D, por la industria automotriz para análisis de accidentes, para hacer pruebas de línea de ensamblaje antes de construirlas. Muchas veces las facilidades de visualización son una parte separada o dedicada del *cluster* o incluso un *cluster* separado si se soporta el movimiento de los datos entre ellos. La figura 7 presenta la arquitectura de un *cluster* de visualización, en el que se incluyen elementos ya mencionados anteriormente, y algunos nuevos como nodos extra para redundancia y para pruebas. Obsérvese la presencia de las distintas redes.

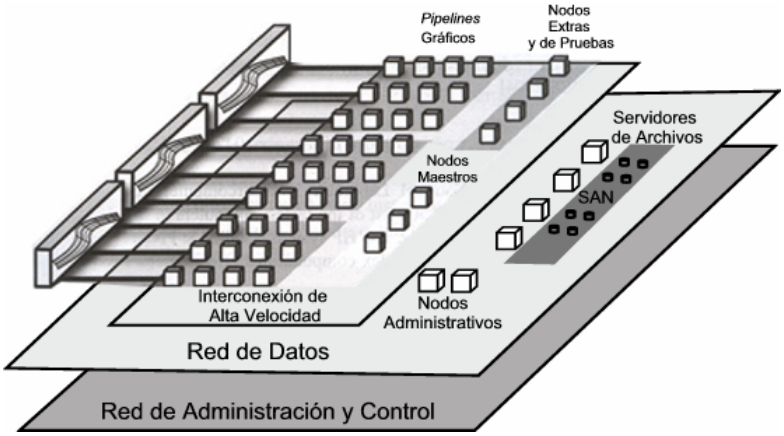

Figura 7 Arquitectura de un *cluster* para visualización 3D

6. Nodo Maestro y Nodos de Cómputo

De la gran cantidad de opciones existentes para los nodos de cómputo en un *cluster*, la selección depende del presupuesto disponible, de las características deseada en el *cluster* y de las aplicaciones a ejecutarse. A nivel de sistema se debe evaluar un gran número de características: costo, número de CPUs por nodo, rendimiento de las unidades de enteros y punto flotante, cantidad de RAM soportada, opciones de 32 y 64 bits para direccionamiento de memoria, ancho de banda del CPU y de I/O, presencia de puerto serial para administración por consola, facilidades incorporadas para LAN, etc. A continuación se describen algunos de ellos.

Una facilidad muy útil es la de disponer de un puerto integrado iLO (*integrated Lights Out*) de administración remota, lo que permite conectar al nodo a una red de administración y monitoreo independiente, permitiendo por ejemplo controlar encendido y apagado del nodo.

Una tarea a realizar durante la selección será el comparar sistemas similares en cuanto a las características de rendimiento (*performance*) importantes para la aplicación. Las principales categorías de rendimiento a evaluarse son las relacionadas a enteros y punto flotante, para lo cual se utilizan aplicaciones de prueba existentes denominadas *benchmarks*. La Tabla 1 presenta una comparación de ejemplo entre sistemas que utilizan procesadores "Pentium 4 Xeon" e "Itanium 2" de Intel y "Opteron" de AMD.

El precio de un nodo es fundamental si se está construyendo *clusters* de decenas o cientos de nodos; la Tabla 1 fue tomada de Lucke (2005) y los precios son a Diciembre 30 de 2004. Para los procesadores del ejemplo, existen otros aspectos a considerar, así el Xeon, de 32 bits, puede manejar hasta 4GB de direcciones, pero en una configuración dual (SMP) debe manejar 8GB; los procesadores Intel de 32 bits incorporan una característica denominada PAE (*Physical Address Extension*) para manejar hasta 64 GB de memoria. Los nodos con Itanium 2 y el Opteron, ambos de 64 bits, no aprovechan las 64 líneas en su totalidad; así con el Opteron se utilizan solo 40 líneas para manejar hasta 1 terabyte de RAM; lo que principalmente se debe al costo y limitaciones físicas para acomodar toda la memoria que podría manejarse con 64 bits. Se han empleado también procesadores Alpha (Henesey y Patterson, 1996) que ofrecen mejores características de punto flotante que los Pentium, pero son más costosos.

Tabla 1 Comparación de precios y rendimiento para tres nodos de cómputo

Tipo de Sistema	Precio Base	Precio Final	SPECint_2000	SPECfp_2000
Hewlett-Packard rx2600 1.5 GHz Itanium 2 3M	$5,730	$22,990	1322	2119
Hewlett-Packard DL-360g3 3.2 GHz Pentium Xeon	$4,448	$7,546	1319	1197
IBM eServer 325 2.0 GHz AMD Opteron	$5,959	$8,746	1226	1231

La cantidad de RAM es un factor fundamental, y determinado por la aplicación o aplicaciones que se ejecutaran en el mismo nodo simultáneamente; se debe considerar en este último caso el espacio de direcciones virtual de un proceso, y la posibilidad de que el "*paging*" (Tanenbaum, 2001; Lucke, 2005) se convierta en un cuello de botella; basta con recordar que cada proceso asume que tiene todo el procesador y RAM a su disposición, pero que en realidad no todo el proceso está en memoria física en un momento dado, y puede ser necesario mover instrucciones y datos entre memoria y el disco duro, lo que seriamente afecta el rendimiento del sistema.

El utilizar hardware de 64 bits en los nodos implica utilizar sistemas operativos de 64 bits y aplicaciones de 64 bits, y se las utiliza cuando se va a trabajar con conjuntos de datos extremadamente grandes, en cada proceso.

Finalmente, dado que el nodo maestro es el punto de acceso al *cluster*, éste debe poseer suficientes recursos para soportar el número de usuarios simultáneos esperados, garantizando además alta disponibilidad. Este nodo necesitará más RAM, un número mayor de CPUs, más ranuras (*slots*) de I/O para garantizar conexiones a las redes internas y de forma redundante.

7. HSI (*High Speed Interconnects*)

En cuanto a tecnologías de interconexión se han realizado numerosos intentos por optimizar su funcionamiento, algunos de ellos cristalizados en productos comerciales. Existen alternativas que oscilan desde las de bajo costo hasta aquellas sumamente eficientes y optimizadas pero con un costo mucho mayor, a tal punto que ya no son consideradas *commodities*. Se puede escoger entre: Ethernet, Fast Ethernet, Gigabit Ethernet, SCI (Scalable Coherent Interface), Myrinet, Infiniband, Dolphin, Quadrics QsNet-II (Buyya, 1999, Vol. II; Lucke,2005).

Muchos de los HSIs están basados en redes de conmutación que tienen elementos de conmutación especializados, y formas distribuidas y topologías especiales como toroides, *hypercubes*, y *meshes* (Henesey y Patterson, 1996). Dichos componentes buscan simplificar el direccionamiento de nodo a nodo, reducir la cantidad de tiempo para tomar decisiones de enrutamiento, y transportar los datos rápidamente.

Es importante considerar, al momento de la selección, que para el *HSI* estén disponibles las librerías (MPI, PVM) y *drivers* necesarios.

8. Software

El software a utilizar en los *clusters* es un elemento primordial a considerarse. En relación a sistemas operativos, sobre todo en *clusters* ensamblados por los propios operadores, una opción común es utilizar alguna distribución de Linux; ciertos sectores (Brauss et al., 1999) consideran que eso se debe a su amplia aceptación en el mundo académico, bajo costo y disponibilidad del código fuente, antes que por poseer características que favorezcan el desarrollo de computación de alto rendimiento. Además, la disponibilidad del código fuente permite hacer variaciones a nivel de sistema y preparar *drivers* de bajo nivel cuando se considere necesario. A pesar de no ser un serio candidato para *clusters* de alto rendimiento (Brauss et al., 1999), Windows NT ha sido empleado en *clusters* IT. El NT *Supercluster* en NCSA es un *cluster* de 192 procesadores, construido con estaciones de trabajo de procesador dual HP Kayak XU y Compaq Professional Workstation 6000, utilizando Myrinet.

Para la comunicación entre nodos se puede utilizar protocolos de red estándar (TCP/IP) o protocolos de bajo nivel como *Active Messages* (Buyya, 1999, Vol.I). Existen también implementaciones que tratan de reducir el número de copias de los datos y frecuentes chequeos de error, encontrados en el interfaz de red, y memoria de usuario y *kernel*.

El paradigma de comunicación entre los componentes de un *cluster* es típicamente el paso de mensajes y la unificación de herramientas de desarrollo paralelas y de dominio público, muchas de ellas gratis, es también un factor que debe ser analizado. Con la madurez y la robustez de Linux, el software GNU y de la estandarización de envío de mensajes vía PVM y MPI, los programadores ahora tienen una garantía que los programas que escriban correrán en *diversos clusters*, sin importar quien fabricó los procesadores o el HSI. La versión abierta de la librería MPI, y varios utilitarios, se denomina MPICH (*http://www-unix.mcs.anl.gov/mpi/mpich*).

Sin la habilidad de leer datos y escribir resultados, las facilidades de cómputo son de poco valor. Se debe proveer un sistema de archivos paralelo, que podría ofrecer a los diversos nodos un sistema de archivos compartido, manteniendo una vista consistente de los datos entre los clientes (del sistema de archivos). De lo mencionado, puede ser que NFS (*Network File System*) sea suficiente solamente para *clusters* pequeños. Para *clusters* grandes, existen varias opciones, entre ellas PVFS (*Parallel Virtual File System, http://www.pvfs.org/pvfs2*), OpenGFS (*Open Global File System, http://opengfs.source-forge.net*), Lustre (*http://www.lustre.org*).

Herramientas de soporte para instalación, monitoreo y administración, bajo la modalidad de software libre, son numerosas, cada una tratando de mejorar los cuellos de botella encontrados en los *clusters.* Sterling (2002) y Gropp et al. (2003) mencionan algunas herramientas: Condor, un planificador distribuido de tareas; Maui, un planificador de tareas; PBS (*Portable Batch System*), un sistema para planificar tareas por lotes; Ganglia es un sistema de monitoreo y ejecución de tareas.

Existen también a disposición librerías para manipulación de vectores y matrices, como BLAS (*Basic Linear Algebra Subprograms*), LAPACK (*Linear Algebra PACKage*) y LINPACK, todas disponibles en (http://www.netlib.org), que junto con MPI y PVM facilitan el desarrollo de aplicaciones.

Las herramientas mencionadas en los párrafos anteriores son subsistemas que se instalan para estructurar un cluster. Para facilitar estas tareas de instalación existen los denominados *toolkits* que guían al usuario en el proceso de instalación. De entre las opciones disponibles, se pueden mencionar algunos que han tenido gran aceptación: OSCAR, NPACI Rocks y openMosix (Sloan, 2004).

Finalmente, mientras los recursos computacionales son cada vez más accesibles, grandes esfuerzos se han realizado en la invención y evolución de algoritmos y aplicaciones para utilizar eficientemente dichos recursos. Los *clusters* han traído a escena un gran número de usuarios y desarrolladores en diversas áreas, así algoritmos genéticos paralelos, solución de ecuaciones FDTD (*Finite-Difference Time-Domain*), simulación de líneas de fabricación de semiconductores, aplicaciones militares, síntesis de imágenes con *ray tracing*, modelos para simulación de clima y océanos, algoritmos para electromagnetismo, etc. Buyya *(1999, vol.II)* provee numerosos escenarios de aplicaciones y detalles sobre su solución con *clusters*.

Un tipo de problema que puede beneficiarse de un *cluster* es aquel que permita ser fraccionado en piezas que sean computacionalmente independientes, o en los que existan pocas dependencias (*loosely coupled*). El programa maestro (*master)* fracciona el problema en piezas, distribuye estas piezas a las tareas esclavas (*slaves*), espera recibir los resultados parciales, y finalmente los integra. Si queda trabajo por realizar, el programa *master* envía una nueva pieza de trabajo a las tareas esclavas que estén disponibles. La existencia de dependencias marcadas (*tightly coupled*) entre los conjuntos de datos entregados a las tareas esclavas, requerirían compartir y enviar datos entre tareas esclavas hasta cuando se concluyan los cálculos. El tiempo que toma mover los datos entre las tareas esclavas o entre esclavas y *master*, debe ser pequeño en relación al tiempo empleado en los cálculos, para que el tiempo empleado en el cálculo global sea mejorado. Esta dependencia o independencia es un factor importante a considerar en el diseño de los *clusters*, tanto del software (modelos de programación) como del hardware (conexiones de alta velocidad, bajo retardo).

9. Comentarios

Para incursionar de forma efectiva en nuestro país en las áreas comerciales y científicas, mencionadas anteriormente, y que éstas se beneficien de los *clusters*, existe la necesidad de un adecuado entendimiento de lo que la computación con *clusters* puede ofrecer: cómo los *clusters* de computadoras pueden construirse (identificando configuraciones y tecnologías de interconexión), cómo pueden desarrollarse las aplicaciones (herramientas e infraestructuras de desarrollo), cuáles son las limitaciones de rendimiento de los *clusters*, cómo se pueden administrar los *clusters* de forma efectiva, cuáles aplicaciones pueden beneficiarse de esta tecnología.

Con el financiamiento de La Escuela Politécnica Nacional (EPN) y FUNDACYT, el Departamento de Electrónica, Telecomunicaciones y Redes de Información de la EPN está trabajando en un proyecto para disponer un *cluster* básico, con un número reducido de nodos, que podrá ampliarse y mejorarse, y que servirá para el desarrollo y ejecución de diversos prototipos. Con la experiencia adquirida, y con el personal formado, se espera ofrecer soporte para que otras unidades académicas de la EPN, y empresas públicas y privadas, desarrollen no solo aplicaciones en sus respectivos campos, sino que se espera proporcionar soporte para que ellas instalen y administren sus propios *clusters*.

Es claro entonces la importancia de adquirir, adaptar, aplicar, difundir y propiciar el empleo de la tecnología de *clusters* en diversas áreas, sean éstas académicas, científicas o comerciales. Se puede entonces obtener una mejor relación costo/beneficio de las inversiones en infraestructuras y acceder a campos que resultaban prohibitivos por los elevados montos de inversión involucrados anteriormente.

Bibliografía

Buyya, R. (1999). *High Performance Cluster Computing: Architectures and Systems,* Volume I, Prentice Hall, Upper Saddle River, New Jersey.

Buyya, R. (1999). *High Performance Cluster Computing: Programming and Applications*, Volume II, Prentice Hall, Upper Saddle River, New Jersey.

Brauss, S., Frey, M., Gunzinger, A., Lienhard, M. y Nemecek J. (1999). *Swiss-Tx Communication Libraries*, HPCN Europe 1999**,** Springer-Verlag.

Clark, T. (2003). *Designing Storage Area Networks: A Practical Reference for Implementing Fibre Channel and IP SANs*, 2nd edition, Addison Wesley.

Culler, D. y Singh J. (1999). *Parallel Computer Architectures: A hardware/Software Approach*, Morgan Kaufmann, San Francisco.

Gordon, B. y Gray J. (2001). "High Performance Computing: Crays, Clusters, and Centers. What Next?" Technical Report MSR-TR-2001-76, Microsoft Research, Microsoft Corporation, USA.

Gropp, W., Lusk, E. y Sterling,T. (2003). *Beowulf Cluster Computing with Linu*x, 2nd edition, The MIT Press.

Henesey, J. y Patterson D. (1996), *Computer Architecture: A quantitative Approach,* Morgan Kaufmann, San Francisco.

Kvasnicka D., Hlavacs H. y Ueberhuber C. (2001). "Cluster Configuration Aided by Simulation". Proceedings of Computational Science – ICCS 2001, USA, 2073, 243--252.

Lucke, R. (2005). Building *Clustered Linux Systems*, Prentice Hall, Upper Saddle River, New Jersey.

Pacheco P. (1997). *Parallel Programming with MPI*, Morgan Kaufmann, San Francisco.

Pfister G. (1995). *In Search of Clusters: The Coming Battle in Lowly Parallel Computing,* Prentice Hall, Upper Saddle River, New Jersey.

Quinn, M. (2003). *Parallel Programming in C with MPI and OpenMP,* McGraw-Hill.

Sloan, J. (2004). *High Performance Linux Clusters: With Oscar, Rocks, openMosix, And MPI*, O'Reilly & Associates.

Sterling, T. (2002). *Beowulf Cluster Computing with Windows*, The MIT Press.

Tanenbaum, A. (2001). *Modern Operating System*, 2nd edition, Prentice Hall, Upper Saddle River, New Jersey.