Introducción a las tecnologías de clustering en GNU/Linux

Rosa María Yáñez Gómez

Versión 1.0

Dedicado a Javi,
por todo lo que nos une

Índice de contenido

1._CLUSTERS DE COMPUTADORAS

Figura 2_Los clusters basados en GNU-Linux son flexibles y económicos.

El término cluster se aplica no sólo a computadoras de alto rendimiento sino también a los conjuntos de computadoras, construidos utilizando componentes de hardware comunes y software libre. Entre estos dos extremos situaremos también las soluciones comerciales modulares, que permiten ajustar la inversión a los requerimientos concretos del problema al que nos enfrentemos y expandir el cluster conforme las necesidades vayan aumentando. Estas soluciones, sin embargo, van a imponer modelos específicos de programación que permitan explotar al máximo las capacidades de la máquina y por otra parte, encadenan al comprador a un fabricante concreto. Es por ello que este documento se centrará en el estudio de la tecnología de clustering dentro del entorno del software libre, sobre todo en el uso del sistema operativo GNU/Linux y otras herramientas libres asociadas a él. Aunque, hemos de señalar que se han presentado soluciones de clustering también para otros sistemas operativos libres (FreeBSD, NetBSD...).

Los clusters juegan hoy en día un papel muy importante en la solución de problemas de las ciencias, las ingenierías y en el desarrollo y ejecución de muchas aplicaciones comerciales. Los clusters han evolucionado para apoyar actividades en aplicaciones que van desde la supercomputación hasta el software adaptado a misiones críticas, pasando por los servidores web, el comercio electrónico y las bases de datos de alto rendimiento.

La computación basada en clusters surge gracias a la disponibilidad de microprocesadores de alto rendimiento más económicos y de redes de alta velocidad, y también gracias al desarrollo de herramientas de software para cómputo distribuido de alto rendimiento; todo ello frente a la creciente necesidad de potencia de cómputo para aplicaciones en las ciencias y en el ámbito comercial, así como de disponibilidad permanente para algunos servicios. Por otro lado, la evolución y estabilidad que ha alcanzado el sistema operativo GNU/Linux, ha contribuido de forma importante, al desarrollo de muchas tecnologías nuevas, entre ellas las de clustering.

Según la aplicabilidad de los clusters, se han desarrollado diferentes líneas tecnológicas. La primera surge frente a la necesidad de supercomputación para determinadas aplicaciones, lo que se persigue es conseguir que un gran número de máquinas individuales actúen como una sola máquina muy potente. Este tipo de clusters se aplica mejor en problemas grandes y complejos que requieren una cantidad enorme de potencia computacional. Entre las aplicaciones más comunes de clusters de alto rendimiento (computacionales, de supercomputación) se encuentra el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, simulación militar, simulación de recombinaciones entre moléculas naturales y el análisis de imágenes.

Un segundo tipo de tecnología de clusters, es el destinado al balanceo de carga. Surge el

concepto de "cluster de servidores virtuales", cluster que permite que un conjunto de servidores de red compartan la carga de trabajo y de tráfico de sus clientes, aunque aparezcan para estos clientes como un único servidor. Al balancear la carga de trabajo en un conjunto de servidores, se mejora el tiempo de acceso y la confiabilidad. Además como es un conjunto de servidores el que atiende el trabajo, la caída de uno de ellos no ocasiona una caída total del sistema. Este tipo de servicio es de gran valor para compañías que trabajan con grandes volúmenes de tráfico y trabajo en sus webs, servidores de correo... Hemos de pensar que la imagen y el prestigio de una empresa que ofrece sus servicios por Internet se compromete en la velocidad, la calidad y la disponibilidad de estos servicios.

El último tipo importante de tecnología de clustering trata del mantenimiento de servidores que actúen entre ellos como respaldos de la información que sirven. Este tipo de clusters se conoce como "clusters de alta disponibilidad" o "clusters de redundancia". La flexibilidad y robustez que proporcionan este tipo de clusters, los hacen necesarios en ambientes de intercambio masivo de información, almacenamiento de datos sensibles y allí donde sea necesaria una disponibilidad continua del servicio ofrecido.

Los clusters de alta disponibilidad permiten un fácil mantenimiento de servidores. Una máquina de un cluster de servidores se puede sacar de línea, apagarse y actualizarse o repararse sin comprometer los servicios que brinda el cluster. Cuando el servidor vuelva a estar listo, se reincorporará y volverá a formar parte del cluster.

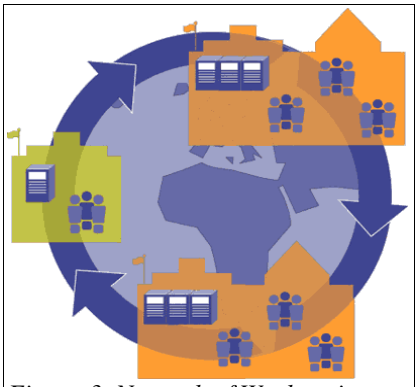

Figura 3_Network of Workstations (NOW)

Además del concepto de cluster, existe otro concepto más amplio y general que es el Cómputo en Malla (Grid Computing). Una Malla (Network Of Workstation o NOW) es un tipo de sistema paralelo y distribuido que permite compartir, seleccionar y añadir recursos que se encuentran distribuidos a lo largo de dominios administrativos "múltiples". Si los recursos distribuidos se encuentran bajo la administración de un sistema central único de programación de tareas, entonces nos referiremos a un cluster. En un cluster, todos los nodos trabajan en cooperación con un objetivo y una meta común y la asignación de recursos la lleva a cabo un solo administrador centralizado y global. En una Malla, cada nodo tiene su propio administrador de recursos y política de asignación.

2._¿CÓMO FUNCIONA UN CLUSTER?

Desde un punto de vista general, un cluster consta de dos partes. La primera es el software: un sistema operativo confeccionado especialmente para esta tarea (por ejemplo un kernel Linux modificado), compiladores y aplicaciones especiales, que permite que los programas que se ejecutan en el sistema exploten todas las ventajas del cluster. En el entorno de GNU/Linux hay que destacar la PVM (Paralell Virtual Machine) y la MPI (Message Passing Interface), librerías que abstraen la componente hardware de la componente software.

La segunda componente es la interconexión hardware entre las máquinas (nodos) del cluster. Se han desarrollado interfaces de interconexión especiales muy eficientes, pero es común realizar las interconexiones mediante una red Ethernet dedicada de alta velocidad. Gracias a esta red de interconexión los nodos del cluster intercambian entre sí las tareas, las actualizaciones de estado y los datos del programa. En un cluster abierto, existirá una interfaz de red que conecte al cluster con el mundo exterior (Internet)[1].

Cuando se trata de resolver un problema en paralelo, el software debe ser capaz de dividir el problema en tareas más pequeñas, repartirlas entre los nodos y elaborar los resultados. Puesto que las subtareas van a ejecutarse en paralelo se consigue un aumento de velocidad, aunque hay que tener en cuenta el retardo que la división, reparto y transmisión de mensajes (resultado, coherencia, estados...) supone.

En el caso de los clusters de balanceo de carga, el hardware y el software deben actuar conjuntamente para que el tráfico se distribuya entre los nodos del cluster. De esta forma, se pueden ofrecer los servicios a mayor velocidad o se realiza una tarea más rápidamente.

Los servidores de un cluster de alta disponibilidad normalmente no comparten la carga de procesamiento que tiene un cluster de Alto Rendimiento. Tampoco comparten la carga de tráfico como lo hacen los clusters de Balanceo de Carga. Su función es la de estar preparados para entrar inmediatamente en funcionamiento en caso de que falle algún otro servidor.

3._¿POR QUÉ CONSTRUIR UN CLUSTER?

Construir un cluster puede aportar importantes ventajas en gran variedad de aplicaciones y ambientes:

- Incremento de velocidad de procesamiento ofrecido por los clusters de alto rendimiento.
- Incremento del número de transacciones o velocidad de respuesta ofrecido por los clusters de balanceo de carga.
- Incremento de la confiabilidad y la robustez ofrecido por los clusters de alta disponibilidad.

Por ejemplo, en la investigaciones meteorológicas y pronóstico numérico del estado del tiempo, se requiere el manejo de cantidades masivas de datos y cálculos muy complejos. Al combinar el poder de muchas máquinas del tipo estación de trabajo o servidor, se pueden alcanzar niveles de rendimiento similares a los de las supercomputadoras, pero a menor costo.

Otra situación de aplicabilidad de un cluster sería en un sitio web que soportara mucho tráfico. Si no se cuenta con un plan de alta disponibilidad, cualquier problema menor de una tarjeta de red, puede hacer que un servidor quede completamente inutilizado. Pero al contar con servidores redundantes y servidores de respaldo instantáneos, se puede reparar el problema mientras el sitio

1 En el caso de los clusters de alto rendimiento, no es común que estos se conecten al exterior debido a las implicaciones de seguridad que esto supone. En estos clusters se suele elegir la velocidad frente a la seguridad.

sigue funcionando sin suspensión de servicio.

4._PUNTOS A CONSIDERAR A LA HORA DE CONFIGURAR UN CLUSTER

Por sus características especiales, hay varias cuestiones particulares asociadas a esta tecnología que deben ser tenidas en cuenta.

Uno de los principales problemas a los que hay que hacer frente cuando se construye un cluster es buscar y eliminar los puntos de fallo únicos (single points of failure). Si se trabaja en un cluster de supercomputación que depende de un servidor central para repartir las tareas, si este servidor cae, todo el cluster quedará inservible. Igualmente, si se trata de un cluster de balanceo de carga o de alta disponibilidad, se deben establecer garantías de que los servidores seguirán funcionando, pero si estos servidores están conectados a una red corporativa o a Internet, mediante una sola interfaz, un fallo en ella dejaría aislado al sistema. Es importante perseguir la redundancia para evitar que el fallo de un sólo componente hardware (recordemos que en un cluster van a integrarse gran número de elementos con lo que la probabilidad de fallo crece) anule la funcionalidad de todo el sistema.

Otra cuestión importante, es elegir correctamente la tecnología que vamos a utilizar en función de nuestras necesidades. Mantener un cluster sobre una red Ethernet de 10 Mb, puede resultar una buena decisión si el cluster sólo tiene unos cuantos nodos, pero en el momento en que se inserten más nodos, la red va a convertirse en un cuello de botella que obligará a los servidores a estar desocupados en espera de los datos durante, quizá, demasiado tiempo.

5._CLUSTERS COMPUTACIONALES

Las simulaciones en computadora son vitales para el estudio de muchos problemas, desde el diseño en Ingeniería hasta el estudio de procesos complejos en la Naturaleza. Sin embargo, el alcance y la precisión de estas simulaciones están limitados por la potencia computacional de las supercomputadoras más potentes.

La historia de los clusters computacionales en Linux comenzó cuando Donald Becker y Thomas Sterling construyeron un cluster para la NASA, su nombre fue Beowulf. El modelo de clusters tipo Beowulf se basa en componentes y periféricos para la plataforma x86 común para obtener un rendimiento sin precedentes a un costo muy bajo. A partir de este proyecto, han surgido numerosas iniciativas en este sentido.

Estos clusters se utilizan para cualquier tarea que requiera enormes cantidades de cómputo: data mining, simulaciones científicas, renderización de gráficos, modelado meteorológico...

5.1._BEOWULF

Figura 4_Beowulf, el primer cluster basado en tecnología COST y GNU-Linux revolucionó la tecnología de clustering.

En el verano de 1994, Thomas Sterling y Don Becker, trabajando en el CESDIS[2] (Center of Excellence in Space Data and Information Sciences) bajo la tutela del proyecto ESS(Earth and Space Sciences), construyeron un cluster computacional consistente en procesadores de tipo x86 comerciales conectados por una red Ethernet de 10Mb. Llamaron a su máquina Beowulf, nombre de un héroe de la mitología danesa relatado en el libro La Era de las Fábulas, del autor norteamericano Thomas Bulfinch (Beowulf derrotó al monstruo gigante Grendel). Inmediatamente, aquello fue un éxito y su idea de basar sistemas en el concepto de COTS (Commodity off the shelf) pronto se difundió a través de la NASA y las comunidades académicas y de investigación. El desarrollo de esta máquina pronto se vio enmarcado dentro de lo que hoy se conoce como "The Beowulf Project". Los clusters Beowulf están hoy reconocidos como un tipo de clusters dentro de los HPC (High Performance Computer)[3].

La industria COST proporciona hoy una gran variedad de elementos hardware compatibles (microprocesadores, placas base, discos, tarjetas de red...) siendo la competencia en el mercado la que ha llevado a esta interrelación. El desarrollo del software libre, en particular del sistema operativo GNU/Linux, y del resto de aplicaciones GNU: compiladores, herramientas de programación y sobre todo, de la MPI (Message Passing Interface) y la librería PVM (Paralell Virtual Machine), junto a esta disponibilidad de hardware compatible, han logrado independizar el software del hardware. Esto, unido a los resultados obtenidos por diversos grupos de investigación, ha fundamentado la idea de adoptar una actitud do-it-yourself[4] para mantener el trabajo de software independiente del hardware específico de cada fabricante.

Figura 5_Wiglaf, el primer cluster de tipo Beowulf.

2 El CESDIS es una división de la USRA (University Space Research Association) localizada en Maryland. El CESDIS trabajaba para la NASA en el proyecto ESS. Este proyecto determinó la aplicabilidad de las computadoras MPP (Massively Parallel Processors) a los problemas que surgían en aquel momento de la investigación de la tierra y el espacio.

3 Clusters de alto rendimiento.

4 Esta actitud puede resumirse en la pregunta "¿ Por qué comprar lo que estás capacitado para fabricar tu mismo? (Why buy what you can afford to make?) La realidad es que aprender a construir, administrar y desarrollar para clusters Beowulf o de cualquier otro otro tipo siempre que estén basados en tecnología COST, es una excelente inversión; aprender las peculiaridades de un fabricante concreto sólo sirve para encadenarte a él.

Wiglaf fue el primer cluster de tipo Beowulf. Era un cluster de 16 nodos con procesadores DX4 a 100MHz (un híbrido entre el 80486 y el Intel P5 Pentium). La placa base estaba basada en el chipset SiS 82471, que era el de más altas prestaciones en bajo coste en aquel momento. Cada procesador disponía de 16M de DRAM algo más rápida y cara que la usual en el mercado (60ns). Y cada nodo poseía también 540 M de 1G EIDE disk. Además el sistema contenía tres tarjetas de red Ethernet a 10Mb.

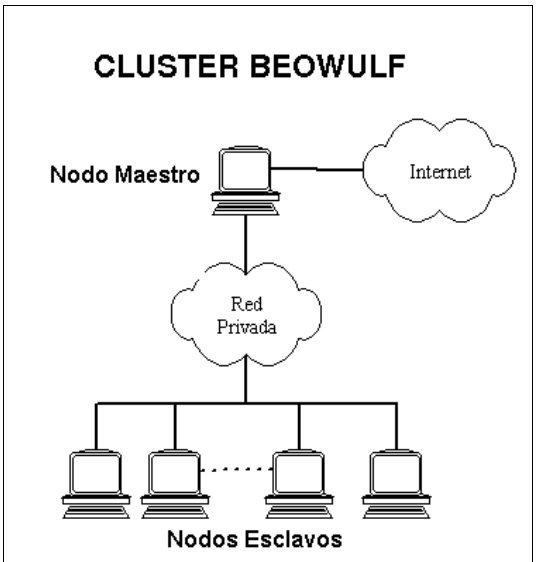

Figura 6_Diagrama general de un cluster Beowulf

Un cluster de tipo Beowulf no es más que una colección de computadoras personales (PC's) interconectadas por medio de una red privada de alta velocidad, corriendo algún sistema operativo libre: FreeBSD, NetBSD,... Nos centraremos en este documento en los clusters de tipo Beowulf basados en GNU-Linux. Los nodos en el cluster están dedicados exclusivamente a ejecutar tareas asignadas al cluster. Por lo general el cluster se encuentra comunicado al mundo exterior a través de un solo nodo, llamado nodo maestro, el cual también esta reservado para acceder, compilar y manejar las aplicaciones a ejecutar. Si bien hay que señalar que en los clusters de computación se elige la rapidez frente a la seguridad con lo que se relajan las medidas de seguridad entre nodos hasta el punto de hacer poco recomendable la conexión del cluster a Internet; incluso el sistema utilizado para mantenimiento y administración debe estar desconectado de la red.

A medida que la tecnología ha ido avanzando se han podido construir clusters con procesadores más rápidos, mejores tecnologías de red, coste más bajo frente a las prestaciones... Sin embargo, una característica importante de los clusters Beowulf es que todos estos cambios no modifican el modelo de programación. A esta independencia del hardware, tal como indicamos anteriormente, ha contribuido en gran medida la madurez de GNU-Linux y de la utilización del paso de mensajes via PVM y MPI. Así los programadores tienen la garantía de que los programas que escriben hoy seguirán funcionando en el futuro sobre los clusters Beowulf. La crítica histórica de que el software para computadores de alta computación es dedicado y poco reciclable deja de ser real desde el momento en que, con Beowulf, se escapa de la dependencia de los modelos de software forzados por los fabricantes del hardware de alta computacion.

Si intentáramos clasificar los clusters Beowulf, podríamos decir que se encuentran en un punto intermedio entre los MPP (Massively Parallel Processors) y las NOWs (Networks of Workstations). Los clusters Beowulf se benefician de distintos beneficios de ambos tipos de sistemas. Los MPPs (nCube, CM5, Convex SPP, Cray T3D, Cray T3E...) son más grandes y con menos latencia en la red de interconexión que los clusters Beowulf. Sin embargo, los programadores deben preocuparse acerca de la localidad, el balanceo de carga, la granularidad, y los overheads de comunicación para obtener los mejores resultados. Incluso en máquinas con memoria compartida, muchos programadores, por comodidad, desarrollan sus programas en un estilo de paso de mensajes. Este último tipo de programas se podría portar perfectamente a clusters de tipo Beowulf.

Por otro lado, en una NOW se suelen ejecutar programas poco ajustados, tolerantes a problemas de balanceo de carga y alta latencia de comunicación. Cualquier programa que funcione sobre una NOW va a funcionar, al menos con la misma eficiencia, en un cluster Beowulf. Sin embargo, las diferencias de concepto entre un cluster Beowulf y una NOW son significativas. Primero, los nodos en un cluster están dedicados exclusivamente al cluster, es decir, la labor de cada nodo no está sujeta a factores externos, lo que facilita la resolución de problemas de balanceo de carga. Además, como la red de interconexión de un cluster está aislada del resto de la red, el tráfico que soporta es sólo el generado por la aplicación que está siendo ejecutada sobre el cluster. Esto evita el problema de latencia impredecible que se da en las NOWs. Además, el aislamiento de la red evita problemas de seguridad, que sí han de ser tenidos en cuenta en las NOWs. Asimismo, en un cluster se pueden realizar ajustes en parámetros del sistema operativo para mejorar el funcionamiento de los nodos en el cluster, cosa que no se puede hacer en las NOW ya que los nodos realizan otras tareas independientes y propias. Por último, en las NOWs, el trabajo se replica en los nodos para luego ser contrastado. Esto se lleva a cabo por cuestiones de seguridad ya que los nodos forman parte de entornos de administración distintos.

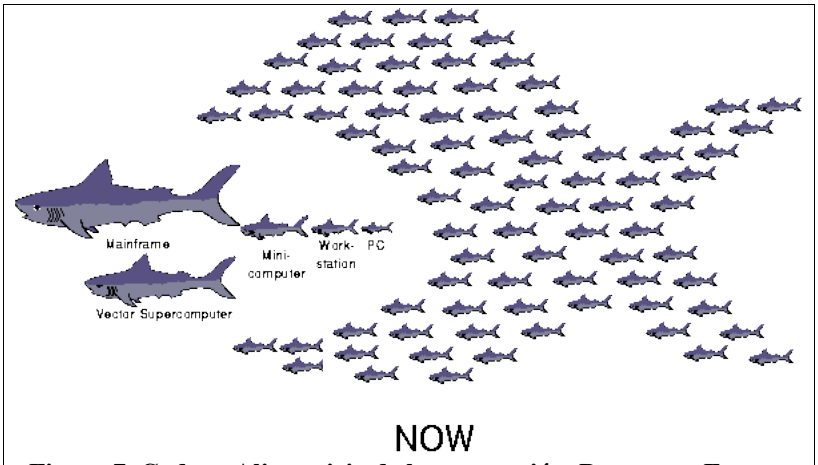

Figura 7_Cadena Alimenticia de la comutación. Presente y Futuro.

5.2._MOSIX

Esta tecnología basada en Linux, permite realizar balanceo de carga para procesos particulares en un cluster. Sin embargo, la clasificamos entre los clusters de alto rendimiento ya que está diseñado para tomar ventaja del hardware más rápido disponible en el cluster para cualquier tarea que le sea asignada. Aumenta así la capacidad y velocidad de cómputo, pero, internamente tan sólo balancea la carga de las tareas en varias máquinas.

Debemos comentar que MOSIX se convirtió en producto comercial en 2001, aunque existe una alternativa libre, OpenMosix, desde febrero de 2002. Durante este apartado nos referiremos a MOSIX ya que a efectos de este documento, que no ahonda en cuestiones técnicas, la diferenciación es prescindible.

Una de las grandes ventajas de MOSIX es que no requiere la confección especial de software como lo requiere los clusters tipo Beowulf. Los usuarios ejecutan sus programas normalmente y el sistema MOSIX se encarga del resto. El núcleo de MOSIX está formado por algoritmos que monitorizan y dan repuesta a las actividades requeridas del cluster mediante migración automática de los procesos a los nodos mejor capacitados. Estos algoritmos están diseñados manteniendo los principios de facilidad de uso, optimización y escalabilidad. Los

algoritmos de MOSIX usan la migración de procesos para:

- Distribución y redistribución automática de procesos.
- Transferencia de procesos desde los nodos más lentos a los más rápidos.
- Balanceo de carga.
- Migración de procesos desde los nodos que funcionan fuera de memoria, para evitar swapping o thrashing.
- Entrada/Salida paralelizada para migrar los procesos de E/S a los servidores de ficheros (al contrario de lo usual).

6._CLUSTERS DE ALTA DISPONIBILIDAD

La alta disponibilidad ha sido un tradicionalmente un requerimiento exigido a aquellos sistemas que realizaban misiones críticas. Sin embargo, actualmente, está siendo cada vez más importante exigir esta disponibilidad en sistemas comerciales y en áreas académicas donde el objetivo de prestar los servicios en el menor tiempo posible es cada vez más perseguido.

El concepto de cluster de disponibilidad continua, se basa en la idea de mantener la prestación del servicio en todo momento. Esto representa una situación ideal, sería necesario que el sistema estuviera compuesto de componentes perfectos que no fallaran nunca, tanto en hardware como en software. Realmente no hay sistemas que puedan asumir este tipo de disponibilidad.

Necesitamos que el cluster sea tolerante a los fallos. En este caso se encubre la presencia de los fallos del sistema empleando redundancia en el hardware, el software e incluso redundancia temporal. La redundancia en el hardware consiste en añadir componentes replicados para encubrir los posibles fallos. La redundancia software incluye la administración del hardware redundante para asegurar su correcto funcionamiento al hacer frente a la caída de algún elemento. La redundancia en el tiempo hace referencia a la repetición de la ejecución de un conjunto de instrucciones para asegurar el comportamiento correcto en caso de que ocurra un fallo.

Definiremos un cluster de alta disponibilidad como un sistema capaz de encubrir los fallos que se producen en él para mantener una prestación de servicio continua. En este caso nos centraremos en los clusters de este tipo que utilizan componentes hardware COST de forma redundante y software capaz de unir estos componentes y enmascarar los fallos de manera que los servicios ofrecidos al usuario no sean interrumpidos.

El cluster de alta disponibilidad va a poder diseñarse con distintas configuraciones. Una posible configuración es activo-pasivo: las aplicaciones se ejecutan sobre un conjunto de nodos, los activos, mientras que los nodos restantes actúan como backups redundantes para los servicios ofrecidos. En el otro extremo, tenemos otra posible configuración, activo-activo: en este caso, todos los nodos actúan como servidores activos de una o más aplicaciones y potencialmente como backups para las aplicaciones que se ejecutan en otros nodos. Cuando un nodo falla, las aplicaciones que se ejecutaba en él se migran a uno de sus nodos backup. Esta situación podría producir una sobrecarga de los nodos de respaldo, con lo que se ejecutarían las aplicaciones con más retardo.

Generalmente, sin embargo, es aceptable una degradación del servicio y también suele ser preferible a una caída total del sistema. Otro caso particular de cluster de alta disponibilidad sería el de un cluster de un solo nodo, tratándose en este caso, simplemente, de evitar los puntos únicos de fallo.

Los conceptos de alta disponibilidad y de clustering están intimamente relacionados ya que el concepto de alta disponibilidad de servicios implica directamente una solución mediante clustering. La principal prestación de un sistema de alta dispnibilidad es que el fallo de un nodo derive en que las aplicaciones que se ejecutaban en él sean migradas a otro nodo del sistema. Este migrado puede ser automático (failover) o manual (switchover).

Generalmente este tipo de cluster integra un número relativamente pequeño de nodos (entre 2 y 8), aunque existen soluciones comerciales que trabajan en clusters de mayor tamaño. En este caso, la escalabilidad no venía siendo un objetivo prioritario, en contraste con el caso de los clusters computacionales. Las aplicaciones usadas para el diseño y la implementación de este tipo de clusters no tienen porqué escalar. Sin embargo, actualmente, existe una cierta tendencia a perseguir la escalabilidad también en los clusters de alta disponibilidad. Cada vez se busca más tener un cluster de mayor número de nodos pero más pequeños en tamaño y prestaciones.

Desde un punto de vista general, una solución de alta disponibilidad consiste en dos partes: la infraestructura de alta disponibilidad y los servicios. La infraestructura consiste en componentes software que cooperan entre sí para permitir que el cluster aparezca como un único sistema. Sus funciones incluyen monitorizar los nodos, los procesos de interrelación entre nodos, controlar el acceso a los recursos compartidos y asegurar la integridad de los datos y la satisfacción de los requerimientos del usuario. La infraestructura debe proveer de estas funcionalidades cuando el cluster está en estado estable y, lo que es más importante, cuando algunos nodos dejan de estar operativos.

Los servicios de alta disponibilidad son clientes de la infraestructura, y usan las facilidades que exporta ese nivel para mantener en todo momento el servicio a los usuarios. Normalmente hay una degradación del sistema cuando algún nodo cae, pero no una interrupción del servicio. Los servicios que se mantienen típicamente sobre este tipo de clusters son las bases de datos, los sistemas de ficheros, los servidores de correo y los servidores web. Y en general, serán utilizados para tareas críticas o servicios ofrecidos por una empresa, grupo de investigación o institución académica, que no puedan, por sus características concretas, interrumpir el servicio.

La adaptación más común que debe sufrir una aplicación para poder ser ejecutada en un cluster de alta disponibilidad implementado sobre GNU/Linux, es añadir scripts. Existen APIs para trabajar cómodamente con alta disponibilidad; todas ellas incluyen métodos que permiten el switchover y el failover y que permiten arrancar, parar o monitorizar una aplicación por mencionar algunas de sus funcionalidades.

La infraestructura puede ser parte del núcleo del sistema operativo o puede ser una capa situada encima. Integrar la infraestructura en el sistema operativo tiene como ventaja la eficiencia y la facilidad de uso. La principal ventaja de una capa separada es que se independiza la solución de alta disponibilidad del sistema operativo, con lo que resulta más portable. En cualquier caso el sistema debe tener la capacidad de aparecer como un único sistema (SSI Single System Image). En caso de un cluster de alta disponibilidad los elementos más importantes para asegurar esta

apariencia única son el sistema de ficheros global, dispositivos globales y la red global.

El balanceo de carga -concepto que desarrollaremos en el siguiente apartado- también se convierte en una importante cuestión cuando se habla de clusters de alta disponibilidad. El integrar soluciones para ambos requerimientos acarrea un aumento de tamaño y complejidad en el cluster. La principal cuestión relacionada íntimamente con el concepto de balanceo de carga en un cluster de alta disponibilidad, es la toma de la decisión de qué nodo debe hacerse cargo de las aplicaciones de otro que ha fallado. La decisión puede ser dirigida-por-recurso (resource-driven), cuando la decisión se toma en función de los recursos de que dispone el nodo de apoyo, o dirigida-por-usuario (user driver), cuando es el usuario quien define las prioridades. Si los nodos se encuentran en estado estable, el balanceo de carga puede implementarse de manera similar al balanceo de carga de un cluster sin alta disponibilidad. En el siguiente apartado veremos el proyecto Linux Virtual Server que ofrece una solución de cluster de servidores basada en la combinación del balanceo de carga con la alta disponibilidad.

7._CLUSTERS DE BALANCEO DE CARGA

Con el crecimiento de Internet en los últimos años el tráfico en la red ha aumentado de forma radical y con él, la carga de trabajo que ha de ser soportada por los servidores de servicios, especialmente por los servidores web. Para soportar estos requerimientos hay dos soluciones: o bien el servidor se basa en una máquina de altas prestaciones, que a largo plazo probablemente quede obsoleta por el crecimiento de la carga, o bien se encamina la solución a la utilización de la tecnología de clustering para mantener un cluster de servidores. Cuando la carga de trabajo crezca, se añadirán más servidores al cluster.

Hay varias formas de construir un cluster de balanceo de carga. Una solución basada en mantener varios servidores -aunque no se trate de un cluster propiamente- es utilizar un balanceo de carga por DNS. En este caso, se asigna un mismo nombre a distintas direcciones IP y se realiza un round-robin a nivel de DNS entre ellas. En una situación ideal la carga se repartirá equitativamente entre los distintos servidores. Sin embargo, los clientes "cachean" los datos del DNS, con lo que la carga no va a repartirse equitativamente y quedaría desbalanceada. Además, aún cuando el balanceo de los accesos de los clientes a los servicios se haga de forma balanceada, estos clientes pueden solicitar cargas muy distintas de trabajo al servidor al que se conectan: mientras un cliente puede querer tan sólo ver una página, otro puede solicitar un buen número de ellas. Por otra parte, si un servidor cae, se van a seguir redirigiendo peticiones a él.

Una solución mejor es utilizar un balanceador de carga para distribuir ésta entre los servidores de un cluster. En este caso el balanceo queda a nivel de conexión, con una granularidad más fina y con mejores resultados. Además, se podrán enmascarar más fácilmente las posibles caídas de los nodos del cluster.

El balanceo de carga puede hacerse a dos niveles: de aplicación y a nivel IP. Sin embargo, el balanceo a nivel de aplicación puede provocar efectos de cuello de botella si el número de nodos es grande. Cada solución debe elegirse en función de las necesidades del problema al que hacemos frente. El Linux Virtual Server utiliza balanceo a nivel IP.

Figura 8_ El proyecto Linux Virtual Server combina la alta disponibilidad y el balanceo de carga.

El proyecto Linux Virtual Server (LVS) ofrece parches y aplicaciones de mantenimiento y gestión que permiten construir un cluster de servidores que implementa alta disponibilidad y balanceo de carga sobre el sistema operativo GNU/Linux.

El sistema aparece al usuario externo como un único servidor (en realidad, como un único servidor virtual). Cada uno de los servidores reales que forman el cluster, es controlado por un nodo director que se encarga de realizar el balanceo de carga. Este director corre un sistema operativo GNU/Linux con un kernel parcheado para incluir el código ipvs, que es la herramienta más importante ofrecida por el proyecto LVS. El director puede ser entendido en términos generales como un router con un conjunto concreto de reglas de enrutamiento.

Cuando un cliente solicita un servicio al servidor virtual, el director escoge un servidor real para que lo ofrezca. Desde ese momento el director debe mantener la comunicación entre el cliente y el servidor real. Esta asociación entre cliente y servidor real va a durar sólo lo que dure la conexión tcp establecida; cuando se inicie una nueva conexión el director escogerá de nuevo un servidor real que puede ser distinto del anterior. Así, si el servicio ofrecido es una página web, el cliente podría obtener las imágenes o los textos desde distintos servidores reales ocultos por el servidor virtual.

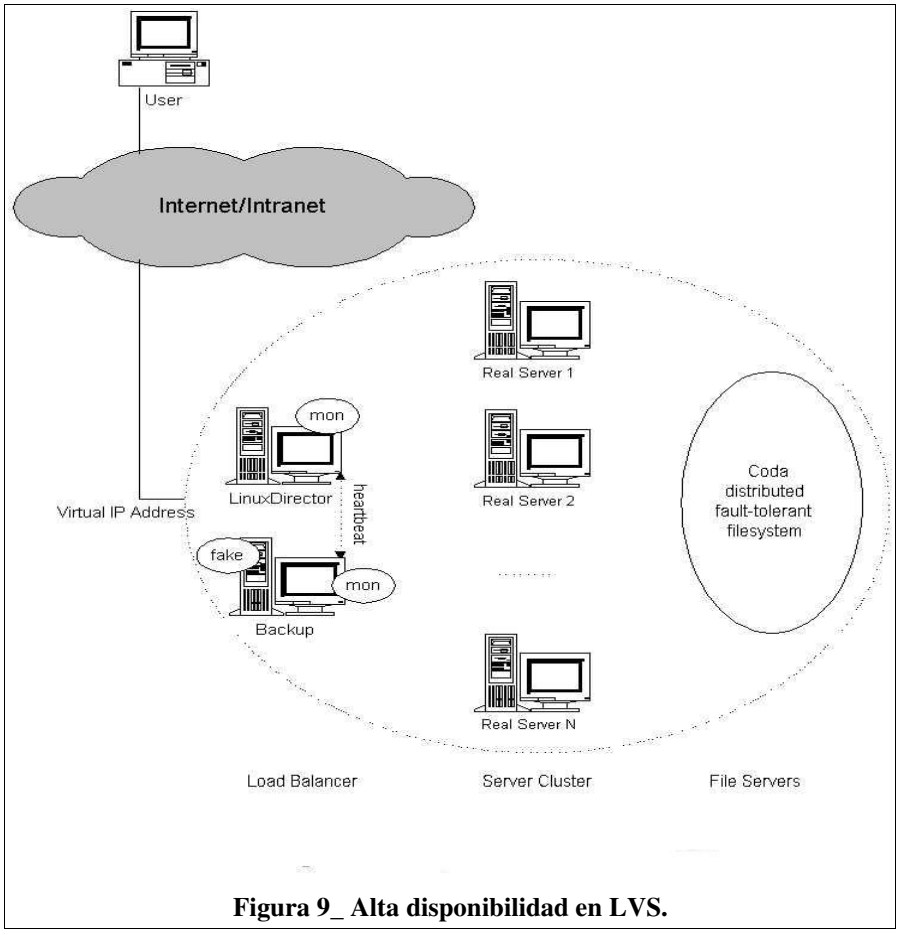

Figura 9_ Alta disponibilidad en LVS.

Con esta arquitectura, además del balanceo de carga, estamos permitiendo que los servidores reales individuales puedan ser extraídos del LVS, actualizados o reparados y devueltos al cluster sin interrumpir la prestación de servicio. Asimismo, el sistema es fácilmente escalable. La alta disponibilidad en LVS se diseña utilizando software mon, heartbeat, fake y coda, que se utilizan para gestionar la alta disponibilidad y para mantener una gestión segura, la comunicación (hearbeat) entre máquinas y un sistema de ficheros tolerante a fallos, respectivamente.

8._RAID

El RAID (Redundant Array of Inexpensive Disks) consiste en organizar varios discos como si de uno solo se tratara pero haciendo que trabajen en paralelo para aumentar la velocidad de acceso o la seguridad frente a fallos del hardware o ambas cosas. Para el sistema operativo, un RAID aparenta ser un sólo disco duro lógico. Hablaremos del RAID en este documento ya que podemos entender un RAID como un "cluster de discos".

Por una parte el RAID permite balancear las operaciones de entrada/salida entre los distintos discos físicos, lo que permite un acceso más rápido a los datos. Por otra, el RAID puede almacenar los mismos datos en distintos lugares (por tanto de modo redundante) en múltiples discos duros. Al colocar los datos en discos múltiples, las operaciones de entrada/salida pueden superponerse de un modo equilibrado, mejorando el rendimiento del sistema y, dado que los discos múltiples incrementan el tiempo medio entre errores, el almacenamiento redundante de datos incrementa la tolerancia a fallos.

El RAID emplea la técnica conocida como "striping" (bandeado o creación de bandas), que incluye la partición del espacio de almacenamiento de cada disco en unidades que van de un sector (512 bytes) hasta varios megabytes. Las bandas de todos los discos están interpaginadas (interleaved) y se accede a ellas en orden. En un sistema de un solo usuario donde se almacenan grandes registros (como imágenes médicas o de otro tipo), las bandas generalmente se establecen para ser muy pequeñas (quizá de 512 bytes) de modo que un solo registro esté ubicado en todos los discos y se pueda acceder a él rápidamente leyendo todos los discos a la vez. En un sistema multiusuario, un mejor rendimiento demanda que se establezca una banda lo suficientemente ancha para contener el registro de tamaño típico o el de mayor tamaño. Esto permite operaciones de entrada/salida superpuestas en los distintos discos.

8.1._¿POR QUÉ USAR RAID?

Las operaciones de entrada/salida a disco son relativamente lentas debido a su carácter mecánico. Una lectura o una escritura involucra, normalmente, dos operaciones. La primera es el posicionamiento de la cabeza lectora/grabadora y la segunda es la transferencia desde o hacia el propio disco[5]. Una forma de mejorar el rendimiento de la transferencia es el uso de varios discos en paralelo, esto se basa en el hecho de que si un disco solitario es capaz de entregar una tasa de transferencia dada, entonces dos discos serían capaces, teóricamente[6], de ofrecer el doble de la tasa anterior. La adición de varios discos debería extender el fenómeno hasta un punto a partir del cual algún otro componente empezará a ser el factor limitante.

Muchos administradores o encargados de sistemas intentan llevar a cabo esta solución en forma básicamente manual, distribuyendo la información entre varios discos intentando asegurar una carga de trabajo similar para cada uno de ellos. Este proceso de "sintonía" podría dar buenos resultados de no ser por dos factores principales:

- No consigue mejorar las velocidades de transferencia de archivos individuales, sólo mejora la cantidad de archivos accedidos en forma concurrente.
- Es obvio que el balanceo es imposible de mantener en el tiempo debido a la naturaleza dinámica de la información.

Una forma bastante más efectiva de conseguir este objetivo es el uso de un RAID. El RAID dividirá en forma automática los requerimientos de lectura/escritura entre los discos. Ante una operación de lectura/escritura, un RAID de 4 discos podría, teóricamente, entregar cuatro veces la tasa de operación de un disco único. En la práctica, sin embargo, esto no es cierto debido, principalmente, a la carga de trabajo inherente al control del propio RAID. Además el uso de varios discos se suele emplear también para mantener cierto nivel de redundancia de los datos, lo que también puede producir un cierto retardo añadido.

5 El posicionamiento de la cabeza está limitado por dos factores: el tiempo de búsqueda (seek time) y el retardo por el giro del disco hasta la posición de inicio de los datos (latencia rotacional). La transferencia de datos, por su parte, ocurre de a un bit por vez y se ve limitada por la velocidad de rotación y por la densidad de grabación del medio.

6 Hay que tener en cuenta el overhead.

8.2._RAID HARDWARE Y SOFTWARE

Existen dos posibilidades de realizar un sistema basado en la tecnología RAID: RAID Hardware o RAID Software.

Las soluciones hardware gestionan el subsistema RAID independientemente del host, presentándole a este un solo disco. Un ejemplo de RAID hardware podría ser el conectado al controlador SCSI que presenta al sistema un único disco SCSI. Un sistema RAID externo se encarga de la gestión del RAID con el controlador localizado en el subsistema externo de los discos. Existen también controladores RAID en forma de tarjetas que se comportan como un controlador SCSI con el sistema operativo, pero gestionan todas las comunicaciones reales entre los discos de manera autónoma.

El RAID software implementa, en el código del kernel, la gestión del disco para los distintos niveles de RAID. Ofrece además la solución menos costosa: el RAID software funciona, además de con discos SCSI, con discos IDE, menos costosos. La potencia de una CPU común actual permite que las prestaciones de un RAID software puedan competir con las de un RAID hardware.

El driver MD del kernel de Linux es un ejemplo de que la solución RAID es completamente independiente del hardware. Las prestaciones de un RAID basado en el software dependen a su vez de las prestaciones de la CPU y de la carga que ésta soporta.

8.3._RAIDs PARALELOS Y RAIDs INDEPENDIENTES

Un RAID paralelo es aquel en el que cada disco participa en todas las operaciones de entrada/salida. Ofrece tasas muy altas de transferencia debido a que las operaciones son distribuidas y ocurren en forma prácticamente simultánea. La tasa de transferencia será muy cercana, 95%, a la suma de las tasas de los discos miembros, mientras que los índices de operaciones de entrada/salida serán similares a las alcanzadas por un disco individual. Resumiendo, un RAID paralelo accederá a un solo archivo al mismo tiempo pero lo hará a muy alta velocidad. Algunas implementaciones requieren de actividades adicionales como la sincronización de discos[7].

Por otro lado, en un RAID independiente cada disco integrante opera en forma autónoma, aún en el caso de que le sea solicitado atender varias peticiones de forma concurrente. Este modelo ofrece operaciones de entrada/salida sumamente rápidas debido a que cada disco está en posición de atender un requerimiento por separado. De esta forma las operaciones de entrada/salida serán atendidas a una velocidad cercana, 95%, a la suma de las capacidades de los discos presentes, mientras que la tasa de transferencia será similar a la de un disco individual debido a que cada

7Los RAIDs de niveles 2 y 3 son paralelos.

archivo está almacenado en sólo un disco[8].

8.4._NIVELES DE RAID

Dentro de un RAID los discos integrantes pueden adoptar distintas configuraciones. La elección de una configuración u otra dependerá de las prestaciones y funcionalidad perseguidas al construir el RAID.

8.4.1._MODO LINEAL

En este caso simplemente uno o más discos físicos se combinan en un mismo dispositivo lógico. La información se almacena consecutivamente, cuando un disco se llena se pasa al siguiente. La ventaja de esta configuración es que se va a conseguir un disco de mayor capacidad mediante discos de pequeño tamaño. Incluso se tienden a integrar clusters computacionales con soluciones de alta disponibilidad.

Sin embargo, tiene varias desventajas:

- No hay redundancia. Si un un disco falla, se perderá la mayoría de los datos o incluso todo el sistema de ficheros.
- No se va a optimizar el tiempo necesario para las operaciones de lectura/escritura (al menos en el caso general[9]).

8.4.2._RAID 0

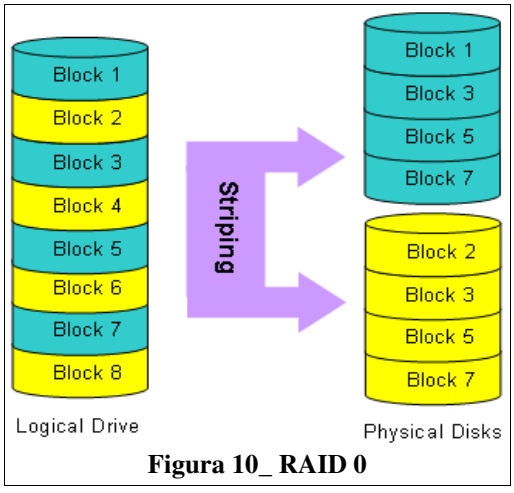

Figura 10_ RAID 0

También es conocido como modo "stripe". Los dispositivos deberían tener la misma capacidad (aunque no es una restricción). Las operaciones sobre el disco virtual se dividirán sobre dos discos físicos: por ejemplo una escritura de gran tamaño se puede dividir en trozos que se escribirán cada uno en un disco físico. Si uno de los discos es más grande que el otro, aún se puede utilizar este espacio pero esto implica hacer lecturas/escrituras de forma independiente en uno de los discos, con lo que se pierde eficiencia.

Las desventajas de esta configuración son:

8Los niveles 4 y 5 de RAID se implementan como independientes, mientras que los niveles 0 y 1 pueden ser implementados tanto independientes como en paralelo.

9 Si varios usuarios están trabajando sobre el mismo dispositivo lógico, quizá uno esté trabajando sobre el primer disco físico, otro sobre el segundo... con lo que sí se lograría alguna mejora

- Como en lineal, aquí tampoco hay redundancia y además, si falla un disco, en este caso, no va a poder recuperarse ni un sólo dato. Si se quita un disco de un RAID-0, el RAID no va a limitarse a omitir un bloque de datos consecutivo, sino que va a llenar de pequeños huecos todos los discos físicos integrantes. Las herramientas de recuperación para los distintos sistemas de ficheros, no van a ser capaces de recuperar demasiada información.
- Las lecturas/escrituras se hacen más eficientes, ya que se están haciendo en paralelo en los dispositivos. Esta es la principal razón para construir un RAID-0.

8.4.3._RAID-1

Este nivel de RAID, mantiene un imagen exacta de la información de un disco en el otro. Por supuesto, los discos deben tener el mismo tamaño. Si un dispositivo es más grande que el otro, el RAID será del tamaño del más pequeño.

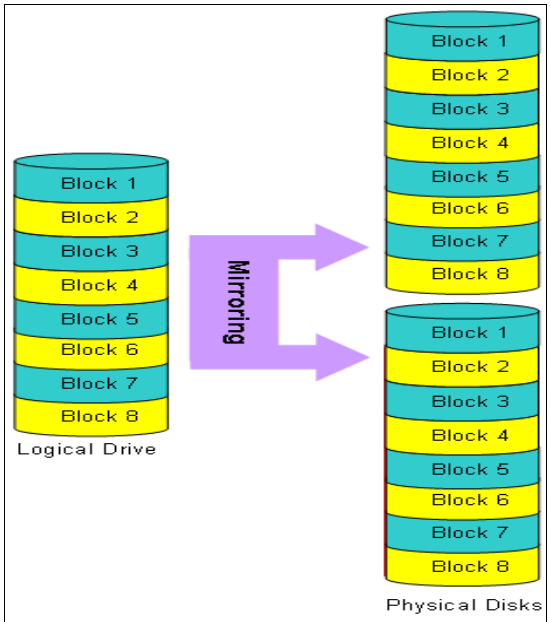

Figura 11_ RAID 1

Obtenemos las siguientes ventajas:

- En este caso sí existe redundancia. RAID-1 puede ser usado en dos o más discos con posibilidad de tener discos de repuesto (spare-disks).
- Si se trata de un RAID-1 de N discos, podrían fallar hasta N-1 sin que perdiéramos ningún dato. Si existen discos de repuesto y el sistema (drivers SCSI, chipset IDE,...) sobrevive, la reconstrucción de la imagen copiada comenzará inmediatamente en uno de los discos de repuesto, después de la detección del fallo.
- Las lecturas tienen buena eficiencia, especialmente si se tienen múltiples lectores trabajando con lecturas que hacen muchas búsquedas de bloque. El código del RAID emplea un algoritmo de balanceo de carga, de manera que se toma el dato del disco cuyos cabezales están más cerca del dato buscado.

Sin embargo, esta configuración también tiene desventajas. La eficiencia de las escrituras es menor que en el caso de un sólo dispositivo: hay que enviar diversas copias de los datos que se quieren escribir a cada disco en el array. Cuando se trata de RAIDs-1 de gran tamaño, esto puede llegar a ser un problema, ya que puede llegar a saturarse el bus PCI con estas copias extra. En este caso las soluciones hardware son una buena ayuda.

8.4.4._RAID 4

Este nivel de RAID no es muy usado. Puede realizarse con tres o más discos. En lugar de copiar completamente la imagen de la información, en este caso se mantiene en un disco una información de códigos de error y en los discos restantes se construye un RAID-0. Si se tienen N discos y S es el tamaño del más pequeño, el dispositivo virtual tendrá un tamaño de (N-1)*S.

Esta configuración tiene como ventaja que si un dispositivo falla, se puede usar la información de control de errores para restaurar los datos. Pero un fallo que afecte a un disco del RAID-0 y al disco con la información para restaurar, dejará inservible el conjunto.m

Sin embargo, la razón por la que este nivel no es muy utilizado es que la información copiada se mantiene en un sólo dispositivo. Esta información debe ser actualizada cada vez que se escribe en los otros discos. De este modo, es fácil que el disco de copia se convierta en un cuello de botella si no es mucho más rápido que los otros. Sin embargo, si se dispone de varios discos lentos y uno rápido, esta es una buena solución.

8.4.5._RAID-5

Quizá este sea el RAID más útil cuando se quiere combinar un gran número de discos físicos y mantener alguna redundancia. RAID-5 puede usarse con tres o más discos y con discos de repuesto. El tamaño resultante se calcula del mismo modo que en RAID-4 ((N-1)*S). La diferencia principal entre RAID-5 y RAID-4 es que la información copiada es distribuida entre varios dispositivos de los participantes. Con esto, se evita el cuello de botella que podía producirse en RAID-4.

En este caso, tanto las lecturas como las escrituras van a ser más eficientes, pero puede ser difícil predecir cuánto mejores. Las lecturas serán similares en eficiencia al caso del RAID-0. Las escrituras, sin embargo, pueden llegar a ser bastante pesadas o similares a las escrituras de un RAID-1. La eficiencia de las escrituras va a depender fuertemente de la cantidad de memoria de la máquina y el patrón de uso del array.

8.4.6._DISCOS DE REPUESTO[10]

Son discos que no forman parte del RAID hasta que uno de los discos activos falla. Cuando se detecta un fallo de disco, se marca el dispositivo que ha fallado con una etiqueta que lo indique y se comienza inmediatamente la reconstrucción en el primer disco de repuesto disponible.

Los discos de repuesto añaden seguridad extra, especialmente para los RAID-5 que son difíciles de mantener físicamente. Se puede permitir al sistema funcionar un tiempo con un dispositivo caído, porque los datos se están preservando en el disco de repuesto. Sin embargo, no se

10 spare-disks

tiene seguridad de que el sistema vaya a seguir funcionando: el dispositivo virtual puede manejar perfectamente fallos de disco, pero los drivers SCSI pueden generar errores, o el chipset IDE puede bloquearse,... o podría ocurrir casi cualquier otra cosa.

Por otra parte, cuando se comienza una reconstrucción en caliente en un disco de repuesto, el RAID puede comenzar a leer información desde los otros discos para reconstruir la información redundante. Si varios discos contienen bloques fallidos, los errores se van a transmitir con la reconstrucción. Esto puede llevar a una caída completa del RAID. Si se realizan frecuentes backups de todo el sistema de ficheros contendido en el RAID haremos más improbable esta situación. Es importante tener claro que el RAID no es un sustituto de los backups.

9._SOFTWARE PARA CLUSTERs

Como vimos anteriormente, en términos generales, un cluster se va a poder dividir en dos partes principalmente. Primero, el hardware de interconexión de los elementos del cluster y segundo, la parte software: un sistema operativo adaptado (por ejemplo GNU Linux con un kernel modificado), compiladores especiales y aplicaciones; en general, elementos software que permitan a los programas que se ejecutan sobre el cluster aprovechar todas sus ventajas.

9.1._SOFTWARE PARA CLUSTERs COMPUTACIONALES

Aunque por la naturaleza de los problemas a los que dan solución este tipo de clusters lo normal es desarrollar aplicaciones específicas, existen algunas soluciones generales.

Dentro del proyecto Beowulf se puede encontrar un kernel modificado y algunas herramientas y librerías usadas para presentar el cluster como un único sistema. La idea es que los procesos que se ejecutan en los nodos esclavos son manejados por un nodo maestro, dando la impresión de que el cluster es un único sistema. Ya que el kernel que mantiene este proyecto está modificado, es el propio proyecto Beowulf quien se encarga de la distribución de nuevas versiones y parches.

Hay otros proyectos en los que también se utiliza la idea de un único sistema. OpenMosix es un conjunto de extensiones del kernel estándar, así como herramientas desarrolladas para que el uso del cluster sea más eficiente. SCE (Scalable Cluster Environment) es un conjunto de herramientas que permiten construir y usar un cluster Beowulf. Bproc, el programa núcleo del proyecto Beowulf tiene la habilidad de presentar el sistema como único y es usado en ClubMask, Kickstart, cfengine, the Maui scheduler, LAM/MPI, ...

Existen otros clusters computacionales que no modifican la forma en que funciona el kernel. Éstos utilizan otros medios para ejecutar tareas y mostrar información sobre ellas. Cplant, el Ka Clustering Toolkit, y OSCAR permiten construir, usar y administrar clusters de este modo.

Existen algunos sistemas operativos. que han surgido en base a las necesidades de los clusters computacionales. Proporcionan distintas funcionalidades, desde herramientas HPC generales hasta ofrecer instalaciones específicas de clusters. Warewulf es una distro que se configura y copia en un CD y con la que se puede arrancar los nodos esclavos y que también puede utilizarse ejecutándola de forma independiente para tener al instante un cluster.

Otra herramienta de este tipo es ClusterKnoppix: ClusterKnoppix es básicamente una Knoppix, un fantástico CD arrancable, con un kernel openMosix. Incluye toda la funcionalidad básica de openMosix, y basta con arrancar desde el CD en el servido y ejecutar unos cuantos clientes por red para tener al instante un cluster.

9.2._SOFTWARE PARA CLUSTERs DE ALTA DISPONIBILIDAD

La herramienta Kimberlite está especializada en almacenamiento compartido de datos y mantenimiento de la integridad de los mismos, con lo que es interesante para este tipo de clusters.

El paquete Piranha permite a los servidores Linux proveer alta disponibilidad sin la necesidad de invertir cantidades mayores en hardware, ya que basa el clustering en software. Puede ser utilizado de dos modos, bien como una solución de alta disponibilidad en dos nodos o una solución de balanceo de carga multinodo. Piranha puede configurar un servidor de respaldo en caso de fallo de la contraparte. También puede hacer que el cluster aparezca como un servidor virtual.

Uno de los proyectos más conocidos en este campo es el Linux-HA (High Availability Linux Project), dentro de este proyecto se proporcionan herramientas variadas para la construcción y explotacion de este tipo de clusters. Otro proyecto es Poor Man' sHigh Availability, que usa el servicio de DNS para dar una solución simple para disponibilidad de servidores Web.

9.3._SOFTWARE PARA CLUSTERs DE BALANCEO DE CARGA.

Uno de los proyectos más conocidos en este campo es el Linux Virtual Server Project (LVS). Usa balanceo de carga para pasar las peticiones a los servidores y puede virtualizar casi cualquier servicio TCP o UDP, tales como HTTP(S), DNS, POP, IMAP, SMTP, y FTP. Muchos proyectos de balanceo de carga están basados en LVS (UltraMonkey, Piranha, Keepalived).

El Zeus Load Balancer no está basado en LVS pero ofrece una funcionalidad similar. Pen es otro proyecto no basado en LVS y que es un simple balanceador de carga para servicios basados en TCP como HTTP o SMTP.

Por último, el balanceo de carga y software de monitorización de Turbolinux Cluster Server permiten la detección y recuperación de errores de hardware y software (si la recuperación es posible)

9.4._SOFTWARE USADO EN CLUSTERs Y PARA USAR CLUSTERs

9.4.1._SISTEMAS DE ARCHIVO

Los sistemas de archivo más utilizados son: OpenAFS (Opened Andrew FileSystem), Coda, GFS(Global FileSystem) e InterMezzo.

9.4.2._SOFTWARE DE CONFIGURACIÓN E INSTALACIÓN

Cuando se tiene que tratar con un cluster de cientos de nodos, instalarlos y configurarlos es una tarea pesada y aburrida, sobre todo cuando cada nodo es una copia de los demás. Cualquier herramienta que ayude en este proceso debe formar parte del arsenal del administrador del cluster.

FAI (Fully Automatic Installation) es un sistema no interactivo para instalar la distribución Debian.

System Installation Suite es una herramienta de instalación basada en imágenes. Las imágenes se crean y son almacenadas en un servidor de donde los nodos las copiaran cuando las necesiten. Red Hat, Mandrake, SuSE, Conectiva y Turbolinux están soportados por SIS. Con este proyecto están relacionados otros tres: SystemImager, System Instaler y System Configurator.

9.4.3._SOFTWARE DE MONITORIZACIÓN Y ADMINISTRACIÓN

En este apartado debemos señalar ClusterIt, Ganglia, Performance Co-Pilot, MOSIXVIEW, LVSmon, Syncopt, Fsync, Ghosts y pconsole.

9.4.4._ENTORNOS DE PROGRAMACIÓN Y EJECUCIÓN

La PVM (Parallel Virtual Machine) es una interfaz de paso de mensajes que permite a un conjunto heterogéneo de máquinas funcionar como un cluster. Las aplicaciones que usan esta interfaz pueden estar escritas en C, C++ o Fortran. PVM++ proporciona una librería de uso sencillo para programar para PVM en C++.

Otra interfaz de paso de mensajes es MPI (Message Passing Interface), de la que hay varias implementaciones LAM/MPI y MPICH por ejemplo.

10._EJEMPLOS DE CLUSTERS

Todos conocemos Google. Este buscador debe atender, en promedio, más de 200 millones de consultas por día (2,315 consultas por segundo). Debe almacenar la información indexada de más de 3 mil millones de páginas web en 36 idiomas y más de 35 millones de documentos en formatos distintos a HTML accesibles a través de su máquina de búsqueda (web profundo). Google opera con un Cluster de más de 10,000 sistemas Linux.

WebArchive.Org es una hemeroteca digital de Internet, donde se puede encontrar la información de más de 5 años de la web, representando esto una base de datos con más de 100TB de información. A través de su Máquina de Tiempo (The Wayback Machine) se pueden consultar más de 10 mil millones de páginas. La tecnología de cluster utilizada por la Máquina del Tiempo consta de un Cluster de cerca de 400 máquinas.

La NASA usa Beowulf, desde 1994 cuando surgió el modelo hasta hoy, que siguen utilizando este tipo de clusters para el trabajo en laboratorios universitarios y equipos de investigación.

NOAA (The National Oceanic and Atmospheric Administration) utiliza varios clusters de diversas tecnologías en sus proyectos. Su grupo HPCC (High Performance Computing and Communications) trabaja sobre diversas áreas de supercomputación, balanceo de carga y alta disponibilidad.

11._REFERENCIAS Y BIBLIOGRAFÍA

11.1._REFERENCIAS

11.1.1._CLUSTERS CON GNU/LINUX

-Sobre multiprocesadores en general:
http://icaro.eii.us.es/index.php?op=allnews&id=36&PHPSESSID=43a2eba7c12339d197650504c1baf516
-Paralell-Processing-HOWTO
-Esta página contiene información y links sobre clusters con GNU/Linux. http://lcic.org/
-Linux Documentation Project http://www.tldp.org
-Sobre clusters con Linux: http://www.fisica.uson.mx/carlos/LinuxClusters/
-Linux-Cluster-HOWTO
-¿Cómo construir un cluster?
http://www.mcsr.olemiss.edu/bookshelf/articles/how_to_build_a_cluster.html
-Un documento interesante sobre clusters computacionales sobre pc's:
http://www.fisica.uson.mx/carlos/LinuxClusters/cluster-computing2000.pdf
-Sobre alta disponibilidad: http://www.linuxvirtualserver.org/HighAvailability.html
-Artículos de Linux Magazine: http://www.linux-mag.com/1999-05/extreme_01.html
http://www.linux-mag.com/2000-10/clustering_01.html
http://www.linux-mag.com/depts/extreme.html
-High-Availability Linux Project: http://linux-ha.org/
-The Open Cluster Framework (OCF) opencf.org/documents/OCF.pdf
-Linux-High-Availability-HOWTO
-Diseñando sistemas de alta disponibilidad:
http://jo.morales0002.eresmas.net/pdfs/disponibilidad.pdf

11.1.2._SOFTWARE Y PROYECTOS DE CLUSTERING

-Software libre para construir clusters. En este enlace se encuentran referencias a los distintos productos. http://freshmeat.net/articles/view/458/
-Sobre Beowulf http://www.beowulf.org/
-Sobre el proyecto alfa (cluster Beowulf) de la Universidad de Sonora: http://alfa.acarus.uson.mx/
-Articulos de Thomas Sterling, uno de los creadores del primer Beowulf:
http://beowulf.gsfc.nasa.gov/tron1.html
http://beowulf.gsfc.nasa.gov/tron2.html
-De dónde proviene el nombre de nombre Beowulf:
http://www.bulfinch.org/fables/bull42.html
http://www.bulfinch.org/fables/welcome.html
-¿Cómo construir un Beowulf Linux Cluster?: http://www.mcsr.olemiss.edu
- Sobre Mosix http://www.mosix.org/
-The Message Passing Interface (MPI) standard: http://www-unix.mcs.anl.gov/mpi/
- Warewulf. Distro para clusters: http://freshmeat.net/projects/warewulf

-Turbo Linux Cluster Server http://www.turbolinux.com/products/tlcs8/
-Linux PC Clustering Project http://hp-linux.cern.ch/
-Sobre distros para clusters : http://lcic.org/distros.html#cd
-Sobre el Linux Scalability Project
http://www.citi.umich.edu/projects/linux-scalability/index.html-
-Linux-Virtual-Server-HOWTO
-Sobre el software OSCAR: http://oscar.sf.net/
-Sobre MPI: http://www-unix.mcs.anl.gov/mpi/

11.1.4._RAID

-Software-RAID-HOWTO

-Sobre los distintos niveles de RAID, un resumen muy bueno:
http://www.smdata.com/NivelesRAID.htm
- También sobre RAID: http://www.monografias.com/trabajos14/discosraid/discosraid.shtml

11.2._BIBLIOGRAFÍA

- Parallel Computer Architecture. A hardware/software approach. David E. Culler Jaswinder Pal Singh, Morgan Kaufmann 1998. ISBN 1-55860-343-3

X._ANEXOS

A._MODELOS DE COMPUTACIÓN EN MÁQUINAS MULTIPROCESADOR

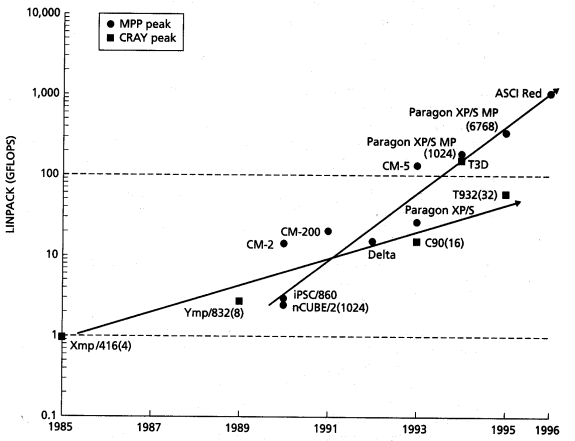

Figura 12_Comparativa entre el CRAY (supercomputador vectorial) y los sistemas MPP (Massively parallel processors) más rápidos

En los sistemas monoprocesador, la CPU es un todo monolítico, está separada de la memoria principal y de la E/S, alcanzando grandes velocidades. En el dominio de los multiprocesadores se va a perseguir el aprovechamiento de las capacidades de varios monoprocesadores para aumentar las prestaciones totales de un sistema, presentándose éste como una generalización de una computadora. Se ha declarado en distintas oportunidades el cercano estancamiento de la tecnología monoprocesador: continuar con el incremento en prestaciones de los monoprocesadores supone cada vez un mayor coste e incluso podría obligar a un cambio completo de paradigma tecnológico. Ante esta perspectiva, los multiprocesadores incrementan el rendimiento de forma menos costosa que lo que supone el diseño de un nuevo procesador. Esta solución, además, es más flexible para el software que un nuevo diseño.

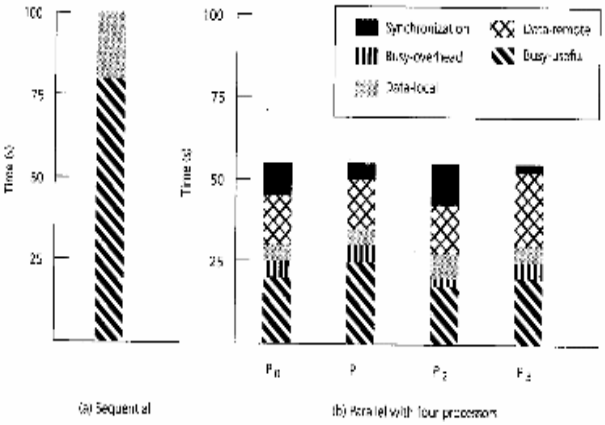

Figura 13_ Tiempo de ejecución en monoprocesadores y multiprocesadores.

Por otra parte, sin embargo, el anuncio repetido del fin de los sistemas monoprocesador se ve interrumpido periódicamente por el nacimiento de alguna nueva arquitectura: segmentación, vectorización, superescalaridad, hyperthreading... Estos avances vienen motivados por la fuerte inversión en investigación para estos sistemas. Además, los sistemas multiprocesador son objeto de distintas críticas: bien por la lentitud natural de las comunicaciones en estos sistemas, bien por la limitación de la capacidad de paralelización de las aplicaciones. Idealmente, podríamos pensar que el tiempo de ejecución de una aplicación en un sistema monoprocesador va a experimentar una mejora lineal en relación al número de multriprocesadores, al ejecutarse en un sistema multiprocesador: $t_{eje,mult}=t_{ejec,uni}/N$. Sin embargo, esta relación lineal es imposible en la práctica debido a los retardos de la red de interconexión y las instrucciones de control de la ejecución, que se van a superponer a las instrucciones de la aplicación que estamos ejecutando. Asimismo, vamos a tener distintas

configuraciones de memoria: global, local o algún sistema mixto. La forma de organizar la memoria también va a influir notablemente en el retardo añadido (overhead) al ejecutar cualquier aplicación en un sistema multiprocesador.

La principal limitación de los multiprocesadores viene siendo la necesidad de software paralelo, difícil de desarrollar, depurar y mantener. Es necesario adaptar las aplicaciones para trabajar en entornos multiprocesador, sobre todo si se quieren aprovechar al máximo sus ventajas. Aunque ésta ha sido una limitación tradicionalmente, cada vez se avanza más en software paralelo, se crean más herramientas y lenguajes y se realizan más estudios sobre algoritmos.

Las líneas de investigación actuales para estos sistemas vienen siendo:

- Adaptar el software a estas arquitecturas: nuevos algoritmos, nuevos lenguajes paralelos, compiladores capaces de llevar a cabo la paralelización de forma automática, mejora de la portabilidad.
- Ocultar y/o reducir la latencia remota, bien por hardware o por software.
- Investigación de la relación entre topologías y rendimiento de aplicaciones específicas.

Podemos realizar distintas clasificaciones de los multiprocesadores en función de los distintos criterios escogidos. En la figura 14 se muestran algunas de ellas, dentro de la clasificación más general de computadores paralelos. A efectos de ampliar y profundizar los conceptos sobre los que trata este documento, señalaremos la clasificación basada en la organización de la memoria, ya que esta clasificación va a imponer distintos modelos de programación. Esta diferenciación entre modelos de programación ilustrará debidamente la necesidad de una separación entre software y hardware y las ventajas obtenidas al crear librerías y herramientas que actúen como interfaz entre el software de aplicación y el hardware de un sistema multiprocesador.

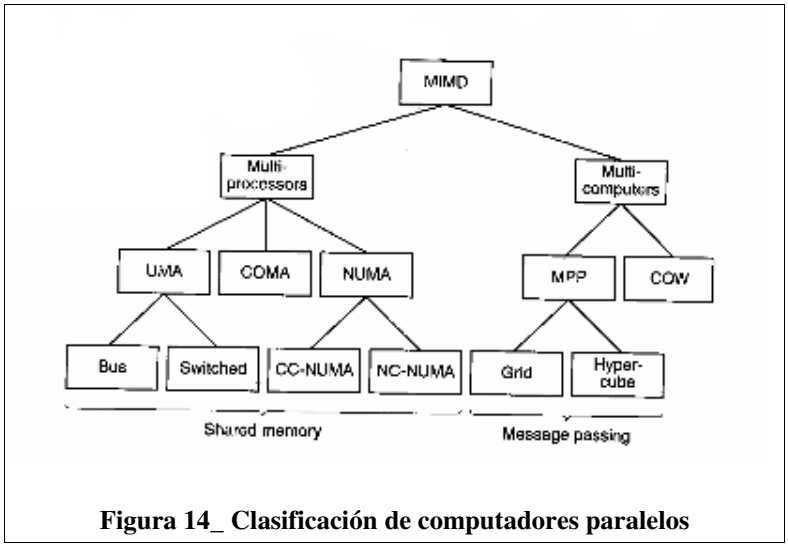

Figura 14_ Clasificación de computadores paralelos

A.1._CLASIFICACIÓN EN FUNCIÓN DE LA ORGANIZACIÓN DE LA MEMORIA.

En general, vamos a distinguir tres tipos de organización de la memoria en sistemas multiprocesador. Aunque los sistemas comerciales van a emplear modelos mixtos, esta clasificación sencilla, ayuda suficientemente a comprender las implicaciones que se derivan de cada modelo.

- UMA (Uniform Memory Access). En estos sistemas la memoria es global para todos los procesadores y los accesos tienen los mismos retardos para todos los nodos. Requiere un hardware sencillo aunque su principal desventaja es el cuello de botella en que se convierte el acceso a memoria y los problemas de escalabilidad que ello supone.
- NUMA (NonUniform Memory Access). El espacio de direcciones sigue siendo global pero no así la memoria, que va a ser local a cada procesador aunque accesible por todos los demás. Los accesos tienen retardos no uniformes dependiendo de si el dato buscado se encuentra o no en memoria local. El hardware de la red es complejo ya que se van a necesitar interconexiones especiales que minimicen los retardos en el acceso a memoria remota. Pero, por otra parte, se mejora la escalabilidad.
- MPM (Message Passing Machines). En este caso la memoria además de pertenecer a cada procesador, va a ser vista mediante un mapa de direccionamiento distribuido. Cada procesador es un computador independiente del resto. Se mejora la escalabilidad notablemente, siendo la red la parte más compleja de escalar. El overhead de comunicación, en cambio, es el más alto. Las arquitecturas MPM son las que hemos venido relacionando con los clusters.

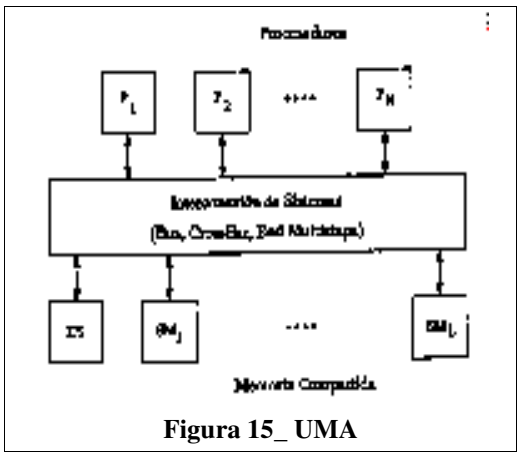

Figura 15_ UMA

Los distintos modelos implican consideraciones particulares a la hora de programar software. Las máquinas UMA y NUMA, al tener el espacio de direcciones compartido, se adaptan fácilmente a la programación multihilo (multithread). También se puede usar para la programación multiprocesos, realizándose la comunicación entre procesos mediante el intercambio de variables en la memoria compartida. En este caso, al igual que en el de las máquinas NUMA, surge un problema en la coherencia de los datos almacenados en las cachés de los procesadores; para las máquinas UMA este problema es fácilmente controlable mediante hardware, no así en NUMA, cuya arquitectura complica el hardware necesario para el mantenimiento de la coherencia. Esta coherencia se mantiene a veces vía software, aunque este software debe permanecer en un nivel inferior al del software de aplicación. En cualquier caso, se exija o no al programador tener en cuenta el mantenimiento de la coherencia de forma explícita, sí que es inevitable el retardo para accesos remotos en máquinas NUMA, con lo que el programador debe tratar de minimizar estos accesos.

La programación en arquitecturas MPM se orienta a multiprocesos. La comunicación se realiza mediante mensajes entre los nodos, para ello se requieren librerías y herramientas que abstraigan esta comunicación. La comunicación se hace de forma explícita con llamadas a métodos de estas librerías, lo que obliga a los programadores a tener en cuenta la latencia de las comunicaciones. El overhead de comunicación es muy alto, por otra parte.

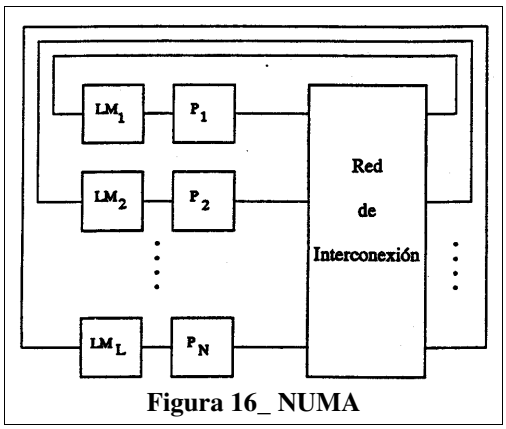

Figura 16_ NUMA

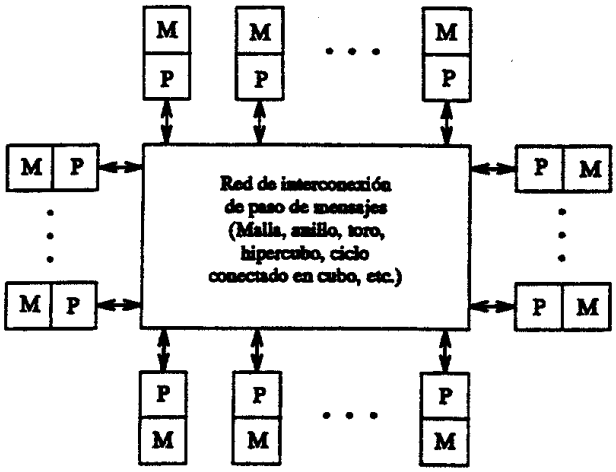

Figura 17_ MPM

Como vemos, la existencia de librerías de comunicación minimiza los detalles de bajo nivel que tienen que ser tenidos en cuenta por los programadores, si bien no los eliminan. Se asegura, además la portabilidad del software desarrollado, siempre y cuando las librerías abstraigan convenientemente la arquitectura sobre la que trabajan. Ya vimos, como la separación entre software y hardware en los clusters, había supuesto un fuerte impulso para su difusión y desarrollo, convirtiéndolos en una opción versátil y flexible en muchos casos.