UNIVERSIDAD TECNICA DE AMBATO

FACULTAD DE INGENIERIA EN SISTEMAS

Tema:

"Estudio y diseño de cluster Beowulf bajo la plataforma Linux para la elaboración de un Webcluster"

AUTORES:

EDIT MARILI CORREA BAEZ

MARIANELA DEL PILAR MOREJON ABRIL

DIRECTOR :

ING. DAVID GUEVARA

ASESOR:

ING. JAIME RUIZ M. Sc.

Tesis de grado, previa a la obtención del Título de Ingeniero en Sistemas.

Ambato-Ecuador

ENERO/2005

AGRADECIMIENTO

A Dios por darnos el don maravilloso de la vida y con ello la oportunidad de poder realizarnos como personas y como profesionales.

A la Facultad de Ingeniería en Sistemas, al personal docente por sus conocimientos transmitidos y al personal administrativo por su colaboración.

A nuestros Padres por brindarnos su amor, apoyo económico, moral y por creer siempre en nosotras, a nuestros hermanos y amigos por darnos su apoyo para cumplir nuestra meta.

Al Ing. David Guevara y al Ing. Jaime Ruiz M. Sc por ser las personas que nos han guiado y brindado sus conocimientos dentro de desarrollo de nuestra tesis.

DEDICATORIA

A mis Padres por brindarme su apoyo incondicional, además que han sido el eje fundamental de mi vida y mi carrera estudiantil.

A mi esposo y mi hija por estar siempre a mi lado y brindarme su amor, comprensión para lograr esta meta.

A mis hermanos y amigos por estar siempre prestos a ayudarme en todos los momentos de mi vida.

Al Señor del Consuelito por ser mi amigo y confidente incondicional, por estar siempre a mi lado y ser la luz que ilumina mi camino.

Marianela Morejón

A mi Madre ya que sin su apoyo y sin su amor incondicional no hubiera culminado la carrera y llegar a ser una profesional

A mi Hermana y Familiares que han sido un pilar fundamental los cuales han estado en los momentos de alegría y tristeza

A la Virgen de Baños por haberme colmado de bendiciones y guiado por el camino de Fe

A todos mis amigos que de una u otra forma han sido parte importante de mi vida estudiantil y profesional

Edit Correa

DECLARACION DE AUTENTICIDAD

Nosotras, Edit Marilí Correa Báez y Marianela del Pilar Morejón Abril.

Números de cédula de identidad 050233118-4 y 180323866-4

Declaramos que la investigación enmarcada en el diseño de la tesis es absolutamente original, auténtica y personal. En tal virtud, declaramos que el contenido, efectos legales y académicos que se desprenden del trabajo de tesis son y serán de nuestra sola y exclusiva responsabilidad legal y académica.

---------- **----------**

Edit M. Correa B **Marianela del P. Morejón A.**

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

Debido al avance de la tecnología y de la Informática, en la actualidad los equipos informáticos se quedan anticuados en un plazo corto de tiempo.

Para aprovechar estos viejos equipos; se puede construir un cluster, de forma que la capacidad que obtengamos pueda llegar a compensarnos ante la inversión a realizar en equipos más potentes.

Los programas con gran carga tardarían bastante tiempo en realizarse en uno de estos viejos ordenadores. Se pretende usar estos ordenadores para construir un cluster y con él reducir este tiempo, mediante el reparto de la carga entre sus nodos. El surgimiento de plataformas computacionales de comunicación y procesamiento estándares de bajo costo, nos han brindado la oportunidad de crear herramientas computacionales del dominio público o de costo razonable.

Una de las herramientas que está surgiendo en la actualidad son los llamados cluster Beowulf, los cuales presentan diversas capacidades para el cómputo paralelo con un relativo alto rendimiento.

El software también ha evolucionado mucho y en particular Linux, un sistema operativo que ha sido creado bajo una licencia libre. Linux es un conjunto de software que se puede utilizar, modificar y compartir sin tener que pagar las grandes cantidades de dinero que cuesta su contraparte privada. Además de ser de fácil adquisición el software Linux ha mejorado mucho su rendimiento y hoy es considerado como uno de los sistemas operativos para aplicaciones criticas como la del cálculo intensivo.

CAPITULO I

1. GENERALIDADES

1.1 PLANTEAMIENTO DEL PROBLEMA

En muchas ramas de la Ciencia la complejidad de los problemas que se estudian, requieren contar con el acceso a una supercomputadora que pueda desarrollar varios miles de millones de operaciones por segundo. Las supercomputadoras tradicionales emplean procesamiento en paralelo, contienen arreglos de microprocesadores ultra rápidos que trabajan en sincronía para resolver problemas complejos como pronósticos numéricos del estado del tiempo o modelar estructuras complejas de la materia.

Esto dio origen a los clusters tipo Beowulf que se producen a partir de lo inadecuado de los superordenadores clásicos y de las oportunidades ofrecidas por las tecnologías emergentes de hardware y software.

El concepto de cluster nació cuando los pioneros de la supercomputación intentaban difundir diferentes procesos entre varias computadoras, para luego poder recoger los resultados que dichos procesos debían producir.

Con un hardware más barato y fácil de obtener se pudo perfilar que podrían conseguirse resultados muy parecidos a los producidos con aquellas máquinas mucho más costosas, como se ha venido probando desde entonces.

Los clusters han evolucionado para apoyar las actividades en aplicaciones que van desde supercomputo y software de misiones críticas, servidores web y comercio electrónico, bases de datos de alto rendimiento etc. El cómputo en clusters surge como resultado de la convergencia de varias tendencias que incluye, la disponibilidad de microprocesadores de alto rendimientos más económicos, redes de alta velocidad el desarrollo de la herramienta de software para cómputo distribuido de alto rendimiento y la creciente necesidad de potencia computacional para aplicaciones en las ciencias computacionales y comerciales.

Por otro lado, la evolución y escalabilidad que ha alcanzado el sistema operativo Linux, ha contribuido al desarrollo de muchas tecnologías nuevas, entre ellas la del cluster.

Esto hace que en la actualidad las empresas e instituciones compitan en un escenario que evoluciona con muchísima rapidez y en el que, ante el aumento

del número de competidores, la adaptación de los procesos productivos y de negocio a esa realidad cambiante resulta crítica a la hora de obtener y mantener la ventaja competitiva.

Este nuevo escenario obliga a someter continuamente a revisión y reinventar la forma en que las empresas hacen su trabajo, con el objetivo de mejorar la productividad y reducir los costos de software, hardware, mantenimiento, administración.

1.2 JUSTIFICACION

El presente Trabajo de Investigación tiene una importancia fundamental dentro del campo de la computación porque es una innovación Informática presentada como cluster.

Esta tecnología no es conocida ni explotada en nuestro medio, razón por la cual se hace necesario realizar una demostración de un tipo de cluster como es el de alto rendimiento Webcluster que sirva como base de futuras implementaciones tanto en empresas como en instituciones

Al ser un campo no difundido, se hace indispensable dar a conocer los nuevos paradigmas y los beneficios que estos ofrecen, como es el bajo costo de implementación lo que hace que las empresas que tiene los recursos económicos limitados y que no puede adquirir un Mainframe tengan acceso a una nueva alternativa, que es la implementación los clusters con la que tendrían los siguientes beneficios de acuerdo a las aplicaciones y ambientes:

- Incremento de velocidad de procesamiento, número de transacciones, tiempo de respuesta ofrecido por los clusters de alto rendimiento.
- Incremento de confiabilidad ofrecido por los clusters de alta disponibilidad.

El equilibrio de carga de red mejora la disponibilidad y la escalabilidad de los programas basados en el servidor de Internet, como son los servidores Web, los servidores de multimedia de transmisión por secuencia y los Servicios de Terminal Server, actuando como la infraestructura de equilibrio de carga y proporcionando información de control para aplicaciones de administración.

De hecho, el Beowulf en sí catalizaría un cambio de paradigma de la computación de alto rendimiento que dominaría el comienzo del siglo XXI.

1.3 OBJETIVOS

1.3.1 General

- Estudiar y Aplicar Clusters Beowulf bajo la plataforma Linux para la elaboración de un Webcluster.

1.3.2 Específico

- Analizar un Cluster Beowulf bajo la plataforma Linux
- Instalar, implementar, administrar un cluster de alto rendimiento.
- Aplicar el cluster de alto rendimiento bajo la plataforma Linux.

1.4 FORMULACION DE HIPOTESIS

El estudio y diseño de cluster Beowulf bajo la plataforma Linux para la elaboración de un Webcluster determinará el máximo rendimiento de los recursos existentes con un mínimo costo.

CAPITULO II

2. CLUSTERS

2.1 CLUSTERS NOCIONES GENERALES

2.1.1 Concepto De Clusters

Un cluster es un grupo de equipos independientes interconectados o unidos por una red de comunicación, que ejecutan una serie de aplicaciones de forma conjunta y aparecen ante clientes y aplicaciones como un solo sistema. Los clusters permiten aumentar la escalabilidad, disponibilidad y fiabilidad de múltiples niveles de red.

2.1.2 Como Funciona Un Cluster

En su parte central, la tecnología de Clusters consta de dos partes:

1.- **Un sistema operativo** que sirve para hacer modificaciones al kernel de Linux, un conjunto de compiladores y aplicaciones especiales, que permiten

que los programas que se ejecutan sobre esta plataforma tomen las ventajas de esta tecnología de Clusters.

2.- **La interconexión de hardware entre las máquinas (nodos) del Cluster**. Las interconexiones se realizan mediante una red Ethernet dedicada de alta velocidad. Es mediante esta interfaz que los nodos del Cluster intercambian entre si asignación de tareas, actualizaciones de estado y datos del programa.

2.2 BENEFICIOS DE LOS CLUSTERS

Los beneficios de elaborar un Cluster se dan en varios aspectos en una 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 cluster de balance de carga.
- Incremento de confiabilidad ofrecido por los clusters de alta disponibilidad. Por ejemplo en un sitio Web de mucho tráfico, si no se cuenta con un plan de alta disponibilidad cualquier problema menor de una tarjeta de red, puede hacer que el servidor quede completamente inutilizado.

2.3 CARACTERISTICAS DE UN CLUSTER

1. Para crear un cluster se necesitan al menos dos nodos.

Una de las características principales de estas arquitecturas es que exista un medio de comunicación (red) donde los procesos puedan migrar para computarse en diferentes estaciones paralelamente. Un solo nodo no cumple este requerimiento por su condición de aislamiento para poder compartir información.

2. Un cluster consta de 2 o más nodos conectados entre sí por un canal de comunicación funcional.

Cada nodo únicamente necesita un elemento de proceso, memoria, disco y un interfaz para comunicarse con la red del cluster.

3. Los clusters necesitan software especializado.

Cada clusters requiere un modelado y diseño del software distinto.

El software se debe dedicar a la comunicación entre los nodos.

Existen dos tipos de software que pueden conformar un cluster:

1.- Software a nivel de aplicación

Software generado por bibliotecas especiales dedicadas a clusters. Este tipo de software se sitúa a nivel de aplicación, se utilizan generalmente bibliotecas de

carácter general que permiten la abstracción de un nodo a un sistema conjunto, permitiendo crear aplicaciones en un entorno distribuido de manera lo más abstracta posible. Este tipo de software suele generar elementos de proceso del tipo rutinas, procesos o tareas, que se ejecutan en cada nodo del cluster y se comunican entre sí a través de la red.

2.- Software a nivel de sistema.

Este tipo de software se sitúa a nivel de sistema, suele estar implementado como parte del sistema operativo de cada nodo, o ser la totalidad de este.

Un cluster puede tener implementado a nivel de kernel parte del sistema y otra parte estar preparada a nivel de usuario.

2.4 ACOPLAMIENTO DE UN CLUSTER

2.4.1 Definición de Acoplamiento.

A menudo el nivel de acoplamiento se confunde las características que posee el cluster, ya que dependiendo de este acoplamiento se considera cluster a un sistema y no a otro.

Lo habitual es que a medida que el acoplamiento de un sistema decrezca deje de ser tomado en cuenta como cluster. El problema está en que no existe un límite inferior de acoplamiento definido, lo cual causa mucha confusión.

El concepto de acoplamiento debe ser definido en todos los clusters. Esto implica que a pesar de la diferencia de funcionalidad entre unos y otros, solamente entrarán en la categoría de clusters, aquellos cuyo software pueda definir de una forma única y clara su nivel de acoplamiento.

El acoplamiento de un sistema de tipo cluster se basa principalmente en las colaboraciones que existen entre los elementos de proceso, es decir, a más colaborativa, más acoplado.

Se entiende por acoplamiento del software a la integración que tengan todos los elementos software que existan en cada nodo.

Por ejemplo una aplicación GNOME que utiliza CORBA[1] puede llegar a ser tan acoplada como OpenMosix en el caso de que el número de colaboraciones sea mayor.

[1]CORBA(Common Object Request Broker Architecture)

Gran parte de la integración del sistema, es producida por la comunicación entre los nodos, y es por esta razón se define el acoplamiento.

En cualquier caso, el acoplamiento del software es una medida subjetiva basada en la integración de un sistema cluster a nivel general.

Se distingue entre 3 tipos de acoplamiento:

- Acoplamiento fuerte
- Acoplamiento medio
- Acoplamiento débil

2.4.2 Acoplamiento fuerte.

El caso de acoplamiento más fuerte que se puede dar es que solamente haya una imagen del kernel del sistema operativo, distribuido entre un conjunto de nodos que la compartirán. Es fundamental poder acceder a todas las partes de este sistema operativo, que están estrechamente relacionadas entre sí y distribuidas entre los nodos.

Este caso es el que se considera como más acoplado, de hecho no está catalogado como cluster, sino como sistema operativo distribuido.

Por ejemplo los cluster SSI[2] (Sistema de una sola Imagen) es aquel que ve una misma imagen del sistema en todos los nodos, pero los nodos tienen su propio sistema operativo, aunque estos sistemas están estrechamente relacionados para dar la sensación a las aplicaciones que los nodos son idénticos y se acceda de una manera homogénea a los recursos del sistema total. Si arranca o ejecuta una aplicación, ésta verá un sistema homogéneo, por lo tanto los kernels tienen que conocer los recursos de otros nodos para presentarle al sistema local los recursos que encontraría si estuviera en otro nodo.

2.4.3 Acoplamiento medio.

A este grupo pertenece un software que no necesita un conocimiento tan exhaustivo de todos los recursos de otros nodos, pero que sigue usando el software de otros nodos para aplicaciones de muy bajo nivel.

Por ejemplo tenemos: OpenMosix y Linux-HA.

Un cluster OpenMosix necesita que todos los kernels sean del mismo sistema operativo y que usen un parche compatible.

[2]SSI(Single System Image)

2.4.4 Acoplamiento débil.

Son los casos en los que los programas se dividen en diversos nodos y por tanto se necesitan pero que no están a un nivel tan bajo. Generalmente se basan en aplicaciones construidas por bibliotecas preparadas para aplicaciones distribuidas, agrupadas en un conjunto de aplicaciones específicos que generan el cluster en sí. Es el software que tiene mayor número de ejemplos, como PVM[3], MPI[4], CORBA[5], *etc.* los cuales por sí mismos no funcionan solos, y hay que dotarles de una estructura superior que utilice las capacidades del cluster para que éste funcione.

Existen otras características necesarias, por ejemplo:

- Mejora sobre la disponibilidad
- Mejora del rendimiento

Otra de las características que se requiere generalmente de un cluster es que presente un entorno de trabajo confiable.

[3]PVM(Parallel Virtual Machine)
[4]MPI(Message Passing Interface)
[5]CORBA(Common Object Request Broker Architecture)

2.5 FACTORES DE CLASIFICACION DE LOS CLUSTERS

En general la clasificación de los clusters se hace en base a factores de diseño:

- Acoplamiento
- Control
- Homogeneidad

2.5.1 Control

Este parámetro implica el modelo de gestión que propone el cluster. Este modelo de gestión hace referencia a la manera de configurar el cluster y es dependiente del modelo de conexión o colaboración que surgen entre los nodos.

Puede ser de dos tipos:

- **Centralizado:** en un modelo centralizado, se hace uso de un nodo llamado maestro desde el cual se puede configurar el comportamiento de todo el sistema, por otro lado, este nodo es un punto crítico del sistema.
- **Descentralizado:** en un modelo distribuido cada nodo debe administrarse y gestionarse en un principio por sí mismo, también pueden ser gestionados mediante aplicaciones de más alto nivel de manera centralizada, pero la mayoría de la gestión que hace el nodo local

es leer archivos de configuración de su propio nodo. Es propio de sistemas distribuidos, como ventaja tiene que presenta más tolerancia a fallos como sistema global, y como desventajas que la gestión y administración de los equipos requiere más tiempo.

Por otro lado, el factor de gestión o control de un cluster es importante a la hora de monitorizar su comportamiento ya que estos sistemas deben estar bien controlados para ver que realmente hacen su trabajo correctamente.

2.5.2 Homogeneidad Del Cluster

- **Homogéneos:** formados por equipos de la misma arquitectura. Todos los nodos tienen una arquitectura y recursos similares, de manera que no existen muchas diferencias entre cada nodo. Una mala gestión o diseño de red puede romper esta homogeneidad del cluster haciendo que se tarde más tiempo en acceder a un nodo, esto también representa un fallo en la homogeneidad.

- **Hetereogéneos:** formados por nodos con distinciones que pueden estar en los siguientes puntos.

- Tiempos de acceso distintos
- Arquitectura distinta
- Sistema operativo distinto
- Rendimiento de los procesadores o recursos sobre una misma arquitectura distintos

El uso de arquitecturas distintas o distintos sistemas operativos impone que exista una biblioteca que haga de interfaz, e incluso una sintaxis de notación abstracta en la capa de presentación que utilice la interfaz de comunicación del sistema distribuido o cluster, esto hace que este tipo de cluster se consideran implementados a nivel de aplicación.

Entre estos factores de diseño que implementan algunos clusters tenemos:

- Soporte de alta disponibilidad y confiabilidad.
- SSI o Sistema de una Sola Imagen
- Manejo y gestión coordinada de los trabajos o tareas que corre el cluster.
- Implementación de comunicaciones eficientes.

Las metas que se proponen en un cluster SSI[6] son proveer simultáneamente de:

- Alta disponibilidad

[6] SSI(Sistem Single Image)

- Escalabilidad
- Facilidad de gestión

Los cluster SSI no solamente se van pareciendo cada vez más a clusters del tipo HA[7], sino que también se encargan de balancear la carga a varios niveles y obtener mejoras de rendimiento del sistema.

2.6 CLASIFICACIÓN SEGÚN EL SERVICIO PRIORITARIO

Generalmente el diseño de un cluster se realiza para solucionar problemas de:

- Mejora de rendimiento
- Abaratamiento del coste
- Distribución de factores de riesgo del sistema
- Escalabilidad

El modelo de los clusters permite que la mejora de rendimiento sea evidente respecto a grandes Mainframes a un precio realmente asequible. Lo que explica a su vez el segundo punto, acerca del coste de los cluster, que permite relaciones rendimiento precio que se acercan a un margen lineal dependiendo del cluster implementado.

[7] HA(High Available)

Por otro lado esta la distribución de riesgos. La mayoría de las empresas o usuarios, tiene sus servicios, aplicaciones, bases de datos o recursos en un solo ordenador, o dependientes de un solo ordenador. La distribución de factores de riesgo a lo largo de un cluster o la distribución de funcionalidad (en casos más generales) a lo largo de un cluster nos permite de una forma única, obtener la funcionalidad de una manera más confiable, ya que si una máquina cae, otras pueden hacer el servicio, o funcionalidad por esta.

La **escalabilidad**. Cuanto más escalable es un sistema, menos cuesta mejorar el rendimiento, lo cual abarata el coste, y en el caso de que el cluster lo implemente distribuye más el riesgo de caída de un sistema.

- Clusters de alto rendimiento
- Clusters de alta disponibilidad
- Clusters de alta fiabilidad

CAPITULO III

3. CLUSTER HA

3.1 INTRODUCCIÓN

Con el actual ritmo de crecimiento del comercio y el movimiento de datos de todo tipo en Internet (más de un 100% anual) y la incuestionable importancia de la informática en las empresas actuales de cualquier tamaño, es cada día más importante que los sistemas informáticos de éstas puedan funcionar de forma ininterrumpida y sin errores las 24 horas del día, 7 días a la semana y 365 días al año, ya sea para dar soporte interno (contabilidad, control de personal, desarrollo...) como para ofrecer servicios a través de Internet (comercio electrónico, correo, portales, etc.). A esta necesidad de un servicio confiable se le conoce como alta disponibilidad.

La principal técnica para obtener estos sistemas tolerantes a fallos es la redundancia, estrategia utilizada en la industria aeronáutica prácticamente desde sus principios, que consiste en replicar las zonas críticas del sistema, teniendo una unidad activa y varias copias inactivas que, tras el fallo de la

principal, sean capaces de retomar su labor en el punto que aquella falló, en el menor tiempo posible y de forma transparente para el usuario.

3.2 INTERÉS COMERCIAL

Desde hace unos anos Heartbeat está en las distribuciones SuSE Linux, Conectiva Linux y Mandrake, incluso Mission Critical Linux está haciendo un producto basado en él. Todo esto es así porque el mercado de clusters HA es un mercado con muchos potenciales clientes y, lo que es quizás más importante, para los intereses comerciales de muchas empresas.

Por ejemplo una empresa de grandes almacenes que tiene ordenadores carísimo validando las operaciones de tarjeta de crédito. Estos ordenadores no deben caer nunca porque si lo hicieran todo el sistema de tarjetas de créditos se vendría abajo con lo que se podrían ocasionar grandes pérdidas económicas.

Por todo esto se desarrollan proyectos que intentan conseguir esta disponibilidad pero no gracias a un soporte hardware carísimo sino usando clusters. Las empresas que necesitan esta alta disponibilidad suelen pagar a la empresa que le ofrece este servicio aun cuando los programas sean de libre

distribución, porque quieren unas garantías. Esto está haciendo que muchas empresas estén colaborando en proyectos libres de clusters HA, cosa que no deja de ir en pro de la mejora del software en cuestión.

3.3 CONCEPTOS IMPORTANTES

Un buen cluster HA necesita proveer fiabilidad, disponibilidad y servicio RAS. Por tanto debe existir una forma de saber cuándo un ordenador ha caído y cuándo vuelve a funcionar.

Los clusters HA solucionan el problema de la disponibilidad con una buena capa de software.

Para conseguir la alta disponibilidad en un cluster los nodos tienen que saber cuándo ocurre un error para hacer una o varias de las siguientes acciones:

- **Intentar recuperar los datos del nodo que ha fallado.**

  Cuando un nodo cae tenemos que recuperar de un disco duro, compartido por los nodos, la información para poder seguir con el trabajo. Generalmente hay scripts de recuperación para intentar recuperarse del fallo.

- **Continuar con el trabajo que desempeñaba el nodo caído.**

  Aquí no se intenta recuperar del fallo sino que cuando se descubre que ocurrió un fallo otro nodo pasa a desempeñar el puesto del nodo que falló.

Esta es la opción que toma por ejemplo Heartbeat que permite que 2 ordenadores mantengan una comunicación por un cable serie o Ethernet, con lo que cuando un ordenador crítico cae el ordenador que no recibe respuesta ejecuta las órdenes adecuadas para ocupar su lugar.

Las ventajas de escalabilidad y economía de los clusters tienen sus desventajas. Una de ellas es la seguridad. Cuando se diseña un cluster se busca que haya ciertas facilidades de comunicación entre las estaciones y en clusters de alta disponibilidad el traspaso de información puede ser muy importante.

Recordando el ejemplo anterior de las tarjetas de crédito, se ha visto que se podría crear un cluster de alta disponibilidad que costará varias veces menos que el ordenador centralizado. El problema podría sobrevenir cuando ese cluster se encargará de hacer operaciones con los números de las tarjetas de crédito y transacciones monetarias de la empresa. Las facilidades de

comunicación podrían ocasionar un gravísimo problema de seguridad. Un agente malicioso podría hacer creer al cluster que uno de los nodos ha caído, entonces podría aprovechar el traspaso de la información de los nodos para conseguir los números de varias tarjetas de crédito.

3.3.1 Servicio RAS

En el diseño de sistemas de alta disponibilidad es necesario obtener la suma de los tres términos que conforman el acrónimo RAS[8].

Reliability.

El sistema debe ser **confiable** en el sentido de que éste actúe realmente como lo hemos programado. Por un lado está el problema de coordinar el sistema cuando éste está distribuido entre nodos, por otro lado hay el problema de que todo el software que integra el sistema funcione entre sí de manera confiable.

En general se trata de que el sistema pueda operar sin ningún tipo de caída o fallo de servicio.

[8] Reliability Availability Serviceability

Availability.

Es lógicamente la base de este tipo de clusters. La disponibilidad indica el porcentaje de tiempo que el sistema esta disponible en su funcionalidad hacia los usuarios.

Serviceability.

Referido a cómo de fácil es controlar los servicios del sistema y qué servicios se proveen, incluyendo todos los componentes del sistema.

El concepto de disponibilidad es el que prima por encima de los anteriores. La disponibilidad de un sistema es dependiente de varios factores. Por un lado el tiempo que el sistema está funcionando sin problemas, por otro lado el tiempo en el que el sistema esta fallando y por último el tiempo que se tarda en reparar o restaurar el sistema.

3.3.2 Técnicas para proveer disponibilidad

Están las técnicas basadas en reducir el tiempo de reparación. Este tipo de técnicas se componen a base de scripts o programas que detectan donde fallo el sistema y tratan de recuperarlo sin necesidad de un técnico especializado. En general son técnicas de automatización de tareas basadas en sistemas expertos.

Al reducir el tiempo de recuperación, el sistema puede no solamente funcionar activamente sin fallos más tiempo sino que también aumentamos la confiabilidad.

3.4 SOLUCIONES LIBRES

Este es el mayor proyecto de software libre de clusters HA que existe, parte de este proyecto es Heartbeat y trabajan conjuntamente con el grupo encargado de LVS.

Han desarrollado varias aplicaciones comerciales sobre este proyecto y se está utilizando en varios servicios con éxito. Como parte de los objetivos que se persiguen se encuentran:

- **Servicios de membership.**

  Estos servicios permiten añadir y quitar miembros a un cluster.

- **Servicios de comunicación.**

Comunicar información crítica de forma que una caída en un sistema no haga que se pierda la información y a la vez enviar la información de una forma suficientemente segura para evitar posibles ataques externos.

- **Manejo del cluster.**

Una serie de servicios que hagan sencillo el manejo del cluster en general y de los nodos y procesos en particular. Al igual que un sistema operativo provee de servicios para administrarlo, un cluster también debe proveer de instrucciones para gestionar su funcionamiento.

- **Monitorización de los recurs**

Este punto está muy unido al anterior. Para que el administrador detecte prematuramente posibles fallos y pueda ver qué ocurre en el cluster necesita algunas facilidades de monitorización.

- **Replicación y/o compartición de datos.**

Para conseguir que los datos que estuviera modificando uno de los nodos no se pierda cuando caiga se puede replicar la información y/o mantenerla en lugares compartidos por todos los nodos con lo que cualquier nodo podría continuar con los datos compartidos. Para

conseguir tener unos discos compartidos se necesita un hardware caro como es SCSI[9] y fibra óptica.

HeartBeat

Esta tecnología implementa heartbeats, cuya traducción directa sería latidos de corazón. Funciona enviando periódicamente un paquete que si no llegara indicaría que un servidor no está disponible, por lo tanto se sabe que el servidor ha caído y se toman las medidas necesarias.

Dichos latidos se pueden enviar por una línea serie, por UDP o por PPP/UDP. De hecho los desarrolladores de Heartbeat recomiendan el uso de puertos serie por varias razones, entre las que destacan que están aislados de las tarjetas de red.

3.5 LVS(LINUX VIRTUAL SERVER)

El proyecto Linux Virtual Server nos provee con la información y todos los programas necesarios para montar un "servidor virtual" fácilmente escalable sobre un cluster de máquinas Linux. De cara al usuario final solamente habrá un servidor, aún que de puertas adentro lo que tendremos será un cluster que

[9] SCS(Source Control System)

servirá las peticiones que le lleguen como si se tratara de una única máquina. Linux Virtual Server se basa únicamente en PCs corriendo Linux con su software, tanto para los servidores como para los equipos que hagan de balanceadores de carga (el punto de entrada al cluster que redirigirá el tráfico hacia cada uno de los servidores reales). Es decir, no necesitaremos de ningún router/firewall/balanceador hardware, ni ningún software propietario de terceras personas. Todo el software de Linux Virtual Server está disponible bajo la licencia GNU General Public License (GPL).

CAPITULO IV

4. CLUSTER ALTO RENDIMIENTO (HIGH PERFORMANCE)

4.1 INTRODUCCION

El término "High Performance Computing" (Computación de Alto Rendimiento) tiene su origen en el trabajo desarrollado por Seymour Cray al diseñar y construir las computadoras que llevan su nombre.

El rápido incremento tanto por la velocidad de los procesadores de computadoras personales como por la relación rendimiento/precio que las mismas suministran, sumado al desarrollo de tecnologías de redes de alta velocidad, conducen necesariamente a una profunda transformación del concepto original de High Performance Computing.

Actualmente, pensar en un sistema para efectuar computación de alto rendimiento es pensar en un cluster de computadoras personales utilizando el sistema operativo Linux y efectuando sus tareas bajo un entorno de programación en paralelo.

Muchas son las disciplinas que requieren llevar a cabo diariamente computación de alto rendimiento como una parte importante de sus actividades. La astronomía, la biología, la física, muchas ramas de la ingeniería, y la química son algunas de esas disciplinas.

El procesamiento de imágenes, la determinación de la secuencia de los nucleótidos en el ADN y la simulación numérica de procesos de interés biológico, químico o físico son algunas de las actividades que requieren de la computación de alto rendimiento para alcanzar resultados en tiempos razonables.

Podemos decir entonces que la "computación de alto rendimiento" es aquella dedicada a la resolución de problemas bien determinados por medio de sistemas de altísimo poder de cómputo.

4.2 CONCEPTOS

Un cluster de alto rendimiento es un conjunto de computadoras utilizadas como un recurso unificado de procesamiento que comparte una administración común.

Los clusters alto rendimiento son clusters dedicados a dar el mayor rendimiento posible y existen multitud de formas de implementarlos.

Rendimiento

Es la efectividad del desempeño de una computadora, sobre una aplicación. En las mediciones de rendimiento están involucrados velocidad, costo y eficiencia.

Flops

Es una medida de la velocidad del procesamiento numérico del procesador. Son operaciones de punto flotante por segundo. Se utilizan en unidades de millones de flops: MegaFlops; y en Miles de Millones de flops: GigaFlops.

Alto Rendimiento (HPC)

Gran demanda de procesamiento de datos en procesadores, memoria y otros recursos de hardware, donde la comunicación entre ellos es muy rápida.

Nodo

Se refiere a una computadora sola, que contiene recursos específicos, tales como memoria, interfaces de red, uno o más CPU, etc.

Escalabilidad

Generalmente se mide la eficiencia de un problema, utilizando un tamaño y un número de procesadores fijo, lo cual es insuficiente, pues los resultados serán diferentes cuando aumentamos o disminuimos el tamaño del problema y el número de procesadores. Es decir, existe un problema de escalabilidad.

Cuando se aumenta el número de procesadores para el mismo tamaño del problema, la sobrecarga debido al paralelismo aumenta. Similarmente, podemos tener casos en donde el tamaño del problema es muy pequeño para tener una evaluación real del problema sobre cierta máquina.

Memoria Compartida

En una máquina paralela existe una sola memoria que puede ser accesada por todos los procesadores.

Memoria Distribuida

Cada uno de los procesadores de un multiprocesador tiene asociado a él una unidad de memoria.

Balanceo de carga

Lo ideal en el procesamiento en paralelo es que cada procesador realice la misma cantidad de trabajo, y además, se espera que los procesadores trabajen al mismo tiempo. La meta del balanceo de carga es minimizar el tiempo de espera de los procesadores en los puntos de sincronización.

4.3 DIVISION CLUSTER ALTO RENDIMIENTO

Su división es:

- soluciones que funcionan a nivel de aplicación
- soluciones que funcionan a nivel de kernel.

Nivel de aplicación.- Suele tomar forma de librería, se realiza programas para aprovechar esta librería por lo tanto cualquier programa ya existente para que pueda ser usado en un cluster y mejore su rendimiento, tiene que ser reescrito al menos parcialmente.

Nivel de kernel.- Es que el software que se encarga del cluster de Alto Rendimiento se encuentre en el kernel del sistema operativo, en este caso no se necesitan cambiar las aplicaciones de usuario, sino que éstas usan las llamadas

estándar del kernel por lo tanto internamente es el que se encarga de distribuir el trabajo de forma inteligente.

4.4 FORMAS DE MIGRAR LOS PROCESOS

Una forma de conseguir Alto Rendimiento es migrando procesos, dividiendo las aplicaciones grandes en procesos y ejecutando cada proceso en un nodo distinto. Lo que se quiere conseguir es el máximo uso de los recursos en todo momento, especialmente los procesadores. Para conseguirlo hay dos aproximaciones:

- **Asignar estáticamente cada proceso a un nodo en particular.**

  En esta aproximación es importantísima la política de localización. Se elige estáticamente el nodo donde el proceso vivirá toda su vida. Por tanto es muy importante elegir correctamente estos nodos. Muchas veces esta solución necesita un administrador que decida dónde debe ir cada proceso. El caso más simple es tener todos los nodos con la misma potencia de cálculo y dividir el único programa al que se quiera dedicar los recursos en un número de procesos igual al número de nodos de los que se disponen. Así se asignaría cada proceso a uno de los nodos. Hay que tener en cuenta

que esta configuración es lejana a la normal y que ya que un fallo en el algoritmo de elección de nodo puede infrautilizar mucho de los recursos la configuración manual es normal en estos algoritmos. Esto no es tan malo como pueda parecer a primera vista porque es también muy corriente en estos casos que una vez hecha la configuración inicial.

- **Asignar dinámicamente procesos a los nodos.**

  Los procesos una vez iniciados en un nodo pueden migrar a otro nodo dinámicamente, en estos casos aunque es importante la política de localización para minimizar el gasto de recursos, también es importante la política de migración. Se puede ubicar los procesos manualmente, con la ventaja de que se puede ubicar en cualquier momento durante la vida del proceso. Si la política de migración es correcta y los procesos tienen una vida larga y se ha dividido correctamente la aplicación, debería haber al comienzo de la ejecución de los procesos un periodo de reordenación de los procesos, con varias migraciones, una vez el sistema llegara a una condición estable, no deberían producirse apenas migraciones hasta que los procesos finalizaran.

No es necesaria la configuración manual (si el algoritmo de migración es suficientemente bueno). Cuando se desbalancea el sistema éste se encarga de que se vuelva a balancear, de tal forma de que se aprovechen los recursos al máximo.

4.5 PVM y MPI

4.5.1 Paso de mensajes

Tanto PVM[10] como MPI[11] se basan en el concepto de paso de mensajes, los mensajes son pasados entre los procesos para conseguir que se ejecuten de manera conjunta y de forma sincronizada. Se ha elegido mensajes pues se puede implementar de forma más o menos efectiva en un cluster, los mensajes se pueden enviar en forma de paquete IP[12] y el ordenador destino desempaqueta el mensaje y decide a que proceso va dirigido, una vez hecho esto, envía la información al proceso en cuestión. Por tanto para comprender este tipo de clusters, MPI en particular se necesita conocer de forma básica los mecanismos de paso de mensajes. Hay tres mecanismos básicos de paso de mensajes:

[10] PVM(Paraell Virtual Machine)
[11] MPI(Message Passing Interface)
[12] IP(Internet Protocol)

- **Paso de mensajes Síncrono**

  Cuando un proceso P ejecuta un envío sincrónico a un proceso Q, tiene que esperar hasta que el proceso Q ejecuta el correspondiente recibo de información sincrónico. Ambos procesos no volverán del envío o el recibo hasta que el mensaje está a la vez enviado y recibido. Cuando el enviar y recibir acaban, el mensaje original puede ser inmediatamente sobrescrito por el nuevo mensaje de vuelta y éste puede ser inmediatamente leído por el proceso que envió originariamente el mensaje. No se necesita un buffer extra en el mismo buffer donde se encuentra el mensaje de ida, se escribe el mensaje de vuelta.

- **Enviar/Recibir bloqueante**

  Un envío bloqueante es ejecutado cuando un proceso lo alcanza sin esperar el recibo correspondiente. Esta llamada bloquea hasta que el mensaje es efectivamente enviado, lo que significa que el mensaje puede ser reescrito sin problemas. Cuando la operación de enviar ha acabado no es necesario que se haya ejecutado una operación de recibir. Sólo sabemos que el mensaje fue enviado, puede haber sido recibido o puede estar en un buffer del nodo que lo envía, o en un buffer de algún lugar de

la red de comunicaciones o puede que esté en el buffer del nodo receptor.

Un recibo bloqueante es ejecutado cuando un proceso lo alcanza, sin esperar a su correspondiente envío. Sin embargo no puede acabar sin recibir un mensaje. Quizás el sistema esté proveyendo un buffer temporal para los mensajes.

- **Envío/recibo no bloqueante**

  Un envío no bloqueante es ejecutado cuando un proceso lo alcanza, sin esperar al recibo. Puede acabar inmediatamente tras notificar al sistema que debe enviar el mensaje. Los datos del mensaje no están necesariamente fuera del buffer del mensaje, por lo que es posible incurrir en error si se sobre escriben los datos.

  Un recibo no bloqueante es ejecutado cuando un proceso lo alcanza, sin esperar el envío. Puede volver inmediatamente tras notificar al sistema que hay un mensaje que se debe recibir. El mensaje puede que no haya llegado aún, puede estar todavía en transito o puede no haber sido enviado aún.

4.5.2 PVM (Parallel Virtual Machine)

PVM Es un conjunto de herramientas y librerías que emulan un entorno de propósito general compuesto de nodos interconectados de distintas arquitecturas. El objetivo es conseguir que ese conjunto de nodos pueda ser usado de forma colaborativa para el procesamiento paralelo.

El modelo en el que se basa el PVM es dividir las aplicaciones en distintas tareas. Son los procesos los que se dividen por las máquinas para aprovechar todos los recursos. Cada tarea es responsable de una parte de la carga que conlleva esa aplicación. El PVM soporta tanto paralelismo en datos, como funcional o una mezcla de ambos.

El PVM permite que las tareas se comuniquen y sincronicen con las demás tareas de la máquina virtual, enviando y recibiendo mensajes, muchas tareas de una aplicación puede cooperar para resolver un problema en paralelo.
Cada tarea puede enviar un mensaje a cualquiera de las otras tareas, sin límite de tamaño ni de número de mensajes.

El sistema PVM se compone de dos partes:

La primera es un demonio, llamado pvmd que residen en todas los nodos que forman parte de la máquina virtual. Cuando un usuario quiere ejecutar una aplicación PVM, primero crea una máquina virtual para arrancar PVM. Entonces se puede ejecutar la aplicación PVM en cualquiera de los nodos. Muchos usuarios pueden configurar varias máquinas virtuales aunque se mezclen unas con las otras y se pueden ejecutar varias aplicaciones PVM simultáneamente. Cada demonio es responsable de todas las aplicaciones que se ejecutan en su nodo. Así el control está totalmente distribuido excepto por un demonio maestro, que es el primero que se ejecuto a mano por el usuario, los demás nodos fueron iniciados por el maestro y son esclavos. En todo momento siempre hay un pvmd maestro. Por tanto la máquina virtual mínima es de un miembro, el maestro.

La segunda parte del sistema es la librería de PVM. Contiene un repertorio de primitivas que son necesarias para la cooperación entre los procesos o threads de una aplicación. Esta librería contiene rutinas para inicialización y terminación de tareas, envío y recepción de mensajes, coordinar y sincronizar tareas, broadcast, modificar la máquina virtual.

Cuando un usuario define un conjunto de nodos, PVM abstrae toda la complejidad que tenga el sistema y toda esa complejidad se ve como un gran computador de memoria distribuida llamada máquina virtual. Esta máquina virtual es creada por el usuario cuando se comienza la operación. Es un conjunto de nodos elegidos por el usuario. En cualquier momento durante la operación puede elegir nuevos nodos para la máquina virtual. Esto puede ser de gran ayuda para mejorar la tolerancia a fallos pues se tiene unos cuantos nodos de reserva si alguno de los nodos fallara. Es un conjunto de nodos de una determinada red están fallando se pueden habilitar nodos de otra red para solucionarlo. Para conseguir abstraer toda la complejidad de las diferentes configuraciones, soporta la heterogeneidad de un sistema a tres niveles:

- **Aplicaciones:** las subtareas pueden estar hechas para aprovechar las arquitecturas sobre la que funcionan. Por tanto como se puede elegir en que conjunto de nodos se ejecutarán unas tareas específicas, podemos hacer nuestras aplicaciones con la arquitectura al máximo por lo que se puede optimizar y hacer que funcionen aplicaciones hechas para arquitecturas específicas con PVM.

- **Máquinas:** nodos con distintos formatos de datos están soportados, incluyendo arquitecturas secuenciales, vectoriales, SMP[13].

- **Redes:** la máquina virtual puede ser interconectada gracias a distintas tecnologías de red. Para PVM, bajo él existe una red punto a punto, no fiable y no secuencial. Esto abstrae cualquier tecnología de red. Utiliza UDP y implementa toda la confiabilidad y todas las demás operaciones como broadcast en la propia librería PVM.

Tiene un conjunto de interfaces que está basado en la observación de las necesidades de la mayoría de las aplicaciones, que están escritas en C y Fortran. Los enlaces para C y C++ para la librería PVM están implementados como funciones, siguiendo las reglas usadas por la mayoría de los sistemas que usan C, incluyendo los sistemas operativos tipo UNIX. Los enlaces para Fortran están implementados como subrutinas más que funciones.

4.5.3 Problemas Pvm (Paralel Virtual Machine)

PVM no tiene requisa de procesos dinámico, esto quiere decir que una vez que un proceso empieza en una determinada máquina seguirá en ella hasta que se

[13] SMP(Symetric MultiProcessor)

muera. Esto tiene graves inconvenientes como explicamos en las características de asignar estáticamente un proceso a un nodo en concreto.

Hay que tener en cuenta que las cargas suelen variar y que, a no ser que todos los procesos que se estén ejecutando sean muy homogéneos entre sí, se está descompensando el cluster. Por lo tanto tenemos unos nodos más cargados que otros y seguramente unos nodos terminen su ejecución antes que otros, con lo que se podrían tener nodos muy cargados mientras otros nodos están libres. Esto lleva a una pérdida de rendimiento general.

Otro problema de PVM es que está implementado a nivel de usuario, esto no es malo de por sí pero teniendo en cuenta el tipo de operaciones que lleva, sí lo expuesto son operaciones de bastante bajo nivel como puedan ser paso de mensajes entre aplicaciones y la capa sobre UDP. Esto añade complejidad y latencia a las comunicaciones que se tienen que producir sobre las comunicaciones del kernel.

4.5.4 MPI (Message Passing Interface)

MPI es una especificación estándar para una librería de funciones de paso de mensajes. El MPI fue desarrollado por el *MPI* Forum, un consorcio de

vendedores de ordenadores paralelos, escritores de librerías y especialistas en aplicaciones.

Consigue portabilidad proveyendo una librería de paso de mensajes estándar independiente de la plataforma y de dominio público. La especificación de esta librería está en una forma independiente del lenguaje y proporciona funciones para ser usadas con C y Fortran.

El MPI tiene que ser implementado sobre un entorno que se preocupe del manejo de los procesos y la E/S por ejemplo, puesto que MPI sólo se ocupa de la capa de comunicación por paso de mensajes. Necesita un ambiente de programación paralelo nativo.

Todos los procesos son creados cuando se carga el programa paralelo y están vivos hasta que el programa termina. Hay un grupo de procesos por defecto que consiste en todos esos procesos, identificado por MPI_COMM_WORLD.
Los procesos MPI son procesos como se han considerado tradicionalmente, del tipo pesados, cada proceso tiene su propio espacio de direcciones, por lo que otros procesos no pueden acceder directamente a las variables del espacio de

direcciones de otro proceso. La intercomunicación de procesos se hace vía paso de mensajes.

Las desventajas de MPI son similares a la del PVM. Además aunque es un estándar y debería tener un API[14] estándar, cada una de las implementaciones varía, no en las llamadas sino en el número de llamadas implementadas (MPI tiene unas 200 llamadas). Esto hace que en la práctica los diseñadores del sistema y los programadores tengan que conocer el sistema particular de MPI para sacar el máximo rendimiento. Además como sólo especifica el método de paso de mensajes, el resto del entorno puede ser totalmente distinto en cada implementación con lo que otra vez se impide esa portabilidad que teóricamente tiene.

4.6 EJEMPLOS DE CLUSTERS DE ALTO RENDIMIENTO

4.6.1 Beowulf

El proyecto Beowulf fue iniciado por Donald Becker, también famoso por crear numerosos drivers para tarjetas de red en Linux en 1994 para la NASA. Este proyecto se basa en usar PVM y MPI, añadiendo algún programa más que se usan para monitorizar, realizar benchmarks y facilitar el manejo del cluster.

[14] API(Application Programming Interface)

Entre las posibilidades que integra este proyecto se encuentra la posibilidad de que algunos equipos no necesiten discos duros, por eso se consideran que no son un cluster de estaciones de trabajo, sino que dicen que pueden introducir nodos heterogéneos. Esta posibilidad la da otro programa y Beowulf lo añade a su distribución.

Beowulf puede verse como un empaquetado de PVM/MPI junto con más software.

4.6.2 OpenMosix

Mosix es un software para conseguir clustering en Linux, migrando los procesos de forma dinámica. Consiste en unos algoritmos de compartición de recursos a nivel de kernel, que están enfocados a conseguir alto rendimiento, escalabilidad con baja sobrecarga y un cluster fácil de utilizar.

La idea es que los procesos colaboren de forma que parezca que están en un mismo nodo.

Los algoritmos de OpenMosix son dinámicos lo que contrasta y es una fuerte ventaja frente a los algoritmos estáticos de PVM/MPI, responden a las variaciones en el uso de los recursos entre los nodos migrando procesos de un nodo a otro, con requisa y de forma transparente para el proceso, para balancear la carga y para evitar falta de memoria en un nodo. Las fuentes de openMosix han sido desarrolladas 7 veces para distintas versiones de Unix y BSD.

OpenMosix, al contrario que PVM/MPI, no necesita una adaptación de la aplicación ni siquiera que el usuario sepa nada sobre el cluster. Como se ha visto, para tomar ventaja con PVM/MPI hay que programar con sus librerías, por tanto hay que rehacer todo el código que haya para aprovechar el cluster.
El OpenMosix puede balancear una única aplicación si esta está dividida en procesos lo que ocurre en gran número de aplicaciones hoy en día. Y también puede balancear las aplicaciones entre sí, lo que balancea OpenMosix son procesos, es la mínima unidad de balanceo. Cuando un nodo está muy cargado por sus procesos y otro no, se migran procesos del primer nodo al segundo. Con lo que OpenMosix se puede usar con todo el software actual si bien la división en procesos ayuda al balanceo gran cantidad del software de gran carga ya dispone de esta división.

El usuario en PVM/MPI tiene que crear la máquina virtual decidiendo qué nodos del cluster usar para correr sus aplicaciones cada vez que las arranca y se debe conocer bastante bien la topología y características del cluster en general.

Sin embargo en OpenMosix una vez que el administrador del sistema que es quien realmente conoce el sistema, lo ha instalado, cada usuario puede ejecutar sus aplicaciones y seguramente no descubra que se está balanceando la carga, simplemente verá que sus aplicaciones acabaron en un tiempo record.

PVM/MPI usa una adaptación inicial fija de los procesos a unos ciertos nodos, a veces considerando la carga pero ignorando la disponibilidad de otros recursos como puedan ser la memoria libre y la sobrecarga en dispositivos E/S.

4.6.3 Top 500

La máquina Earth Simulator, de NEC, continúa siendo el computador más rápido del mundo con su capacidad de 35.86 teraflops por segundo.
Earth Simulator se encuentra en Yokohama, Japón, donde analiza el clima de todo el mundo y simula situaciones meteorológicas globales. La máquina de

NEC relevó del sitial del supercomputador más rápido del mundo a una máquina de IBM en junio pasado.

El segundo computador más rápido del mundo es de marca Hewlett-Packard y está destinado a tareas quizás menos pacíficas que la máquina de NEC. Ubicado en el centro de investigaciones militares en Los Alamos, Estados Unidos, el supercomputador HP simula explosiones y ataques nucleares.

En el tercer lugar también se ubica una máquina Hewlett-Packard, de características similares a las de su predecesora en la lista.
La lista de los 500 supercomputadores más rápidos del mundo es elaborada por la Universidad Mannheim, de Alemania, y por National Energy Research Scientific Computing Center, de EEUU. La lista es actualizada dos veces al año, en los meses de junio y noviembre, respectivamente.

"La lista fue publicada por primera vez en 1993 y es considerada como la fuente más confiable en materia de supercomputadores"[15]

[15] Información tomada de http://www.top500.org

CAPITULO V

5. CONCEPTOS BASICOS

5.1 TECNOLOGÍA DE REDES

La eficiencia de un cluster de alta performance depende en gran parte de la tecnología utilizada para interconectar sus nodos. Esto se debe a que los programas paralelos utilizan intercambio de mensajes entre los nodos involucrados en la resolución de un problema.

5.1.1 Tecnología Ethernet

En un principio bastó con poder conectar los nodos del cluster mediante una red compartida. Cada nodo con su placa de red ethernet de 10 Mbps, se conectaba con los demás mediante un Hub a través de cables de cobre de par trenzado según norma de transmisión 10BaseT (10 Mbps). La tecnología Ethernet es un bus Broadcast que transmite paquetes a 10 Mbps. Es un bus porque todas las máquinas comparten el mismo canal de comunicación, y es broadcast porque todas las máquinas reciben todas las transmisiones. El modo de transmisión es half duplex.

5.1.2 Protocolo IP, Internet Protocol (MTU)

El protocolo estándar TCP/IP define al datagrama IP como la unidad de información enviada a través de la red y provee las bases para la distribución de paquetes. El MTU[16], es el tamaño de paquete más grande que puede ser transferido a través de una red física. En teoría, puede ser de hasta 64 Kbytes, pero en la práctica está determinado por el hardware de red. Actualmente el estándar de los dispositivos para interconectar redes (switches, routers, etc.) es de 1500 bytes, pero algunas placas de red gigabit permiten configurar el MTU de hasta 9000 bytes logrando así incrementar el throughput. Se debe tener en cuenta que si el MTU es muy pequeño, la máquina estará respondiendo seguido mientras que si es muy grande puede haber errores y los paquetes tendrían que ser retransmitidos.

5.1.3 Protocolo TCP, Transmisión Control Protocol

TCP es un protocolo confiable pues garantiza que los datos llegarán al destino y orientado a conexión pues simula un túnel entre el emisor y el receptor. Básicamente una conexión TCP consta de las siguientes partes: emisor con un buffer para el envío de mensajes cuyo tamaño está dado en bytes receptor con

[16] MTU(Maximum Transfer Unit)

un buffer para la recepción de mensajes cuyo tamaño está dado en bytes y puede ser diferente al anterior Dichos buffers, también llamados ventana de congestión, cambian de tamaño dinámicamente en el transcurso de una conexión TCP controlando así el flujo de datos.

La performance depende del tamaño de las ventanas y de la velocidad en que la red transmita los paquetes. Incrementando el tamaño de la ventana es posible reducir por completo los momentos ociosos de la red. Una buena configuración del tamaño de la ventana deslizante, mantiene a la red completamente saturada de paquetes, obteniendo un mejor throughput. Luego, el máximo throughput está condicionado por el tamaño de ambos buffers. La siguiente fórmula permite calcular el máximo ancho de banda dado el tamaño del búfer emisor y el tiempo de respuesta del enlace:

$$\text{Ancho de banda} = \frac{\text{tamaño del búfer de envío}}{\text{RTT}}$$

Por la misma fórmula, sabiendo el ancho de banda teórico de la red y el RTT podemos calcular el tamaño óptimo de la ventana de congestión para lograr un buen throughput.

5.1.4 Ancho de Banda, Latencia y RTT

La Performance de la red se mide preferentemente en bits por segundo 'bps'. Sus características están dadas por el ancho de banda y la latencia.
El ancho de banda indica la cantidad de datos que pueden ser transferidos a través de un enlace por unidad de tiempo. La latencia es el tiempo necesario para transferir un dato desde un origen al destino, generalmente se refiere a las demoras debidas al procesamiento del dato de red. El tiempo de ida y vuelta se denomina Round Trip Time (RTT).

El ancho de banda y la latencia están relacionados. Mientras que el máximo ancho de banda teórico es fijo (dado por el hardware de red utilizado), el ancho de banda real se ve afectado por la latencia de la red, es decir por el tiempo mínimo necesario para transferir un paquete de dato desde un punto a otro.

Mucha latencia en un corto período de tiempo puede causar cuellos de botella que no permiten llenar el enlace, por lo tanto decrece el ancho de banda. También se ve afectado por el overhead del hardware y del sistema operativo. El ancho de banda práctico lo llamamos throughput. La percepción de la velocidad de la red es una síntesis de estas dos unidades de medida.

5.2. Arquitectura del nodo

A grandes rasgos, el camino seguido en una transferencia de datos a través de una red implica tres cargas en memoria. Del lado del receptor, el adaptador de red accede directamente a memoria (DMA, Direct Memory Access) para cargar los datos en el búfer receptor, el CPU lee los datos de dicho búfer y los escribe al de la aplicación. En una transmisión se realiza el paso inverso. Las cargas en memoria dependerán de la aplicación que éste siendo utilizada, tamaños de buffers y el comportamiento de la cache.

El tráfico total de memoria, lecturas / escrituras, es a través del Front Side Bus - FSB. La Figura 5.1 muestra el tráfico de memoria durante la transmisión de un mensaje.

La Figura 5.2 muestra el tráfico de memoria durante la recepción de un mensaje.

Envío de mensajes.

1. El CPU lee el búfer de la aplicación
2. El CPU escribe los datos al búfer del socket

3. La placa de red (NIC) accede directamente a memoria y lee el búfer del socket

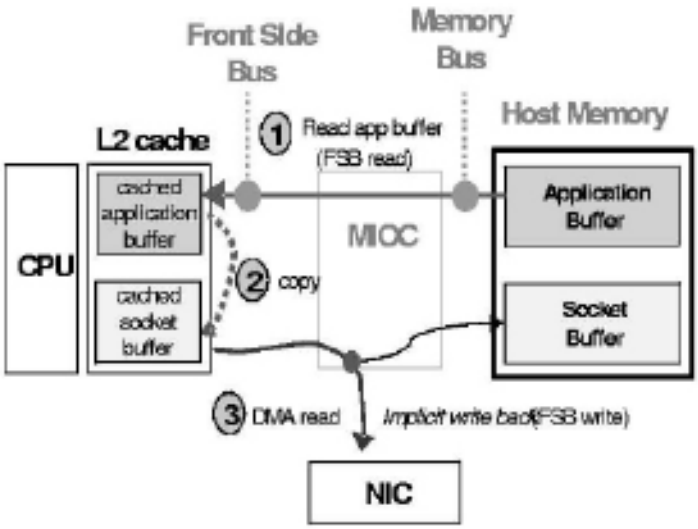


Figura 5.1 Envió De Mensajes

Recepción de mensajes.

1. La placa de red (NIC) accede directamente a memoria y escribe el búffer del socket

2. El CPU lee el búfer del socket

3. El CPU escribe los datos al búfer de la aplicación

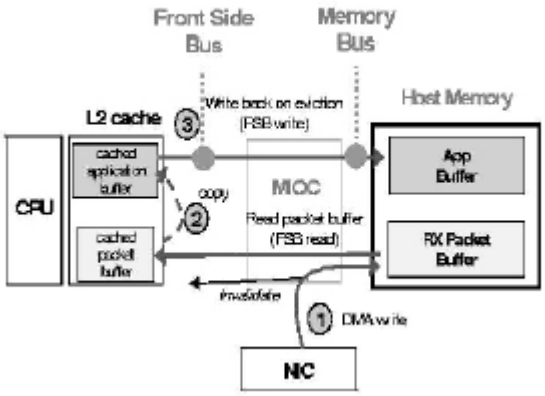


Figura 5.2: Recepción de Mensajes

5.3. IMPLEMENTACIÓN DE GIGABIT ETHERNET EN CLUSTERS DE PC

Dado un cluster de PCs interconectado mediante placas de red y switch Fast Ethernet que presenta buena performance de comunicación, no es trivial migrar su medio de conexión a Gigabit Ethernet. No se obtiene con facilidad un aumento de rendimiento utilizando placas de red y switch Gigabit Ethernet.

Esto se debe en gran parte a la arquitectura del bus PCI (Periphera Component Interconnect) que poseen los motherboards que en general se encuentran en uso y que poseen placas de red fast ethernet. Otra posible fuente de dificultad son los potenciales cuellos de botella asociados a la gestión del FSB como así también la configuración de los parámetros TCP/IP.

El bus PCI es compartido entre varios dispositivos, por su naturaleza paralela.

Aquellos dispositivos que se encuentren conectados al bus harán uso del mismo rotando cíclicamente. No solo comparten el bus sino que la frecuencia quedará definida en base al dispositivo de menor frecuencia, disminuyendo así la performance de los dispositivos más rápidos. Por ejemplo, el bus PCI está compartido por el CPU, la memoria, la placa de video, las placas de red, etc.
En particular, las placas de red negocian la velocidad con el bus PCI. Para una placa con un bus PCI 32 bits/33 Mhz, ver Tabla 5.1, se observa que nominalmente soporta 1 Gbps, pero no es recomendable. En el caso del cluster Speedy, éste posee motherboards PC-Chip M810 con PCI 2.2 y placas de red 3Com 3c2000 que negocian a 32-bit/66 MHz, alcanzando un ancho de banda teórico de 2 Gbps. Existe una arquitectura más moderna para bus PCI que consiste en no compartir el bus con muchos dispositivos a través de un bridge PCI, sino que se conectan directamente con el Memory I/O Control, MIOC alcanzando así mayor throughput para las placas Gigabit.

Arquitectura	Ancho del Bus	Frecuencia del Bus	Ancho de banda del Bus (Bytes)	Ancho del banda del Bus (bits)
PCI 1.x 2.x	32-bit	33 MHz	133 MBps	1 Gbps
PCI 2.2	32-bit	66 MHz	264 MBps	2 Gbps
PCI 2.2/PCI-X	64-bit	66 MHz	533 MBps	4.2 Gbps
PCI-X	64-bit	100 MHz	800 MBps	6.4 Gbps
PCI-X	64-bit	133 MHz	1 GBps	8 Gbps

Tabla 5.1: Características del throughput nominal del Bus PCI/PCI-X

Se puede perder hasta el 50% en la capa de pasaje de mensajes. El sistema operativo y los drivers usualmente aumentan la latencia y disminuyen el ancho de banda debido a que realizan muchas copias de dato memoria a memoria al empaquetar los datos para su transmisión. Estos movimientos extras de datos saturan el bus de memoria, típicamente antes de saturar el bus PCI. Por otro lado, también se debe tener en cuenta que los sistemas operativos actualmente calculan los parámetros TCP/IP suponiendo el uso de placas fast ethernet. Con lo cual, se deben configurar los parámetros por defecto de las ventanas TCP/IP como así también los buffers para pasaje de mensajes de MPICH.

5.4. SISTEMA DE ARCHIVOS

Dada la arquitectura, diseño y composición del hardware de Speedy se tiene la necesidad de utilizar la capacidad de disco de cada uno de los nodos en forma

global. Se quiere aplicar un sistema de archivos que soporte el acceso a los mismos en forma paralela. Cada sistema de archivos será evaluado con respecto a su velocidad de acceso de entrada / salida y su escalabilidad.

También se tendrá en cuenta el tipo de aplicaciones que se quiere ejecutar en el cluster, aquellas con gran cantidad de operaciones de entrada / salida y las que tienen poca cantidad de operaciones de entrada / salida.

Se evaluarán los siguientes sistemas de archivos:

- Local (ext3) tipo de fichero utilizado en Linux
- NFS - Network File System
- PVFS - Parallel Virtual File System
- MFS - Mosisx File System
- GFS - Global File System

Cuando recibe un pedido, el software del cliente utiliza el protocolo correspondiente para conectarse al / los servidor / es remotos apropiados y lleva a cabo la operación requerida. Cuando el servidor remoto contesta, el software del cliente devuelve los resultados a la aplicación que inicio el pedido.

5.4.1. Sistema de archivo local - ext3

El tipo de fichero local ext3, utilizado actualmente en Linux, posee la ventaja de tener journaling (registros de eventos periódicos), esto garantiza disponibilidad e integridad de los datos y velocidad de acceso.

Disponibilidad pues luego de una caída del sistema inesperada ext3 no requiere un chequeo total del sistema de archivos gracias al journal.
Integridad garantiza integridad de los datos luego de una caída abrupta del sistema.

Velocidad a pesar de escribir los datos más de una vez, es más rápido que ext2, su predecesor, pues el journaling optimiza el movimiento de las cabezas del disco.

5.4.2. Network File System - NFS

NFS es un protocolo desarrollado por Sun Microsystems, Incorporated. Permite que un conjunto de máquinas accedan a sistemas de archivos remotos en forma transparente.

Cuando un cliente monta un sistema de archivos, el servidor de NFS le devuelve un file handle (FH) que contiene el tipo de archivo, el disco, el i-nodo e información de seguridad. La implementación de NFS, ver Figura 5.3, independiente del protocolo, consiste en tres capas. La primer capa de llamadas al SO (OPEN, READ, CLOSE) luego de examinar el pedido invoca a la segunda capa, el sistema de archivos virtual (VFS). Ésta se encarga de mantener una tabla con una entrada por archivo abierto llamado v-nodo. Si el archivo es local, la entrada contendrá un i-nodo y la tercer capa será la que manipula archivos locales y si el archivo es remoto, la tercera capa estará a cargo del cliente NFS. Analizaremos las llamadas al sistema: MOUNT, OPEN y READ.

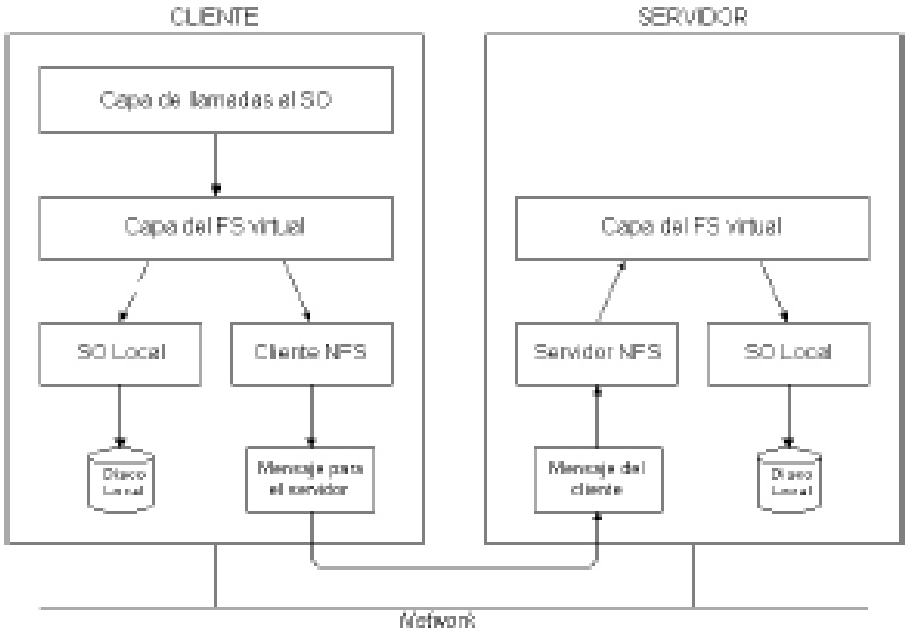


Figura 5.3: Estructura De Capas NFS

MOUNT Para montar un fichero remoto se utiliza el comando mount, se especifica el directorio remoto y el directorio local donde será montado, entre otra información. El programa mount analiza el nombre del directorio remoto, descubre la máquina a la cual pertenece, la contacta y le pide el FH.

Si el directorio existe y está disponible para su acceso remoto, el servidor le da el FH. Finalmente, el programa realiza una llamada al sistema MOUNT y le pasa el FH al kernel. El kernel construye el v-nodo y le pide al cliente NFS que cree el r-nodo (i-nodo remoto) en su tabla interna para guardar el FH.

Luego el v-nodo apunta al r-nodo.
Cada v-nodo de la capa VFS contendrá un puntero o un r-nodo en el código del cliente NFS o bien, un puntero a un i-nodo del SO local.

Luego desde la capa VFS se puede saber si el directorio o archivo es remoto o local y si es remoto, se puede encontrar el FH.

OPEN Durante la examinación del "path" del archivo, el kernel descubre que el directorio en el cual está montado el fichero es remoto. Busca en la tabla de v-nodos el puntero al r-nodo. Luego pide al cliente NFS que abra el archivo. El

cliente NFS busca el resto del "path" en el servidor remoto asociado con el directorio montado y devuelve el FH. Crea un r-nodo para el archivo remoto en sus tablas y reporta a la capa VFS, la cual pone en sus tablas el v-nodo que tendrá un puntero al r-nodo.

Entonces vemos que todo archivo o directorio abierto tiene un v-nodo que apunta a un r-nodo o i-nodo.
Notemos que el que guarda la información del FH es el cliente, el servidor no guarda ningún tipo de información sobre el archivo. El servidor provee los FH ante un pedido, pero no lleva ningún registro de los FH entregados. Cuando un cliente le envía un FH para acceder a un archivo, el servidor verifica que el mismo exista y lo usa.

READ Cuando el FH es utilizado en llamadas posteriores al sistema, por ejemplo para una lectura, la capa VFS localiza el v-nodo correspondiente y a partir de el determinara si es local o remoto y también que i-nodo o r-nodo lo describe.

Se utilizan distintas técnicas para mejorar la eficiencia del protocolo. Las transferencias entre el cliente y el servidor se realizan en bloques de 4096 bytes,

incluso en el caso en que se pidan menos bytes. Una vez que el cliente recibe el bloque de 4 KB, pide el siguiente, este procedimiento se llama look ahead e incrementa la performance de NFS. Para las escrituras se procede del mismo modo, sólo se envía la escritura al servidor cuando el cliente junta un bloque de 4 KB. Se recomienda elevar el tamaño del bloque a 8192 bytes para lograr mayor performance.

Otra técnica utilizada para mejorar la performance es el uso de cache, los servidores guardan datos en cache para evitar accesos al disco. Los clientes, mantienen dos caches, una para los atributos del archivo (i-nodos) y otra para los datos. Cuando se necesita un i-nodo o un bloque de archivo, se verifica primero si está en cache, si es así se evita el tráfico de red.

Mientras que la cache en el cliente ayuda a mejorar la performance, introduce problemas de consistencia. Si dos clientes tienen en cache la misma porción de un mismo archivo, y uno de ellos la modifica, cuando el otro lea el bloque leerá información vieja. La cache no es coherente. NFS trata de evitar este problema del siguiente modo:

Asocia a cada bloque en cache un contador de tiempo, cuando el mismo expira (normalmente 3 segundos para bloques de datos y 30 segundos para directorios) se descarta la entrada; cuando se abre un archivo que esta en cache envía un mensaje al servidor para averiguar cuando fue modificado el archivo por última vez, si la misma ocurrió luego de la copia en cache la misma se descarta y se trae la nueva información; cada 30 segundos expiran los datos de la cache y todos los bloques que han sido modificados se envían al servidor.

El NFS ha sido muy criticado pues mientras que un cliente escribe un archivo puede o no ser visto por otro que lo está leyendo, dependiendo del tiempo. Cuando se crea un archivo puede no ser visto por el resto de los clientes hasta pasados 30 segundos. La semántica del acceso a archivos no está bien definida, si un conjunto de programas cooperativos se ejecutan varias veces podrían dar distintos resultados dependiendo del tiempo.

5.4.3. Parallel Virtual File System - PVFS

PVFS es un sistema de archivos paralelo cliente / servidor en el cual los archivos se distribuyen en forma transparente en discos de múltiples servidores.

Su característica principal es que a través de él las aplicaciones paralelas pueden acceder velozmente a los datos. Provee tres servicios básicos a los usuarios: un espacio de nombres consistente entre los nodos del cluster que permite a los programadores acceder a los archivos desde múltiples nodos; distribución física de los datos entre los discos de los nodos que permite evitar cuellos de botella tanto en la interfase del disco como así también en la red proveyendo mayor ancho de banda a los recursos de entrada / salida (E/S); interfase de E/S que permite que los usuarios controlen cómo serán distribuidos los datos y habilitar modos de acceso.

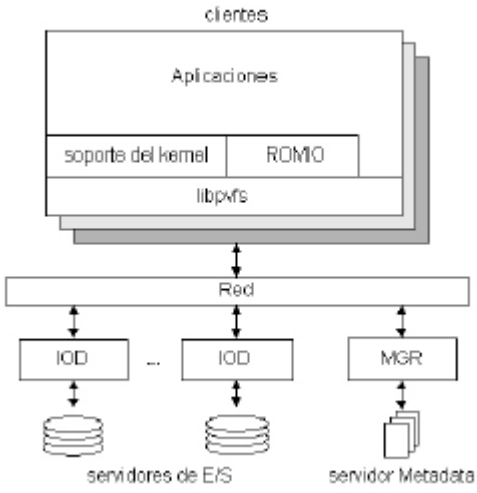


Figura 5.4: Diagrama del sistema PVFS

Como se observa en la Figura 5.4 las máquinas que integran el sistema PVFS pueden tomar uno o más de los siguientes roles:

Servidor Metadata. Existe un único servidor metadata por sistema de archivos PVFS en el cual corre el demonio MGR. Contiene información correspondiente a los archivos y directorios que posee como son: permisos, dueño, ubicación de los datos distribuidos en los servidores de E/S. Es contactado por los clientes cuando necesitan leer un directorio o bien crear, eliminar, abrir o cerrar un archivo.

Servidor de E/S. Puede haber uno o más. Cada servidor de E/S (IOD) aporta una porción de su disco local para integrar la partición PVFS. Lleva a cabo las operaciones de acceso a los archivos sin intervención del servidor metadata.
Cliente. Puede haber uno o más clientes. En ellos se corren las aplicaciones que acceden a los archivos y directorios de la partición PVFS. Cuando una aplicación desea abrir, cerrar, crear o eliminar un archivo se comunica directamente con el MGR a través de la biblioteca libpvfs. Una vez que el servidor metadata localiza el archivo, le devuelve la ubicación a la aplicación.

Luego ésta puede utilizar la biblioteca para acceder directamente al servidor de E/S correspondiente para leer o escribir sin necesidad de comunicarse con el MGR (**Figura 5.5**).

PVFS utiliza el protocolo TCP/IP para transmitir los datos. Las particiones PVFS pueden ser directamente accedidas por las aplicaciones a través de las bibliotecas libpvfs.a, archivos de include y el archivo de configuración pvfstab que indica cómo acceder a los ficheros PVFS. También se pueden utilizar las funciones ROMIO como interface de E/S de MPICH las cuales:

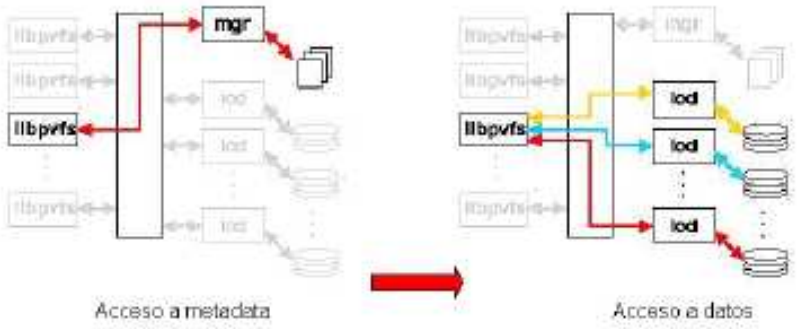


Figura 5.5: Flujo de Metadata y datos en PVFS

Por otro lado, se puede acceder indirectamente a través del kernel, ver Figura 5.6. En este caso los clientes deben utilizar: el demonio pvfsd que se encarga de las transferencias a través de la red, comunicación entre clientes; el módulo del kernel pvfs.o registra en el kernel este tipo de fichero PVFS permitiendo que los archivos PVFS sean accedidos a través de las clásicas llamadas al SO. Este módulo es el que permite que los programas comunes accedan a particiones PVFS que estén previamente montadas; el dispositivo / dev / pvfsd es

utilizado como punto de comunicación entre el módulo pvfs.o y el demonio pvfsd; el ejecutable mount.pvfs es utilizado por el comando mount para llevar a cabo el proceso específico de montar un fichero PVFS.

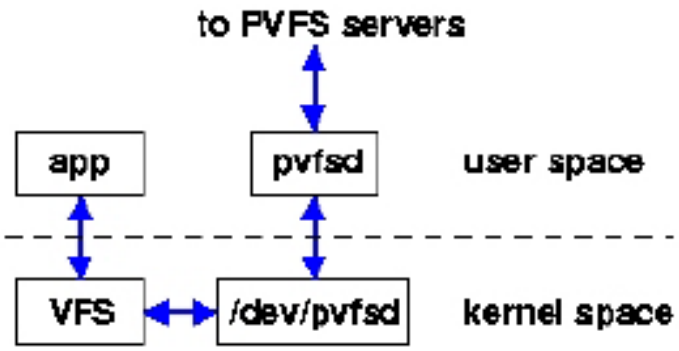


Figura 5.6: Flujo de datos a través del kernel

Los clientes hacen uso transparente de la partición PVFS a cambio de pérdida de performance debido al movimiento de datos entre el kernel y el demonio pvfsd. Por eso existe la opción de utilizar directamente las bibliotecas PVFS, o a través de las funciones ROMIO, para que el programador pueda especificar la distribución física del archivo y configurar particiones lógicas, logrando así mayor performance sobre todo en aplicaciones paralelas que posean operaciones colectivas.

Los programas que estén desarrollados para utilizar funciones de entrada y salida propias de UNIX (read, write), correrán también sobre una partición

PVFS sin necesidad de hacerles ningún cambio. Si se crea un archivo de este modo será distribuido entre todos los servidores de E/S con las variables default, porciones de 64 KB. Notar que en este caso, cada escritura o lectura que se quiera realizar sobre dicho archivo involucrará a todos los servidores de E/S. Con muchos accesos a pequeñas porciones no dará un buen resultado.

Por eso es conveniente utilizar las funciones de la biblioteca estándar (fread, fwrite) que permiten definir el tamaño del búfer a utilizar. Generalmente el PVFS tendrá mayor performance con tamaños de buffers grandes.

5.5. Administración de Procesos

El cluster es utilizado por un grupo de usuarios heterogéneo, alumnos y profesores de diversas áreas. Esto implica que el tipo de problemas que se ejecutan en el cluster es variado y por lo tanto se necesita un método que permita balancear la carga de procesos que se genera a partir de un conjunto de usuarios con distintos requerimientos:

- Mucha memoria
- Poca memoria
- Mucho uso de CPU

- Poco uso de CPU
- Muchas horas de procesamiento
- Pocas horas de procesamiento
- Gran cantidad de pasaje de mensajes
- Poca cantidad de pasaje de mensajes

Es necesario aplicar políticas de administración de procesos para optimizar el 'throughput' de procesos y así utilizar el cluster en forma óptima aumentando la disponibilidad de una máquina de alta performance para un mayor número de investigadores. Para esto se utilizaron las aplicaciones 'Portable Batch System' (PBS) como administrador de procesos y 'MAUI Scheduler' que permite definir políticas para decidir qué proceso de la cola se ejecutará y en qué momento.

5.5.1. Portable Batch System - PBS

PBS es un sistema de control de ejecución de procesos y de asignación de recursos a los mismos (cpu, memoria, cantidad de procesadores, tiempo de ejecución de CPU, tiempo total de ejecución de un proceso 'walltime').

Como tal, aceptará procesos 'batch', un script escrito en shell, atributos de control y se encargará de preservar y proteger el proceso hasta que el mismo comience su ejecución, se ejecute y devuelva la salida a quien le solicitó la ejecución del proceso.

5.5.2. MAUI Scheduler

El encolador MAUI es un sistema avanzado de encolamiento de procesos por lotes, adecuado para la computación científica en gran escala o, en su versión actual, computación de alta performance (High Performance Computing - HPC). Fue diseñado para correr sobre plataformas Alpha e Intel, con sistemas operativos IBM SP y Unix/Linux y puede utilizar distintos administradores de recursos entre ellos PBS (Portable Batch System). 'MAUI Scheduler' provee un mecanismo para enviar a ejecutar procesos, ejecutarlos y hacer un seguimiento del estado del mismo en un sistema con recursos compartidos. Esta tarea es muy importante en un 'batch system' que provee acceso centralizado a recursos distribuidos. Además de proveer una visión global de los recursos del cluster, MAUI puede administrar dichos recursos de manera justa y efectiva. Permite definir políticas de asignación de los recursos, qué proceso corre, dónde y cuando maximizando el 'throughput' del sistema. MAUI soporta un sistema de

reserva de recursos avanzada, niveles de calidad de servicio (QOS), métodos para optimizar la distribución de procesos en la arquitectura del sistema ('backfill') y administración de asignación de los recursos en general ('allocation management'). Su esquema de 'scheduling' está basado en un concepto avanzado de reserva de tiempo ('walltime') combinada con 'backfill'. La principal diferencia con respecto a otros encoladores (e.g. NQS, DQS) es que MAUI permite que algunos procesos de menor prioridad se ejecuten antes que otros de mayor prioridad si y solo si no se produce un retraso la ejecución de mayor prioridad.

5.5.3. Interacción entre PBS y MAUI

MAUI Scheduler es un conjunto de funciones que permiten un 'schedule' racional y capacidades de 'backfill' utilizando un administrador de recursos del sistema, PBS. PBS es un sistema de administración de procesos que provee información de los recursos de la máquina al MAUI y se encarga de ejecutar los procesos según las indicaciones del mismo.

La Figura 5.7 muestra un esquema que explica en forma sintética la interacción entre PBS y MAUI.

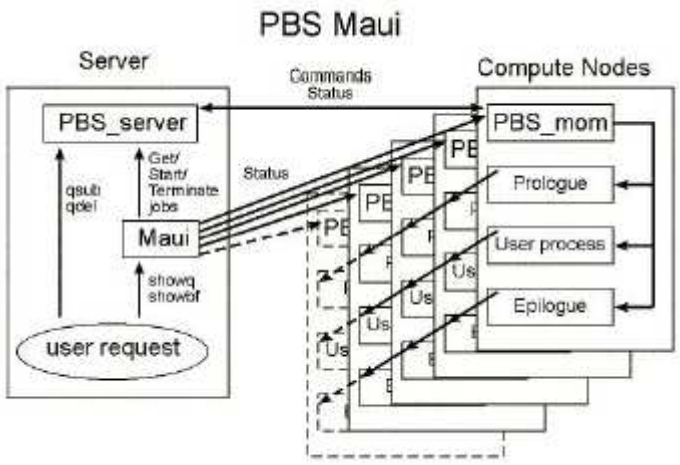


Figura 5.7: Interacción entre PBS y MAUI

En síntesis: el usuario envía a encolar un proceso con el comando qsub al PBS server.

MAUI consulta al PBS server en forma periódica acerca de los procesos encolados.

MAUI consulta a los PBS moms para saber el estado de los recursos del sistema MAUI en base a los requerimientos de los usuarios y a sus reglas predefinidas, asigna prioridades y ordena la ejecución de los procesos.

5.6 HERRAMIENTAS DE PROGRAMACIÓN

5.6.1 MPICH

El paradigma de pasaje de mensajes Message Passing Interface - MPI, es utilizado usualmente en distintas clases de computadoras paralelas, especialmente en aquellas que poseen memoria distribuida. Es un estándar defunciones para llevar a cabo el pasaje de mensajes en la programación paralela.

Puede invocarse desde C y Fortran. Una función de pasaje de mensajes se encarga simplemente de transmitir datos de un proceso a otro a través de la red. MPICH es una de las tantas implementaciones de MPI que actualmente se encuentran disponibles y la que está instalada en el cluster Speedy.

Provee los siguientes tipos de operaciones descriptas en detalle en:
Combinación de contexto y grupos de mensajes a través de un comunicador comunicación punto a punto

- Estructuras de buffers y tipos de datos

• Modos de comunicación: normal (bloqueante y no bloqueante), sincrónica, buffered

Comunicación Colectiva

• Gran variedad de rutinas para pasar datos entre procesos

• Definición de grupos

Control de errores

En los sistemas UNIX, MPICH utiliza la biblioteca llamada p4 como interface sobre la capa TCP. Las transferencias de datos sólo se llevan a cabo durante las llamadas a las funciones de la biblioteca MPICH.

CAPITULO VI

6. CLUSTER DE ALTO RENDIMIENTO BEOWULF

6.1 INTRODUCCION

El surgimiento de plataformas computacionales de comunicación y procesamiento estándares de bajo costo, ha brindado la oportunidad a los programadores de crear herramientas computacionales de costo razonable. Estas realidades permiten la implantación de códigos paralelizados sobre este tipo de plataformas obteniendo un rendimiento competitivo en relación a equipos paralelos especializados cuyos costos de operación y mantenimiento son elevados. Una de las herramientas de más auge en la actualidad son los llamados cluster Beowulf, los cuales presentan diversas capacidades para el cómputo paralelo con un relativo alto rendimiento.

En el verano de 1994 Thomas Sterling y Don Becker, trabajando en CESDIS (Center of Excellence in Space Data and Information Sciencies) bajo el patrocinio del Proyecto de la Tierra y Ciencias del Espacio (ESS), construyeron un Cluster de Computadoras que consistía en 16 procesadores DX4 conectadas por un canal Ethernet a 10Mbps. Ellos llamaron a su máquina Beowulf. La

máquina fue un éxito inmediato y su idea de proporcionar sistemas de base COTS (Material de Aparador) para satisfacer requisitos de cómputo específicos se propagó rápidamente a través de la NASA y en las comunidades académicas y de Investigación. El esfuerzo del desarrollo para esta primera máquina creció rápidamente en lo que ahora llamamos el proyecto de Beowulf.

6.2 HISTORIA

El proyecto Beowulf Cluster tuvo inicio en el verano de 1994 cuando se ensamblaron 16 nodos clusters, para el desarrollo en ciencias de la tierra y el espacio, el proyecto rápidamente se extendió ala NASA algunos laboratorios, también diversas universidades al rededor del mundo.

El primer cluster se formo por 16 nodos DX4 conectados en una canal ethernet el máximo poder de procesamiento fue de un GigaFLOP (Floting Point Operation / sec.)

Los actuales sistemas cluster cuentan con procesadores y tecnología de red cada vez más eficiente y rápida, consisten en cientos, rara vez en miles de nodos con el poder de procesamiento en varios TeraFLOPS.

6.3 CONCEPTO BEOWULF

"Cluster Beowulf no es un paquete software especial, ni una nueva topología de red, ni un núcleo modificado. Beowulf es una tecnología para agrupar computadores basados en el sistema operativo Linux para formar un supercomputador virtual paralelo"[17].

6.4 ARQUITECTURA DE BEOWULF

Beowulf posee una arquitectura basada en multicomputadores el cual puede ser utilizado para la computación paralela. Este sistema consiste de un nodo maestro y uno o más nodos esclavos conectados a través de una red Ethernet u otra topología de red. Esta construido con componentes hardware comunes en el mercado, similar a cualquier PC capaz de ejecutar Linux, adaptadores de Ethernet y switches estándares. Como no contiene elementos especiales, es totalmente reproducible.

Una de las diferencias principales entre Beowulf y un cluster de estaciones de trabajo (COW, Cluster Of Workstations) es el hecho de que Beowulf se comporta más como una sola máquina que como muchas estaciones de trabajo

[17] Información tomada de http://www.clusters.unam.mx/conceptos/index.php

conectadas. En la mayoría de los casos los nodos esclavos no tienen monitores o teclados y son accedidos solamente vía remota o por terminal serie. El nodo maestro controla el cluster entero y presta servicios de sistemas de archivos a los nodos esclavos. Es también la consola del cluster y la conexión hacia el exterior. Las máquinas grandes de Beowulf pueden tener más de un nodo maestro, y otros nodos dedicados a diversas tareas específicas, como por ejemplo, consolas o estaciones de supervisión. En la mayoría de los casos los nodos esclavos de un sistema Beowulf son estaciones simples. Los nodos son configurados y controlados por el nodo maestro, y hacen solamente lo que éste le indique.

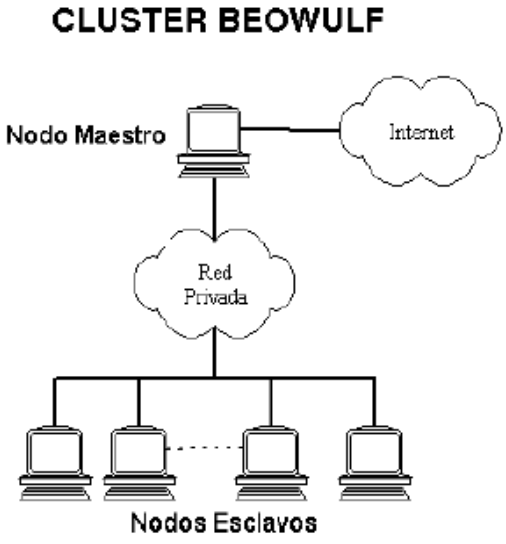


Figura 6.1 Arquitectura de Cluster Beowulf

6.5 POR QUÉ BEOWULF

Durante mucho tiempo, se utilizó la típica historia sobre la búsqueda de un nombre basándonos en la metáfora del tío chiquitín que le da una paliza al grandullón, tal y como ocurría en "David y Goliat". La historia seguía diciendo que "Goliat" ya había sido utilizado para un famoso ordenador histórico y, de todas maneras, el gigante estaba en el lado equivocado de la metáfora.

En la búsqueda de otras culturas para folklore equivalentes, recordó la épica saga de Beowulf, el héroe escandinavo quien salvó el reino danés de Hrothgar al derrotar el monstruo Grendel.

6.6 EVOLUCIÓN DE LA ARQUITECTURA DEL SISTEMA BEOWULF

Los Beowulf se distinguen de otros sistemas de clusters, en que no imponen una arquitectura del sistema. Los componentes primarios del sistema que manejan la arquitectura se pueden descomponer en el procesador, memoria, red de trabajo y sistema de almacenamiento secundario. Pero el procesador y la red de trabajo han tenido el impacto más visible en los cambios de la arquitectura Beowulf.

El poder de procesamiento fue un factor dominante en el desempeño inicial de los sistemas Beowulf. La aparición sucesiva de generaciones de hardware, hicieron que el procesador disminuyera su impacto en el funcionamiento total del sistema. El ancho de banda de memoria ahora ha reemplazado el rol del procesador como un cuello de botella del funcionamiento. La red de trabajo siempre ha sido un factor limitante desde los primeros días. Sin embargo, la red de trabajo todavía impone un límite estricto en el escalamiento de los sistemas Beowulf. Una vez que se mueve entre los 100 y 200 nodos, la red de trabajo fácilmente se satura. La disponibilidad de compra de la Ethernet de 1 Gbps eliminará este problema, y permitirá escalar a un gran número de nodos para aplicaciones que puedan incorporar mucho paralelismo.

6.7 QUE USOS TIENEN LOS CLUSTERS

Los sistemas cluster han empezado a ser usados en donde los sistemas mainframes o supercomputadoras eran utilizadas en el pasado.

Los clusters han formado parte significativa en el avance de las siguientes industrias.

- Investigación científica y defensa
- Procesos de energía sísmica

- Efectos visuales y entretenimiento
- Internet

En algunos puntos específicos

- Identificar nuevas partículas sub-atómicas
- Investigación para descubrir una droga para el SIDA
- Exploración de recursos petróleo, gas
- Investigación del genoma humano

En el momento de cargar el sistema operativo, los constructores del Beowulf tienen que tomar una serie de decisiones acerca de la configuración. Estas decisiones tienen que pensarse cuidadosamente, ya que cambiarlas requiere reinstalar todos los nodos. La mayoría de los clusters Beowulf basados en Linux usan la distribución de Red Hat Linux, pero cualquier distribución tendría que soportar una clusterización básica.

Cargar Red Hat es muy sencillo y puede hacerse vía CD-ROM o a través de una tarjeta de red desde Internet o desde el primer nodo en el cluster, que tendría que tener una copia de la distribución. Hemos encontrado

beneficioso cargar el sistema operativo en los discos de cada nodo vía FTP desde el nodo primario en lugar de montar las particiones raíz vía NFS.

Así sacrificamos un poco de facilidad de mantenimiento por un mejor rendimiento en la ejecución. Esta práctica elimina tráfico de red innecesario, preservado el ancho de banda para el paso de mensajes mientras las aplicaciones se ejecutan.

El entorno de ejecución de Red Hat Linux require solo unos 100 MB de espacio de disco en cada nodo; no obstante, es útil incluir compiladores y otras utilidades en todos los nodos para posibilitar la compilación paralela. Con esto, nuestra configuración requiere 175 MB de espacio en disco para el sistema operativo. Mientras algunos clusters se configuran para usar como memoria virtual un archivo del sistema de archivos normal, es más eficiente usar una partición dedicada a la memoria virtual en el disco local.

Se acepta normalmente que la cantidad de espacio razonable para la memoria virtual es el doble que el tamaño de la memoria, hasta los 64 MB. Para los nodos con más de 64 MB de memoria, el espacio para la memoria virtual tendría que ser igual a la cantidad de memoria. En la práctica, uno

puede desear reservar 128 de espacio para la memoria virtual si el tamaño de la memoria va de 64 a 128 MB. Como resultado, si un nodo previsor con 32MB de memoria tiene dos discos de 202 MB, uno podría cargar Linux en el disco principal (como una partición única) y usar el segundo disco para espacio de memoria virtual (64MB) y espacio local para la ejecución (138 MB).

6.8 SISTEMA OPERATIVO

Otro componente dominante para remitir compatibilidad, es el software del sistema usado en Beowulf. Con la madurez y la robustez de Linux, el software GNU() y de la estandarización de envío de mensajes vía PVM y MPI.

El primer Beowulf fue construido para tratar un requisito de cómputo determinado de la comunidad de ESS. Fue construido por y para el investigador con experiencia en programación paralela. Muchos de estos investigadores han pasado años luchando con los vendedores de MPP(Massively Parallel Prosessor), y administradores de sistemas sobre la información detallada del funcionamiento y luchando con las herramientas subdesarrolladas y nuevos modelos de programación.

6.9 PORQUE UTILIZAR LINUX PARA CONSTRUIR UN BEOWULF

Linux es el Sistema Operativo de código abierto más popular en el mundo, su éxito se debe a diversos factores pero la estabilidad madurez y fácil diseñó han sido sus claves para el crecimiento en el mercado. La estabilidad y disponibilidad de Linux ha creado un booming en el mercado comercial para productos que no son compatibles con otros Sistemas Operativos de código abierto, la mayoría de compañías han incorporado en su modelo de negocios a Linux y otras compañías lo utilizan porque hace más prácticos a sus negocios.

La razón más importante para usar Linux en un Beowulf es su flexibilidad debido a que es de código abierto puede ser reestructurado, modificado fácilmente sin importar el tipo de trabajo.

6.9.1 Cuál es La Mejor Distribución

Se puede escoger una distribución considerando tres factores:
Soporte, Lenguaje y facilidad de uso.

Soporte.- Aquel que se lo puede obtener por Internet, algunos incluyen librerías adicionales a bajo costo.

Lenguaje.- Se refiere al lenguaje nativo por ejemplo en Alemania Suse tiene un excelente soporte de lenguaje, los alemanes utilizan Suse hasta el punto que si nosotros tuviéramos algunas inquietudes por e-mail ellos contestarían las mismas.

6.9.2 Un Ejemplo

Uno de los motores de búsqueda más aclamados el Google el cual usa miles de servidores utilizando Linux para indexar y proveer capacidades de búsqueda avanzadas en la web Google no es un cluster científico, su tamaño demuestra flexibilidad y adaptabilidad de Linux.

Se utiliza en computadoras desde el tamaño de un Palm hasta clusters, han demostrado su utilidad y estabilidad para casi todas sus tareas.

Otra razón para escoger Linux es el soporte para muchos tipos de procesadores.

6.10 REQUERIMIENTOS MÍNIMOS

Un nodo tendría que contener, como mínimo, una CPU 486 y una placa madre Intel. Los procesadores Intel 386 funcionarían, pero su rendimiento raramente valdría la pena. Los requerimientos de memoria dependen de los requerimientos de datos de la aplicación y del paralelismo, pero un nodo tendría que contener como mínimo 16MB de memoria. La mayoría de aplicaciones necesitaran 32MB o más por nodo. Usando un espacio de disco centralizado, los nodos se pueden inicializar desde un disquete, un pequeño disco duro o un sistema de archivos en red. Entonces los nodos pueden acceder a su partición raíz desde un servidor de archivos a través de la red, normalmente usando el Network File System (NFS). Esta configuración funciona mejor en un entorno con mucho ancho de banda en las conexiones y con un gran rendimiento en el servidor de archivos. Para un mejor rendimiento bajo otras condiciones, cada nodo tendría que tener suficiente espacio en el disco local para el sistema operativo, la memoria virtual y los datos. Cada nodo tendría que tener como mínimo 200 MB de espacio de disco para los componentes del sistema operativo y la memoria virtual, pero 400 MB o más permite tener espacio libre que puede ser usado para las aplicaciones en tiempo de ejecución. Como mínimo cada nodo tiene que incluir una tarjeta de red

Ethernet o Fast Ethernet. Alternativamente, interconexiones de más alto rendimiento, incluyendo la Gigabit Ethernet y la Myrinet, podrían ser usadas en conjunción con CPUs más rápidas.

Finalmente, cualquier tarjeta de vídeo, una lectora de disquetes, la caja y la batería, completan un nodo funcional.Los teclados y los monitores solo se necesitan para la carga y configuración inicial del sistema operativo, a no ser que las máquinas individuales sean usadas interactivamente además de servir como nodos en el sistema paralelo.

Todos los componentes hardware escogidos necesitan el soporte de drivers o módulos en Linux. Generalmente esto no es un problema si el hardware tiene algunos meses. Muchas fuentes de información acerca del hardware soportado están disponibles en la WWW. Particularmente interesante es la completa lista de drivers y la excelente documentación para una amplia variedad de tarjetas de red hecha por Donald Becker's. El soporte de las tarjetas de vídeo no es importante en los nodos, ya que normalmente no ejecutan un servidor X; no obstante, el nodo master puede gestionar el cluster. Un servidor X seria útil para esta máquina. Si un componente particular presenta un problema,

preguntar en los grupos de noticias puede ayudar a encontrar información acerca del hardware soportado por Linux.

6.11 CLASIFICACIÓN CLUSTER BEOWULF

Dentro de la taxonomia de computadores paralelos Beowulf esta en algún lugar entre:

- **Red de estaciones de trabajo.**

  Existen diferencias sutiles entre un cluster Beowulf y una red de estaciones de trabajo, pero son muy significativas en relación al rendimiento.

- **Red interna aislada.**

  La carga de la red sólo depende de las aplicaciones.

- **Nodos dedicados al cluster**

  Fácil manejo del balanceo de carga de los procesadores.

  Para establecer las diferencias entre los distintos tipos de sistemas Beowulf se presenta la siguiente clasificación.

- **Clase I**. Son sistemas compuestos por máquinas cuyos componentes cumplen con la prueba de certificación ``Computer Shopper'' lo que significa que sus elementos son de uso común, y pueden ser adquiridos muy fácilmente en cualquier tienda distribuidora. De esta manera, estos clusters no están diseñados para ningún uso ni requerimientos en particular.

- **Clase II.** Son sistemas compuestos por máquinas cuyos componentes no pasan la prueba de certificación ``Computer Shopper'' lo que significa que sus componentes no son de uso común y por tanto no pueden encontrarse con la misma facilidad que los componentes de sistemas de la clase anterior. De tal manera, pueden estar diseñados para algún uso o requerimiento en particular. Las máquinas ubicadas en esta categoría pueden presentar un nivel de prestaciones superior a las de la clase I.

Latencia .- El tiempo que se toma para un bloque específico de datos en una pista de datos rotar alrededor de la cabeza lectora/escritura.

Significando el lapso de tiempo entre el envío y recepción de un mensaje, medido en décimas de segundo. Los clusters son colecciones débilmente

acopladas de computadoras stand-alone usualmente ubicadas en el mismo sitio y "dominio administrativo" con latencias entre 10 y 100 microsegundos. Los MPPs comprende varios procesadores fuertemente conectados dentro de la misma unidad exhibiendo latencias entre: medio microsegundo y 5 microsegundos.

6.12 BENEFICIOS DEL BEOWULF

Además de la ventaja significante precio/ desempeño de los sistemas de clase Beowulf. Beowulf proveen varias ventajas importantes que van desde una inclusión rápida de la tecnología COTS hasta los lenguajes de programación y API, ampliamente usados.

En el mercado de la tecnología dinámica de hoy son los sistemas "low-end" los que primero incorporan los dispositivos de más alto desempeño y más actuales. Esto es porque representan el mercado compartido más amplio y proveerán los mayores ingresos a corto plazo. Ya que Beowulf no requieren un tiempo de desarrollo tan pronto como aparecen nuevos avances en la tecnología de chips en el mercado estos pueden ser incorporados inmediatamente en las

configuraciones del Beowulf. Lleva mucho más tiempo para los mismos dispositivos encontrar su camino hacia la MPP comerciales.

De ahí una ventaja del modelo Beowulf es la respuesta rápida a la tecnología de punto.

Estos son ensamblados a partir de subsistemas que pueden ser adquiridos.

No existe una arquitectura que sea fija o especifica en amplio rango de procesadores y de topologías de sistemas, son posibles y pueden ser optimizados a un más pueden ser cambiados de acuerdo a nuevas necesidades y a la aparición de nuevas tecnologías.

La forma del sistema no se limita a la especificación del vendedor sino solo a la imaginación de los usuarios por tanto otro beneficio del Beowulf es su propiedad llamada configuración (Just in Place) justo en el lugar.

Los sistemas Beowulf son escalables, existen varios tamaños de sistemas desde un pequeño número de nodos conectados por un simple Hub hasta un sistema de topología compleja de ciento de procesadores los cuales pueden ser

expandidos a medida que aparezcan nuevos recursos o el tamaño de los requerimientos aumente. Beowulf usa la misma semántica para el paso de mensajes que el MPP (Massively Parallel Procesor) de manera que proveen la misma interfase programadora de la aplicación de usuario.

6.13 TIPOS DE CONFIGURACIONES

Las computadoras en un cluster se comunican mediante una red de interconexión. Existen distintos tipos de red disponibles. Un Hub ethernet es un dispositivo de red que actúa como un bus broadcast, en el cual una señal de entrada es amplificada y distribuida a todos los puertos. Sólo un par de computadoras se pueden comunicar correctamente a la vez. Si dos o más computadoras envían paquetes a la vez ocurrirá una colisión. Por tanto, el ancho de banda de un Hub Ethernet es equivalente al ancho de banda del link de comunicación, 10Mbps para estándard Ethernet, 100Mbps para Fast Ethernet, y 1Gbps para Gigabit Ethernet. Un Switch Ethernet ofrece más ancho de banda acumulado permitiendo múltiples comunicaciones simultáneas. Si no hay conflictos en los puertos de salida, el Switch Ethernet puede enviar múltiples paquetes simultáneamente. El coste por puerto de un Switch Ethernet es aproximadamente cuatro veces más grande que un Hub Ethernet

6.14 CARACTERÍSTICAS CLUSTER BEOWULF

- **Flexibilidad**

  Hardware común:

  Red, procesador, etc.

  Software de dominio público

  Linux, MPI, PVM, etc.

- **Escalabilidad**

  Los cluster permiten agregar nuevos componentes para aumentar el nivel de prestaciones sin necesidad de eliminar los elementos ya existentes.

- **Disponibilidad**

  Existe redundancia natural, cada nodo posee sus propios componentes: bus, memoria, procesador. Se puede implementar políticas para el reemplazo rápido en caso de falla del servidor maestro.

- **Rendimiento**

  En principio las aplicaciones paralelas son más rápidas.

Ejemplos

Cluster de 1000 nodos utilizado para la investigación de algoritmos genéticos

www.genetic-programing.com

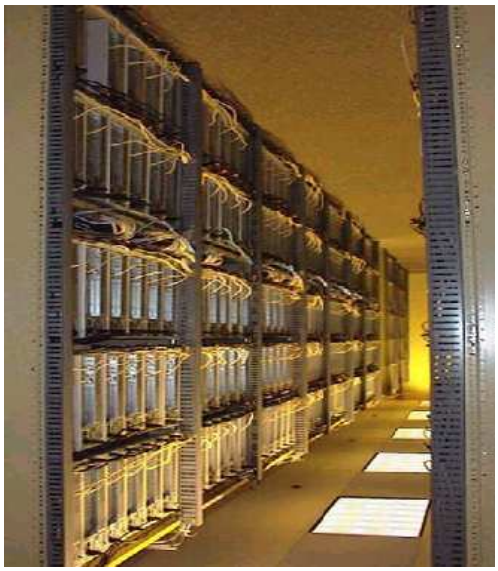


Figura 6.2 Cluster de 1000 Nodos

"PIRUN Beowulf Cluster Kasetsart University, Thailand PIRUN(Piles of Inexpensive and Redundant Universal) Nodes 72 nodos PIII 500 MHz, 128 MB Provee cálculo intensivo en general y funciona como superservidor de Internet"[18].

[18] Información tomada de http://www.cecalc.ula.ve/HPCLC/slides/day_02/Cluster_School.pdf

Figura 6.3 PIRUM Beowulf

Chama Centro de Supercomputación ULA, 28 nodos: PIII dual, 2TB de disco, 20GB RAM Cálculo paralelo, Química, Física, etc[19].

Figura 6.3 Chama Centro de Supercomputación

[19] Información tomada de http://www.cecalc.ula.ve/HPCLC/slides/day_02/Cluster_School.pdf

6.15 DISEÑO

Es necesario tomar en cuenta diversos factores para el diseño de un cluster Beowulf que contribuyan a un mejor desempeño:

- Disco
- Memoria
- Motherboard
- Procesador
- Multiprocesadores simétricos
- Red

OPIUM - OSCAR Password Installer and User Management. Para sincronizar los usuarios del cluster y configurar el ssh de los mismos.

Programación Bibliotecas que permiten que un conjunto de máquinas heterogéneo (Windows/Unix) colaboren entre si a través de una red para ser usadas como una única máquina paralela. De este modo se pueden resolver problemas costosos en menor tiempo. Las herramientas que se pueden instalar con Oscar Cluster son:

- MPI
- LAM MPI

- PVM

Por otro lado, se instala el PVFS para manejo de ficheros distribuidos. También se instalaron herramientas de compilación como son:

Fortran 77 (gcc-g77-3.3.1)

C (gcc-3.3.1)

C++ (gcc-c++-3.3.1)

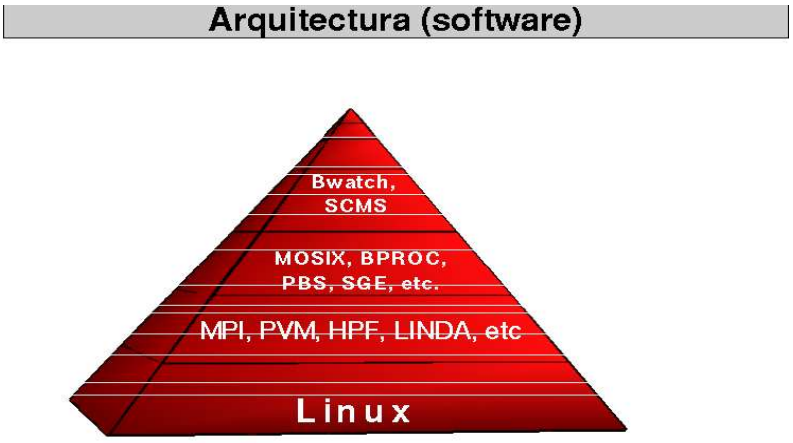


Figura 6.4 Arquitectura de Software Beowulf

CAPITULO VII

7. OSCAR CLUSTER

7.1 INTRODUCCIÓN

Oscar (Open Source Cluster Application Resource), el paquete de aplicación es propuesto para simplificar tareas complejas requeridas para instalar un clúster.

Varios paquetes relacionados con el HPC están instalados por defecto, tales como MPI, PVM, etc.

7.2 TERMINOLOGÍA

Cada máquina individual de un clúster es mencionado como un nodo. Existen dos tipos de nodos: servidor y cliente. Un nodo de servidor es responsable de atender las peticiones de nodos de clientes. Un nodo cliente está dedicado a procesar.

Un clúster de Oscar consiste de un nodo de servidor y uno o más nodos clientes, donde todos los nodos de cliente deben tener hardware homogéneo.

7.3 DISTRIBUCIONES SOPORTADAS

Oscar se ha examinado para trabajar con varias distribuciones. Tabla 7.3 las listas cada distribución y versión y especifica el nivel de soporte para cada uno. A fin de asegurar una instalación exitosa, la mayor parte de los usuarios deben utilizar una distribución que es listada como enteramente soportado.

Distribución y la versión

Distribución	Soporte
RedHat 9.0	Enteramente soportado
RedHat 8.0	enteramente soportado
Mandráke 9.0	enteramente soportado

Tabla 7.3: Oscar soporte de Distribuciones

7.4 REQUISITOS MÍNIMOS DE SISTEMA

El siguiente es una lista de requisitos mínimos de sistema para el nodo de servidor de Oscar:

- **Nodo Servidor:**

La siguiente es una lista de requisitos mínimos de sistema para el nodo servidor Oscar:

- Cpu 586 en adelante.
- Tarjeta de Red que soporte el protocolo TCP/IP.
- Si se utiliza en el nodo servidor una red publica y otra red privada se necesita 2 tarjetas de Red que soporte el protocolo TCP/IP.
- 4 Gigas de espacio libre, 2Gigas para la raíz y 2 Gigas para el under /var
- Una versión de Linux instalada como:

  Red Hat 8.0

  Red Hat 9.0

  Mandrake 9.0

- **Nodo Cliente**

La siguiente es una lista de requisitos mínimos de sistema para los nodos cliente de Oscar:

- Cpu 586 en adelante.
- Un disco para cada nodo cliente al menos 2 GB (Oscar formateará el disco durante la instalación)

- Tarjeta de Red que soporte el protocolo TCP/IP.
- La misma distribución que la del nodo servidor
- Todos los clientes deben tener la misma arquitectura
- Monitor, Teclado no son requeridos
- Floppy o inicio con la opción del PXE que permita el arranque desde la red.

Los paquetes de distribución de Oscar pueden descargarse de: http://oscar.sourceforge.net/, La versión que hemos utilizado es: Oscar versión 3.0, cada paquete de Oscar tiene su propia instalación

El instalador de Oscar instala el paquete de MySQL en el nodo de servidor.

7.5 VISIÓN GENERAL DEL DE INSTALACIÓN DE SISTEMA (SSI)

Fue escogido para ser el mecanismo de instalación para Oscar por múltiples razones:

- Los SSI no requieren que los nodos clientes tengan instalado Linux
- Los SSI mantienen una base de datos que contiene la información de

instalación y configuración de cada nodo en el clúster.

- Los SSI usan rpm como un estándar para instalación de software
- Los SSI soportan hardware heterogéneo

Los conceptos usados dentro de SIS son los siguientes: el primer concepto es el de una imagen. Está definida para el uso de los nodos del cluster es una copia de los archivos del sistema operativo almacenada en el servidor. Los nodos clientes instalan duplicando esta imagen a sus particiones de disco locales. Otro concepto importante del SSI es la definición de cliente, un cliente SSI está definido para cada uno de sus nodos del clúster. Estas definiciones de cliente mantienen la información perteneciente sobre cada cliente. El nodo de servidor es responsable de crear la base de datos de información y para atender las peticiones de instalación de cliente. La información que es guardada para cada cliente incluye:

• La información IP como: hostname, dirección IP, ruta.

• Nombre de la imagen.

7.6 PROCEDIMIENTO DETALLADO DE LA INSTALACIÓN DEL CLÚSTER OSCAR

Todos los pasos que se muestran son obligatorios a menos que explícitamente esté marcado como opcional.

a)Instalación y configuración del servidor

Preparamos la máquina para ser usada como el nodo servidor en el clúster Oscar.

a.1) Instalación de Linux en la máquina servidor

Si posee una máquina, instalar Linux pero debe asegurar que tenga los requerimientos de distribuciones que se ha mencionado anteriormente.

Es mejor no instalar las actualizaciones de la distribución después que se instala Linux; primero hay que instalar el Oscar Cluster , y entonces instalar las actualizaciones de la distribución.

a.2) Espacio en el disco y las consideraciones de directorio

Oscar Cluster tiene ciertos requerimientos de espacio en disco del servidor. El espacio será necesario para almacenar los rpms de Linux y la o las imágenes.

Los rpms serán almacenados en /tftpboot/rpm. 2GB es suficiente para almacenar los rpms. Las imágenes son guardadas en /var/lib/systemimager y necesitamos aproximadamente 2GB por imagen. Aunque sólo una imagen es requerida por Oscar Cluster, se puede crear más imágenes en lo sucesivo.

a.3) Descargar una copia de Oscar Cluster y desempaquetar en el servidor

Ubicar el paquete de distribución de Oscar Cluster en el directorio raíz en el nodo servidor.

Aunque no existe ningún directorio de instalación requerido (no puede usar el directorio /usr/local/oscar, /opt/oscar, /var/lib/oscar, o /var/cache/oscar estos son reservados para ser usados por el OSCAR)
Donde <filename>sea oscar-3.0.tar.gz (distribución regular) u oscar-including--srpms-3.0.tar.gz (distribución extra rizada).

El directorio	Los contenidos
~/oscar-3.0/	el directorio de Oscar bajo
~/oscar- 3.0/COPYING	la licencia pública general
~/oscar-3.0/dist	scripts de la distribución para configure/install
~/oscar-3.0/doc	directorio de documentación
~/oscar-3.0/images	images auxiliares usadas en el GUI
~/oscar-	script principal de la instalación
3.0/install_clustes	librerias y rutinas auxiliares
~/oscar-3.0/lib	ejemplos de archivos de configuración
~/oscar-3.0/oscarsamples	RPM e archivos de instalación
~/oscar-3.0/packages	Texto y documentos README
~/oscar-3.0/readme	Contiene scripts que hacen la mayor parte del
~/oscar-3.0/scripts	trabajo
~/oscar-3.0/share	Más archivos auxiliares
~/oscar-3.0/testing	Contiene scripts para el test del cluster
~/oscar-3.0/VERSION	Archivo que contiene la versión del OSCAR

Tabla 7.2: distribución del Oscar y esquema de directorio.

b) Configuración e instalación de Oscar Cluster

Después de desempaquetar necesita configurar e instalar Oscar Cluster.

b.1) El primer paso para ejecutar la configuración es:

```
#  cd /root/oscar-3.0
# ./configure
```

Una vez que el script se haya completado satisfactoriamente el siguiente paso es:

```
# make instale
```

b.2) Copiar los rpms de la distribución instalada en el directorio /tftpboot/rpm

En este paso, copiar todos los rpms incluidos en la distribución de Linux en el directorio /tftpboot/rpm

b.3) Ejecutar el instalador del Oscar Cluster

Cambiar al directorio OSCARHOME y arrancar el asistente de instalación de Oscar Cluster:

OSCAR HOME

./install_clúster <eth0>

El asistente, como se muestra en la figura 7.1, suministra una guía para el resto de la instalación de clúster. Para el uso del asistente, completará todos los pasos en orden.

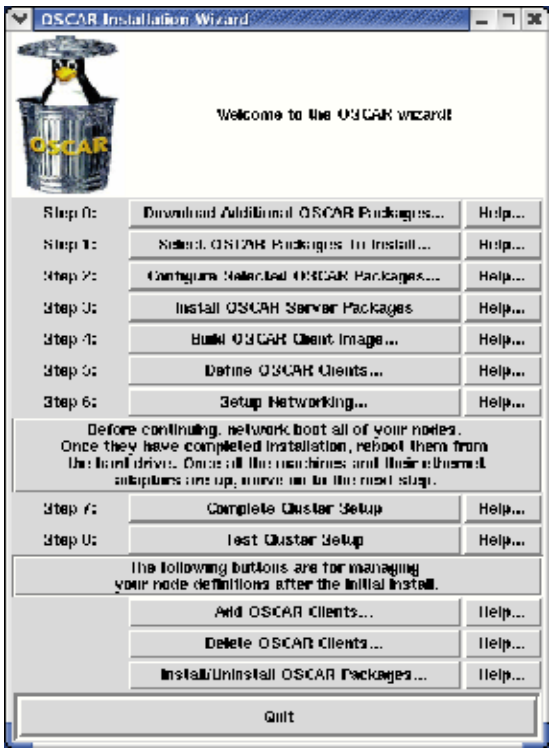


Figura 7.1: Oscar Wizard

b.4) Descargando los paquetes de Oscar Cluster adicionales

Este paso es opcional.

El primer paso del asistente, "paso 0", activar para descargar paquetes adicional de modo que serán instalados durante la instalación principal de Oscar Cluster, si desea agregar direcciones adicionales como el Ganglia que es un paquete que no está incluido en la distribución principal del Oscar Cluster.

b.5) Seleccionando paquetes para instalar

Este paso es opcional.

Si quiere cambiar la lista de paquetes que serán instalado, puede cambiar con la opción <Select Oscar Pack to install> Este paso es opcional porque todos los paquetes por defecto están directamente incluidos en el Oscar Cluster.

b.6) Configurando paquetes de Oscar Cluster

Este paso es opcional.

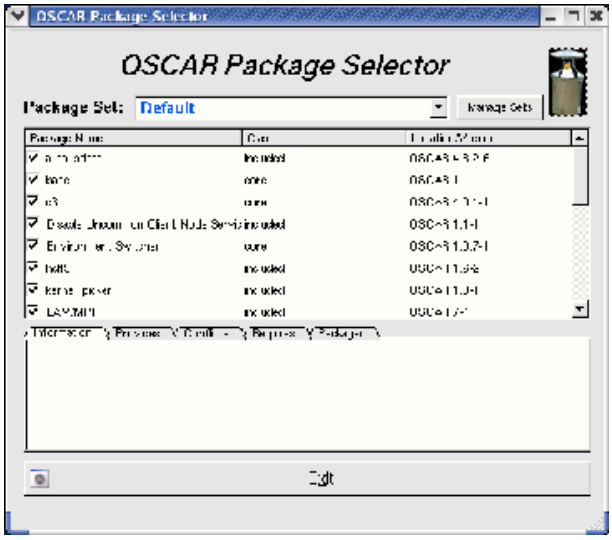


Figura 7.2 Selección de Paquetes Oscar

Ciertos paquetes de Oscar permiten configurarse pero no es necesario.

b.7) Instalar paquetes del nodo Servidor Oscar Cluster

Pulsar el botón de <Install Oscar Server Packages>. Este invocará la instalación de varios rpms y configuración de los mismos en el nodo servidor. La ejecución tarda varios minutos; el texto de salida y mensajes de estatus aparecerán en otra ventana emergente indicando el éxito o fallo de este paso.

b.8) Construir la imagen cliente

Antes presionar <Build Oscar Client Image>debe asegurar que las siguientes condiciones en el servidor se cumplan:

• Que el archivo de configuración del demonio de SSH (/etc/ssh/sshd_config) la línea del PermitRootLogin esté YES. Después de la instalación de Oscar Cluster, establecer como NO, pero aquella línea necesita estar como YES durante la instalación porque la configuración de este archivo es copiada a los nodos clientes.

• Por la misma razón, asegurar que la configuración de demonio TCP wappers con los archivos etc/hosts.allow y /etc/hosts.deny deben permitir el tráfico en la red privada. Si estas condiciones no son tomadas en cuenta, la instalación puede fallar durante los pasos posteriores.

Pulse el botón de <Build Oscar Client Image> asegurar que esté en la inicialización de la unidad de disco duro local antes de alzar el servicio de red.

Construir la imagen toma varios minutos, la barra roja indicadora en el fondo de la ventana indica la variación del proceso:

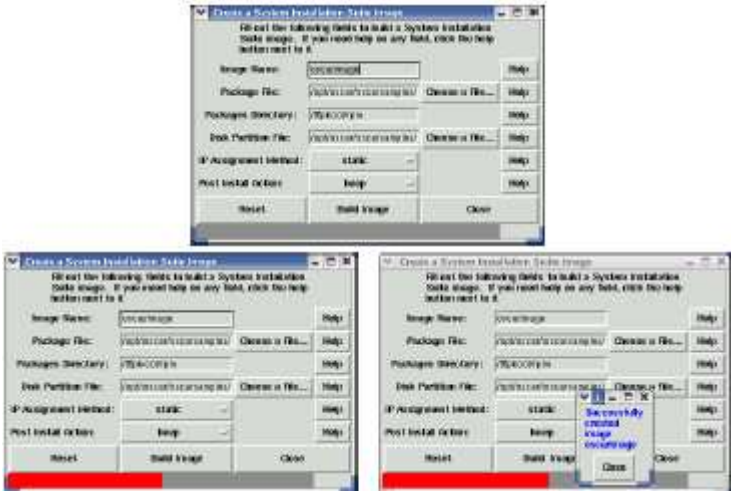


Figura 7.3: Construcción de la imagen

b.9) Disk partitioning

El archivo de partición del disco contiene una línea para cada partición deseado, donde cada línea tiene el siguiente formato

<partition> <size en megabytes> <type> <mount point> <options>

Esté es un ejemplo del archivo para un disco SCSI:

/dev/sda1 24 ext2 /boot defaults

/dev/sda5 128 swap

/dev/sda6 * ext2 / defaults

Nfs_oscar:/home - nfs /home rw

Un * en la columna llena la el disco entero.

b.10) Definición de los clientes de Oscar Cluster

Pulsar el botón <Define Oscar Clients>. Una ventana de diálogo es mostrada, la información de entrada puede ser cambiada, pero con la información que se muestra por defecto es suficiente para este caso lo único que necesitamos ingresar es el number of host para especificar cuántos clientes vamos a crear.

1. El campo imagen name fied especifica el nombre de imagen con la que va a trabajar.

2. El campo Domain named field especifica el nombre de dominio IP de los clientes. Ello debe contener el dominio del nodo servidor (si tiene uno); si el servidor no tiene un nombre de dominio, el valor predeterminado **oscardomain** será puesto en el campo, este campo debe tener un valor no puede estar en blanco.

3. El campo base named field especifica la primera parte del nombre del cliente y hostname. Este nombre no puede contener un carácter "_".

4. **El number of host field** aquí especifica cuántos clientes va a crear. Este número debe ser mayor **que 0.**

5. El starting number especifica el índice para añadir al nombre de la base bajo para derivar el nombre del primer cliente el incremento para cada cliente es subsecuente.

6. El padding especifica el número de dígitos para los nombres de los clientes, para 3 dígitos tendrá oscarnode001.

7. El starting ip especifica la dirección IP del primer cliente.

Los clientes se tendrán direcciones IP comenzando con esta dirección, el incremento por cada cliente es sucesivo.

8. El subnetmask especifica la mascara de las IP para los clientes.

9. El default gateway especifica la ruta predeterminada para todos los clientes.

Cuando terminá de ingresar toda la información pulsamos el botón **<addclients>**los clientes son creados en la base de datos.

Figura 7.4. Definición de los clientes

b.11)Configuración de la red

La dirección MAC de un cliente contiene 12 dígitos hexadecimales por ejemplo, " 00:0A:CC:01:02:03 " estas direcciones especifican los clientes dentro de la red antes de asignar la direcciones IP, utilizamos DHCP para asignar las direcciones IP a los clientes, al reunir todas la direcciones IP coleccionándolas o asignándolas desde un archivo presionamos el botón de <Setup Networking>.

El botón <Floppy> construirá un disco de arranque para los nodos de clientes que no soportan inicializar desde la red El botón <setup networkt> configurará

el servidor las peticiones de inicialización de PXE si el hardware de los clientes soporta inicio desde la red.

Oscar 3.0 posee una nueva característica la cual configura automáticamente la opción de multicast.

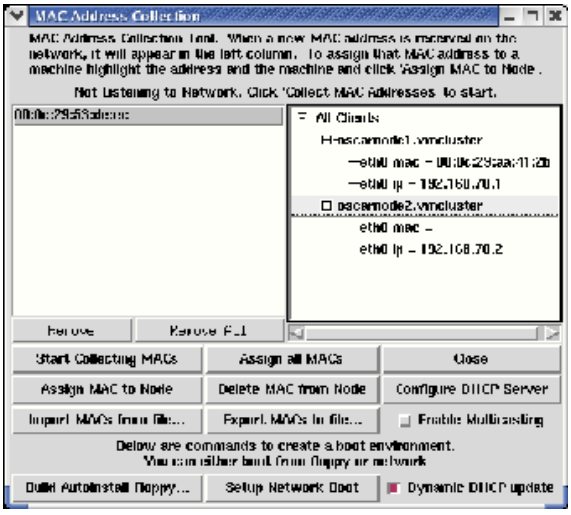


Figura 7.5: Colección de las direcciones de los clientes

b.12) Instalación de los clientes

Durante esta fase, debe inicializar los nodos clientes ya sea desde la red o desde el diskette para luego de forma automática ser instalados y configurados.

b.13) Verificación del estatus de terminación de los nodos

Después de varios minutos, los clientes deben completar la instalación. Puede mirar las consolas de los clientes hasta que se pare, reinicie, o emita un sonido corto e incesantemente cuando la instalación es completada.

El tiempo requerido por la instalación depende de las capacidades de su servidor, y sus clientes.

b.14) Reiniciar los nodos de cliente

Después de confirmar que un cliente ha completado su instalación, debe reiniciar el nodo desde su unidad de disco duro.

b.15) Completación de la instalación del clúster Oscar

Asegurar que todos los nodos clientes estén inicializados desde el disco duro para proceder con el siguiente paso.

Presionar el botón <Complete Cluster Setup>. Entonces se ejecutan las configuraciones de la instalación final scripts en cada nodo. Una ventana emergente nos indica el éxito o fallo de este paso.

b.16) Prueba de la instalación del cluster

Una simple prueba realiza en este paso en el cual nos asegura que los componentes como (SSH, PBS, MPI, PVM, etc.) esté funcionando correctamente. Los servicios básicos del clúster se verifican y entonces son puestos en el root.

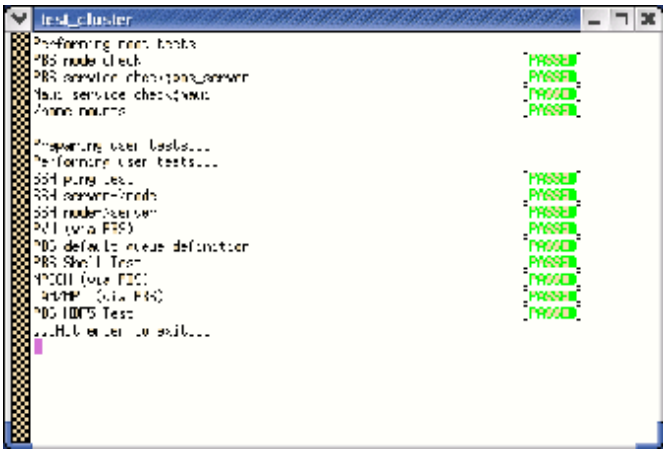


Figura 7.6: Test de la instalación del cluter

La instalación de clúster esta ahora completa. Los nodos de clúster están listos para el trabajo.

CAPITULO VIII

8. GANGLIA TOOLKIT

8.1 INTRODUCCIÓN

Ganglia es un software que provee monitoreo en tiempo real y ejecución de ambientes gráficos usado por cientos de universidades tanto privadas como gubernamentales en implementaciones de cluster comerciales alrededor del mundo. Ganglia es tan fácil de usar que al igual como puede correr en 16 nodos de un cluster puede correr en 512 o mas nodos en un sistema de ambiente distribuido.

Inicialmente Ganglia fue desarrollada en la Universidad de Berkeley por la división de ciencias computacionales como manera de enlazar cluster entre los campus de manera lógica. Ganglia fue desarrollado en un ambiente universitario y es completamente Open Source y no tiene componentes adicionales que pertenezcan a un propietario. Todos los datos son intercambiados definidamente por XML y XDR para un máximo de extensibilidad y portabilidad.

El monitor core permite que monitoree cualquier número de métrica de un host en tiempo real. En el presente el monitor core puede correr bajo sistemas operativos como Linux, FreeBSD, Solaris, AIX, Tru64, y IRIX. En Windows esta en versión Beta.

Ganglia no enlaza nodos de un cluster con otro, la manera lógica es que enlaza cluster con otros cluster. Ganglia enlaza líneas de cluster y computación distribuida por lo que llamamos "cluster to cluster" (C2C).

CONCEPTO

"Ganglia es un software que provee monitoreo en tiempo real y ejecución de ambientes usado por cientos de universidades"

8.2 COMO FUNCIONA GANGLIA

Para entender como funciona Ganglia es importante saber que este corre con dos demonios los cuales son el Ganglia Monitoring Daemon (gmond), el Ganglia Meta Daemon (gmetad) y el Ganglia Web Frontend.

El Ganglia Monitoring Daemon (**gmond**), es un demonio multi-threaded el cual corre en cada uno de los nodos del cluster que se desea monitorear. Su instalación muy fácil. No se tiene porque tener un sistema de archivo NFS en común ni una base de datos. Gmond tiene su propia base de datos distribuida y su propia redundancia.

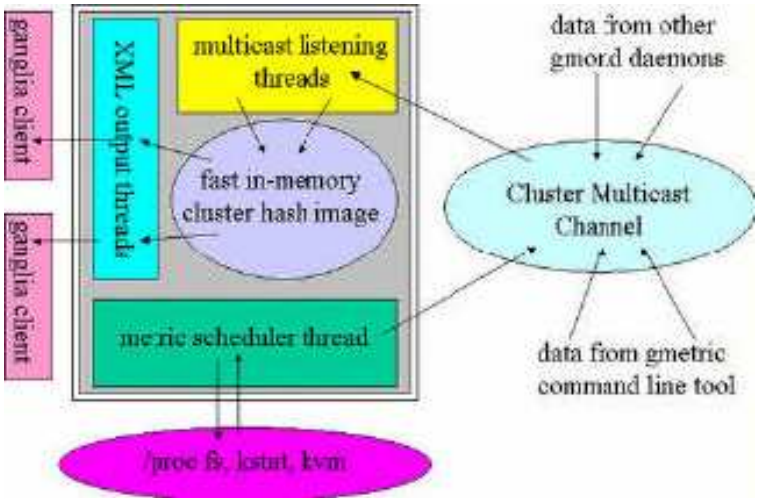


Figura 8.1 Base de Datos Distribuida

Ganglia Meta Daemon (**gmetad)**, Este demonio permite obtener la información vía XML en intervalos regulares desde los nodos, el gmetad toma la información y la salva en una base de datos Round-Robin y concatena los XML de los nodos para compartir la información con el servidor Web u otro Frontend que corra el demonio gmetad

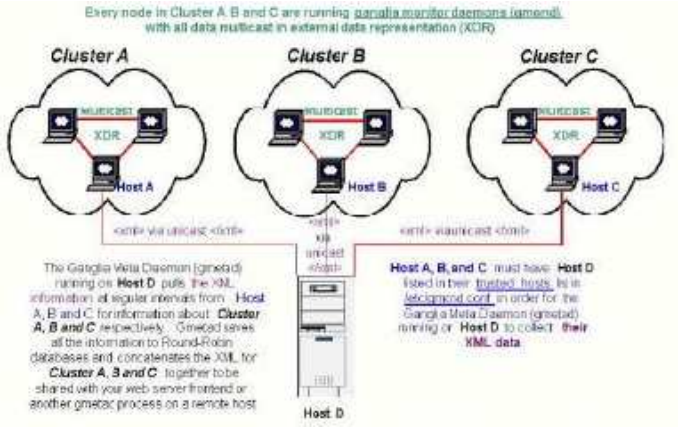


Figura 8.2 Ganglia Información de Clusters

Ganglia Web Frontend*:* Provee una vista de información de una pagina Web dinámica en tiempo real a los administradores y usuarios de sistemas, a través del Web Frontend Ganglia arranca como una html pero viene de un árbol xml, este fue echo bajo un sistema donde el historial se presenta de manera grafica.

Un ejemplo del Ganglia Web Frontend es que deja ver la información (CPU, Memoria, etc.) de los nodos del cluster en tiempo real y de forma grafica.

8.3 INSTALACIÓN

La maquina y la red debe estar habilitada de manera multicast para poder correr Ganglia.

El Gmond esta listo para ser usado sin ninguna configuración previa. Si las maquinas del Cluster están separadas por un router, entonces se necesitara configurar la opción mcast_ttl en el /etc/gmond.conf. Esto es para elevar el 1 que esta por omisión. Configurar el multi-cast TTL para ser un gran número de hops (router) entre los hosts. También hay que asegurarse de que los router estén configurados para pasar tráficos multicast.

Ganglia descargar de http://ganglia.sourceforge.net/downloads.php

* Unzip de la distribución

prompt> gunzip ganglia -monitor-core-2.5.7.

tar.gz | tar -xvf - ganglia-monitor-core -2.5.7/

ganglia-monitor-core -2.5.7/Makefile.in

ganglia-monitor-core -2.5.7/README

ganglia-monitor-core -2.5.7/stamp-h.in

Entonces ejecute las siguientes líneas de código:

ganglia-monitor-core -2.5.7/AUTHORS

ganglia-monitor-core -2.5.7/COPYING

ganglia-monitor-core -2.5.7/ChangeLog

ganglia-monitor-core -2.5.7/INSTALL

ganglia-monitor-core -2.5.7/Makefile.am

ganglia-monitor-core -2.5.7/NEWS

ganglia-monitor-core -2.5.7/acconfig.h

ganglia-monitor-core -2.5.7/acinclude.m4

ganglia-monitor-core -2.5.7/aclocal.m4

ganglia-monitor-core -2.5.7/config.h.in

Entrar al directorio de la distribución

prompt> cd ganglia-monitor-core-2.5.7

* Correr el script de configuración

NOTA: Se debe agregar **./configure –with-gmetad**, esto es por si desea compilar e instalar el ganglia Meta daemon (gmetad).

Para compilar el gmetad se debe terner instalado la librería y los header de la base de datos Round-Robin.

Esta librería se puede descargar de las siguientes direcciones:

http://www.rrdtool.com/download.html

ftp://ftp.pucpr.br/rrdtool/

Entonces ejecutamos las siguientes líneas:

```
prompt> ./configure
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal... found
checking for working autoconf... found
checking for working automake... found
checking for working autoheader... found
checking for working makeinfo... found
checking host system type... i686-pc -linux-gnu
checking for gcc... gcc

creating ./config.status
creating Makefile
creating lib/Makefile
creating gmond/Makefile
creating gmetric/Makefile
```

creating ganglia.spec

creating config.h

linking ./gmond/machines/linux.c to gmond/ma chine.c

Corriendo el Make.

prompt> make

make all-recursive

make[1]: Entering directory `/tmp/mas/ganglia-monitor-core-2.5.0'

Making all in lib

make[2]: Entering directory `/tmp/mas/ganglia-monitor-core-2.5.0/lib'

/bin/sh ../libtool --mode=compile gcc -DHAVE_CONFIG_H -I. -I. -I.. -I.

-O2 -D_REENTRANT -

Wall -Wshadow -Wpointer-arith -Wcast-align -Wstrict-prototypes –

D_GNU_SOURCE -c daemon_inetd.

mkdir .libs

gcc -DHAVE_CONFIG_H -I. -I. -I.. -I. -O2 -D_REENTRANT -Wall -

Wshadow -

Wpointer-arith -Wcast-a

```
lign -Wstrict-prototypes -D_GNU_SOURCE -c daemon_inetd.c -fPIC -
DPIC -o
.libs/daemon_inetd.l

gcc -DHAVE_CONFIG_H -I. -I. -I.. -I. -O2 -D_REENTRANT -Wall -
Wshadow -
Wpointer-arith -Wcast-a
lign -Wstrict-prototypes -D_GNU_SOURCE -c daemon_inetd.c -o
daemon_inetd.o >/dev/null 2>&1
mv -f .libs/daemon_inetd.lo daemon_inetd.lo
...
```

Corriendo el Make Install

```
prompt> make install
Making install in lib
make[1]: Entering directory `/tmp/mas/ganglia-monitor-core-2.5.0/lib'
make[2]: Entering directory `/tmp/mas/ganglia-monitor-core-2.5.0/lib'
/bin/sh ../config/mkinstalldirs /usr/lib
/bin/sh ../libtool --mode=install /usr/bin/install -c libganglia.la
/usr/lib/libganglia.la
```

Arranca el gmond y debe haber reiniciado la maquina. Activa el gmetad si es necesario.

Estas instrucciones son específicas para la utilización en el terminal de la consola Linux:

```
prompt> cp ./gmond/gmond.init /etc/rc.d/init.d/gmond
prompt> chkconfig --add gmond
prompt> chkconfig --list gmond
gmond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
prompt> /etc/rc.d/init.d/gmond start
Starting GANGLIA gmond: [ OK ]
```

En caso de que se instale en el web server es necesario activar el gmetad.

Si se tiene especificado una ruta diferente de almacenamiento RRDs por omisión (/var/lib/ganglia/rrds) entonces sustituya el directorio por uno nuevo.

```
prompt> mkdir -p /var/lib/ganglia/rrds
prompt> chown -R nobody /var/lib/ganglia/rrds
```

```
prompt> cp ./gmetad/gmetad.init /etc/rc.d/init.d/gmetad
prompt> chkconfig --add gmetad
prompt> chkconfig --list gmetad
gmond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
prompt> /etc/rc.d/init.d/gmetad start
Starting GANGLIA gmetad: [ OK ]
Probando la instalación Gmond.
```

Probando la instalación Gmond.

```
prompt> telnet localhost 8649
Probando la instalación Gmetad
prompt> telnet localhost 8651
```

8.4 INSTALACIÓN RPM (PARA LINUX)

La maquina y la red debe estar habilitada de manera multicast para poder correr Ganglia. El Gmond esta listo para ser usado sin ninguna configuración previa. Si las maquinas del Cluster están separadas por un router, entonces se necesitara configurar la opción mcast_ttl en el /etc/gmond.conf. Esto es para elevar el 1 que esta por omisión. Configurar el multi-cast TTL para ser un gran

número de hosts (router) entre los hosts. También hay que asegurarse de que los routers estén configurados para pasar tráficos multicast.

Descargar:

rpm: gmond

http://ganglia.sourceforge.net/downloads.php

ganglia-monitor-core-gmond-2.5.7-1.i386.rpm

rpm: gmetad

http://ganglia.sourceforge.net/downloads.php

ganglia-monitor-core-gmetad-2.5.7-1.i386.rpm

* Instalar el gmond RPM en todas la maquinas (Nodo Servidor y los demás Nodos Clientes)

prompt> rpm -ivh ganglia-monitor-core-gmond-2.5.7-1.i386.rpm

1:ganglia -monitor -core

##

[100%]

Starting GANGLIA gmond: [OK]

Luego es necesario levantar el gmond

* Instalar el gmetad RPM en el Frontend, puede estar en los nodos de calculo, pero generalmente el servicio del gmetad debe instalarse en el webserver(Nodo Servidor).

```
prompt> rpm -Uvh ganglia -monitor-core-gmetad-2.5.7-1.i386.rpm
Preparing...                ########################################### [100%]
  1:ganglia -monitor -core  ########################################### [100%]
Starting GANGLIA gmond: [  OK  ]
```

Este paquete debe instalarse en los archivos de arranque para levantar el gmetad de manera automática. Se necesitaría definir el data-source en el archivo de configuración /etc/gmetad.conf para decirle al gmetad por donde va a monitorear el gmond.

```
## /etc/gmetad.conf ##
```

The data_source tag must immediately be followed by a unique

string which identifies the source then a list of machines

which service the data source in the format ip:port, or name:port.

If a # port is not specified then 8649 (the default gmond port) is

assumed.

default: There is no default value

data_source "Nodoservidor" localhost

8.5 GANGLIA WEB FRONTEND

Instalación de Ganglia Web Frontend

Descargar desde la página **http://ganglia.sourceforge.net/downloads.php**

ganglia-webfrontend-2.5.7.tar.gz

Descomprimir la distribución del webfrontend en la raíz del sitio web. Se recomienda que el sitio sea **/var/www/html**, es importante que se observe el DocumentRoot en la configuración de Apache. Todos los Archivos script PHP usan relativamente enlaces URLs, también se puedes construir los archivos del webfrontend donde se desee.

```
prompt> cd /var/www/html
prompt> tar xvzf gmetad-webfrontend-2.5.7.tar.gz
```

Hay que asegurarse que el webserver entienda bien los scripts PHP, ya que ganglia en el webfrontend trabaja con solo archivos PHP. Para Apache hay que habilitarle el modulo mod_php, estas líneas aparecen en cualquier lugar del archivo httpd.conf de apache tal como:

```
<IfDefine HAVE_PHP4>
LoadModule php4_module    extramodules/libphp4.so
AddModule mod_php4.c
</IfDefine>
 AddType  application/x-httpd-php        .php .php4 .php 3 .phtml
AddType  application/x-httpd-php-source  .phps
```

El webfrontend requiere de la existencia del gmead, y de los directorios rrdtool y del rrds.

Para hacer un test del funcionamiento del webfrontend se despliega un browser y se introduce:

http://localhost/gmetad-webfrontend/

http://192.168.100.1/gmetad-webfrontend/

8.6 INSTALACIÓN USANDO RPM

Se descarga el RPM de la página de http://ganglia.sourceforge.net

Instalación del gmetad-webfrontend RPM en la maquina webserver

```
prompt> rpm -Uvh gmetad-webfrontend-2.5.0-1.i386.rpm
Preparing...                ########################################### [100%]
   1:gmetad-webfrontend     ########################################### [100%]
```

Prueba la instalación
http://localhost/gmetad-webfrontend/

8.6.1 Configuración de los archivos gmetad.conf y gmond.conf

El comportamiento de Ganglia Meta Daemon (gmetad) es completamente controlado por un archivo de configuración que esta en /etc/gmetad.conf. El gmetad para hacer cualquier cosa verifica las líneas data_source en la configuración. Ejemplo.

```
data_source "Cluster A" 127.0.0.1  1.2.3.4:8655  1.2.3.5:8625
```

data_source "Cluster B" 1.2.4.4:8655

Como en el ejemplo, estos dos único data_source : Cluster A y cluster B.

Los datos de recurso del cluster A tienen recursos redundantes. Si el gmetad no puede colocar los datos desde el primer recurso entonces este trata de continuar con el otro ordenadamente.

Nota: Si no se especifica nada en el data_source el gmetad asume de que todos se conectan por el puerto de omisión que es 8649.

8.6.2 Ejemplo de la configuración de gmetad.conf

```
# This is an example of a Ganglia Meta Daemon configuration file
#                http://ganglia.sourceforge.net/
#  $Id: gmetad.conf,v 1.3 2002/09/19 18:56:42 sacerdoti Exp $
#  Setting the debug_level above zero will make gmetad output
# debugging information and stay in the foreground
# default: 0
# debug_level 10
#  The data_source tag must immediately be followed by a unique
```

```
# string which identifies the source then a list of machines
# which service the data source in the format ip:port, or name:port.
# If a # port is not specified then 8649 (the default gmond port) is
# assumed.
# default: There is no default value
# data_source "my box" localhost  my.machine.edu:8655  1.2.3.5:8655
# data_source "another source"  1.3.4.7:8655  1.3.4.8
#  List of machines this gmetad will share XML with
# default: There is no default value
# trusted_hosts 127.0.0.1 169.229.50.165
#  If you don't want gmetad to setuid then set this to off
# default: on
# setuid off
#  User gmetad will setuid to (defaults to "nobody")
# default: "nobody"
# setuid_username "nobody"
#  The port gmetad will answer requests for XML
# default: 8651
 # xml_port 8651
# The number of threads answering XML requests
```

```
# default: 2
# server_threads 4
# Where gmetad stores its round-robin databases
# default: "/var/lib/ganglia/rrds"
# rrd_rootdir "/some/other/place"
```

El Ganglia Monitoring Daemon (gmond), el gmond es un demonio muy flexible el cual su configuración por lo general se deja por omisión. Un ejemplo de este:

```
# $Id: gmond.conf,v 1.2 2002/09/19 00:37:18 sacerdoti Exp $
# This is the configuration file for the Ganglia Monitor Daemon (gmond)
# Documentation can be found at http://ganglia.sourceforge.net/docs/
#  To change a value from it's default simply uncomment the line
# and alter the value
###################### #
# The name of the cluster this node is a part of
# default: "unspecified"
# name  "My Cluster"
#  The owner of this cluster. Represents an administrative
# domain. The pair name/owner should be unique for all clusters
```

```
# in the world.
# default: "unspecified"
# owner "My Organization"
#  The latitude and longitude GPS coordinates of this cluster on earth.
# Specified to 1 mile accuracy with two decimal places per axis in Decimal
# DMS format: "N61.18 W130.50".
# default: "unspecified"
# latlong "N32.87 W117.22"
#  The URL for more information on the Cluster. Intended to give purpose,
# owner, administration, and account details for this cluster.
# default: "unspecified"
# url "http://www.mycluster.edu/"
#  The location of this host in the cluster. Given as a 3D coordinate:
# "Rack,Rank,Plane" that corresponds to a Euclidean coordinate "x,y,z".
# default: "unspecified"
# location "0,0,0"
#  The multicast channel for  gmond to send/receive data on
# default: 239.2.11.71
# mcast_channel 239.2.11.71
#  The multicast port for gmond to send/receive data on
```

```
# default: 8649
# mcast_port    8649
#  The multicast interface for gmond to send/receive data on
# default: the kernel decides based on routing configuration
# mcast_if  eth1
#  The multicast Time -To-Live (TTL) for outgoing messages
# default: 1
# mcast_ttl  1
#  The number of threads listening to multicast traffic
# default: 2
# mcast_threads 2
#  Which port should  gmond listen for XML requests on
# default: 8649
# xml_port     8649
#  The number of threads answering XML requests
# default: 2
# xml_threads   2
#  Hosts ASIDE from "127.0.0.1"/localhost and those multicasting
# on the same multicast channel which you will share your XML
# data with.  Multiple hosts are allowed on multiple lines.
```

```
# Can be specified with either hostnames or IP addresses.
# default: none
# trusted_hosts 1.1.1.1 1.1.1.2 1.1.1.3  \
# 2.3.2.3 3.4.3.4 5.6.5.6
#  The number of nodes in your cluster.  This value is used in the
# creation of the cluster hash.
# default: 1024
# num_nodes  1024
#  The number of custom metrics this gmond will be storing.  This
# value is used in the creation of the host custom_metrics hash.
# default: 16
# num_custom_metrics 16
#  Run gmond in "mute" mode.  Gmond will only listen to the multicast
# channel but will not send any data on the channel.
# default: off
# mute on
#  Run gmond in "deaf" mode.  Gmond will only send data on the multicast
# channel but will not listen/store any data from the channel.
# default: off
# deaf on
```

```
# Run gmond in "debug" mode.  Gmond will not background.  Debug messages
# are sent to stdout.  Value from 0-100.  The higher the number the more
# detailed debugging information will be sent.
# default: 0
# debug_level 10
# If you don't want gmond to setuid, set this to "on"
# default: off
# no_setuid  on
# Which user should gmond run as?
# default: nobody
# setuid     nobody
# If you do not want this host to appear in the gexec host list, set
# this value to "on"
# default: off
# no_gexec   on
# If you want any host which connects to the gmond XML to receive
# data, then set this value to "on"
# default: off
# all_trusted on
```

Si se quiere configurar el gmond es posible poder entrar al /etc/gmond.conf Para poder crear multiples configuraciones de este.

8.7 GANGLIA WEB FRONTEND

Los parámetros de configuración se hacen en el gmetad -webfrontend/conf.php, dentro de este archivo se puede configurar la ubicacion del gmetad.conf, la ubicación de RRDtool y la configuración de los tiempos, rangos y métricas de las graficas.

8.8 HERRAMIENTAS DE LÍNEAS DE COMANDO.

- Ganglia Metric Tool (gmetric): Permite un facil monitoreo en cualquier host, tal como expandir los Metric Core que se encuentra por omisión en el gmond. Si se quiere una ayuda de la sintaxis del gmetric, escriba la opción:

prompt> gmetric --help

gmetric 2.5.0

Purpose:

The Ganglia Metric Client (gmetric) announces a metric

value to all Ganglia Monitoring Daemons (g monds) that are listening

on the cluster multicast channel.

Usage: ganglia-monitor-core [OPTIONS]...

-h --help Print help and exit

-V --version Print version and exit

-nSTRING --name=STRING Name of the metric

-vSTRING --value=STRING Value of the metric

-tSTRING --type=STRING Either string|int8|uint8|int16|uint16|int32|uint32|float|double

-uSTRING --units=STRING Unit of measure for the value e.g. Kilobytes, Celcius

-sSTRING --slope=STRING Either zero|positive|negative|both (default='both')

-xINT --tmax=INT The maximum time in seconds between gmetric calls (default=60)

-cSTRING --mcast_channel=STRING Multicast channel to send/receive on (default='239.2.11.71')

-pINT --mcast_port=INT Multicast port to send/receive on (default=8649)

-iSTRING --mcast_if=STRING Network interface to multicast on e.g. 'eth1' (default='kernel decides')

-lINT --mcast_ttl=INT Multicast Time -To-Live (TTL)
(default=1)

El gmetric tool formatea mensajes multicast y envía estos a todos los demonios gmond que estan escuchando.

Todas las metricas en ganglia tiene un nombre, valor, tipo y unidades opcionales.

8.9 GANGLIA CLUSTER STATUS TOOL (GSTAT)

Es una utilidad de líneas de comando que permite obtener reporte del Status de cluster.

Si se quiere las opciones de la línea de comando, simplemente corra:
prompt> gstat --help
gstat 2.5.0

El gstat s conecta con un gmond y da una lista de salida load-balance de los host del cluster.

```
Usage: gstat [OPTIONS]...

 -h        --help            Print help and exit

 -V        --version         Print version and exit

 -a        --all             List all hosts.  Not just hosts running

gexec (default=off)

 -d        --dead            Print only the hosts which are dead

(default=off)

 -m        --mpifile         Print a load-balanced mpifile

(default=off)

 -1        --single_line     Print host and information all on one

line (default=off)

 -l        --list            Print ONLY the host list (default=off)

 -iSTRING   --gmond_ip=STRING  Specify the ip address of the gmond to

query (default='127.0.0.1')

 -pINT     --gmond_port=INT   Specify the gmond port to query

(default=8649)
```

8. 10 ESQUEMA DE GANGLIA NODO SERVIDOR

Ejecutando Ganglia Web Frontend desde el nodo Servidor, en el cual da las características del CPU.

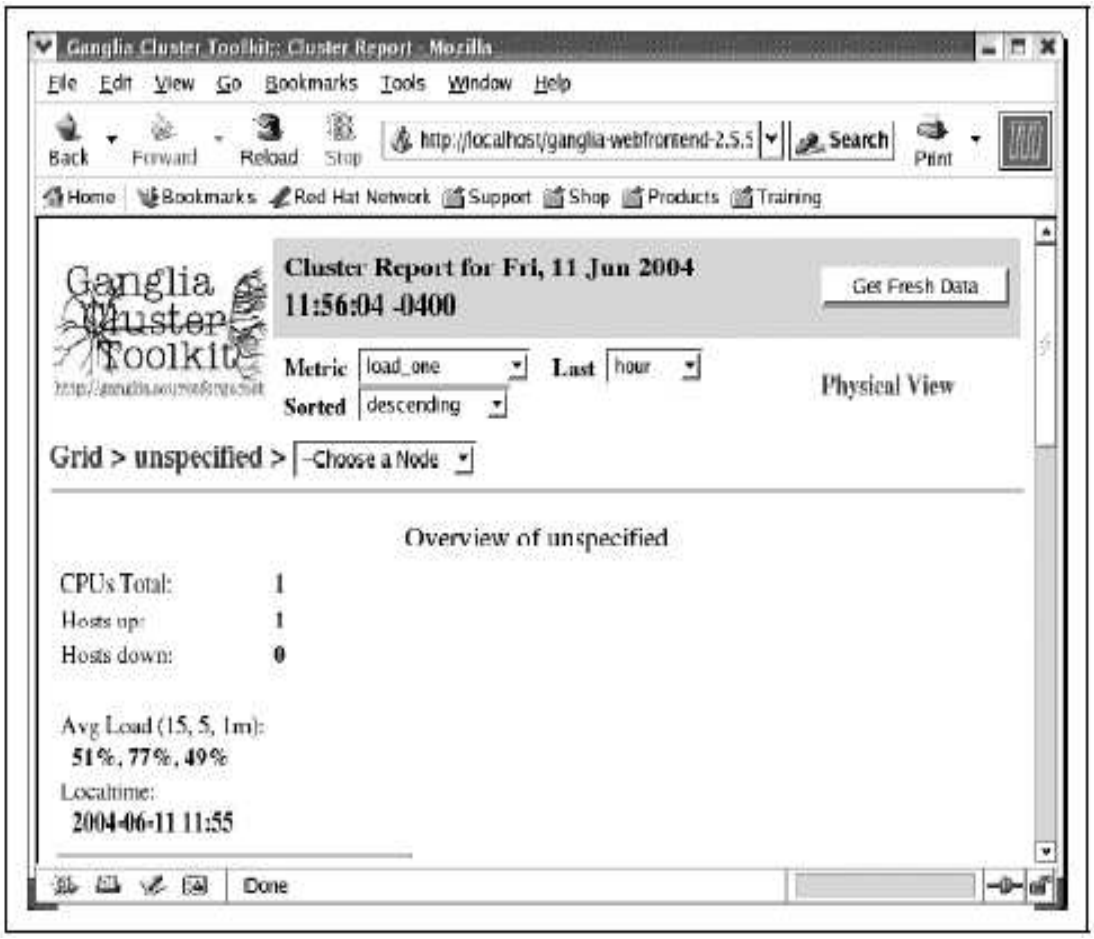


Figura 8.3 Ganglia Servidor

Ejecutando Ganglia Web Frontend desde el nodo Cliente, en el cual da las características de los CPU de los Nodos.

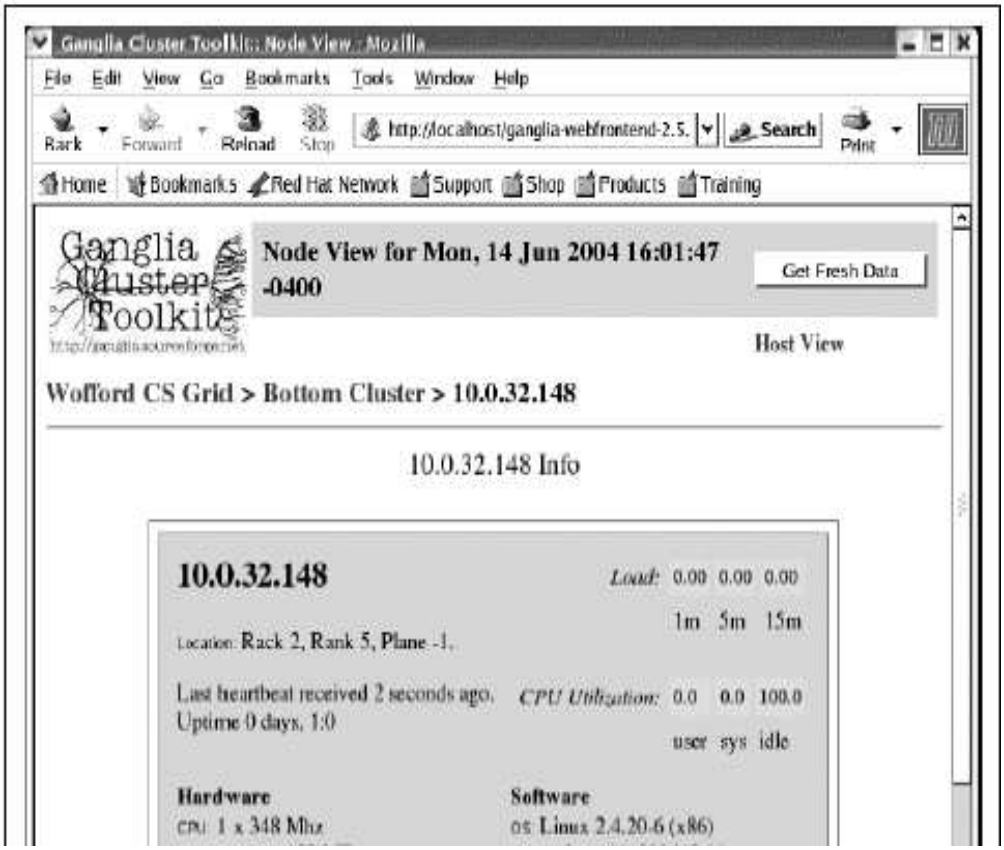


Figura 8.4 Ganglia Nodos Clientes

8.11 CLUSTER WEB SERVER

Un Webcluster es una colección de máquinas que sirven páginas WEB. Estas máquinas se unen dentro de una pequeña "granja" o familia de servidores.

Apache posee un módulo, libproxy.so, que posee la misma funcionabilidad que un proxy. Un Proxy es una máquina instalada en una red cuya función es conectarse a otra red, obtener información y entregarla a los clientes conectados al Proxy. Un proxy no es un Gateway ni un Router, aunque generalmente se le confunde. Un Router o un Gateway desvían paquetes TCP/IP de acuerdo a una serie de reglas, llamadas generalmente reglas de enrutamiento. Estas reglas funcionan en el nivel más básico de TCP/IP. Un proxy es un como un "router de alto nivel". No desvía paquetes y funciona como un intermediario de contenidos entre un cliente y una red.

```
[cliente1]---+
[cliente2]---+---[ PROXY ]----[INTERNET]
[cliente3]---+

        Cliente-(+)->PROXY
                        PROXY --GET(*)--->WWW SERVER
                        PROXY <-----(*)---WWW SERVER
        Cliente<--(*)PROXY
```

Un servidor Web de una intranet, la gran mayoría de las veces esta conectado a Internet. Por ejemplo, Oscar posee una intranet y un servidor Web que sirve al dominio **www.oscar.com.**

Esto hace que las páginas FÍSICAMENTE no residan en el servidor, sino en un computador de la Intranet. Este tipo de conexión es totalmente transparente para el cliente.

Aquí es donde entra otro modulo de Apache. Se llama mod_rewrite. Su tarea principal es reescribir las peticiones de acuerdo a cierto parámetro de búsqueda.

Reescribiendo URL (URL Rewrite)

Hay dos maneras de reescribir un URL:

Client Rewriting: esta es la forma menos transparente de todas. Si un cliente HTTP se conecta, por ejemplo, a www.oscar.com, el servidor puede responder de varias maneras para reescribir el URL:

- usando Javascript y document.location
- usando CGIs
- HTML y contenidos META
- reescribiendo las cabeceras usando PHP y la función header()

Suponiendo el mismo ejemplo:

1. Cliente se conecta a http://www.oscar.com

2. El servidor responde con un index.php

3. El contenido de index.php es

4. El cliente es redirigido a http://oscar.com/moodle/index.html

Server Rewriting: Es la forma más transparente. En vez de reescribir el URL en el cliente, el URL es transformado DENTRO del servidor. El servidor reinterpreta la conexión, hace la conexión donde FÍSICAMENTE residen las páginas y entrega el resultado al cliente.

Suponiendo el mismo ejemplo de www.oscar.com:

1. El cliente se conecta a http://www.oscar.com

2. El servidor detecta que debe haber una reescritura de URL

3. Como el cliente no pide ninguna pagina, asume que es la pagina índice configurada en Apache o en su VirtualHost En este caso, index.html

4. El servidor reinterpreta el URL : http://www.oscar.com/

5. El servidor se conecta a http://oscar.com/moodle//index.html

Preparación

La distribución elegida fue RedHat 9.0, aunque es 100% factible .

TECNICAS PARA CONFIGURAR EL WEB CLUSTER

Para probar el correcto funcionamiento del Web Cluster fue necesario configurar los siguientes servicios:

1. DNS Round Robin

2.Squid acelerar la navegación web

DNS Round Robin

DNS RR es una forma de hacer "rotar" en una "ronda" de números IP que corresponden a un mismo servidor. Para ello se configura el archivo de zona de la siguiente manera:

```
@       IN      SOA  ....
..
; zona del cluster
cluster0        IN      A       192.168.1.150
cluster1        IN      A       192.168.1.151
cluster2        IN      A       192.168.1.152
; el cluster
cluster         60      IN      A       192.168.1.150
                60      IN      A       192.168.1.151
                60      IN      A       192.168.1.152
; eof
```

Y configurar Apache a todos los servidores (cluster0-2) de la siguiente manera:

/etc/httpd/conf/httpd.conf

NameVirtualHost <ip>
<VirtualHost <nombre_cluster> >

```
        DocumentRoot <ruta1>
        CustomLog <ruta_log> vcommon
</VirtualHost>

<VirtualHost cluster.dominio>
        DocumentRoot <ruta2>
        CustomLog <ruta_log> vcommon
</VirtualHost>
# eof
```

Reiniciar Apache y Named en todas las máquinas

Al hacer entonces

```
# nslookup www.oscar.com
Name:www.oscar.com
Address:  192.168.1.150
Name:cluster.dominio
Address:  192.168.1.151
Name:cluster.dominio
Address:  192.168.1.152
```

Al escribir entonces en el navegador de algún cliente Web de la intranet:

http://www.oscar.com

Respondería cualquiera de los tres, dependiendo cual este primero en el DNS, sin reescribir el URL en el cliente.

El sistema del DNS Round Robin no es una forma de compartir carga entre servidores, sino una forma de destinar iguales conexiónes entre varias máquinas. No hay un balance de carga ni tampoco comparten carga entre ellas. Es el sistema más básico y efectivo.

Squid acelerar la navegación web

Con la finalidad de probar el tiempo de respuesta que el Web Cluster los usuarios que acceden aceleran la navegación.

Para lo cual se necesita la configuración del archivo /etc/squid/ vi squid.conf

Y dar permisos para la utilización del Proxy a los usuarios que utilicen el Internet.

CONCLUSIONES

- Con la instalación de un Cluster Beowulf de alto rendimiento se puede minimizar el tiempo de respuesta a un programa o tarea que se ejecute como por ejemplo un Servidor Web, pues la carga y procesos se dividen a todos los nodos, mientras más nodos tengamos menor será el tiempo de respuesta.

- El WebCluster permite que usuarios que acceden a las páginas o sitios que se encuentran en el Servidor Web del nodo servidor tengan un tiempo de respuesta menor ya sea en páginas estáticas o dinámicas, esto es una ventaja respecto de los Proxy Cache o Proxy Reversibles ya que estos solo permiten disminuir el tiempo de respuesta que el usuario tiene de páginas estáticas.

- El uso de software libre como es Linux permite que el costo de elaboración de un Cluster sea bajo y su funcionalidad sea más confiable.

- Con la utilización de la herramienta de monitoreo de un Cluster como es el Ganglia podemos visualizar de manera muy amigable el

funcionamiento del Cluster y su monitoreo es casi en tiempo real se ve el trabajo que los nodos van realizando, como se agrupan los recursos que posee en Cluster.

- Mediante la utilización del C3 que es una herramienta de administración para Cluster, nos ayuda a reiniciar, apagar, matar procesos directamente desde el Nodo Servidor.

RECOMENDACIONES

- Los Usuarios deben tener conocimientos profundos sobre Linux en las distribuciones de Rethat 8.0 y 9.0, Mandrake 10.0 para poder realizar una correcta instalación, administración, mantenimiento del cluister.

- Se debe utilizar la distribución Rethat 9.0 para un correcto funcionamiento del Oscar Cluster versión 3.0, ya que se acoplan de mejor manera.

- Cumplir con los requerimientos mínimos que mencionamos anteriormente para la instalación del Oscar.

- Seleccionar el idioma Ingles de Rethat 9.0 para correr el wizard de instalación del Oscar cluster, ya que los paquetes rpm tienen soporte para el idioma Inglés.

- Ejecutar paso a paso el wizard de instalación del Oscar con precaución de los mensajes que se van presentando durante la instalación.

- Utilizar como medio de comunicación entre los nodos un Switch ya que este evita colisiones entre los nodos

GLOSARIO DE TERMINOS

Beowulf: Es una clase de computador masivamente paralelo de altas prestaciones principalmente construido a base de un cluster de componentes hardware estándard.

Cluster: Conjunto de ordenadores conectados a través de la red

C3: Cluster Command Control es una herramienta que permite administrar y ejecutar comandos en todos los nodos del cluster. Por ejemplo un comando como cshutdown permite detener y reiniciar todos los nodos del cluster.

Escalabilidad vertical: Cuando tratamos con un gran sistema, con múltiples CPUs, con una gran velocidad de interconexión entre nodos, tradicionalmente empleando elementostales como backplanes. La mayoría de las aplicaciones escalan muy bien verticalmente hasta cierto número de CPUs. Actualmente, debido a su alto precio, solo se emplea en aplicaciones muy transaccionales y/o minería de datos. El problema de estas máquinas es que suponen una gran inversión que pocas organizaciones pueden afrontar, además de los problemas de escalabilidad que provocan.

Escalabilidad horizontal: Consiste en emplear muchas máquinas pequeñas e inteconectadas para realizar una tarea determinada. Con un coste proporcionalmente bajo por máquina, es una solución muy barata para algunos tipos de aplicaciones, como servicios web, correo o supercomputación. Su principal ventaja es que el crecimiento a un mayor número de nodos se puede hacer cuando sea necesario, sin tener que realizar un gran desembolso inicialmente como ocurre en un sistema vertical. La principales desventajas de un sistema horizontal son que escalan mal para aplicaciones transaccionales (bases de datos, por ejemplo), y que la administración de muchas máquinas interconectadas implica una mayor atención y conocimiento que la administración de una gran máquina con muchos procesadores.

Dynamic Host Configuration Protocol (DHCP*):* El protocolo de configuración dinámica de host permite:

Administración más sencilla.

Configuración automatizada.

Permite cambios y traslados.

Posibilidad de que el cliente solicite ciertos parámetros. El software necesario para el arranque se puede obtener a través de tftp

Ganglia*:* Es un programa que permite monitorear en tiempo real todo el cluster. De esta forma es posible verificar el estado de carga de los nodos. Los datos pueden ser accesados vía web.

Licencia GPL: Los programas de ordenador suelen distribuirse con licencias propietarias o cerradas. Estas licencias son intransferibles y no exclusivas, es decir, no eres propietario del programa, sólo tienes derecho a usarlo en un ordenador o tantos como permita expresamente la licencia y no puedes modificar el programa ni distribuirlo.

LAM-MPI*:* Es una de las implementaciones de la norma MPI que se caracteriza por se más amigable. Permite el paso de mensajes sobre TCP/IP o en memoria compartida.

MPI (Message Passing Interface) es una biblioteca de funciones y macros que permite explotar la existencia de múltiples procesadores mediante el paso de mensajes.

MPICH: Es otra de las implementaciones de MPI. Es importante notar que Oscar instala tanto LAM-MPI como MPICH, la única consideración es elegir una por defecto, lo cual no impide ocupar la otra versión de MPI.

Maui*:* Permite programar, administrar y priorizar con sofisticados algoritmos los trabajos a través del cluster. Funciona como parte de OpenPBS, aun cuando originalmente no forma parte de él(Es agregado por el equipo de desarrollo de Oscar para mejorar las prestaciones de OpenPBS).

Mosix: Es una herramienta desarrollada para sistemas tipo UNIX, cuya característica resaltante es el uso de algoritmos compartidos, los cuales están diseñados para responder al instante a las variaciones en los recursos disponibles, realizando el balanceo efectivo de la carga en el cluster mediante la migración automática de procesos o programas de un nodo a otro en forma sencilla y transparente.

Network File System (NFS): (Network File System) Es una arquitectura de Servidores de archivos portable a través de diferentes hardware, sistemas operativos, protocolos de transporte, y tecnologías de red. En este caso NFS permite montar el directorio /home del pc maestro en los nodos.

OPIUM: Este permite sincronizar cuentas y configurar ssh a través de todo el cluster. Copia los archivos de cuentas a todos los nodos y los sincroniza cada cierto tiempo.

OpenPBS: Es un administrador de trabajos que permite administrar colas y ejecución de trabajos.

OpenSSH: Es un sustituto total del venerable y obsoleto telnet. Permite establecer sesiones remotas, las cuales pueden ser encriptadas, esto significa un nivel de seguridad muy superior a telnet. La versión 2 del protocolo ssh permite además el traslado de archivos al igual que ftp.

Pfilter: Es un filtro de paquetes al igual que ipchains e iptables(pfilter es incompatible con ipchains e iptables). Permite un manejo sofisticado de un cortafuegos para mejorar la seguridad.

PVM: Esta biblioteca de paso de mensajes viene disponible con Oscar y se puede ocupar sin problemas para paralelizar.

PXE: Permite a los nodos instalar todo el sistema operativo utilizando la red.

SIS*:* System Instalation Suite es una herramienta que permite la instalación de Linux a través de una red. Se usa para instalar los nodos clientes.

Switcher: La instalación de los programas parte de Oscar necesitan modificar varios de los script de configuración propios de las cuentas (.bashrc, .login, .logout). Switcher facilita esta tarea, evitando la manipulación manual por parte del usuario de estos script. Es necesario tomar en cuenta que configurar mal alguno de estos archivos puede inutilizar una cuenta.

UDP: es un protocolo sencillo que implementa un nivel de transporte orientado a datagramas:

NO orientado a conexión, NO fiable.

Los datagramas UDP se encapsulan dentro de la parte de datos de un datagrama IP.

Una aplicación que utilice UDP para transmitir datos, producirá exactamente un datagrama UDP por cada operación de salida que precise, el cual originará un datagrama IP encapsulándolo.

BIBLIOGRAFIA

REFERENCIAS BIBLIOGRAFICAS

- STERLING L Thomas, SALMON John, BECKER J Donald, SAVARESE F Daniel. "How to Buid a Beowulf A Guide to the Implemetation and Aplication of PC Clusters". Primera Edición, London, England, The MIT Press, 1999.
- VRENIOS Alex. "Linux Cluster Architecture".Primera Edición, United States of America, Sams Publishing, 2002.
- PETERSEN Richard. "Manual de Referencia LINUX". Segunda Edición, España, McGraw Hill, 2001.
- GROPP William, LUSK Ewing, STERLING . "Beowulf Cluster Computing with Linux". Segunda Edición, London, England, The MIT Press, 2003.

INTERNET

CLUSTER

http://linux.ubiobio.cl/documentacion/cluster_ufro/proyecto.html

http://bulma.net/body.phtml?nIdNoticia=1708

http://clusters.unam.mx/

http://www.linuxdevcenter.com/pub/a/linux/2004/12/29/lnxclstrs_10.html

http://es.tldp.org/Manuales-LuCAS/doc-cluster-computadoras/doc-cluster-computadoras-html/node8.html

http://www.fisica.uson.mx/carlos/LinuxClusters/

http://www.redes-linux.com/manuales.php?catId=Cluster

CLUSTER ALTO RENDIMIENTO

http://www.google.com.ec/search?q=cache:iMIF7GV4jQcJ:cq-ing.tij.uabc.mx/docentes/cristobal/clases/paralelo/10_paradigmas.pdf+lam+mpi+ejemplos&hl=es

http://www.google.com.ec/search?q=cache:8MkfXEgplUwJ:www.ii.uam.es/~fjgomez/CursoVerano/instal/cluster.pdf+lam+mpi+ejemplos&hl=es

http://es.wikipedia.org/wiki/Cluster_de_alto_rendimiento

http://www.fisica.uson.mx/carlos/LinuxClusters/ClusterACARUS.htm

http://supercomputo.izt.uam.mx/inicio/masinformacionclustes.htm

CLUSTER ALTA DISPONIBILIDAD

http://es.tldp.org/Manuales-LuCAS/doc-instalacion-cluster-alta-disponibilidad/instalacion-cluster-alta-disponibilidad/

http://www.linuxfocus.org/Castellano/November2000/article179.shtml

http://www.fisica.uson.mx/carlos/LinuxClusters/clusters_de_Linux.htm

http://www.bisente.com/documentos/clustering/memoria.html

BEOWULF

http://www.cecalc.ula.ve/HPCLC/slides/day_02/Cluster_School.pdf

http://www.beowulf.org/archive/2004-November/011158.html

http://www.hispacluster.org/modules.php?file=article&name=News&op=mo
dload&sid=546

http://studies.ac.upc.es/EPSC/TFC/Beowulf/beowulf.html

http://www.dc.uba.ar/people/proyinv/fra/cluster_speedy/software.html

http://www.fysik.dtu.dk/CAMP/cluster-howto.html

http://www.fisica.uson.mx/carlos/Linux/Docs/HOWTO/Beowulf-
HOWTO.html

http://clusters.top500.org/

OSCAR TOOL KIT

http://www.openclustergroup.org/

http://sourceforge.net/project/showfiles.php?group_id=9368

http://oscar.openclustergroup.org/tiki-list_file_gallery.php?galleryId=2

http://oscar.openclustergroup.org/tiki-index.php

http://nubes.uspnet.usp.br/lcca/lcca2/clusters/transparencias/Cluster_Management.pdf

http://www.hispacluster.org/modules.php?file=article&name=News&op=modload&sid=531

http://www.csm.ornl.gov/DOE/mics2003/oscar.doc

GANGLIA:

http://ganglia.sourceforge.net/

http://ganglia.sourceforge.net/downloads.php

http://www.cecalc.ula.ve/HPCLC/slides/day_06/Monitoring/Exercises_Monitoring/Slides_Ganglia_ToolKit.pdf

http://grid.ifca.unican.es/cursos/presentaciones/Monitoring_curso_postgrado.pdf

http://www.cecalc.ula.ve/HPCLC/slides/day_06/Monitoring/Exercises_Monitoring/Slides_Ganglia_ToolKit.pdf

RRDTOOL

http://www.si.uji.es/bin/ponencies/irrdtool.pdf

http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/manual/rrdtutorial.es

http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/download.html

PVFS

http://www.cecalc.ula.ve/HPCLC/informes/Laboratotio_B/ASIA_B/reporte6-asiaB_04_11.pdf

C3

http://www.csm.ornl.gov/torc/C3/

http://www.csm.ornl.gov/torc/C3/C3documentation.shtml

http://www.csm.ornl.gov/torc/C3/C3softwarepage.shtml

LAM/MPI

http://www.lam-mpi.org/

http://www.urjc.es/cat/hidra/manuales/basicos/Desarrollo_basico.pdf

http://www-copa.dsic.upv.es/docencia/iblanque/lcp/transparencias/tema2(1)_04_6p.pdf

http://www.lam-mpi.org/using/docs/

http://www.generacio.com/cluster/cluster1.html

http://www.inf.utfsm.cl/~lnunez/documentos/parte1.pdf

http://cq-ing.tij.uabc.mx/docentes/cristobal/clases/paralelo/10_paradigmas.pdf

Maui Cluster Scheduler

http://www.clusterresources.com/products/maui/

MPICH

http://www-unix.mcs.anl.gov/mpi/mpich/

http://www.cesca.es/supercomputacio/usrecursos/e_desenvolupament/mpich_beowulf.html

http://www.super.unam.mx/ayuda/guias/guia_cluster/node21.html

OpenSSH

http://www.openssh.com/es/

http://www.openssh.com/es/manual.html

SSI

http://clusters.unam.mx/Proyectos/white_paper/cap4.html

OpenSSL

http://www.openssl.org/

http://www.google.com.ec/search?q=cache:-5VOalXMeKQJ:www.urjc.es/cat/hidra/manuales/basicos/Manual_Acceso_Hidra.pdf+OpenSSH+CLUSTER&hl=es&lr=lang_es

PVM

http://www.csm.ornl.gov/pvm/

http://www.orcero.org/irbis/disertacion/node289.html

http://www.orcero.org/irbis/disertacion/node265.html

http://www.ii.uam.es/~fjgomez/intropvm.html

http://www.ii.uam.es/~fjgomez/tutorial.html

http://iio.ens.uabc.mx/~jmilanez/escolar/sistemas_operativos/expo-9.html

System Installation Suite

http://www.sisuite.org/

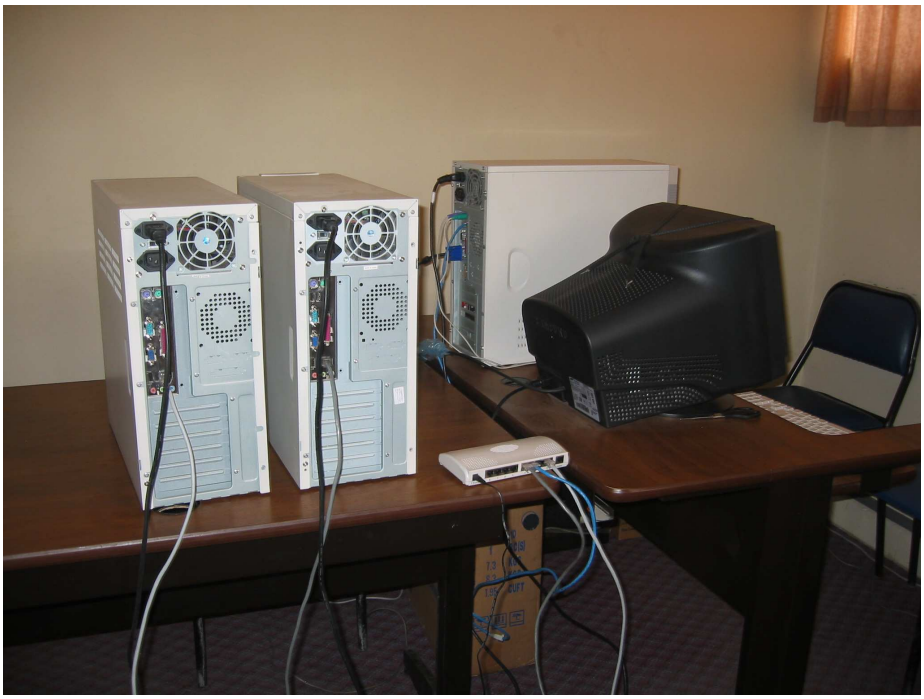
http://sourceforge.net/project/showfiles.php?group_id=28855

Torque

http://www.clusterresources.com/products/torque/ANEXOS

ANEXOS

IMAGENES DE CLUSTER BEOWULF CON SOFTWARE OSCAR TOOLKIT



ELEMENTOS QUE CONFORMAN EL CLUSTER BEOWULF INSTALADO

- Hub
- Cables Utp categoría 5
- Nodo Servidor
- Nodos Clientes
- Monitor en el Nodo Servidor
- Teclado y Mouse en el Nodo Servidor

CARACTERISTICAS DEL NODO SERVIDOR

- Disco Duro de 40 Gb
- Memoria RAM 256
- Tarjeta Madre Pc Chip P4
- Tarjeta de Red 10/100
- Floppy
- Cdrom
- Teclado, Mouse, Monitor

CARACTERISTICAS DE LOS NODOS CLIENTES

NODO CLIENTE 1

- Disco Duro de 60 Gb
- Memoria RAM 256
- Tarjeta Madre INTEL P4
- Tarjeta de Red 10/100
- Floppy
- Cdrom

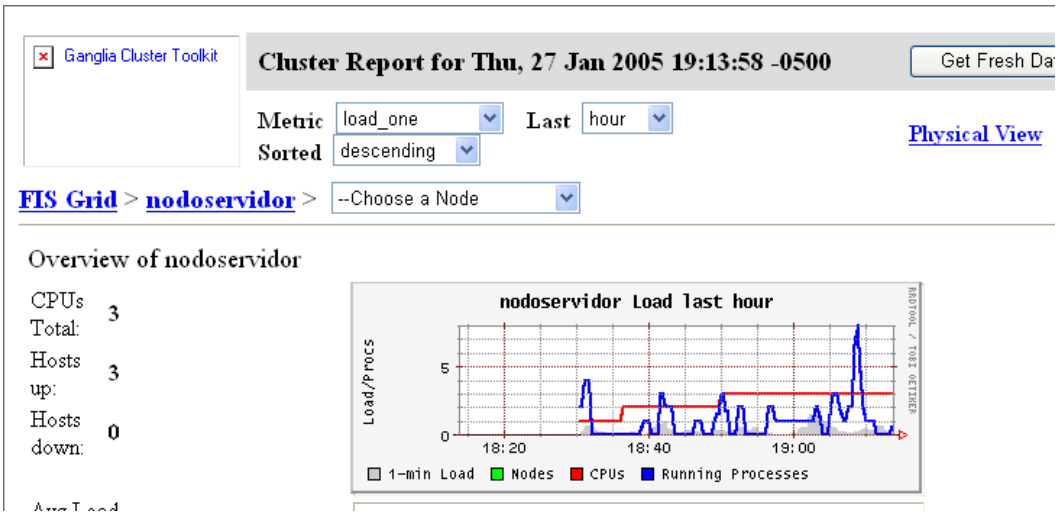
NODO CLIENTE 2

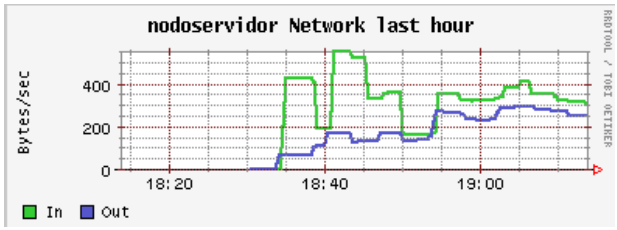
- Disco Duro de 80 Gb

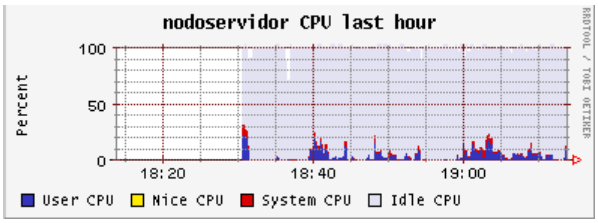
- Memoria RAM 256
- Tarjeta Madre INTEL P4
- Tarjeta de Red 10/100
- Floppy
- Cdrom

IMÁGENES DEL FUNCIONAMIENTO DEL GANGLIA

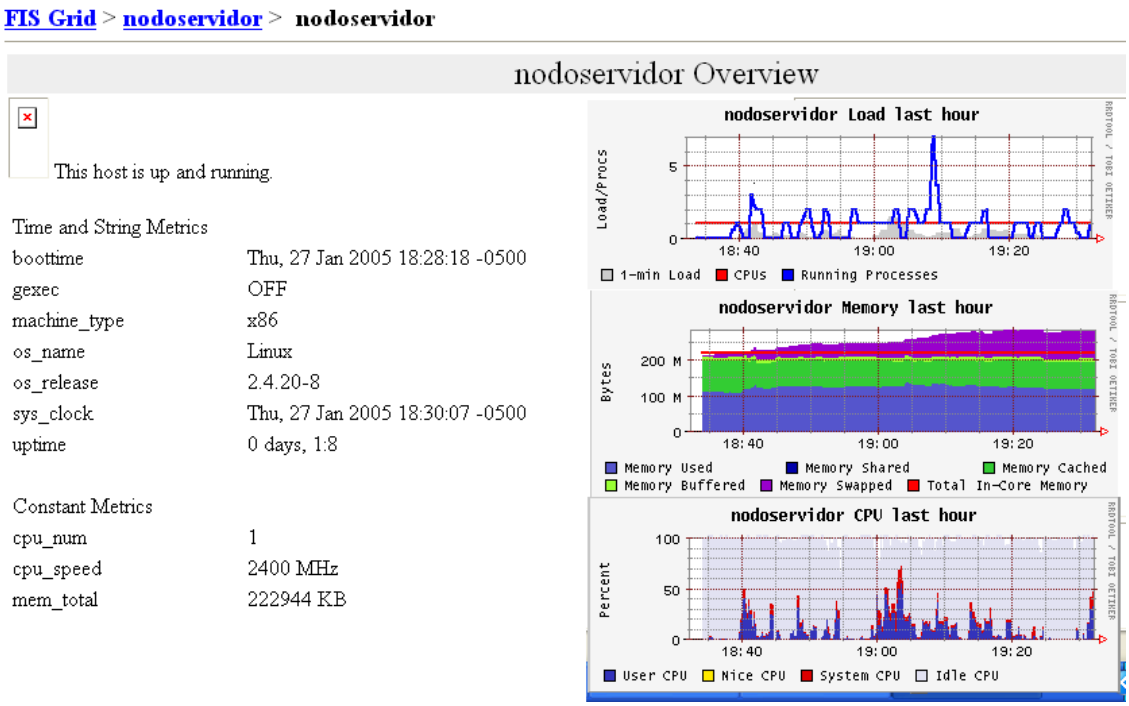
NODO SERVIDOR FUNCIONAMIENTO







NODO SERVIDOR CON VISTA GENERAL



FUNCIONAMIENTO CLIENTE FUNCIONAMIENTO

FIS Grid > **nodoservidor** > **oscarnode1.oscardomain**

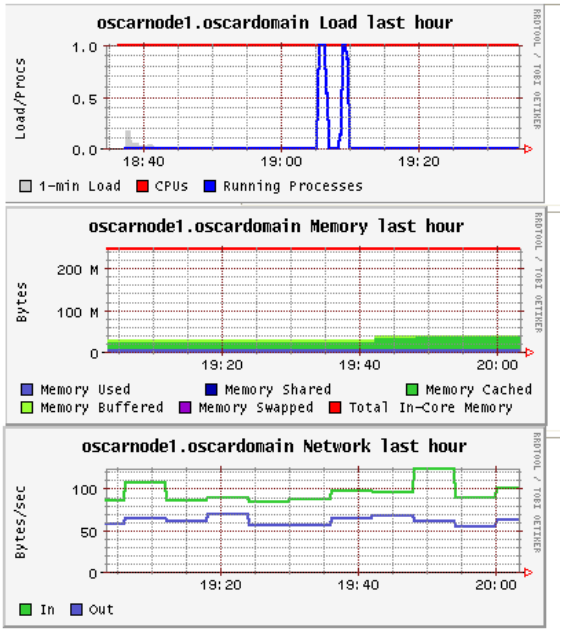
oscarnode1.oscardomain Overview

This host is up and running.

Time and String Metrics	
boottime	Thu, 27 Jan 2005 18:57:10 -0500
gexec	OFF
machine_type	x86
os_name	Linux
os_release	2.4.20-8smp
sys_clock	Thu, 27 Jan 2005 18:57:43 -0500
uptime	0 days, 0:40

Constant Metrics	
cpu_num	1
cpu_speed	1799 MHz
mem_total	252960 KB
mtu	1500 B
swap_total	248968 KB

Gmetrics



CLUSTERS DE COMPUTADORAS

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 que 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 se basa 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.

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. El último tipo 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.

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.

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 ofrecida por los clusters de balanceo de carga.

– Incremento de la confiabilidad y la robustez ofrecido por los clusters de alta Disponibilidad.

Por ejemplo, en laS 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.

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.

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...

BEOWULF

En el verano de 1994, Thomas Sterling y Don Becker, trabajando en el CESDIS2 (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). 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 doityourself4 para mantener el trabajo de software independiente del hardware específico de cada fabricante.

Figura 1 (5_Wiglaf, el primer cluster)

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.

SOFTWARE USADO EN CLUSTERS (SISTEMAS DE ARCHIVO)

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

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.

SOFTWARE DE MONITORIZACIÓN Y ADMINISTRACIÓN

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

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.

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.

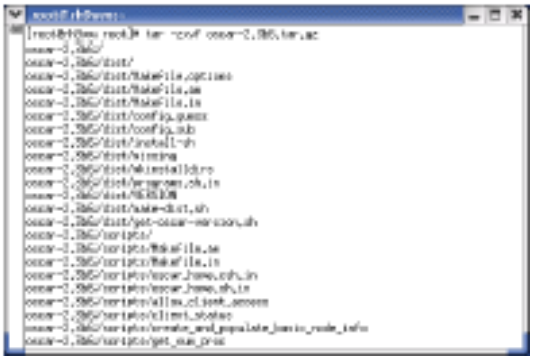
WebArchive.Org es una hemeroteca digital de Internet, donde se puede encontrar la información de más de 5 años de la web, representando esto una base de datos con más de 100TB de información. A través de su Máquina de Tiempo (The Wayback Machine) se pueden consultar más de 10 mil millones de páginas. 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.

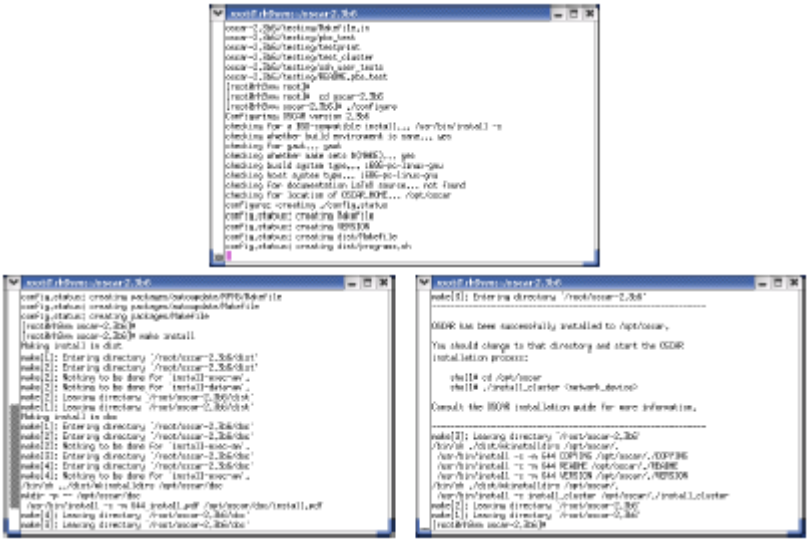
NOAA (The National Oceanic and Atmospheric Administration) utiliza varios clusters de diversas tecnologías en sus proyectos. Su grupo HPCC (High Performance Computing andCommunications) trabaja sobre diversas áreas de supercomputación, balanceo de carga y alta disponibilidad.

PANTALLAS DE OSCAR TOOLKIT

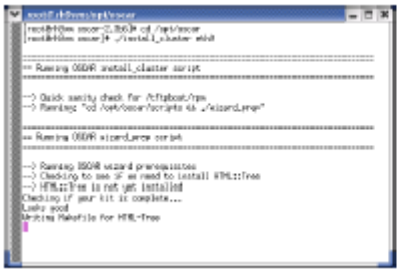
1. Corriendo install cluster



2. Configurando la instalación de oscar toolkit



3. Corriendo el script de Install cluster



4. Ingresar Oscar Data Base

5. Instalación de Oscar Wizard

6. Selección de paquetes adicionales medisnte el ODP

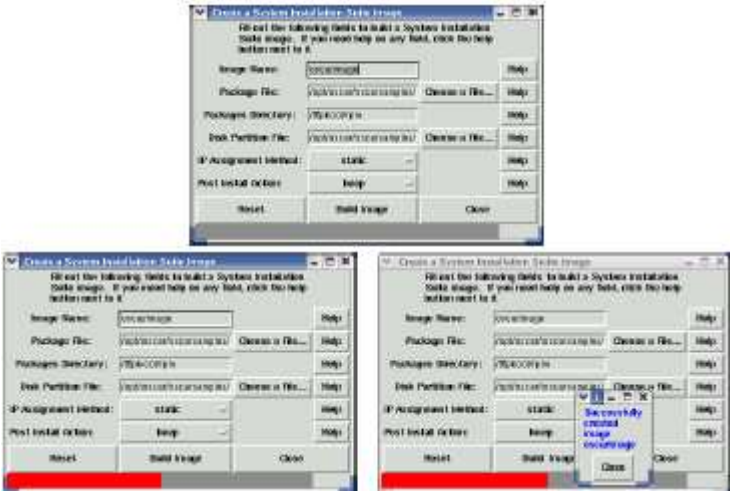
7. Adicionando repositorio ODP

8. Seleccionando paquetes de Oscar

9. Configurando la selección de paquetes de Oscar

10. Instalación de paquetes Oscar Servidor

11. Construyendo la Imagen de Oscar



12. Definiendo Clientes Oscar

13. Iniciando la Red de los Clientes

14. Construyendo un diskette autobooteable para los nodos Clientes

15.Booteando el nodo Cliente para el paso de los archivos por la red

16. Completando el cluster Oscar

17. Test del Cluster Oscar

AÑADIENDO Y BORRANDO LOS NODOS CLIENTES

Describiremos los pasos necesarios para agregar o borrar los nodos.

Una vez construido el clúster con buen resultado y queremos agregar o borrar un nodo de cliente, ejecutamos lo siguiente:

./install_clúster <device Eth xx>

Una vez que el asistente del Oscar aparece

Ejecutamos los dos últimos pasos para agregar o borrar clientes.

AÑADIENDO CLIENTES DE OSCAR

Pulamos el botón del asistente titulado como <Add Oscar Clients>. Un diálogo es mostrado en la Figura 1.

Estos pasos son familiares, pues son los mismos de la instalación.

Al añadir un nodo, en el paso definición de los clientes, necesitamos cambiar lo

siguiente:

- El número de los anfitriones
- El número inicial
- IP de principio

Debemos coleccionar las direcciones MAC de las Tarjetas de cada nodo.

Figura 1: Añadir clientes de Oscar.

Borrando los nodos clientes

Pulsamos el botón del asistente titulado <Delete Oscar Clients>. Un diálogo se mostrará.

Figura 2. Seleccionamos los nodo(s) que desea para borrar y pulsar el botón Delete clients, entonces

pulse <Close>.

Figura 2: Borrar clientes de Oscar.

SISTEMA DE UNA SOLA IMAGEN (SSI SINGLE SYSTEM IMAGE)

INTRODUCCIÓN

La imagen de sistema única (SSI) es la propiedad de un sistema de ocultar la naturaleza heterogénea y distribuida de los recursos disponibles y presentarlos a los usuarios y a las aplicaciones como un sólo recurso unificado de cómputo.

La SSI puede ser habilitada de diferentes formas, éstas varían desde las extensiones de hardware hasta los mecanismos de software. Con la SSI los usuarios tienen una imagen globalizada de los recursos que hay disponibles, sin importar el nodo al cual están físicamente asociados. Más aun, la SSI puede asegurar que un sistema continúe en operación después de algún fallo (alta disponibilidad), así como asegurar que el sistema se mantenga balanceado

proporcionando un multiprocesamiento común (manejo de recursos y calendarización).

Los objetivos en el diseño de una SSI para sistemas clusters se enfocan principalmente en la total transparencia en el manejo de recursos, el rendimiento escalable y la disponibilidad del sistema para soportar las aplicaciones del usuario. Una SSI puede definirse como la ilusión, creada por hardware o por software, que presenta una colección de recursos como un sólo, y más poderoso, recurso unificado.

SERVICIOS Y BENEFICIOS

Los servicios clave de una imagen de sistema única para clusters, son los siguientes:

Punto de entrada único*:* Un usuario puede conectarse al cluster como un host virtual (como telnet beowulf.myinstitute.edu), a pesar de que el cluster pueda tener múltiples nodos físicos para iniciar una sesión de login. El sistema distribuye de manera transparente las solicitudes de conexión de los usuarios a los diferentes hosts físicos para balancear la carga.

Interfase de usuario única*:* El usuario debe ser capaz de utilizar el cluster a través de un solo GUI. La interfase debe tener el mismo "look & feel" que el disponible en las estaciones de trabajo (e.g. Solaris Open Win o Windows NT GUI).

Espacio de procesos único*:* Todos los procesos de usuario, no importando el nodo en que residan, tienen un identificador de proceso único en todo el cluster. Un proceso en cualquier nodo puede crear un proceso hijo en el mismo nodo o en uno diferente. Un proceso debería de ser capaz de comunicarse con cualquier otro proceso de un nodo remoto. Los clusters deberían soportar un manejo de procesos globalizado y permitir el manejo y control de procesos como si estuvieran corriendo en máquinas locales.

Espacio de memoria único: Los usuarios tienen la ilusión de una memoria principal grande y centralizada, que en realidad puede ser un conjunto de memorias locales distribuidas. El software DSM ya ha sido utilizado para alcanzar un único espacio de memoria en los clusters. Otro acercamiento es permitir al compilador distribuir la estructura de datos de una aplicación a través de múltiples nodos. Todavía es un reto, desarrollar un esquema de

memoria única que sea eficiente, independiente de la plataforma y capaz de soportar códigos binarios secuénciales.

Espacio de E/S único (SIOS)*:* Esto permite a cualquier nodo, desempeñar operaciones de E/S en periféricos o dispositivos de disco locales o remotos. En este diseño de SIOS, los discos asociados a los nodos del cluster, los RAIDs conectados por red y los dispositivos periféricos, forman un espacio de direcciones único.

Jerarquía de archivos única: Al entrar al sistema, el usuario ve una imagen única de un sistema de archivos enorme, como una sola jerarquía de archivos y directorios bajo el mismo directorio raíz, que integra de manera transparente los discos locales y globales y otros dispositivos de archivos. Ejemplos de jerarquía de archivos única incluyen NFS, AFS, xFS y Solaris MC Proxy.

Red virtual única*:* Esto significa que cualquier nodo puede acceder a cualquier conexión en el dominio del cluster, aun si la red no está conectada físicamente a todos los nodos del cluster. Redes múltiples soportan operaciones de red virtual única.

Sistema de manejo de trabajos único*:* Bajo un calendarizador de trabajos global, un trabajo de usuario puede ser sometido desde cualquier nodo y requerir cualquier número de nodos para ejecutarlo. Los trabajos pueden ser calendarizados para ejecutarse por lotes, en interactivo o en modo paralelo. Ejemplos de sistemas de manejo de trabajos para clusters, incluyen GLUnix, LSF y CODINE.

Punto de control y manejo único: El cluster completo y cada nodo individual pueden ser configurados, monitoreados, probados y controlados desde una ventana única, usando herramientas con un GUI único, en forma muy parecida a una estación de trabajo NT manejada por la herramienta Task Manager.

Migración de procesos y checkpoint*:* Checkpoint es un mecanismo de software que guarda periódicamente el estado de los procesos y los resultados intermedios de cómputo que residen en la memoria o los discos. Esto permite una recuperación completa después de un fallo. La migración de procesos es necesaria en el balanceo de carga dinámica entre los nodos del cluster y para soportar el checkpoint.

CLUSTER COMMAND CONTROL (C3) OVERVIEW

cexec(s) Una utilidad que ejecuta una línea de comando escrita en consola en cada nodo de un cluster.

cget Una utilidad para refrescar un archivo o archivos específicos en cada nodo de un cluster y localizar estos en un directorio determinado.

ckill Una utilidad para ejecutar la sentencia "kill" en cada nodo del cluster, para un nombre de proceso específico

clist Una utilidad que lista los nombres y tipos de cluster en el archivo de configuración del cluster.

cnum Una utilidad que retorna los nombres de los nodos especificados en un rango determinado en la línea de comando.

cname Una utilidad que retorna la posición de un nodo especificado por el nombre del nodo dado en la línea de comando.

cpush Una utilidad para poner archivos desde la maquina local a los nodos en el cluster.

cpushimage Una utilidad para poder manejar imágenes creadas por el **SystemImager** desde el servidor a los nodos del cluster.

crm Una utilidad que borra archivos desde cada nodo del cluster.

cshutdown Una utilidad que corre shutdown en cada nodo para reiniciar los nodos clientes.

RRDTOOL, HERRAMIENTA DE BASES DE DATOS EN ROUND ROBIN

Qué es RRDtool

• RRDtool es un sistema para almacenar y mostrar datos a través del tiempo. Ej. Tráfico de red, temperatura de la sala de máquinas, carga de servidores.

• Los datos se almacenan de manera compacta, round robin

• La base de datos no crece con el tiempo.

• Se puede mostrar fácilmente en forma de gráficos para distintos periodos de tiempo.

Se utiliza para:

• Poder medir un valor a lo largo del tiempo

• Especificar que resoluciones deben guardarse

• Alimentar a RRDtool con estos valores

• Crear gráficos: PNG, GIF (con esta herramienta)

• Publicarlos en páginas web

El Servicio Informático lo utiliza sobre su infraestructura:

- Monitorización
- Estudio de rendimiento
- Disponibilidad
- Estadísticas de uso
- Alertas

Resultados

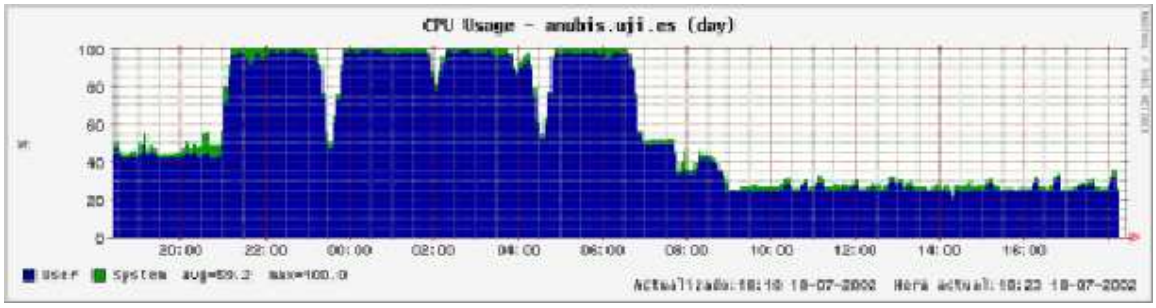


Figura 3. Consumo de CPU en Oscar, 1día

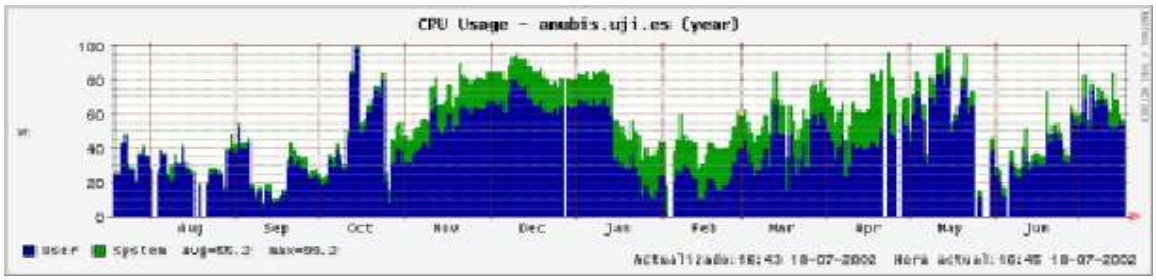


Figura 4. Consumo de CPU en Oscar, un año

Se observa el tanto por cien de cpu usado, tanto por usuarios como por el sistema, a lo largo del periodo de tiempo representado. También se indica el máximo y la media de utilización durante ese periodo.

Figura 5. Número de usuarios.

Número de usuarios en el sistema. Este dato se obtiene a partir del comando who.

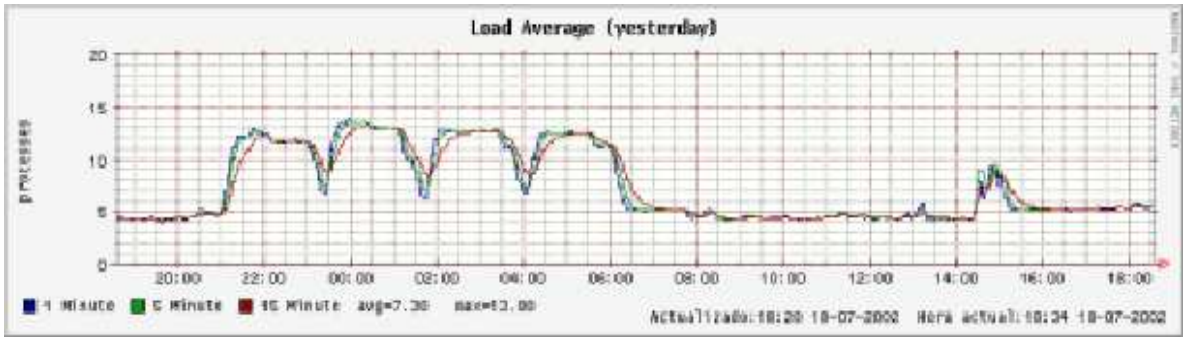


Figura 6. Carga del sistema. Número de procesos esperando CPU.

Cada una de las líneas representa la carga del sistema para el último minuto, cinco minutos y quince minutos respectivamente. Se obtiene a partir de la salida del comando ***uptime***.

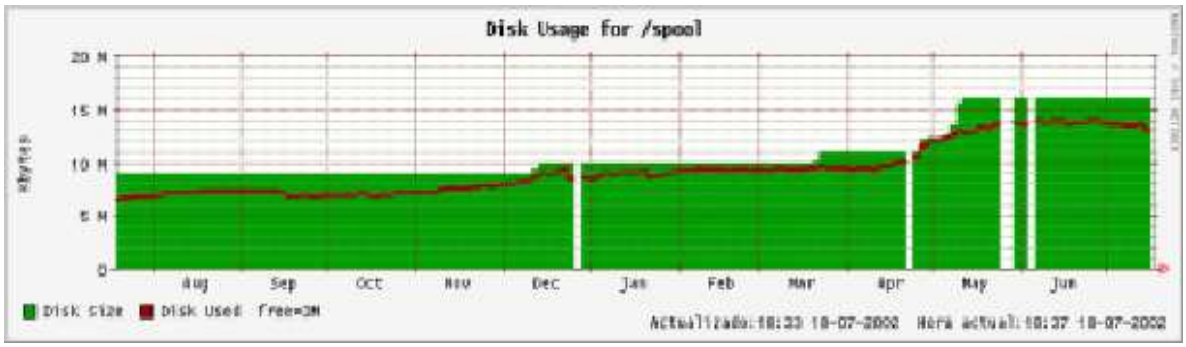


Figura 7. Ocupación de disco.

El área representa la capacidad del disco, mientras que la línea indica el disco que está siendo usado. Se obtiene de la salida del comando ***df***.

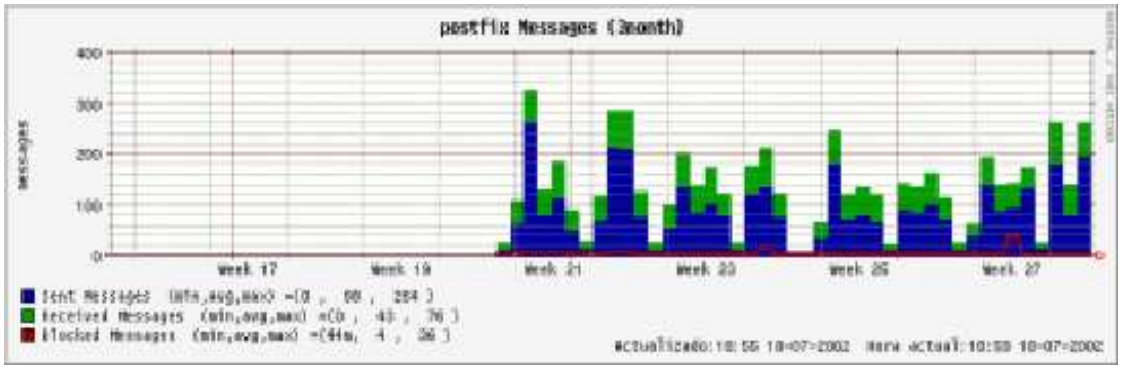


Figura 8. Mensajes de correo.

Muestra el número de mensajes de correo enviados, recibidos y bloqueados. Para ello se procesan las líneas de log del servidor de correo postfix.

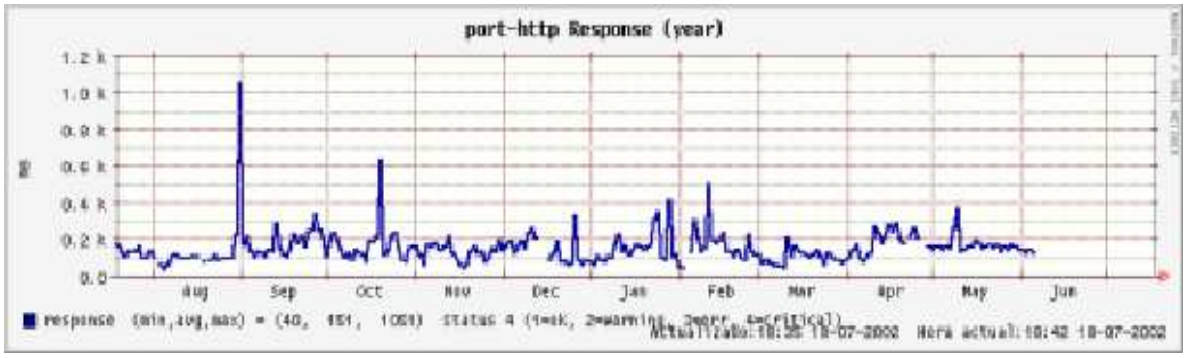


Figura 9. Tiempo de respuesta www.uji.es (antes de migración)

Figura 10. Tiempo de respuesta www.uji.es (después de migración)

Tiempo de respuesta del servidor Web. Se obtiene a partir del tiempo que se tarda en obtener la página principal del servidor.

Funciones de RRDtool

- Interactuamos con la base de datos a través de las funciones de RRDtool
- Están ampliamente documentadas
- Tienen la misma sintaxis con independencia del lenguaje desde donde lo uses:
- Línea de comandos
- Módulo de perl
- PHP
- C

PASO DE MENSAJES

APLICACIONES DISTRIBUIDAS CON MPI

Introducción

MPI es una especificación estándar de una librería que permite comunicar varios procesos mediante paso de mensajes, independientemente de si los procesos ejecutan en la misma máquina o no.

LAM-MPI es la implementación estándar que puede encontrarse en todas las distribuciones de

GNU/Linux, lo cual permite ejecutar en hidra aplicaciones paralelas que hayan sido probadas anteriormente en otras configuraciones. Es posible encontrar documentación y futuras actualizaciones de la librería en la página http://www.lam-mpi.org/

MPICH es una implementación de MPI optimizada para entornos homogéneos y myrinet, lo que proporciona un mayor rendimiento en el paso de mensajes entre nodos. La página del proyecto MPICH es http://www-unix.mcs.anl.gov/mpi/mpich/

De las dos implementaciones disponibles, la más recomendada es MPICH, tanto por su rendimiento (muy superior a LAM) como por facilidad de uso. La razón por la que se ha decidido dejar ambas implementaciones disponibles es porque LAM está más extendida, por lo que es posible encontrar alguna aplicación que haga uso forzoso de ella.

Desarrollo de aplicaciones distribuidas con MPICH

Ejemplo sencillo de programación en C con MPICH

El primer ejemplo que compilaremos es la versión distribuida del famoso "Hola Mundo", que puede encontrarse al final de este documento con el nombre de hello_mpi.c. Para ejecutar el programa hay que seguir los siguientes pasos:

1. Compilar el fichero hello_mpi.c empleando el comando **/opt/scali/contrib/mpich/bin/mpicc** (en lugar del típico cc), o bien **mpicc.mpich**

mpicc.mpich hello_mpi.c -o hello_mpi

2. Ejecutar nuestro programa con el comando **/opt/scali/contrib/mpich/bin/mpirun**, o bien **mpirun.mpich**. Podemos indicar cual es el número de procesos que hay que lanzar con la opción -np, ocupándose MPICH de repartir los procesos entre los nodos del cluster. Por ejemplo, para lanzar 4 procesos:

mpirun.mpich -np 4 ./hello_mpi

La salida del programa debe ser algo parecido a esto:

Soy el nodo 0 de un cluster de 4

Soy el nodo 1 de un cluster de 4

Soy el nodo 2 de un cluster de 4

Soy el nodo 3 de un cluster de 4

Compilación de programas Fortran con MPICH

En caso de que el programa esté escrito en Fortrán, el único cambio necesario es cambiar el nombre del compilador para emplear **/opt/scali/contrib/mpich/bin/mpif77**, o bien **mpif77.mpich**, como puede verse en el siguiente ejemplo:

mpif77.mpich hello_mpi.f -o hello_mpi

mpirun.mpich -np 4 ./hello_mpi

La salida del programa debe ser similar a la del ejemplo anterior.

Compilación de programas C++ con MPICH

Cuando la aplicación se ha realizado en lenguaje C++ debemos emplear **/opt/scali/contrib/mpich/bin/mpiCC** , o bien **mpiCC.mpich**, como puede verse en el siguiente ejemplo:

mpiCC.mpich hello_mpi.cc -o hello_mpi

mpirun.mpich -np 4 ./hello_mpi

Un caso práctico: Cálculo distribuido de una integral numérica

Ejemplo práctico de programación distribuida, que calcula el resultado de una integral aprovechando el número de nodos del cluster que tengamos a nuestra disposición. La solución propuesta en el ejemplo **integral_mpi.c**, que calcula la

integral de la función seno(x) dividiendo el intervalo de integración entre todos los nodos, de forma que cuantos más nodos tengamos más rápida será la respuesta.

La compilación y ejecución del programa es idéntica a los ejemplos anteriores, salvo que necesita la librería matemática (flag -lm) para poder evaluar la función seno. Podemos ejecutar el programa conel comando time para verificar la diferencia de tiempos al variar el número de nodos.

Para ejecutar la aplicación empleando MPICH:

```
mpicc.mpich integral_mpi.c -o integral_mpi -lm
time mpirun.mpich -np 1 ./integral_mpi
time mpirun.mpich -np 4 ./integral_mpi
time mpirun.mpich -np 10 ./integral_mpi
```

Ejecución de programas MPICH de forma óptima con OpenPBS

Aunque en los ejemplos anteriores se ha empleado el comando mpirun.mpich para la ejecución de aplicaciones paralelas con MPICH, no es la manera más recomendable, sino que es mejor realizar las ejecuciones a través del gestor de trabajos OpenPBS. La razón por la cual no se recomienda emplear directamente mpirun.mpich es que este lanzador usa siempre los nodos 01 a n para lanzar sus

procesos, por lo que la ejecución de varias aplicaciones paralelas competirán por el uso de los nodos con numeración más baja del cluster. Para evitar esto, se recomienda emplear el comando **scasub con la opción -mpich** , lo que permite optimizar el uso de los recursos entre todos los usuarios. Así, para ejecutar la aplicación anterior, se recomienda usar:

```
mpicc.mpich integral_mpi.c -o integral_mpi -lm
scasub -mpich -np 1 ./integral_mpi
scasub -mpich -np 4 ./integral_mpi
scasub -mpich -np 10 ./integral_mpi
```

Ejecución de programas MPICH desde el gestor scadestop

Quien prefiera realizar las ejecuciones de sus aplicaciones desde el entorno gráfico **scadesktop,** deberá realizar los siguientes pasos:

1. Primero debe compilar su programa empleando las librerías MPICH tal como se ha explicado en los apartados anteriores, es decir, mediante los compiladores mpicc.mpich, mpiCC.mpich o mpif77.mpich, según el lenguaje en que se haya escrito la aplicación.

2. Ejecute el programa scadesktop y seleccione el cluster deseado (por ejemplo "Hidra") haciendo doble click sobre el icono del mismo.

3. Introduzca su nombre de usuario y contraseña para acceder al cluster

4. Seleccione los nodos sobre los que desea ejecutar la aplicación. Recuerde que puede pulsar la tecla control para añadir nuevos nodos a la selección actual.

5. Ejecute la acción de menú: Run->'MPICH program'

6. Escriba la ruta completa del ejecutable de la aplicación en el campo 'MPI program' y pulse el botón 'Run program'. Recuerde que la aplicación se ejecutará sobre los nodos seleccionados en el paso 4.

Desarrollo de aplicaciones distribuidas con LAM-MPI

A diferencia de MPICH, la ejecución con el sistema LAM-MPI necesita la inicialización de un entorno de ejecución por parte del usuario, lo que complica algo más su uso. Si por alguna razón se prefiriese emplear LAM-MPI, estos son los pasos que hay que seguir para compilar los ejemplos anteriores.

Ejemplo sencillo de programación en C con LAM-MPI

Para compilar y ejecutar el programa hello_mpi.c hay que seguir los siguientes pasos:

1. Compilar el fichero hello_mpi.c empleando el comando **mpicc** (en lugar del típico cc)

mpicc hello_mpi.c -o hello_mpi

2. Escribir un fichero de configuración con el nombre de los nodos que queremos que ejecuten el programa. Por ejemplo, para lanzarlo en los nodos cat01 a cat04 escribiríamos en el fichero

Nodos _1-4.txt:

nodo01

nodo02

nodo03

nodo04

3. Ejecutar el comando **lamboot** con el fichero de configuración anterior, de forma que se crea la infraestructura de comunicación entre los nodos.

lamboot nodos_1-4.txt

4. Ejecutar nuestro programa con el comando **mpirun**. Podemos indicar cual es el número de procesos que hay que lanzar con la opción -np, ocupándose MPI de repartir los procesos entre los nodos indicados mediante el comando lamboot. Por ejemplo, para lanzar 4 procesos:

mpirun -np 4 ./hello_mpi

La salida del programa debe ser algo parecido a esto:

Soy el nodo 0 de un cluster de 4

Soy el nodo 1 de un cluster de 4

Soy el nodo 2 de un cluster de 4

Soy el nodo 3 de un cluster de 4

5. Una vez que hayamos terminado de ejecutar programas distribuidos debemos destruir la infraestructura creada empleando el comando **wipe** con el mismo fichero de configuración empleado en el comando lamboot:

wipe nodos_1-4.txt

NOTA: Es muy IMPORTANTE realizar el paso 5 una vez terminada la ejecución de programas, puesto que de lo contrario los nodos quedan marcados como ocupados para futuras configuraciones de LAM-MPI.

Compilación de programas Fortran con LAM-MPI

En caso de que nuestro programa esté escrito en Fortran en lugar de en C, el único cambio necesario para la compilación y ejecución del programa consiste en emplear el compilador **mpif77** en lugar del mpicc. Por ejemplo, para compilar y ejecutar el programa hello_mpi.f debemos tener preparado el fichero nodos_1-4.txt como en el ejemplo anterior y emplear los siguientes comandos:

```
mpif77 -I/usr/include hello_mpi.f -o hello_mpi
lamboot nodos_1-4.txt
mpirun -np 4 ./hello_mpi
wipe nodos_1-4.txt
```

Nota: Es importante incluir la opción **-I/usr/include** al compilador para que encuentre los ficheros de cabecera necesarios para compilar con MPI

Compilación de programas C++ con LAM-MPI

En caso de que nuestro programa esté escrito C++, debemos emplear el compilador **mpiCC**:

```
mpiCC hello_mpi.cc -o hello_mpi
lamboot nodos_1-4.txt
mpirun -np 4 ./hello_mpi
wipe nodos_1-4.txt
```

Cálculo distribuido de una integral numérica con LAM-MPI

La compilación y ejecución del programa integra_mpi.c es idéntica a los ejemplos anteriores, salvo que necesita la librería matemática (flag -lm).

```
mpicc integral_mpi.c -o integral_mpi -lm
lamboot nodos.txt
time mpirun -np 1 ./integral_mpi
time mpirun -np 4 ./integral_mpi
time mpirun -np 10 ./integral_mpi
wipe nodos_1-10.txt
```

CODIGO DE PROGRAMAS REALIZADOS EN LAM MPI

Programa de paso de Mensajes

```
#include "mpi.h"
#include <stdio.h>
#include <string.h>
int main (int argc, char **argv)
{
int rank, size;
MPI_Init (&argc,&argv);
MPI_Comm_rank (MPI_COMM_WORLD,&rank);
MPI_Comm_size (MPI_COMM_WORLD,&size);
printf ("soy el nodo %d de in cluster de %d \n",rank,size);
```

```
MPI_Finalize();
return 0;
}
```

Programa de Cálculo Distribuido de una Integral numérica

```
#include <stdio.h>
#include <math.h>
#include <mpi.h>
int main(int argc, char **argv)
{
   MPI_Status status;
   int rank, size;
   double ancho, a,b,x,acum;
   double delta=0.00000001;
   int num_iter;

   MPI_Init (&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD,&rank);
   MPI_Comm_size(MPI_COMM_WORLD,&size);
   ancho=M_PI / size;
```

```
a = rank * ancho;
b = (rank + 1) * ancho;
printf("Nodo %d calculando intervalo %f .. %f \n",rank,a,b);
acum=0;
for(x=a; x<b; x += delta)
acum += delta * sin (x);
if(rank > 0)
{
      MPI_Send (&acum,1,MPI_DOUBLE,0,1,MPI_COMM_WORLD);
}
else
{
      int i;
      double data;
      for(i=1; i<size; i++)
      {
         MPI_Recv (&data,1,MPI_DOUBLE,i,1,MPI_COMM_WORLD,&status);
         acum += data;
         }
      printf("El resultado de la integral entre 0 y PI ES: %f\n",acum);
```

```
    }
    MPI_Finalize ();
    return 0;
}
```

PVM vs. MPI. Como aprovechar las ventajas de cada uno

PVM nació en 1.989 como un proyecto universitario y ha evolucionado desde entonces beneficiándose de las experiencias de sus usuarios. El concepto clave en su diseño es la máquina virtual: un conjunto de procesadores conectados por una red es percibido por el usuario como un único ordenador paralelo. El objetivo de este diseño es la portabilidad y flexibilidad, pero con un posible impacto en el rendimiento. Por el contrario, MPI es un estándar formal desarrollado en 1.993 por un comité de 60 expertos. El objetivo de MPI era definir un estándar que cada fabricante pudiera implantar para lograr el máximo rendimiento posible en su hardware. Los distintos orígenes de PVM y MPI así como sus distintas filosofías de diseño se traducen en un conjunto de diferencias:

Estándar: PVM es el estándar de hecho para el procesamiento en paralelo, MPI es el estándar formal. Trabajar con un estándar formal puede ser una ventaja en algunos contextos.

Tolerancia a fallos: PVM es tolerante a fallos en el sentido de que si un nodo falla los demás pueden seguir adelante con la ejecución del programa. Este mecanismo de recuperación no existe en MPI-1

Heterogeneidad: la máquina virtual hace que PVM funcione muy bien en clusters heterogéneos (aquel construido con ordenadores distintos entre sí). MPI espera que el hardware sea homogéneo, una máquina MPP o, al menos, un Beowulf de ordenadores idénticos.

Rendimiento: cabe esperar que MPI sea de mejor rendimiento que PVM, especialmente en máquinas MPP.

Base instalada: al ser bastante posterior, el número de programas que hacen uso de MPI es significativamente inferior a los que emplean PVM.

Control de ejecución y gestión de recursos: PVM especifica cómo ejecutar los programas independientemente del hardware y del sistema operativo.

Asimismo, incluye un mecanismo para la gestión de recursos. MPI ni lo uno ni lo otro. Se espera que las implementaciones de MPI-2 incorporen esta funcionalidad.

Topología: con MPI es posible especificar la topología de conexión entre elementos de proceso, lo que permite optimizar las comunicaciones. PVM no dispone de esta capacidad.

Complejidad: PVM muestra un propósito definido en todo su diseño. En la especificación de MPI 2 se han incluido toda una serie de capacidades al margen del paso de mensajes, como acceso remoto a memoria o un sistema paralelo de entrada/salida. Estas cosas son útiles, pero hacen de MPI una librería compleja.

Flexibilidad en las comunicaciones: el conjunto de funciones de comunicación de MPI es más rico que el de PVM.

ACRONIMOS

BIOS Entrada salida básica del sistema

C3 Cluster Command Control Overview

CORBA common Object requwat brokerar arrchitecture

COTS Commodity off the shelf

DHCP dynsmic host configurtion protocol

GPL General Public License

HPC Computación de alto rendimiento

HPCC Conputación de alto rendimiento y comunicación

IBM internacional business machines

LAN Redes de área local

MPI Interface de paso de mensajes

MPICH Message Passing Interface

NAT network address translator

NFS network file system archivos de sistema de lared

NIS Sistema de informacion de la red

PVM Máquina paralela vitual

SCE Scalable Cluster Environment

UDP User Datagram Protocol