UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLAN

BASES DE DATOS PARALELAS EN CLUSTERS LINUX

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INFORMÁTICA

P R E S E N T A N:

ALMA VALERIA GARCIA BAHENA
ADIEL JESÚS NAVARRO ROSADO

ASESOR: L. C. CARLOS PINEDA MUÑOZ.

CUAUTITLAN IZCALLI, EDO. DE MEX. 2005.

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

U. N. A. M.
FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
DEPARTAMENTO DE EXAMENES PROFESIONALES

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

Bases de Datos Paralelas en Clusters Linux.

que presenta la pasante: Alma Valeria Garcia Bahena
con número de cuenta: 9538427-6 para obtener el título de :
Licenciada en Informática

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 07 de Abril de 2005.

PRESIDENTE	L.C. Carlos Pineda Muñoz	
VOCAL	Ing. Moisés Hernández Duarte	
SECRETARIO	LIA. Conrado Camacho Arteaga	
PRIMER SUPLENTE	Ing. Antonio González Almaguer	
SEGUNDO SUPLENTE	M.A. Jaime Navarro Mejía	

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

Bases de Datos Paralelas en Clusters Linux.

que presenta el pasante: Adiel Jesús Navarro Rosado
con número de cuenta: 09956123-7 para obtener el título de :
Licenciado en Informática

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 07 de Abril de 2005

PRESIDENTE	L.C. Carlos Pineda Muñoz	
VOCAL	Ing. Moisés Hernández Duarte	
SECRETARIO	LIA. Conrado Camacho Arteaga	
PRIMER SUPLENTE	Ing. Antonio González Almaguer	
SEGUNDO SUPLENTE	M.A. Jaime Navarro Mejía	

DEDICATORIAS VALERIA::

GRACIAS A DIOS:

Por permitirme llegar hasta aquí: por darme salud y bienestar. Pero sobretodo por darme a mi familia y pareja que siempre estan conmigo incondicionalmente.

GRACIAS A TI MAMA:

Por tu apoyo, aliento y la seguridad que me has dado constantemente. ¡¡Eres la mejor!!

GRACIAS A TI PAPA:

Porque muy a tu manera siempre has estado presente y también porque siempre me has dado aliento y apoyo.

GRACIAS A MIS HERMANAS ERIKA Y LUPITA:

Porque me han acompañado en este camino y espacio y espero que muy pronto ellas también lleguen hasta aquí ¡¡Las quiero mucho!!

GRACIAS A SALVADOR:

Mi pareja y mi amigo; porque siempre tengo su apoyo incondicional y cariño. Por sus palabras que a veces me hacían falta. ¡¡Te amo!!

GRACIAS A MI GRAN AMIGO ADIEL:

Porque sin él, esto no hubiera sido posible. Por tenerme paciencia y compartir esta experiencia conmigo.

GRACIAS A MI ASESOR, EL L.C. CARLOS PINEDA:

Ya que nos ha dado un gran tesoro, el transmitirnos un poco de sus conocimientos.

DEDICATORIAS ADIEL JESUS:

A DIOS:
¡¡Gracias por todo, Señor!!

A MI MAMÁ ESPERANZA (q.e.p.d.):
Con todo mi amor; gracias por todas tus bendiciones.

A MIS PAPAS:
EDÉN NAVARRO Y NORMA ROSADO.
Con respeto y cariño, ya que gracias a sus consejos, sacrificios, apoyo y amor he podido superarme y llegado hasta donde estoy.

A MI HERMANO
EDGAR NAVARRO ROSADO,
Por su apoyo incondicional. De corazón, te lo agradezco. ¡¡Sigue adelante!!

A MI GRAN AMIGA:
VALERIA GARCIA.
Por tu confianza, apoyo y amistad sincera.

A LA UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO:
Por abrirme sus puertas y por darme la oportunidad para desarrollarme de manera personal y profesional.

AL L. C. CARLOS PINEDA MUÑOZ:
Por su guía y soporte para el desarrollo de este trabajo. Gracias por su tiempo y consejos.

INDICE CAPITULAR

"BASES DE DATOS PARALELAS EN CLUSTERS LINUX"

OBJETIVO GENERAL:

- Instalar y configurar un administrador de base de datos relacional (Mysql) en un cluster.

Objetivos Particulares:

- Definir las características a considerar para la instalación de una base de datos en un cluster.
- Establecer las diferencias entre una base de datos centralizada, distribuida y paralela
- Dar a conocer las características de un cluster y las ventajas que tiene.

INTRODUCCIÓN:

En la actualidad la informática a tomado una mayor importancia, ya que actualmente las empresas para poder estar a la vanguardia necesitan satisfacer requerimientos de información, tanto interna como externa, para lograr un mejor manejo empresarial, eficiente, con datos precisos, en tiempo, veraces y de fácil comprensión, permitiéndole a los ejecutivos y analistas de información tomar decisiones para realizar las acciones pertinentes y definir las estrategias a implementar en el futuro. Pero a raíz de esto ha surgido una necesidad que es tener más y mejores equipos.

En las grandes compañías, cuyos procesos son más complejos y delicados, es necesario tener una supercomputadora la cual puede desarrollar miles de procesos por segundo. Las supercomputadoras tienen dentro de si arreglos de microprocesadores muy rápidos que trabajan de forma paralela, pero hay un inconveniente, este equipo es muy costoso y muy delicado, así que por esta cuestión no es tan fácil que las empresas puedan comprarlo, por lo tanto, para poder tener un equipo capaz de realizar estos procesos complejos y con basta información y que esté dentro de sus posibilidades financieras surgieron los CLUSTERS.

El término Clusters se aplica a "*los conjuntos o conglomerados de computadoras, construidos utilizando componentes de hardware comunes y software*" (http://clusters.fisica.uson.mx/clusters_de_Linux.htm), los cuales juegan hoy en día, un papel importante en la solución de problemas de las ciencias, las ingenierías y aplicaciones comerciales.

Entre las aplicaciones más comunes de clusters se encuentra el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, análisis de imágenes, y más.

En un Cluster, todos los nodos trabajan en cooperación con un objetivo y una meta común pues la asignación de recursos los ejecuta un solo administrador centralizado y global.

Otro componente dominante para permitir compatibilidad, es el software del sistema. Con la madurez y la robustez de Linux, el software GNU y de la estandarización de envío de mensajes vía PVM y MPI, los programadores ahora tienen una garantía que los programas que escriban correrán en un futuro sin importar quien hace los procesadores o las redes de trabajo.

Software GNU es software que es liberado bajo el auspicio del Proyecto GNU (GNU en un acrónimo de GNU's No es UNIX). La mayoría del software GNU está protegido con copyleft, pero no todos; sin embargo, todo el software GNU debe ser software libre.

Algo de software GNU es escrito por el personal de la Fundación para el Software Libre (Free Software Foundation), pero la mayoría del software GNU es aportada por voluntarios. Parte del software aportado está protegido con copyright por la Fundación para el Software Libre; otra parte está protegida con copyright por los programadores que los escribieron.

Software Libre se refiere *"a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software"*. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa, y adaptarlo a sus necesidades. (El acceso al código fuente es una condición previa para esto.)
- La libertad de distribuir copias.
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Empresas de renombre internacional están confiando en esta nueva tecnología, para llevar a cabo sus procesos.

La distribución a nivel continental de los 500 cluster más rápidos (fuente: http://www.clusters.top500.org) se da de la siguiente manera:

	Count	Share	Rmax	Rpeak	Procs
USA/Canadá	256	51.2 %	205156	349880	155834
Europe	154	30.8 %	83173	152044	55518
Japan	40	8 %	62818	80889	18565
South-East Asia	33	6.6 %	15186	32082	8118
Australia	6	1.2 %	3135	5259	1432
South América	4	0.8 %	1438	2739	1568
Arabia	4	0.8 %	1967	7622	3506
África	3	0.6 %	873	2534	2240
Total	500	100 %	373749	633052	246781

Count Distribución a nivel continental.
Share Porcentaje de la distribución.
R_{max} Maximal LINPACK performance achieved
R_{peak} Theoretical peak performance
Procs Número de procesadores

En México, el número de proyectos que involucran a los clusters como opción de cómputo o como experimentación va en aumento, ejemplo de esto es la Universidad Nacional Autónoma de México, la cual, desde hace algunos años, ha iniciado a través de sus grupos de investigación proyectos que involucran el uso de esta tecnología. Sin embargo los foros de discusión y el intercambio de información y experiencia han sido escasos.

En este trabajo vamos a explicar a fondo todo lo relacionado con los clusters, en lo referente al software así como explicar las características a considerar para poder instalar una base de datos en un cluster que va a funcionar de forma paralela, esto con el objetivo de compararla con una base de datos no paralela y explicar así sus diferencias y ventajas, y para lograr esto vamos a llevar a cabo un caso practico.

En el capítulo uno expondremos los avances tecnológicos que han sufrido el hardware y el software hasta llegar a la tecnología de clusters.

En el capítulo dos, nos adentraremos en las características de los clusters; su origen, función y desarrollo, así como los componentes que los integran. .

El capítulo tres, tratará acerca de las bases de datos, sus generalidades, modelos, aplicación de las mismas y dos de los modelos existentes, así como de los manejadores más adecuados para estas.

El contenido del capítulo cuatro, se centra en las bases de datos paralelas y el procesamiento en paralelo. Las modalidades de este último y el diseño de este tipo de sistemas.

Agregado a esto; se anexa un caso práctico, desarrollado en el LABORATORIO del CENTRO DE COMPUTO de la FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN, CAMPO 4, el cual será útil para mostrar el desempeño de los clusters y las bases de datos paralelas.

1.0 GENERALIDADES EN HARDWARE Y SOFTWARE.

La informática como tal ha tomado importancia, debido a que las empresas, para poder estar a la vanguardia, necesitan satisfacer sus requerimientos de información interna y externa, para poder lograr un mejor manejo de sus sistemas de información empresarial, permitiéndole tomar las mejores decisiones para realizar las acciones pertinentes y definir las estrategias a implementar en el futuro.

A raíz de esto, ha surgido la necesidad de tener más y mejores equipos. Para cubrir dichas necesidades, siempre se ha recurrido al rápido desarrollo del hardware y del software, lo que ha permitido a las computadoras convertirse en una buena opción para cubrirlas.

1.1 Desarrollo

Independientemente de la arquitectura física, toda aplicación informática consta de al menos tres niveles funcionales, que son los siguientes:

- **Nivel de presentación**: este nivel trata con la interfase entre el usuario y el sistema. El nivel de presentación es el responsable de aceptar los datos de entrada del usuario y mostrar al usuario la información.

- **Nivel de la lógica de negocio**: este es el nivel que añade realmente significado o valor a los datos subyacentes. Es aquí donde se validan los datos, se realizan los cálculos y tienen lugar otras rutinas de procesamiento de la información.

- **Nivel de acceso de datos**: este nivel es el responsable del almacenamiento físico y la extracción de los datos.

La forma en que estos niveles funcionales estan distribuidos a través del hardware ayuda a definir la arquitectura; antes, incluso de que el termino cliente/servidor fuera empleado para describir una arquitectura informática especifica

La arquitectura informática original era la de ***Mainframe o de host***. En este entorno, prácticamente toda la potencia de procesamiento se encontraba en una máquina host central. El nivel de lógica de negocio y el nivel de acceso de datos estaban centralizados en el host. El usuario de la aplicación interactuaba con los datos a través de una terminal no inteligente, debido a que dicha terminal no tenia la capacidad de procesamiento propia. La única capacidad de procesamiento proporcionada por la terminal consistía en enviar pulsaciones de tecla al host y en mostrar datos al usuario.

Puesto que estas máquinas eran muy caras y los costos de mantenimiento eran igualmente altos, no tenia sentido para las grandes organizaciones centralizar la mayoría de sus datos. Con el tiempo, muchas organizaciones comenzaron a encontrar que este entorno centralizado daba lugar a graves cuellos de botella en lo relacionado al procesamiento de aplicaciones como en el desarrollo de las mismas. Entre estas máquinas figuran:

1.1.1 Mainframe.

Los mainframes son grandes computadoras que poseen una gran rapidez y caros sistemas que son capaces de controlar al mismo tiempo a cientos o miles de usuarios así como cientos de dispositivos de entrada y salida. Su temperatura debe estar siendo controlada constantemente. Su costo puede ser de 350.000 dólares. Si se refiere al número de programas que puede soportar simultáneamente un mainframe es más poderoso que una supercomputadora, pero las supercomputadoras pueden ejecutar un solo programa más rápido que un mainframe.

Los primeros mainframes podían ocupar cuartos completos incluso pisos enteros de un edificio, sin embargo, hoy en día un mainframe es parecido a una hilera de archiveros (como los de una biblioteca) ubicados en un cuarto con un piso falso bajo el cual se ocultan una inmensa cantidad de cables correspondientes a los periféricos. Los mainframes poseen varios procesadores que ejecutan varias tareas a la vez. Por lo general cuentan con varias unidades de disco para procesar y almacenar grandes cantidades de información. A esta clase pertenecen la IBM 390, 430, etc.

1.1.2 Minicomputadoras.

Son la versión más pequeña de un mainframe. Están orientadas a tareas específicas por lo que no necesita de todos los periféricos que necesita un mainframe, con lo que se redujo el costo y el precio de mantenimiento en comparación con los anteriormente nombrados. En cuanto al tamaño y al poder de procesamiento se pueden ubicar entre los mainframes y las estaciones de trabajo. A grandes rasgos una minicomputadora es un sistema multiproceso que puede soportar de 10 hasta 200 usuarios simultáneamente. Fueron diseñadas principalmente para:

- Entornos de múltiples usuarios, apoyando múltiples actividades de proceso al mismo tiempo.
- Ofrecer ciertos servicios más específicos.
- Soportar un número limitado de dispositivos.
- Pequeñas y de bajo costo (en comparación a sus antecesoras).
- Para múltiples aplicaciones.

Actualmente se usan para almacenar grandes bases de datos, automatización industrial y para aplicaciones multiusuario.

1.1.3 Supercomputadoras

Las supercomputadoras son el tipo de computadoras más potentes y más rápidas que existen en un momento dado. Son de gran tamaño, las más grandes entre sus pares. Pueden procesar enormes cantidades de información en poco tiempo, ejecutando millones de instrucciones por segundo, están destinadas a una tarea específica y poseen una capacidad de almacenamiento muy grande. Además son las más caras teniendo un costo que puede superar los 30 millones de dólares. Por su alto costo se fabrican muy pocos durante un año, incluso existen algunas que se fabrican solo por pedidos. Cuentan con un control de temperatura especial para poder disipar el calor que algunos de sus componentes pueden llegar a alcanzar.

Actúa como árbitro de todas las solicitudes y controla el acceso a todos los archivos, lo mismo hace con las operaciones de entrada y salida. El usuario se dirige a la computadora central de la organización cuando requiere apoyo de procesamiento.

Están diseñadas para sistemas de multiprocesamiento, el CPU es el centro del procesamiento y pueden soportar a miles de usuarios en línea. La cantidad de procesadores que puede llegar a tener una supercomputadora depende principalmente del modelo, pueden tener desde alrededor de 16 procesadores hasta 512 (como el modelo SX-4 de NEC de 1997) y más. Como pertenecientes a la clase de las supercomputadoras se pueden nombrar: La CRAY 1, Cyber, Fujitsu ,etc. Más recientemente la supercomputadora SGI encargado por la NASA a SILICON GRAFICS, con un número de 251 procesadores.

Algunos de los usos de las supercomputadoras son:

- Búsqueda y estudio de la energía y armas nucleares.
- Búsqueda de yacimientos de petróleo con grandes bases de datos sísmicos.
- El estudio y predicción de los tornados.
- El estudio y predicción del clima en cualquier parte del mundo.
- La elaboración de maquetas y proyectos de la creación de aviones, simuladores de vuelo, etc.

Cuando los precios de la computadora personal (PC) fueron accesibles, comenzó a tener sentido utilizar la potencia de procesamiento inherente a la PC para descargar de trabajo al host, creándose así la ***arquitectura de redes de área local basadas en PC***. Muchos usuarios empezaron a utilizar su PC para realizar diversas operaciones que antes se llevaban al cabo en el host. Las primeras operaciones serias sobre PC fueron las hojas de cálculo, que permitieron realizar buena parte de las operaciones numéricas y cálculos. Mas adelante, hicieron su aparición las bases de datos basadas en el sistema de archivos, como DBASE y FoxPro, que se extendieron enormemente. Con ellas, los usuarios fueron capaces de crear sus propias aplicaciones dirigidas por bases de datos.

En esta arquitectura, el nivel de presentación y el nivel de lógica de negocios residen típicamente en una PC local. El nivel de acceso a los datos reside en otra máquina situada dentro de la red del área local, posiblemente un servidor de archivos de red.

1.1.4 Workstations o Estaciones de Trabajo (WS).

Entre las minicomputadoras y las PC's (en términos de potencia de procesamiento) se encuentra una clase de computadoras conocidas como estaciones de trabajo. Una estación de trabajo se parece a una computadora personal y generalmente es usada por una sola persona, al igual que una PC, aunque una estación de trabajo es mucho más poderosa que una computadora personal promedio. Internamente, las estaciones de trabajo están construidas de una forma distinta a la de las PC's. Están basadas generalmente en otro tipo de diseño de CPU llamado RISC (procesador de cómputo con un conjunto reducido de instrucciones), con lo que se obtiene un procesamiento más rápido de las instrucciones

Están diseñadas para apoyar a una red permitiendo a los usuarios compartir archivos, programas de aplicaciones y hardware, como por ejemplo las impresoras. Las estaciones de trabajo son usadas para:

- Aplicaciones de ingeniería.
- CAD (diseño asistido por computadora).
- CAM (manufactura asistida por computadora).
- Publicidad.
- Creación de software.

Las bases de datos de sistemas de archivos funcionan bien para aplicaciones aisladas y cuando son usadas localmente en la PC. No estan, sin embargo, bien adaptadas a los entornos multiusuario. La utilización de estas bases de datos de sistemas de archivos sobre una red de área local, para proporcionar acceso compartido a los datos, comenzó a generar una carga creciente de tráfico en la red y demostró ser incapaz de aumentar de escala para construir las grandes aplicaciones de tipo empresarial que son necesarias para llevar un negocio. La solución a este problema radica en un avance tecnológico.

1.1.5 Computadora Personal (Pc).

Las PC's son las computadoras más accesibles para cualquier tipo de usuario, son computadoras personales, de escritorio, de un bajo costo y que pueden ser usadas para múltiples aplicaciones. Son de tal flexibilidad que pueden ser usadas tanto como por un experto en el trabajo como por un niño en casa. En un principio solo podían ser usadas en ambientes monousuario, pero con los avances tecnológicos ya pueden ser utilizadas en ambientes multiusuario e incluso como servidores de una red de computadoras. Hoy en día se pueden encontrar en oficinas, escuelas, hogares, etc.
Las PC's tuvieron su origen con la creación de los microprocesadores por parte de la compañía IBM que más tarde se estandarizó con lo que otras compañías comenzaron a fabricarlos.

Entre los tipos de PC's se pueden nombrar:

- Hand-held
- Palmtop
- Notebook
- Laptop
- Pen computers
- PDA ("personal digital assistant")
- Desktop
- Tower

En un gran avance, la llegada de los sistemas de gestión de bases de datos relacionales fue, realmente, la tecnología clave que hizo posible el modelo cliente/servidor.

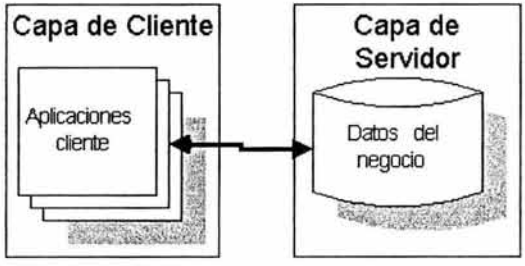
1.2 El Modelo Cliente/Servidor.

En términos generales, la comunicación entre computadoras se lleva a cabo en un sistema cliente/servidor, (algo así como emisor/receptor), donde por cliente se conoce a la computadora que solicita la información y servidor a la que proporciona la información.

En el sentido más estricto, el término cliente/servidor describe ***un sistema en el que una máquina cliente solicita a una segunda máquina llamada servidor que ejecute una tarea específica.***

El programa cliente cumple dos funciones distintas: por un lado ***gestiona la comunicación con el servidor, solicita un servicio y recibe los datos enviados por aquél.*** Por otro, ***maneja la interfaz con el usuario: presenta los datos en el formato adecuado y brinda las herramientas y comandos necesarios para que el usuario pueda utilizar las prestaciones del servidor de forma sencilla.*** El programa servidor en cambio, básicamente ***sólo tiene que encargarse de transmitir la información de forma eficiente, NO TIENE QUE ATENDER AL USUARIO.*** De esta forma un mismo servidor puede atender a varios clientes al mismo tiempo.

En el modelo cliente/servidor existen dos capas, el cliente y el servidor: éste está ubicado normalmente en otra máquina, y suele ser un gestor de base de datos, como DB2, SQL Server, Mysql, Oracle, aunque también puede ser una base de datos más pequeña, como Paradox, dBase, etc., que accedemos directamente desde nuestra aplicación cliente.



El modelo cliente/servidor es realmente una combinación de las mejores funciones del entorno basado en host y del entorno de red de área local basada en PC. Este modelo utiliza la potencia de la PC para realizar la presentación de los datos, junto con el complicado procesamiento relativo a la lógica de negocio que añada valor a dichos datos

Al utilizar el término **servidor** no solo nos referimos al software que realiza ciertas tareas en nombre de los usuarios. El mismo término es también utilizado para referirse a ***la computadora en la cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos*** [1]

Este **uso dual** puede llevar a confusión. Por ejemplo en el caso de un servidor web, este término podría referirse a la máquina que almacena y maneja los sitios web, y en este sentido es utilizada por las compañías que ofrecen hosting u hospedaje. Alternativamente, el servidor web podría referirse al software, como el servidor de HTTP de Apache, que funciona en la máquina y maneja la entrega de los componentes de las páginas Web como respuesta a peticiones de los navegadores de los clientes. Los archivos para cada sitio de Internet se almacenan y se ejecutan en el servidor. Hay muchos servidores en Internet y muchos tipos de servidores, pero comparten la función común de proporcionar el acceso a los archivos y servicios

Una misma computadora servidor puede ofrecer varios servicios a la vez, como puede ser paginas Web, acceso a archivos, e-mail, etc. Cada uno de estos servicios se envía por un canal de comunicación diferente, que se llama puerto. Los puertos son los accesos a los diferentes servicios que ofrece el servidor. Los puertos 0001 al 1024 están reservados para servicios ya conocidos o abiertos por el administrador, por ejemplo, al navegar por Internet se tiene acceso a la red por el puerto 0080 y al bajar archivos se utiliza el puerto 0021.

[1] Cliente/Servidor 2da, edición, ORFALI, HARLEY, EDWARS, McGraw Hill, 2001.

A veces, hay administradores de red que combinan la ubicación de los servicios para así evitar posibles ataques, pero en general, estos puertos ya estan definidos para sus determinados servicios. El administrador puede mantener una lista de nombres y contraseñas de las personas a las que se les permite utilizar los recursos, y el servidor deja que sólo esas personas tengan acceso

Como su nombre implica, un servidor sirve información a las computadoras que se conecten a él. Cuando los usuarios se conectan a un servidor pueden acceder a programas, archivos y otra información del servidor.

Por las operaciones que realizan y los servicios que ofrecen, se clasifican en [2]:

- **Servidores de Comunicaciones**: Realizan todas las operaciones de comunicación requeridas por los usuarios.
- **Servidores de Impresión**: Administra las colas de impresión.
- **Servidores de Bases de Datos**: Maneja la Administración de una Base de Datos común.
- **Servidores de Correo**: Distribuyen el correo electrónico.
- **Servidores Proxy**: Algunas veces se hace referencia a ellos con el nombre de *"gateway"* (puerta de comunicación) o *"forwarder"* (agente de transporte). Es una aplicación que se utiliza a menudo, como sustituto de routers controladores de tráfico, para prevenir el tráfico que pasa directamente entre las redes
- **Servidores Web:** Usan el protocolo HTTP para satisfacer las solicitudes de información que realizan los usuarios finales a través de sus navegadores y presentar páginas de información escrita utilizando el lenguaje HTML.

Los servidores web, servidores de correo y servidores de bases de datos son a lo que tienen acceso la mayoría de la gente.

[2] Cliente/Servidor 2da, edición, ORFALI, HARLEY, EDWARS, McGraw Hill, 2001.

Los servidores se conectan a la red mediante una interfaz que puede ser una red verdadera o mediante conexión vía línea telefónica o digital.

Los servidores pueden ser **dedicados o no dedicados**. Un servidor dedicado es aquel que ***sólo es usado como un servidor de archivos***. Un servidor no dedicado, además de actuar como un servidor de archivos, también ***puede ser usado como una estación de trabajo***. Los servidores dedicados son más estables y tienen mejor rendimiento que los no dedicados debido a que el tiempo del procesador no se tiene que dividir para atender otras tareas. Cabe resaltar que los servidores no dedicados prácticamente han dejado de ser utilizados. .

Los servidores más avanzados añaden seguridad para permitir una conexión encriptada entre el servidor y el navegador así la información de suma importancia como números de tarjetas de crédito pueda ser enviada por Internet.

Aunado al modelo cliente/servidor, podemos mencionar dos nuevas tecnologías, los clusters y los GRID. En este trabajo vamos a explicar a fondo todo lo relacionado con los clusters, así como explicar las características a considerar para poder instalar una base de datos paralela en este.

2.0 CLUSTERS.

El uso de las computadoras dentro de nuestra sociedad ha venido desarrollándose desde su aparición en 1945, cuando comenzaba la era de las computadoras modernas.

Con el actual ritmo de crecimiento del comercio y el movimiento de datos de todo tipo en *Internet* 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 ininterrumpido y fiable se le conoce como *alta disponibilidad.*

Existen gran cantidad de servidores en el mercado fabricados por SUN, IBM y demás empresas del ramo. Son grandes máquinas multiprocesador, con varios controladores de disco, configuraciones RAID, fuentes de alimentación redundantes, y un largo etcétera de circuitos y controladores duplicados para que, en caso de fallo, haya alguna de respaldo. El precio de este tipo de equipos es muy alto. Además, cuando una máquina de este tipo queda obsoleta, no nos queda otro remedio que comprar otra mayor y deshacernos de la antigua.

2.1 Antecedentes

El desarrollo de las redes de alta velocidad y la aparición de la computadora personal (PC), ha permitido romper la barrera de los precios y poder construir eficientes clusters de computadoras a un costo mínimo, que proveen un desempeño comparable a las supercomputadoras. Debido a que la PC como las redes de alta velocidad son de uso común, permiten a la mayoría de las organizaciones comerciales, gobiernos e instituciones educativas tener acceso a supercomputadoras de alto desempeño.

En muchas ramas de las ciencias, la complejidad de los problemas que se estudian requieren contar con acceso a una supercomputadora. Las supercomputadoras tradicionales emplean procesamiento en paralelo; contienen arreglos de microprocesadores ultrarrápidos que trabajan en sincronía para resolver problemas complejos. Los fabricantes de supercomputadoras como Cray, IBM, Silicon Graphics, entre otros, producen modelos por diseño especial y cuyos precios van más allá de los presupuestos de inversión de los grupos de investigación.

En los últimos años, el personal académico de diversas universidades y centros de investigación se han dado a la tarea de aprender a construir sus propias supercomputadoras conectando computadoras personales y desarrollando software para enfrentar tales problemas extraordinarios.

En el verano de 1994, bajo el patrocinio del Proyecto de la Tierra y Ciencias del Espacio (ESS), Thomas Sterling y Don Becker, trabajando en CESDIS (Center of Excellence in Space Data and Information Sciencies), construyeron un Cluster de Computadoras que consistía de 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. El cluster de PCs desarrollado tuvo una eficiencia de 70 megaflops (millones de operaciones de punto flotante por segundo). Los investigadores de la NASA le dieron el nombre de *Beowulf* a este cluster, en honor del héroe de las leyendas medievales, quien derrotó al monstruo gigante Grendel.

Este Beowulf construido en la NASA en 1994 fue el primer cluster de la historia, y su finalidad era el cálculo masivo de datos. Desde entonces, la tecnología de clusters se ha desarrollado enormemente, apareciendo gran cantidad de estudios, teorías, programas y arquitecturas implantando clusters para diversos fines.

Los clusters permiten aprovechar los recursos; las tecnologías de hardware y software que fueron desarrolladas para aplicaciones amplias y mercados comerciales pueden también servir en el área del cómputo de alto rendimiento.

El proyecto Beowulf fue exitoso por haber demostrado que se podían usar máquinas de bajo costo para hacer computo en paralelo, es decir, la relación costo/rendimiento era menor que las computadoras estrechamente acopladas. Los clusters Beowulf se distinguen porque **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.

Fue este aspecto de los clusters lo que inicialmente los hizo posibles y que disparó la primera ola de actividad en el campo. Tanto las redes de estaciones de trabajo como los clusters de PCs tipo Beowulf fueron posibles porque no requerían desarrollos largos y caros previos a su uso inicial. Esos primeros sistemas estaban lejos de la perfección pero eran útiles. Aún estos primeros clusters exhibían una ventaja precio-rendimiento respecto a las supercomputadoras contemporáneas que se acercaba a un factor de 50% en casos especiales mientras que el rendimiento sostenido por nodo para aplicaciones reales era de un factor entre 3 y hasta 50% de los sistemas costosos con el mismo número de procesadores. Pero el rápido mejoramiento en el desempeño de los microprocesadores de las PCs y los avances en las redes locales han llevado a los sistemas a un rendimiento de decenas y aún cientos de Gigaflops mientras mantienen excepcionales beneficios precio-rendimiento como se reconoció en los premios Gordon Bell 1997 y 1998 para Precio-Rendimiento que ganaron los clusters de PCs.

Durante los años 90's empezaron a proliferar los clusters, y para el año 2000 surgieron los primeros superclusters que actualmente compiten con las supercomputadoras.

2.2 Definición:

Originalmente, el término cluster se utiliza para catalogar a los objetos más grandes que se encuentran en el universo, ya sea galaxias o constelaciones.

El término CLUSTER de computadoras se aplica a:

..... *"los conjuntos o conglomerados de computadoras, construidos utilizando componentes de hardware comunes y software"*, [3]..... los cuales juegan hoy en día, un papel importante en la solución de problemas de las ciencias, las ingenierías y aplicaciones comerciales.

..... *"un conjunto de computadoras independientes interconectadas, usadas como un recurso unificado de cómputo"*. [4].....

..... *"sistema de cómputo local que consta de un conjunto de computadoras independientes y una red interconectándolas. Un cluster es local en tanto que todos los subsistemas componentes son supervisados por medio de un solo dominio administrativo, usualmente residiendo en un solo espacio y manejado como un solo sistema de cómputo."* [5].....

Podemos definir a un cluster de computadotas como ***"un grupo de computadoras de bajo costo, unidas por una red de alta velocidad (o un switch), que se utiliza para procesar grandes cantidades de datos en paralelo".*** En lugar de resolver el problema con un solo procesador ultra rápido, este se divide en partes más pequeñas que se resuelven independientemente en procesadores comunes de bajo costo.

Reuniendo un número suficientemente grande de procesadores, la cantidad total de operaciones y toda la RAM acumulada pueden hacer de un cluster una solución competitiva a una fracción del costo de una supercomputadora de igual desempeño.

---

[3] http://clusters.fisica.uson.mx/clusters_de_Linux.htm

[4] http://clusters.unam.mx/

[5] http://clusters.unam.mx/

La arquitectura de cluster mas conocida es la Beowulf, utilizada por docenas de universidades del mundo con necesidad de supercomputadoras pero con presupuestos bajos.

Los clusters han evolucionado para apoyar actividades en aplicaciones que van desde súper cómputo y software de misión crítica, servidores Web y comercio electrónico hasta las bases de datos de alto rendimiento.[6]

Entre las aplicaciones más comunes de clusters se encuentra el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, análisis de imágenes, y más.

El cómputo en clusters surge como resultado de la convergencia de varias tendencias que incluyen, la disponibilidad de microprocesadores de alto rendimiento más económicos y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, y la creciente necesidad de potencia computacional para aplicaciones en las ciencias computacionales y comerciales.[7]

En general, podríamos decir que hay dos tipos de clusters[8], atendiendo a su finalidad:

A. **Clusters para el procesamiento masivo de datos (Alto Rendimiento)**:
   El ejemplo más claro de este tipo es el Proyecto Beowulf, del que ya hemos hablado. Este tipo de clusters, por lo general, aprovechan la posibilidad de paralelización de cierto tipo de operaciones matemáticas (en especial, el cálculo matricial) para repartir los datos entre todos los equipos del cluster y poder así operar varios grados de magnitud más rápido. Para este fin se utilizan librerías como las PVM (Parallel Virtual Machine), que facilitan la distribución de datos entre las máquinas, incluso entre máquinas con distintos sistemas operativos, arquitecturas y lenguajes de programación.

---

[6] http://clusters.fisica.uson.mx/clusters_de_Linux.htm

[7] http://clusters.fisica.uson.mx/clusters_de_Linux.htm

[8] http://www.bisente.com/documentos/clustering/memoria.html

Otro ejemplo de cluster de este tipo sería el caso de MOSIX, unos parches para el núcleo del Sistema Operativo Linux con los que se consigue poder utilizar de forma transparente toda una red de equipos como si fuera una única supercomputadora, permitiendo el migrado transparente de cara al usuario de procesos de una máquina a otra y la compartición de recursos.

Toda la teoría sobre este tipo de clusters se centra en cómo compartir los recursos de procesador, memoria y/o red entre los equipos que forman el cluster para obtener un mejor rendimiento general.

Entre los clusters de alto rendimiento, según el último conteo realizado por clusters@top500.org, encontramos entre los primeros 10 lugares los siguientes[9]:

Rank	Manufacturer Computer/Procs	R_{max} R_{peak}	Installation Site Country/Year	Inst. type Installation Area	Nmax Nhalf	Computer Family Computer Type
2	**Hewlett-Packard** ASCI Q - AlphaServer SC ES45/1.25 GHz/ 8192	**13880.00** 20480.00	**Los Alamos National Laboratory** USA/2002	**Research**	**633000** 225000	**Compaq AlphaServer** Al.Se.-Cluster
3	**Linux Networx** MCR Linux Cluster Xeon 2.4 GHz – Quadrics/ 2304	**7634.00** 11060.00	**Lawrence Livermore National Laboratory** USA/2002	**Research**	**350000** 75000	**NOW - Intel Pentium** NOW Cluster - Intel Pentium -
6	**IBM** xSeries Cluster Xeon 2.4 GHz – Quadrics/ 1920	**6586.00** 9216.00	**Lawrence Livermore National Laboratory** USA/2003	**Research**	**425000** 90000	**NOW - Intel Pentium** xSeries Cluster Xeon - Quadric
8	**Hewlett-Packard** rx2600 Itanium2 1 GHz Cluster – Quadrics/ 1540	**4881.00** 6160.00	**Pacific Northwest National Laboratory** USA/2003	**Research**	**550000** 110000	**HP RX2600 Itanium Cluster** HP rx2600 Itanium2 Cluster

[9] http://www.clusters@top500.org

9	**Hewlett-Packard** AlphaServer SC ES45/1 GHz/ 3016	**4463.00** 6032.00	**Pittsburgh Supercomputing Center** USA/2001	**Academic**	**280000** 85000	**Compaq AlphaServer** Al.Se.-Cluster
10	**Hewlett-Packard** AlphaServer SC ES45/1 GHz/ 2560	**3980.00** 5120.00	**Commissariat a l'Energie Atomique (CEA)** France/2001	**Research**	**360000** 85000	**Compaq AlphaServer** Al.Se.-Cluster

R_{max} and R_{peak} valores en GigaFlops

B. Clusters de alta disponibilidad:

En este caso lo que se busca no es exactamente conseguir una gran potencia de cálculo sino conseguir un conjunto de máquinas que todas realicen la misma función y que, ante el fallo de una de ellas, las demás puedan asumir sus tareas de una forma transparente y rápida. Por supuesto, la escalabilidad también es importante, ya que siempre podremos añadir más máquinas al cluster para así conseguir más potencia, pero el objetivo prioritario no es este si no la resistencia a cualquier fallo imprevisto.

Aquí lo que se busca con la replicación de máquinas es evitar los puntos únicos de fallo, del inglés SPF (Single Point of Failure), que serían aquellas máquinas imprescindibles para el correcto funcionamiento del servicio que queremos dar. Si únicamente tenemos una instancia de cada máquina de este tipo, se convierte en un SPF y ante cualquier fallo en este equipo, todo el cluster queda inutilizado.

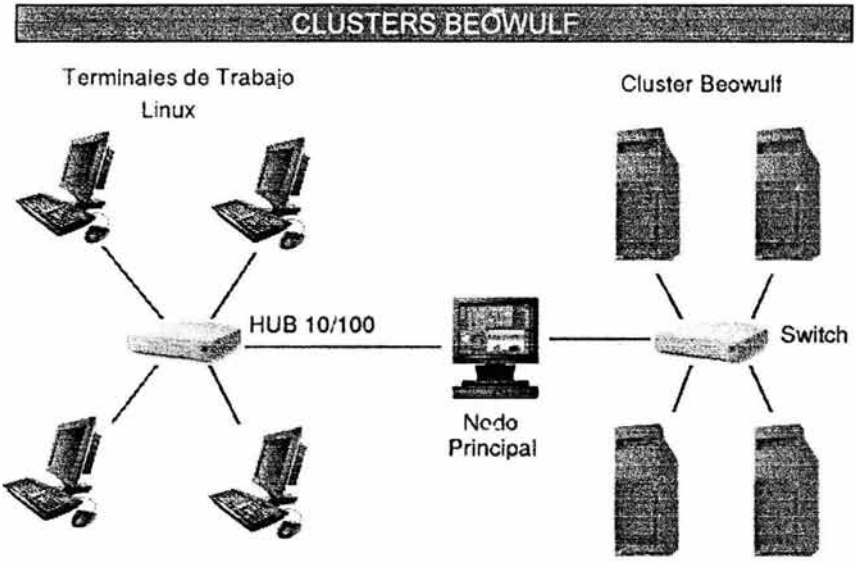
Como ejemplo de este tipo de clusters podemos mencionar:

Google.com. Es una compañía líder en indexar y desarrollo de nuevas tecnologías de la información del Internet. Algunas características importantes:

- Atiende en promedio más de 200 millones de consultas por día (2,315 consultas por segundo).
- 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).

- Opera un Cluster de más de 10,000 sistemas Linux.
- Manejo de una base de datos muy eficiente con cerca de 700 millones de noticias del Usenet (20 años de información y 1 TB de datos)

La teoría sobre este tipo de clusters gira en torno a estos SPF y cómo evitarlos, mediante redundancia del hardware y del software apropiado para controlar el correcto funcionamiento de todos los equipos y, en caso negativo, hacer que una máquina de respaldo suplante a la que acaba de fallar.



Los clusters de alta disponibilidad necesitan de un amplio abanico de componentes que consideren diversos factores, entre otros [10]:

- Control de miembros del cluster.
- Servicios de comunicaciones.
- Control y gestión del clustering, flujo de datos.
- Gestión y monitorización de recursos.
- Compartición o replicación del almacenamiento:
- Compartición:
- Discos SCSI externos.

[10] http://www.bisente.com/documentos/clustering/memoria.html

- Sistemas NAS.
- Sistemas de ficheros compartidos (NFS, SMB, Coda).
- Replicación:
- Propio de la aplicación (DNS, NIS, etc.)
- ftp, rsync, etc.

Todos estos detalles habrá que tenerlos en cuenta a la hora de diseñar el cluster y elegir el software que lo gestionará, ya que este software debe ser capaz por sí mismo de atender todos estos puntos de atención y reaccionar a tiempo ante un fallo en cualquiera de ellos.

Los beneficios de construir un cluster se verían reflejados en varios aspectos y 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

2.3 ¿Cómo funciona un Cluster?

En su parte central, la tecnología de clusters consta de dos partes. La primera; consta de un sistema operativo confeccionado especialmente para esta tarea (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.

La segunda es la interconexión de hardware entre las máquinas (nodos) del cluster. Se han desarrollado interfaces de interconexión especiales muy eficientes, pero comúnmente 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. Existe otra interfaz de red que conecta al cluster con el mundo exterior.

Cuando se habla de resolver un problema en paralelo, se refiere a dividir un trabajo en varias tareas que se pueden desarrollan en paralelo. Esto es lo que sucede en un Cluster de Alto Rendimiento y la solución de los problemas se termina más rápido. Por ejemplo, es más rápido pintar un edificio de 32 cuartos si se cuenta con una brigada de 8 pintores que pinten cada uno 4 cuartos, todos sincronizados a trabajar al mismo tiempo.

La tecnología de Clusters de Alto Rendimiento para Linux más conocida es el la tecnología Beowulf. Los servidores de un Cluster de Alta Disponibilidad normalmente no comparten la carga de procesamiento que tiene un Cluster de Alto Rendimiento. Tampoco comparten la carga de tráfico como lo hacen los Clusters de Balance de Carga. Su función es la de esperar listos para entrar inmediatamente en funcionamiento en el caso de que falle algún otro servidor. La característica de flexibilidad que proporciona este tipo de tecnología de cluster, lo hacen necesario en ambientes de intercambio intensivo de información.

Los Clusters de Alta Disponibilidad permiten un fácil mantenimiento de servidores. Una máquina de un cluster de servidores se puede sacar de línea, apagarse y actualizarse o repararse sin comprometer los servicios que brinda el cluster. Cuando el servidor vuelve a estar listo, se incorpora al cluster y se puede dar servicio al siguiente servidor del grupo.

Adicionalmente a los Clusters tipo Beowulf, existen las siguientes tecnologías similares:

MOSIX. Esta tecnología basada en Linux, permite realizar balance de carga para procesos particulares en un cluster. Trabaja como un Cluster de Alto Rendimiento en el sentido de que está diseñado para tomar ventaja del hardware más rápido disponible en el cluster para cualquier tarea que le sea asignada. Pero, lo realiza balanceando la carga de las varias tareas en varias máquinas. Una de las grandes ventajas de MOSIX es que no requiere la confección especial de software como lo requiere los clusters tipo Beowulf. Los usuarios ejecutan sus programas normalmente y el sistema MOSIX se encarga del resto

Otra tecnología de clusters es el paquete **PIRANHA**, que permite a los servidores Linux proveer alta disponibilidad sin la necesidad de invertir cantidades mayores en hardware. La tecnología de cluster esta basada completamente en software, donde los servidores se comunican en una red de alta velocidad. Se puede configurar para trabajar como Cluster de Alta Disponibilidad o de Balance de Carga.

El Piranha puede configurar un servidor de respaldo en caso de fallo de la contraparte. También puede hacer que el cluster aparezca como un servidor virtual

2.4 Componentes de los Clusters

Procesadores

Procesador RISC (Reduced Instruction Set Computer): Caracterizado por un conjunto pequeño de instrucciones simples, y con ejecución encadenada de ellas (*pipelined*).

Procesador CISC (Complex Instruction Set Computer): Esto corresponde a procesadores que son capaces de ejecutar un gran número de instrucciones predefinidas en lenguaje de máquina (del orden del centenar).

Compiladores

Compiladores y Paralelización Automática: El desarrollo de compiladores que pueden paralelizar automáticamente ha ido evolucionando durante más de una década, con desarrollos significativos en el análisis de dependencia de accesos de datos. Los compiladores son ahora capaces de paralelizar automáticamente programas en FORTRAN basados en manejo de arreglos y tener un rendimiento considerable en multiprocesadores de pequeña escala. Se han logrado también avances en los algoritmos de los compiladores para manejar la localidad de datos en los caches y en la memoria principal, en optimizar la orquestación de la comunicación basándose en las características de desempeño de una arquitectura. Sin embargo, los compiladores no son todavía capaces de paralelizar eficientemente programas más complejos, especialmente aquellos que hacen un uso substancial de apuntadores, pues la dirección que guarda un apuntador no se conoce durante la fase de compilación.

Lenguajes de Programación Paralela

MPI: Message Passing Interface. Biblioteca estándar para programación paralela en el modelo de intercambio de mensajes. En este estándar se han incluido los aspectos más relevantes de otras bibliotecas de programación paralela. Entre las ventajas de MPI se encuentra la disponibilidad de varios modos de comunicación, los cuales permiten al programador el uso de buffers para el envío rápido de mensajes cortos, la sincronización de procesos o el traslape de procesos de cómputo con procesos de comunicación.

Esto último reduce el tiempo de ejecución de un programa paralelo, pero tiene la desventaja de que el programador debe ser más cuidadoso para evitar la corrupción de mensajes. Dos de las principales distribuciones de MPI son: LAM/MPI y MPICH.

PVM: Parallel Virtual Machine. Creado en el verano de 1989 por el Oak Ridge National Laboratory y desarrollado, posteriormente, junto con la Universidad de Tennessee en los EUA. PVM es una biblioteca de envío de mensajes, totalmente libre, capaz de trabajar en redes homogéneas y heterogéneas, que hace uso de los recursos existentes en algún centro de trabajo para poder construir una máquina paralela de bajo costo, obteniendo su mejor rendimiento en "horas muertas".

PVM maneja transparentemente el ruteo de todos los mensajes, conversión de datos y calendarización de tareas a través de una red de arquitecturas incompatibles. PVM está diseñado para conjuntar recursos de cómputo y proveer a los usuarios de una plataforma paralela para correr sus aplicaciones, independientemente del número de computadoras distintas que utilicen y dónde se encuentren localizadas. El modelo computacional de PVM es simple y además muy general. El usuario escribe su aplicación como una colección de tareas cooperativas. Las tareas accesan los recursos de PVM a través de una biblioteca de rutinas. Estas rutinas permiten la inicialización y terminación de tareas a través de la red, así como la comunicación y sincronización entre tareas.

Los constructores de comunicación incluyen aquellos para envío y recepción de estructuras de datos así como primitivas de alto nivel, tales como emisión, barreras de sincronización y sumas globales.

Administración

Job (tarea o trabajo): Secuencia de operaciones, programa o comando requerido por un usuario para correr en un ambiente multiproceso.

Job serial: Caso especial de job donde hay un solo procesador en lugar de múltiples procesos paralelos

Job interactivo: Tarea en la cual el usuario ha iniciado una sesión en la computadora y corre trabajos sin utilizar un sistema de administración de tareas.

Batch job (Proceso en lote): Consiste en una tarea sometida a través de un sistema de administración de tareas y que corre cuando los recursos requeridos se encuentran disponibles

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.

Calendarización: Implica que un trabajo será despachado cuando sea satisfecha una condición de tiempo específica.

Ejecución por lotes: Forma de asegurar que un sistema de cómputo permanezca activo en horas no hábiles o de poca carga de trabajo. Mecanismo efectivo que permite obtener más productividad (*troughtput)* del sistema.

Redes / Conexiones

Switches: Es un conjunto de puertos de entrada, puertos de salida y una red cruzada interna (*crossbar*) que conecta cada entrada a cada salida, el *buffering* interno, y el control lógico para efectuar la conexión de entrada y salida en cada punto de tiempo. Generalmente el número de puertos de entrada es igual al número de puertos de salida, lo cual es llamado el grado del *switch.*

Ethernet: Protocolo de comunicación basado en el estándar IEEE 802.3 cuya velocidad de transmisión es de 10 Mbps. Requiere de 3 elementos básicos para su funcionamiento: un medio físico por el cual van las señales ethernet; un mecanismo de control para identificar el acceso de cada una de las interfaces ethernet hacia el bus y la secuencia de la "trama ethernet" en cuanto a bits.

Entre los medios físicos se destacan el cable coaxial en sus dos variantes (grueso y delgado), el par trenzado (UTP) y la fibra óptica. El par trenzado goza de más popularidad por ser más económico y confiable.

El mecanismo de control que asigna el acceso al bus ethernet es conocido como CSMA/CD (Carrier Sense Múltiple Access with Collision Detection), cuya sencilla explicación se basa en que el bus es compartido por todos los equipos conectados a él; cuando una interfase manda un paquete, si el medio (bus ethernet) se halla libre, entonces es transmitido. Una colisión ocurre cuando dos interfaces ethernet mandan al mismo tiempo y al mismo bus un paquete, es entonces cuando se vuelve a reenviar la información.

El Frame o Trama Ethernet es un conjunto de bits ordenados en distintos campos entre los que destaca el campo de la dirección física de 48 bits o MAC addreess; el campo de datos varía de longitud de 46 a 1500 bytes; 32 bits para identificar la IP (protocolo IP). Existen otros protocolos como ARP y RARP que tienen sus respectivos campos en la trama ethernet

Fast Ethernet: El protocolo Fast Ethernet es similar en cuanto a operación al descrito como ethernet y se basa de igual manera en el estándar IEEE 802.3. La principal novedad es el incremento en la velocidad de transmisión a 100 Mbps. Los medios normalmente utilizados son el par trenzado y la fibra óptica; debido al costo, es mayormente usado el par trenzado debido a su menor costo. Por fortuna, la mayoría de los fabricantes están considerando en sus productos (switches, concentradores, tarjetas de red, etc.) convivan con el ethernet tradicional. Cabe mencionar que para fast ethernet se consideró un mecanismo adicional para el control de acceso aparte del CSMA/CD, dicha novedad se basa en "prioridad por demanda". Esto funciona para las "tramas ethernet" y se hizo extensiva para las tramas de *token ring*. Su aplicación a los clusters es poca debido a los nuevos estándares existentes ya en el mercado.

Gigabit Ethernet: Es el protocolo de la familia ethernet que alcanza la velocidad de 1000 Mbps. El estándar correspondiente es el IEEE 802.3z. Los medios físicos que lo soportan son el par trenzado de categoría 5 y la fibra óptica unimodo y multimodo soportando los modos de full y half-duplex. Sigue utilizando el mecanismo de control de acceso de CSMA/CD. Ya es una buena opción para utilizarlo como *backbone* y para aplicaciones de ancho de banda grande al escritorio. La aceleración de la velocidad de transmisión fue resultado de mezclar los estándares 802.3 y el ANSI X3 T11 Fiber Channel.

Una adición de gigabit ethernet es conocida como "frame bursting" que consiste en permitir a una estación transmitir una ráfaga de frames sin ceder el control. El modo full duplex se está utilizando en mayor medida en los equipos de mayor capacidad como switches, grandes servers o routers, dicha capacidad da la opción de incrementar el ancho de banda a 2 Gbps para enlaces punto a punto y de eliminar el control de flujo CSMA/CD ya que no existen colisiones. Esta última característica es ideal para la conectividad en clusters.

FDDI (Fiber Distributed Data Interface): El estándar FDDI tiene su mayor uso en *backbone* y en conexiones de alta velocidad en redes de área local. La topología de FDDI es similar al token ring con la diferencia que se tiene con un doble anillo. El anillo primario es usado para la transmisión de datos y el secundario como respaldo. El medio de transmisión es la fibra óptica en sus dos modos (unimodo y multimodo). En cuanto a los tipos de trafico en FDDI existe el síncrono y el asíncrono (pueden convivir ambos tipos), el primero de ellos se utiliza para la transmisión continua de información como voz y video; el asíncrono permite manejar el ancho de banda en un esquema de 8 niveles de prioridad. El anillo secundario o de respaldo es el método para tolerar fallas en el sistema. Para su implementación, se cuenta con dispositivos conocidos como "bypass" los cuales hacen la función de aislar fallas y hacer uso del anillo de respaldo para que la comunicación no se pierda. A diferencia del ethernet, la unidad de comunicación es el token, que es quien lleva la información a través del FDDI, entregando y recibiendo paquetes. La trama del FDDI incluye al Token con 3 campos (preámbulo, delimitador de comienzo y el control de la trama). A continuación siguen los campos de dirección destino y dirección origen para seguir con el campo de datos.

Existe un campo que realiza la verificación de la información a través de un CRC (Cyclic Redundancy Check) y que en el FDDI es mejor conocido como el FCS (Frame Check Sequency). Finalmente están los campos que delimitan el fin de la trama y del estado de la trama.

SCI (Scalable Coherent Interface): Es un estándar que fue concebido para incrementar el ancho de banda de los buses en *blackplane*. Está basado en la tecnología de conexiones punto a punto que elimina muchos de los problemas físicos proporcionando velocidades mayores. SCI, de hecho, simula un bus, dando los mismos servicios que un bus (incluso otros que normalmente no se ofrecen) pero sin usar buses. La configuración básica de SCI es un anillo que contiene varios nodos conectados entre si por enlaces (links) unidireccionales punto a punto. Cada aplicación se conecta a SCI a través de un nodo, el cual es visto por la aplicación como un bus convencional. Los nodos, a su vez, se enlazan entre si con links punto a punto a velocidad muy alta ya que no tienen las limitaciones de los blackplanes.

Cada nodo está conectado al anillo SCI por un link (SCI IN) proveniente del nodo precedente y por otro link (SCI OUT) con destino al nodo siguiente y, a su vez, está conectado con la aplicación normalmente por el bus propio. Cuando un paquete que circule por el anillo llegue al nodo, el address decoder verá si el paquete es para ese nodo o no.

En caso de que si lo sea lo encaminará a la Fifo de entrada, en el caso contrario, lo será hacia la "bypass Fifo", o sea, hacia el anillo SCI. El codificador/decodificador de paquetes transformará el paquete entrante en el formato del bus y se usa para comunicar en ambos sentidos con la aplicación. Cuando se mande un paquete de salida a SCI, éste irá a la Fifo de salida y si no hay ningún paquete mandándose en ese momento, desde la bypass Fifo, el paquete será encaminado hacia el link de salida. Es un estándar que permite a sistemas crecer con componentes modulares de diferentes vendedores. El flujo de datos es de 1 Gbyte. El estándar SCI usa memoria compartida.

Herramienta

Depurador (Debugger): Software que permite revisar la ejecución instrucción por instrucción de un programa y se utiliza cuando se detecta un mal funcionamiento en el programa. Depurar programas paralelos es una tarea difícil debido a su naturaleza concurrente. Un depurador para programas paralelos debe proporcionar control sobre todos los procesos del programa, y además ser capaz de presentar la información de forma concisa.

Almacenamiento

RAID (Redundant Array of Inexpensive Disks):. En un cluster existe una gran cantidad de recursos, tradicionalmente cada nodo no puede acceder a la totalidad de recursos. Los procesos corriendo en un nodo sólo pueden acceder a los recursos locales. Antes para accesar a recursos remotos, el mecanismo no era simple ni transparente. Por ejemplo, para acceder a archivos en un disco remoto era a través de ftp.

La idea ahora es distribuir los datos entre los discos de tal forma que se puedan acceder, en lo posible, en varios discos al mismo tiempo. La primera vez que se uso esta idea no fue para un cluster sino para construir un "disco único" con un gran ancho de banda, conectando varios discos a un controlador y dar la impresión de que el "disco" tenía un más alto ancho de banda para transferencia de datos. Este tipo de discos se conoce como RAID.

2.5 Componentes de software

Mientras que los rápidos avances en la capacidad del hardware han colocado a los clusters a la vanguardia de la nueva generación de sistemas, igualmente importante han sido la capacidad evolutiva y la madurez de los sistemas y herramientas de software. El resultado es un ambiente de sistema completo que está convergiendo de generaciones previas de supercomputadoras. Y como estos predecesores, los clusters presentan una oportunidad para la investigación futura y desarrollo avanzado en herramientas de programación y software de manejo de recursos para ampliar su aplicabilidad, disponibilidad, escalabilidad y utilización.

Sin embargo, fuera de estos primeros tipos de sistemas, la amplia y creciente distribución de los clusters está fomentando un trabajo sin precedentes en el campo proveyendo una familia de arquitecturas convergente en la cual la comunidad con aplicaciones puede seguir confiando.

Por lo tanto, los componentes de software, tan necesarios para el éxito de cualquier tipo de sistema paralelo, están mejorando a ritmo acelerado y acercándose rápidamente a niveles estables y sofisticados de funcionalidad que colocarán a los clusters como soluciones duraderas en el cómputo.

Los componentes de software que un cluster comprende pueden ser descritos en dos grandes categorías: **las herramientas de programación y el software de manejo de los recursos del sistema**[11]. Las herramientas de programación proveen lenguajes, bibliotecas y depuradores de rendimiento y corrección para construir programas de aplicación paralelos.

El software de manejo de recursos relaciona la instalación inicial, la administración, la calendarización y la distribución de los componentes de hardware y software a las cargas de trabajo de los usuarios.

Ambientes de programación de aplicaciones

Asegurar el paralelismo en los programas de aplicación para alcanzar ganancias de órdenes de magnitud en el rendimiento ha sido un reto que el cómputo ha atacado desde los 70's. Las supercomputadoras vectoriales han explotado formas variables de algoritmos de paralelización a través de una combinación de mecanismos de hardware y software. Los resultados han sido variados. Los más altos grados de rendimiento alcanzado han sido a través de cómputo paralelo.

[11] http://clusters.fisica.uson.mx/proyectos/ white_paper/cap3.html

Pero en muchos casos, los rendimientos observados han sido bajos y las dificultades para alcanzarlos han sido grandes. Los clusters armados, debido a su ventaja superior en precio-rendimiento para un amplio rango de problemas, sufren ahora de gran presión para atacar los problemas que sus predecesores confrontaron. Mientras se han buscado múltiples modelos de programación, un paradigma ha surgido como la forma predominante, al menos a corto plazo. El modelo de "comunicación de procesos secuenciales" comúnmente llamado modelo de "envío de mensajes" ha evolucionado a través de varias implementaciones distintas, resultando un estándar de la comunidad: MPI o Interfase de envío de mensajes. MPI se encuentra ahora en prácticamente cualquier multiprocesador comercial. MPI no es un lenguaje completo sino una biblioteca que permite a los usuarios de C y FORTRAN acceder a las bibliotecas de envío de mensajes entre procesos concurrentes en los nodos de procesamiento separados pero interconectados.

Varias de las implementaciones de MPI estan disponibles con los vendedores de sistemas y grupos de investigación, proporcionando versiones de código abierto a través de distribuciones libres.

Para que la programación paralela sea efectiva requiere algo más que un grupo de instrucciones. Se necesita de ambiente y herramientas para entender el comportamiento operacional de un programa, para corregir los errores en el cálculo y mejorar el rendimiento. El estado de dichas herramientas para los clusters está aún en la infancia aunque muchos equipos han hecho un esfuerzo significativo.

Varios de los perfiladores de desempeño han sido desarrollados tomando diversas formas. Un buen ejemplo es el conjunto de herramientas incorporadas a la distribución de PVM.

El trabajo continúa pero se requiere de más investigación y ningún estándar verdadero ha surgido en la comunidad. Mientras que el modelo de envío de mensajes en general y MPI en particular domina la programación paralela en los clusters armados, otros modelos y estrategias son posibles y pueden ser superiores a largo plazo. La programación de datos paralelos ha sido soportada a través de BSP (Bulk Synchronius Parallel). El cálculo por medio de mensajes ha sido soportado por Split-C y Fast messages.

La programación para memoria distribuida compartida fue perseguida por el proyecto Shrimp y a través del desarrollo de UPC. Y para la comunidad comercial, un conjunto de herramientas alternativas han soportado la metodología convencional de esclavo-maestro. Es claro que hoy, en la programación para clusters hay un estándar efectivo y altamente portable para construir y ejecutar programas paralelos y que trabajos con posibles técnicas alternativas se llevan a cabo por grupos de investigadores que pueden producir modelos más efectivos y mejorados en el futuro.

Software de manejo de recursos

Mientras que un par de estándares fuertes han sido aceptados en la comunidad de la programación para los clusters armados, no se puede decir lo mismo de los ambientes de software y las herramientas necesarias para el manejo de los recursos, tanto de hardware como de software, que ejecutan los programas. Sin embargo, en los últimos dos o tres años un progreso sustancial ha sido hecho por un conjunto de productos disponibles de los grupos de investigación que empiezan a satisfacer algunos de los requerimientos más importantes. Las áreas donde se requiere soporte son diversas y reflejan los muchos retos en el manejo de los recursos de los clusters.

Calendarización y asignación

La asignación de las aplicaciones en los recursos distribuidos de los clusters armados requiere herramientas para asignar los componentes de software a los nodos y para calendarizar el momento de su operación. En su forma más simple, el programador realiza este trabajo manualmente. Sin embargo, para sistemas grandes, especialmente aquellos compartidos entre múltiples usuarios y con la posibilidad de trabajar con varios programas al mismo tiempo, son requeridos medios más sofisticados para un manejo robusto y disciplinado de los recursos de cómputo. La asignación puede ser realizada a diferentes niveles de granularidad de la tarea incluyendo: trabajos, transacciones, procesos o hebras. Estos pueden ser calendarizados estáticamente de manera que cuando una asignación se hace es conservada hasta la culminación de la tarea o dinámicamente permitiendo al sistema migrar, suspender o reiniciar tareas automáticamente para el mejor aprovechamiento del sistema balanceando la carga de trabajo.

Administración del sistema

La supervisión de sistemas requiere la realización muchas tareas mundanas pero esenciales que incluyen el manejo de cuentas de usuario, colas de trabajo, seguridad, respaldos, almacenamiento, registro de bitácoras, interfaces para el operador, shells de usuario y otras actividades de mantenimiento. Tradicionalmente, a los sistemas de cómputo sofisticados les faltan algunas o muchas de estas utilidades de soporte típicas de un sistema tipo servidor comercial.

Monitoreo y diagnóstico

La operación continua de sistemas que comprenden una gran cantidad de nodos requiere de herramientas de bajo nivel por medio de las cuales el operador pueda monitorear el estado, operación e integridad de todos los elementos del sistema. Dichas herramientas pueden ser tan simples como versiones distribuidas de comandos de Unix como ps y top a imágenes de GUI del sistema paralelo completo actualizado en tiempo real. Mientras que muchas de estas herramientas han sido desarrolladas por diferentes grupos, ninguna de ellas ha alcanzado el dominio de uso por la comunidad.

Disponibilidad

Muchos clusters operan por meses sin ningún cambio y sólo son apagados para alguna actualización de software. Pero especialmente durante el inicio de operaciones, las fallas tanto en hardware (mal funcionamiento) y/o software (instalación y configuración incorrecta) pueden causar fallos graves. Puede ser el caso de algunos sistemas clusters dedicados a una sola aplicación de corrida larga, con ejecuciones continuas de semanas o incluso meses. En todos estos casos, las herramientas de software para mejorar la robustez y maximizar el tiempo de operación se vuelven cada vez más importantes para la explotación práctica de los clusters. Las herramientas de *checkpoint* y reinicio son cruciales para la realización de corridas largas en sistemas imperfectos. Los diagnósticos rápidos, pruebas recursivas, detección de fallas, aislamiento y análisis de mantenimiento requieren de herramientas de software y un marco común para mantener a los sistemas en operación el mayor tiempo posible, reiniciar la operación rápidamente en respuesta a fallos y recobrar cómputos parciales para continuar la ejecución ante fallas.

2.6 Sistema Operativo.

El sistema operativo (SO) de un cluster reside en el corazón de cada nodo, igual que en un sistema convencional de escritorio. Cuando el usuario está abriendo archivos o iniciando procesos, el sistema operativo está omnipresente. Mientras que los usuarios pueden elegir el uso de diferentes paradigmas de programación o capas de middleware, el sistema operativo es casi siempre el mismo para todos.

Un sistema operativo moderno provee a los usuarios de dos servicios fundamentales:

- **Máquina Virtual**: Un programa y una interfase de usuario que oculta las obscuridades de usar/programar los diversos componentes de hardware de un sistema.
- **Recursos de Hardware entre usuarios y procesos ejecutables**: Uno de los más importantes recursos es el procesador. En un sistema operativo multitarea, la computadora desempeña múltiples tareas a través de la calendarización de trabajos usando diversos algoritmos ya calendarizados que permiten correr varios procesos concurrentemente en pseudo paralelo o paralelo.

El sistema operativo ideal debería ayudar siempre al usuario, y nunca estorbarle. Así, ayudaría al usuario a configurar el sistema para la ejecución óptima de su programa, proporcionando un conjunto de funciones consistente y bien orientado que le ofrezca la mayor cantidad de recursos de sistema posibles. Después de configurar el ambiente, es deseable que permanezca fuera del camino del usuario evitando cualquier cambio de contexto que consuma tiempo o configuraciones excesivas en las estructuras de datos para aplicaciones sensibles en rendimiento.

El ejemplo más común de esto es el envío de mensajes de alta velocidad, en el cual el sistema operativo coloca buffers de mensajes en la memoria DMA y después permite a la tarjeta de interfase de red, posiblemente guiada por las llamadas de mensajes a nivel de usuario, a transferir el mensaje directamente a la RAM sin la intervención del sistema operativo. Por otro lado, puede ser deseable que el sistema operativo ofrezca una amplia funcionalidad para seguridad, tolerancia a fallas y facilidades de comunicación, lo cual contrasta con la necesidad de rendimiento que es omnipresente.

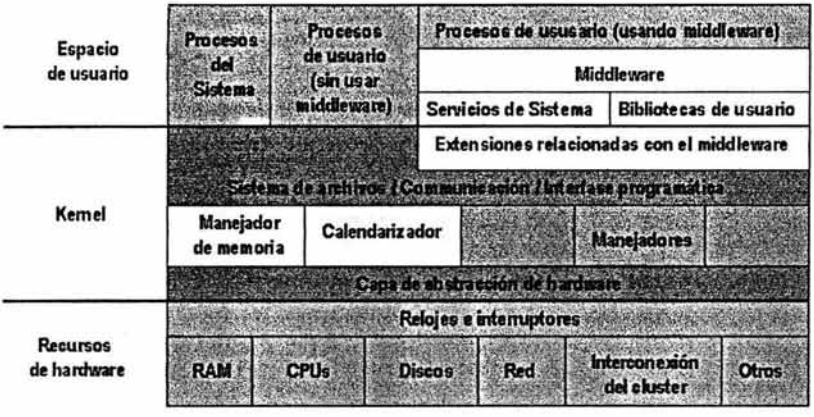
¿Qué atributos exactamente debería poseer un sistema operativo para clusters?, es todavía una pregunta abierta. Aquí se da una lista pequeña de las características deseables[12]:

- ***Manejabilidad***: Una necesidad absoluta es la administración remota e intuitiva del sistema; esto está asociado generalmente con una Imagen Única del Sistema (SSI) que puede realizarse a diferentes niveles, desde el conjunto de scripts especiales de alto nivel, tal vez controlado vía un *front-end* de Java, hasta un estado compartido real a nivel sistema operativo.
- ***Estabilidad***: Las características más importantes son la robustez ante la caída de procesos, la recuperación de fallas por medio reconfiguración dinámica y la funcionalidad bajo grandes cargas de trabajo.
- ***Rendimiento***: Las partes del sistema operativo críticas para el rendimiento, como manejo de memoria, calendarización de procesos y hebras, Entrada/Salida de archivos y protocolos de comunicación, deberían trabajar lo más eficientemente posible. El usuario y el programador deberían ser capaces de modificar de manera transparente los parámetros relevantes para sintonizar el sistema operativo para cumplir con sus demandas específicas.
- ***Extensibilidad***: El sistema operativo debería permitir una fácil integración de las extensiones específicas para clusters, las cuales estarán más enfocadas a la cooperación entre nodos. Esto implica, como mínimo, manejadores de dispositivos cargables por el usuario y documentación profunda de las interfases del kernel, así como en el espacio de usuario para el desarrollo de extensiones específicas. La mejor manera de proveer extensibilidad es probablemente la provisión del código fuente porque muestra todas las interfases y permite modificaciones a la funcionalidad existente (en lugar de tener que diseñar partes de reemplazo desde cero). Un buen ejemplo es el sistema MOSIX basado en Linux.

[12] Clusters Computing Theory, MORRISON Richard S. This document is distributed under the GNU General Public Licence

- ***Escalabilidad***: La escalabilidad de un cluster está influenciada principalmente por cuánto provean los nodos, lo que es dominado por las características de rendimiento de la interconexión. Esto incluye el soporte del sistema operativo para poder hacer uso del rendimiento potencial de la interconexión habilitando llamadas de baja sobrecarga para acceder a las interconexiones (escalabilidad entre nodos). Sin embargo, los clusters son generalmente construidos con nodos SMP con un número creciente de CPU's contenidos en cada nodo. La habilidad del sistema operativo de beneficiarse de éstos, está determinada por la escalabilidad intra nodo. La escalabilidad intra nodo está dominada por los calendarizadores de procesos y hebras y por el grado de paralelismo en el kernel que permite el sistema operativo. También incluye los límites de recursos, que un sistema operativo impone, para el tamaño máximo de espacio de direcciones del usuario.
- ***Soporte***: Muchas soluciones en cómputo, inteligentes y técnicamente superiores, fallan debido a la falta de soporte en varios aspectos: cuales herramientas, manejadores de hardware y ambientes de middleware están disponibles. Este soporte depende principalmente del número de usuarios de un sistema, que en el contexto de clusters está influenciado principalmente por los costos por hardware (ya que usualmente docenas de nodos serán instalados). Adicionalmente está el soporte para la interconexión del hardware: disponibilidad de interfases abiertas o aun código abierto; el soporte al menos demanda de la industria financiamiento para motivar la investigación y desarrollo. Todo esto lleva a la comunidad de usuarios que emplea el middleware requerido, ambientes y herramientas a, por lo menos, habilitar aplicaciones para clusters.
- ***Heterogeneidad***: Los clusters proporcionan un ambiente dinámico y evolutivo en el cual pueden ser extendidos o actualizados con hardware estándar conforme las necesidades o posibilidades del usuario. Por tanto, un ambiente de cluster no consiste necesariamente de hardware homogéneo. Esto requiere que el mismo SO deba correr en múltiples arquitecturas.

El sistema operativo es un factor crítico en el desempeño de sistemas paralelos. En todos los sistemas Beowulf, se requiere una capa de software llamada middleware. Middleware maneja código programado, compilación y ejecución paralela. Las últimas generaciones de sistemas operativos han surgido con la capacidad de manejar paradigmas de programación paralela con la integración de funciones middleware dentro del sistema operativo.



Bosquejo de un sistema operativo genérico para un nodo típico de un cluster

Algunos SO's tienen una capa independiente para abstraer el hardware del kernel. Portar un SO como éste a un hardware diferente requiere sólo de adaptar dicha capa de abstracción. Sobre esta capa, van las funcionalidades básicas que ejecuta el kernel: manejo de memoria, calendarización de procesos y hebras y manejadores de dispositivos, sólo por mencionar las más importantes. Los servicios en turno son ofrecidos a los usuarios por el sistema de archivos, las interfases de comunicación y una interfase general de programación para acceder a la funcionalidad del kernel de una manera segura y protegida.

Los procesos del usuario (aplicaciones) corren en el espacio de usuario, junto con los procesos de sistema. Especialmente en los clusters, los procesos del usuario utilizan muchas veces no sólo las funciones del kernel, sino también la funcionalidad adicional que proporciona el middleware. Esta funcionalidad está localizada usualmente en las bibliotecas de usuario, apoyada muchas veces por servicios de sistema adicionales. Algunas extensiones de middleware requieren de la funcionalidad extendida del kernel, la cual se alcanza usualmente cargando manejadores especiales o módulos al kernel.

Entonces, ¿cuál es el papel del sistema operativo en un cluster? El papel primario es *cumplir las mismas tareas de un sistema de escritorio: multiplexar los procesos de los múltiples usuarios en un solo conjunto de componentes de hardware (manejo de recursos) y proveer abstracciones útiles para el software de alto nivel.* Algunas de estas abstracciones incluyen protección de límites, coordinación de procesos/hebras, y comunicación, así como manejo de dispositivos

Tres de los sistemas operativos más usados (Microsoft Windows, Red Hat Linux y Sun Solaris) han incluido un nivel en el kernel capaz de soportar programación paralela.

La solución más común que los clusters actuales están usando es un sistema operativo convencional, con pocas modificaciones o al menos ninguna en especial. Este sistema operativo es usualmente un derivado de Unix (Linux) principalmente por las siguientes razones[13]:

- Es libre de pago.
- Es un sistema operativo de código abierto, lo que significa que uno es libre de personalizar el kernel al gusto. Hoy en día, esto ha significado manejadores especializados para las redes de gran velocidad involucradas y otras optimizaciones de E/S
- Por razones históricas: Don Becker seleccionó Linux para el cluster Beowulf original (esto fue por razones técnicas y sociales, ya que el kernel de Linux estaba libre de cualquier problema de licencias), y así los sistemas derivados del Beowulf también han usado Linux.

[13] Clusters Computing Theory, MORRISON Richard S. This document is distributed under the GNU General Public Licence

- Una colección de herramientas, bibliotecas de middleware, manejadores de red y extensiones de kernel que habilitan a un cluster para propósitos específicos de alto rendimiento.

2.7 Paralelismo y Computación.

Con el continuo desarrollo de la tecnología y de la metodología de la programación, el dominio de los sistemas informáticos se abre cada vez más. La definición de estos sistemas plantea numerosos problemas, entre ellos podemos citar la gestión de la memoria común y de las memorias locales de los diversos procesadores, la asignación de recursos físicos y virtuales definidos en el sistema, la gestión de los ficheros, la protección y la gestión de la concurrencia. En efecto, resulta frecuente que la gestión de los procesos y de su concurrencia tenga que llevarse a cabo en un marco autentico de paralelismo. Los numerosos problemas suscitados por la gestión de un conjunto de procesos paralelos hacen necesario el control mediante un sistema, ya sea sistema operativo, sistema de base de datos..., de acuerdo con la finalidad de la entidad construida.

El concepto de paralelismo supone la introducción de varios procesadores para resolver un problema. Sabemos que un procesador diez veces más potente que un procesador de potencia normal para una fecha es mucho más caro que diez procesadores de potencia normal para dicha fecha. Por ello, si paralelizamos nuestro programa; es decir, dividimos la carga computacional entre varios procesadores distintos, vamos a obtener una mejora en la relación entre costo y rendimiento. Con menos inversión en hardware estamos obteniendo mucha más potencia computacional.[14]

Sin embargo, existen varios factores tanto de índole computacional como de índole puramente física (las comunicaciones entre procesadores son lentas en comparación con la velocidad de transmisión de datos y de cómputo dentro de un procesador) que hacen que nunca obtengamos un escalado en el rendimiento igual o superior que el escalado en el número de procesadores, es decir, diez procesadores no van a ir diez veces más rápido que un solo procesador.

[14] Clusters Computing Theory, MORRISON Richard S. This document is distributed under the GNU General Public Licence

No obstante, el incremento de velocidad se aproxima al número de procesadores que se integran en el sistema en determinados algoritmos con una paralelización intrínseca muy fuerte.

Además, hay gran cantidad de científicos trabajando para acercar el límite práctico al límite teórico, por lo que, a los avances en hardware, hemos de sumar los avances que se producen en algoritmia.

A todo esto hay que añadir que el factor costo sigue siendo determinante para optar por soluciones paralelas aunque no sean óptimas, ya que, aunque no obtengamos el rendimiento máximo teórico, un incremento lineal en la potencia del procesador de una máquina implica un crecimiento exponencial del precio de una máquina, mientras que, con el incremento del número de procesadores, el incremento del precio es mucho menor.[15]

Algunos ejemplos de aplicaciones en las que el paralelismo es de gran utilidad son:

- Los grandes sistemas gestores de bases de datos,
- Operaciones de búsqueda y proceso de datos a gran escala -el denominado *data minning*-; este problema tiene grandes implicaciones tanto en ciencia de punta como en el proyecto genoma humano, como en análisis de datos estadísticos para problemas de bolsa y de análisis de mercados, es decir, mucho dinero en juego.
- Algoritmos de cálculo numérico pesados.
- Los gráficos por ordenador a gran escala. Aunque este problema realmente es una especialización de los algoritmos de cálculo numérico pesados, problemas como generación de imágenes realistas o efectos especiales son muy pesados en cálculos.
- Problemas de simulación. Estos problemas son de gran interés para la industria, ya que ahorran gran cantidad de dinero en experimentos de campo que no necesitan ser realizados por poder ser analizados en etapas de desarrollo del producto mediante complejas simulaciones por computadora.

---

[15] Algoritmica del paralelismo, RAYNAL Michael, Ed. Omega

- Predicción de fenómenos naturales, como una especialización del campo anterior de gran interés en la actualidad. Tanto fenómenos sísmicos, como grandes fenómenos naturales -inundaciones, tornados, sequías- son centro de estudio en la actualidad, y una parte muy importante de las supercomputadoras paralelas son empleados en la actualidad ya sea parcial o totalmente para este tipo de tareas.

De hecho, máquinas como el Beowulf original fueron desarrolladas para realizar complejas simulaciones atmosféricas.

- Problemas de optimización. Si bien algunos métodos de optimización son bastante complejos de paralelizar, los problemas de optimización son lo suficientemente pesados computacionalmente como para que sea de interés su paralelización. Algunos de estos problemas son el diseño de estructuras, problemas de gran interés en la ciencia actual. Nuestro problema en cuestión es de optimización y puede encuadrarse dentro de esta categoría, aunque el método de optimización empleado, los algoritmos genéticos, es inherentemente paralelo y por lo tanto es interesante enfrentarnos a su paralelización.

El paralelismo es una herramienta cada vez más utilizada en instituciones académicas, empresas e industria, para la resolución de problemas que requieren alto desempeño. Sin embargo, es un tema de estudio que no se ha explorado ni explotado a profundidad. Esto se debe a la falta de una plataforma adecuada para la formación educativa, en la que tanto estudiantes como profesores tengan la oportunidad de experimentar y aprender cómo aplicar el paralelismo para resolver problemas que no serían realistas o costo-efectivos para sistemas uniprocesador.

Linux ha empezado hace poco a tener un impacto importante dentro de la industria de la informática, pero ha sido una herramienta poderosa para los investigadores en informática durante muchos años. Aparte de los beneficios obvios de trabajar con un sistema operativo abierto eficiente, fiable y de libre distribución, la llegada de la computación clusterizada siguiendo el estilo Beowulf extiende la utilidad del Linux al mundo de la computación paralela de alto rendimiento.

Hoy en día, estos valiosos clusters basados en PCs están apareciendo en laboratorios de investigación federales, departamentos de I+D de empresas, universidades e incluso pequeñas facultades.

Hay disponibles para Linux tanto compiladores comerciales como gratuitos. Los compiladores del GNU Project's C (gcc), C++ (g++), y FORTRAN (g77) están incluidos en la mayoría de las distribuciones estándar de Linux .Los compiladores de C y C++ son excelentes y el compilador de FORTRAN mejora constantemente. También hay disponibles compiladores comerciales de Absoft, The Portland Group (PGI), The Numerical Algorithms Group (NAG), y otros.

Algunos de los compiladores comerciales de FORTRAN-90 paralelizan automáticamente algunas operaciones si se configuran adecuadamente. En general, no obstante, desarrollar código paralelo requerirá un paso explícito de mensajes entre los procesadores utilizando la PVM (Parallel Virtual Machine - Máquina virtual paralela), MPI (Message Passing Interface - Interfaz de paso de mensajes), u otras librerías de comunicaciones. Ambas, la PVM y la PMI están disponibles gratuitamente y permiten al programador definir fácilmente los nodos usados para ejecutar el código paralelo y pasar datos entre los nodos durante la ejecución usando simples invocaciones a la librería.

Por supuesto, no todas las tareas informáticas se pueden adecuar al paralelismo. Muchas veces, se tienen que desarrollar nuevas estrategias para resolver problemas informáticos para poder sacar partido del paralelismo. De hecho, puede ser necesario dar un paso atrás en los códigos serie existentes para concebir una aproximación paralela.

Muchos problemas científicos se pueden beneficiar de una descomposición del dominio: una división del espacio o del tiempo en partes relativamente independientes que pueden ser procesados en nodos separados del cluster. Por ejemplo, las tareas de procesamiento de imágenes se pueden dividir normalmente de forma que cada nodo trabaje en una sección de una misma imagen. [16]

[16] Algoritmica del paralelismo, RAYNAL Michael, Ed. Omega

3.0 BASES DE DATOS.

3.1 Introducción.

Las bases de datos constituyen una parte integrante y fundamental de los sistemas de información y tienen su razón de ser en la misma existencia de estos.

Las exigencias de los usuarios respecto a sistemas de información mas flexibles y eficientes ha obligado a dedicar una mayor atención a los datos y a su estructuración, buscándose una gestión mas racional de la información es su conjunto, por lo cual ha pasado a ser considerada como un recurso fundamental de la organización.

A medida que los diseñadores de sistemas de información se van convenciendo de la trascendencia que la gestión racional de los datos tiene para conseguir un desarrollo coherente y eficaz de estos sistemas, las bases de datos empiezan a ocupar un primer plano en las áreas de interés de los informáticos y de los usuarios.

En los últimos años, las bases de datos han experimentado profundos cambios y no son ya, como hace algunos años, competencia exclusiva de grandes instalaciones con sistemas de información que gestionen millones de registros; tampoco el diseño de bases de datos esta reservado actualmente, a unos pocos especialistas en las técnicas de estructuración de datos. Por el contrario, las bases de datos se han extendido, alcanzando a sistemas de tipo medio y pequeño, con moderadas o incluso bajas cargas de trabajo, viéndose implicados en su diseño, administración o manejo de muchos profesionales o multitud de usuarios que reclaman de ellas flexibilidad, sencillez y eficiencia.

Las bases de datos son ampliamente usadas; ya sea por instituciones bancarias, líneas aéreas o universidades, ya que forman ahora una parte esencial en la mayoría de las empresas.

3.1.1 Papel económico, social y cultural de la Información.

Las necesidades de información de nuestra sociedad se dejan sentir de forma cada vez más imperiosa. La información como soporte de la transferencia de conocimientos *es "la clave para el porvenir de la humanidad e indispensable para poder modelar bien este porvenir".* [17](OCDE:).

El problema de la información esta estrechamente relacionado con el desarrollo económico y social. La investigación, la planificación y la toma de decisiones exigen una información precisa, oportuna, completa, coherente y adaptada a las necesidades especificas de cada usuario o de cada circunstancia.

Nadie puede, en estos momentos, poner en tela de juicio la importancia de la información, importancia que provoca una fuerte demanda de este bien, siendo preciso analizar, además de los condicionantes tecnológicos, el marco legal e institucional en el que se inscribe el derecho a la información.

Aunque a veces el deseo de información no es un fin en si mismo, en general, responde a una necesidad de conocer el entorno socioeconómico y cultural en el que nos desenvolvemos con fines de investigación o de toma de decisiones. La disponibilidad de información es precisa para que también el individuo pueda participar en los asuntos públicos.

Son muchos los factores que han influido en la transformación que se ha operado en el papel que desempeña la información en los contextos económicos y sociales. Entre ellos es preciso destacar la elevación del nivel cultural; el afán de desmasificación, que lleva a una mayor diversidad, con el consiguiente crecimiento de las necesidades de información; el deseo de participar en las decisiones públicas; las exigencias de planificación y ordenación del territorio; las tendencias hacia la descentralización lo cual requiere de datos aun cada vez mas detallados para áreas mas pequeñas, la aparición de nuevos métodos de toma de decisiones, etc.

---

[17] Perspectivas de la OCDE sobre las tecnologías de la información 2002: OECD Work on Measuring the Information Society

En la investigación se considera a la información como instrumento *esencial y como elemento para la entrada y la salida en toda la actividad en este campo, sea esta fundamental o aplicada.*

Otra acepción identifica a la información como *conocimiento transmisible,* lo que lleva a abrir el circulo de los beneficios de la información, que ya no se suscribe solamente a los científicos o a los técnicos, sino que amplia a otro conjunto de actividades socioculturales como la educación, los medios de comunicación, etc.

Con una visión mucho más general, se puede considerar que la información es un **recurso fundamental, un bien en el sentido económico del término, que ha de ser utilizado en cualquier actividad humana.**[18] Se convierte así la información en un elemento esencial para la producción. La igualdad de oportunidades exige no limitar arbitrariamente el acceso a este recurso que ha de ponerse al servicio de toda la sociedad.

Los gobiernos de distintos países, al reconocer que la información circula para responder a las necesidades de la sociedad, no ha podido por menos que ocuparse del tema, impulsando y planificando sistemas nacionales de información, a fin de asegurarse un flujo de este recurso.

Sin embargo, es preciso reconocer que aun nos encontramos muy lejos de poder satisfacer las necesidades de información, ya que no solo existen problemas tecnológicos relativos al almacenamiento y acceso a la información, sino que existen también problemas económicos, políticos, administrativos, etc, que impiden el desarrollo de sistemas de información eficientes, capaces de atender debidamente las demandas de información.

---

[18] Fundamentos y Modelos de Base de Datos. CASTAÑO, Ad. de Miguel, Ed. Alfaomega-Rama, 2000.

3.1.2 Cualidades de la Información.

Las cualidades que debe poseer la información, y que hacen de ella un recurso fundamental de las organizaciones y de los individuos, son básicamente las siguientes:[19]:

- ***La precisión:*** es *el porcentaje de información correcta sobre la información total del sistema.* De todas formas, el usuario ha de tener presente que el tratamiento por computadora no puede mejorar la calidad de los datos que son elaborados, lo único que puede hacer la maquina es señalar ciertos errores, e incluso sustituir el dato detectado como erróneo por otro que no tenga error aparente; es decir, que sea coherente. Una precisión baja lleva a una falta de credibilidad hacia la información que se le proporciona.

- ***La oportunidad:*** se refiere al *tiempo trascurrido desde el momento en que se produjo el hecho que originó el dato hasta el momento en que la información se pone a disposición del usuario.* Otras veces, la oportunidad se mide en función del desfase que se produce por máquina. Al igual que la precisión, la oportunidad depende de cada aplicación.

- ***La compleción:*** la información *debe de estar completa, para cumplir con sus fines.* La compleción absoluta en imposible de conseguir, y lo que se suele pretender en los sistemas de información es alcanzar un nivel que se considere suficiente, el cual dependerá de dos factores: *de los datos existentes en el sistema de información y de los que el sistema es capaz de localizar ante una búsqueda concreta.* En este segundo factor influirá la flexibilidad e idoneidad del lenguaje de recuperación y el acierto de la formulación de la consulta.

[19] Fundamentos y Modelos de Base de Datos. CASTAÑO, Ad. de Miguel, Ed. Alfaomega-Rama, 2000.

- ***La información ha de ser significativa:*** es decir, *ha de poseer el máximo contenido semántico posible*, ya que sin el, no constituirá verdadera información. Esto lleva a que ha de ser *comprensible e interesante*, lo que supone no proporcionar a los usuarios grandes masas de información que por su volumen, no puedan ser asimiladas. Un volumen de información justo es condición indispensable para que esta sea significativa. Cuando se realiza el diseño de un sistema es preciso tener en cuenta que la información suministrada por este, además de fácilmente interpretable, *solo deberá de ser la necesaria y suficiente para que se cumplan los fines propuestos.*

- ***La coherencia:*** toda la información contenida en un sistema debe ser coherente en si misma, además de consistente con las reglas semánticas propias del mundo real a que ha de representar lo mas fielmente posible.

- ***La seguridad de la información:*** la información *ha de ser protegida tanto frente a su deterioro, por causas físicas o lógicas, como frente a accesos no autorizados.* La seguridad de la información esta adquiriendo una gran relevancia, muy especialmente con la difusión de las nuevas posibilidades de las comunicaciones y la enorme extensión de redes de conexión como Internet e Intranet. Actualmente el concepto de seguridad comprende **confidencialidad, disponibilidad** e **integridad.**

Cualidades de la Información.

3.2 Definición.

Parte fundamental de una base de datos son los **archivos**. Un archivo es un *elemento de información conformado por un conjunto de registros.* Estos registros a su vez están compuestos por una serie de caracteres o bytes. Los archivos, alojados en dispositivos de almacenamiento conocidos como memoria secundaria, pueden almacenarse de dos formas diferentes: archivos convencionales o bases de datos.

Los archivos convencionales, pueden organizarse como archivos secuenciales o archivos directos. Sin embargo, el almacenamiento de información a través de archivos convencionales presenta una serie de limitaciones que restringen de manera importante la versatilidad de los programas de aplicación que se desarrollan.

El uso de sistemas de información por parte de las organizaciones requiere el almacenamiento de grandes cantidades de información, ya sea para el uso mismo del sistema, para generar resultados o para compartir dicha información con otros sistemas.

Las formas en las cuales pueden organizarse *son archivos secuenciales o archivos directos.* En los archivos secuenciales los *registros están almacenados en una secuencia que depende de algún criterio definido.* Por ejemplo, pueden almacenarse los registros de los empleados de la empresa de manera secuencial de acuerdo al departamento al que pertenecen o de acuerdo a su antigüedad.

Si se desea consultar o modificar información, también es necesario buscar uno por uno en los registros hasta encontrarla.

Los archivos directos *permiten accesar directamente un registro de información sin tener que buscar uno a uno por todos los registros del archivo, utilizando una llave de acceso dentro del archivo.*

Son muy numerosas las definiciones de base de datos, y si se analizan detenidamente, se suele observar en casi todas ellas coincidencias en ciertos elementos.

Colocado de forma sencilla, una base de datos es:

..... *"un conjunto de información almacenada sistemáticamente y ordenadamente para su uso posterior"* [20] .....

En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

En la actualidad, y en gran parte gracias a la tecnología y recursos disponibles provenientes de campos como la informática y la electrónica, las bases de datos pueden adquirir diversas formas, ofreciendo un amplio rango de soluciones al problema de almacenar datos.

Otros conceptos de una base de datos:

..... *"es un sistema de archivos de computadoras que usa una organización de archivos particulares para facilitar la actualización rápida de registros aislados, la actualización simultánea de registros relacionados, fácil acceso de los programas de aplicaciones a todos los registros y acceso rápido a todos los datos almacenados que deben unirse para satisfacer un informe o consulta particular de rutina o de propósito especial"* [21] .....

..... *"una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular"*[22] .....

La aparición de la expresión base de datos se produce a comienzos de los años sesenta. En 1963 tuvo lugar en Santa Mónica (EU) un simposio en cuyo titulo se encontraba la expresión DATA BASE.

Todas las definiciones coinciden en que una base de datos es ***un conjunto, colección o deposito de datos almacenados en un soporte informático, en donde los datos estan interrelacionados y estructurados de acuerdo con un modelo capaz de recoger el máximo contenido semántico.*** Dada la relevancia que tienen en el mundo real las interrelaciones entre los datos, es imprescindible que la base de datos sea capaz de almacenar estas interrelaciones.

---

[20] Introducción a los Sistemas de Base de Datos, ULLMAN Jeffrey. Prentice Hall. 1999.
[21] Fundamentos de Base de Datos 4ta Edición, SILBERSCHATZ, KORTH, SUDARSHAN, McGraw Hill, 2001
[22] Base de Datos desde Chen hasta Codd. CRUZ Irene, GOMEZ NIETO Miguel, Ed. AlfaOmega-Rama, 2001

La redundancia de los datos debe ser controlada, de forma que no existan duplicidades perjudiciales ni innecesarias, y que las redundancias físicas, convenientes muchas veces a fin de responder a objetivos de eficiencia, sean tratadas por el mismo sistema, de modo que no puedan producirse inconsistencias.

Las bases de datos proporcionan la infraestructura requerida para los sistemas de apoyo a la toma de decisiones y para los sistemas de información estratégicos, ya que estos sistemas explotan la información contenida en las bases de datos de la organización para apoyar el proceso de toma de decisiones o para lograr ventajas competitivas. Por este motivo es importante conocer la forma en que están estructuradas las bases de datos y su manejo.

Las bases de datos pretender servir al conjunto de la organización, manejando los datos como otro recurso que viene a añadirse a los ya tradicionales. Por tanto, las bases de datos han de atender a múltiples usuarios y a diferentes aplicaciones.

Un concepto a tener en cuenta cuando hablamos de Bases de Datos es el de los apuntadores, estos establecen uniones entre los registros y son una parte básica de la organización de archivos de todos los sistemas de Base de Datos que veamos, excepto en el sistema relacional.

Con este sistema de apuntadores generalmente se coloca un apuntador en el último campo del registro que contiene la dirección de otro registro relacionado con el que se apunta y el apuntador dirige el sistema de cómputo hacia el registro relacional.

Los componentes principales de una base de datos son:

- **Datos.** Los datos son la Base de Datos (BD) propiamente dicha.
    - *Diccionario de datos:* Depósito centralizado de información en forma computarizada acerca de los datos en una BD (el nombre de cada elemento en la BD y una descripción y definición de sus atributos). El diccionario incluye información acerca de la localización de estos datos en los archivos de una BD y muchos también contienen reglas de acceso y de seguridad y privacidad acerca de los mismos.

- **Hardware.** El hardware se refiere a los dispositivos de almacenamiento en donde reside la base de datos, así como a los dispositivos periféricos (unidad de control, canales de comunicación, etc.) necesarios para su uso.
    - *Las terminales de acceso y actualización en línea*: Estas pueden encontrarse adyacentes en la computadora o a miles de Km. de distancia, pueden ser terminales inteligentes, no inteligentes o micro computadoras.

- **Software.** Está constituido por un conjunto de programas que se conoce como Sistema Manejador (o gestor) de Base de Datos (DMBS: Data Base Management System). Este sistema maneja todas las solicitudes formuladas por los usuarios a la base de datos.
    - *Sistema de Interfase de Lenguaje Anfitrión*: Esta es la parte del DBMS que se comunica con los programas de aplicaciones en lenguaje de alto nivel, como programas en Cobol y Fortran que piden datos de los archivos para que pueda obtenerse la información necesaria.
    - *Programas de aplicación:* Estos realizan las mismas funciones que en sistemas convencionales pero son independientes de los archivos de datos, y usan definiciones estándares de los mismos, los programas de aplicación usando el lenguaje anfitrión de la interfase lo desarrollan por lo general programadores profesionales. (No se definen los datos).
    - *Sistema de Interface de Lenguaje Natural:* Este lenguaje de consultas permite la actualización y las consultas en línea de los usuarios que no son muy ilustrados acerca de los sistemas de cómputo (Lenguajes Query, como SQL).
    - *Sistema Gestor de Interfaces de Salida:* Este proporciona información de trabajos de rutina, documentos o informes especiales.

- **Usuarios**. Existen tres clases de usuarios relacionados con una Base de Datos:
    - *El programador de aplicaciones*, quien crea programas de aplicación que utilizan la base de datos.
    - *El usuario final*, quien accesa la Base de Datos por medio de un lenguaje de consulta o de programas de aplicación.
    - *El administrador de la Base de Datos (DBA: Data Base Administrator)*, quien se encarga del control general del Sistema de Base de Datos.

3.3 Abstracción y Modelo de los datos.

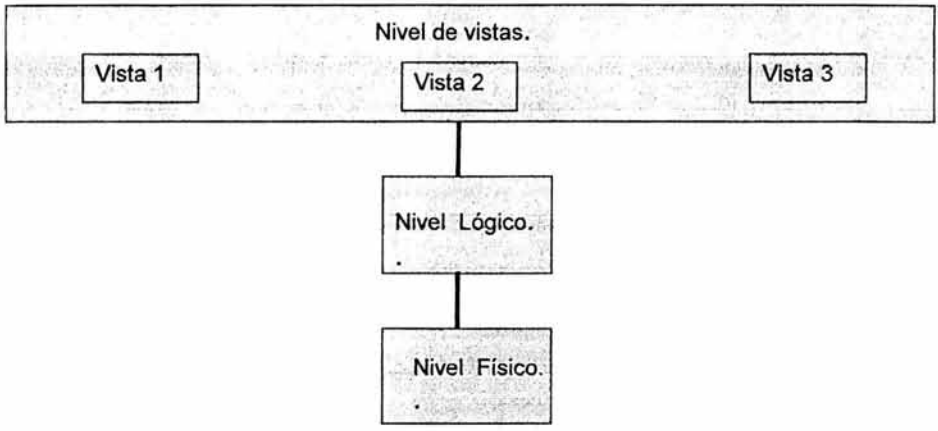
Para que un sistema sea útil debe recuperar los datos eficientemente. Esta preocupación ha conducido al diseño de estructuras de datos complejas para la representación de los datos en la base de datos.

Los tres niveles de abstracción de datos son:

- **Nivel físico:** el nivel más bajo de la abstracción. Describe ***como se almacenan realmente los datos***. En el nivel físico, se describen en detalle las estructuras de datos complejas de bajo nivel.

- **Nivel lógico:** el siguiente nivel más alto de abstracción. Describe ***que datos se almacenan en la base de datos y que relaciones existen entre esos datos.*** La base de datos completa se describe así en términos de un número pequeño de estructuras relativamente simples.

- **Nivel de vistas:** El nivel mas alto de abstracción. Describe ***solo parte de la base de datos completa.*** A pesar del uso de estructuras más simples en el nivel lógico, queda algo de complejidad, debido a la variedad de información almacenada en una gran base de datos. Son partes del esquema lógico. El nivel lógico presenta toda la base de datos, mientras que los usuarios por lo general sólo tienen acceso a pequeñas parcelas de ésta. El nivel visión es el encargado de dividir estas parcelas.



Los tres niveles de abstracción de datos.

Bajo la estructura de las bases de datos se encuentra el modelo de los datos: una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y la restricción de consistencia.

Existen fundamentalmente tres alternativas disponibles para diseñar las bases de datos: el modelo jerárquico, el modelo de red y el modelo relacional.

- **El modelo jerárquico:** La forma de esquematizar la información se realiza a través *de representaciones jerárquicas o relaciones de padre/hijo,* de manera similar a la estructura de un árbol. Así, el modelo jerárquico puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos. En el primer tipo se dice que existe una relación de uno a uno si el padre de la estructura de información tiene un solo hijo y viceversa, si el hijo tiene solamente un padre. En el segundo tipo se dice que la relación es de uno a muchos si el padre tiene más de un hijo, aunque cada hijo tenga un solo padre.

Inconveniente del modelo jerárquico.

Relación maestro-alumno, donde un maestro tiene varios alumnos, pero un alumno también tiene varios maestros, uno para cada clase. En este caso, si la información estuviera representada en forma jerárquica donde el padre es el maestro y el alumno es el hijo, la información del alumno tendrá que duplicarse para cada uno de los maestros.

Otra dificultad que presenta el modelo jerárquico de representación de datos es respecto a las bajas. En este caso, si se desea dar de baja a un padre, esto necesariamente implicará dar de baja a todos y cada uno de los hijos que dependen de este padre.

- **El modelo de red:** El modelo de red evita esta redundancia en la información, a través de la incorporación de un tipo de registro denominado *conector,* que en este caso pueden ser las calificaciones que obtuvieron los alumnos de cada profesor. La dificultad surge al manejar las conexiones o ligas entre los registros y sus correspondientes registros conectores.

- **El modelo relacional:** Se está empleando con más frecuencia en la práctica, debido al rápido entendimiento por parte de los usuarios que no tienen conocimientos profundos sobre Sistemas de Bases de Datos y a las ventajas que ofrece sobre los dos modelos anteriores. En este modelo toda la información se representa a través de *arreglos bidimensionales o tablas.* Las tablas son un medio de representar la información de una forma más compacta y es posible acceder a la información contenida en dos o más tablas.

Debido a su relevancia, este modelo es explicado a continuación con más profundidad.

3.4 Base de Datos Relacional.

El modelo relacional se ha establecido como el principal modelo para las aplicaciones de procesamiento de datos. Ha conseguido la posición principal debido a su simplicidad, que facilita el trabajo del programador en comparación con otros modelos.

Una base relacional consiste en **un conjunto de tablas**, a cada una de las cuales se le **asigna un nombre exclusivo**.

Una Base de Datos Relacional, en informática, es definida como:

..... *"un tipo de base de datos o sistema de administración de bases de datos, que almacena información en tablas (filas y columnas de datos) y realiza búsquedas utilizando los datos de columnas especificadas de una tabla para encontrar datos adicionales en otra tabla* " [23].....

En una base de datos relacional, las filas representan registros (conjuntos de datos acerca de elementos separados) y las columnas representan campos (atributos particulares de un registro). Al realizar las búsquedas, una base de datos relacional hace coincidir la información de un campo de una tabla con información en el campo correspondiente de otra tabla y con ello produce una tercera tabla que combina los datos solicitados de ambas tablas.

Por ejemplo, si una tabla contiene los campos NÚM-EMPLEADO, APELLIDO, NOMBRE y ANTIGÜEDAD y otra tabla contiene los campos DEPARTAMENTO, NÚM-EMPLEADO y SALARIO, una base de datos relacional hace coincidir el campo NÚM-EMPLEADO de las dos tablas para encontrar información, como por ejemplo los nombres de los empleados que ganan un cierto salario o los departamentos de todos los empleados contratados a partir de un día determinado. En otras palabras, una base de datos relacional ***utiliza los valores coincidentes de dos tablas para relacionar información de ambas.*** Por lo general, los productos de bases de datos para microcomputadoras o microordenadores son bases de datos relacionales.

---

[23] Base de Datos desde Chen hasta Codd. CRUZ Irene, GOMEZ NIETO Miguel, Ed. AlfaOmega-Rama, 2001

Las dos características esenciales de una Estructura Relacional son [24]:

- El archivo está en forma de tabla (similar a un archivo secuencial).
- Las asociaciones de los registros están hechas con base en los valores en un campo de los registros, no en las direcciones (apuntadores dentro de los registros) por lo tanto, no hay caminos predefinidos para la obtención de datos.

Cada operación relacional se realiza en una tabla de registros y produce una nueva tabla..

Estas operaciones básicas son:

- Seleccionar renglones de alguna tabla (SELECT)
- Seleccionar columnas de alguna tabla (PROJECT)
- Unir o juntar información de varias tablas (JOIN)

La estructura relacional difiere de una estructura secuencial ordinaria en *que los registros en un sistema relacional pueden estar en cualquier orden,* sin embargo colocarlo secuencialmente puede hacer que el procesamiento rutinario en lotes sea más eficiente

Es importante mencionar que la mayoría de los paquetes que manejan bases de datos disponibles en el mercado poseen las instrucciones SELECT, PROJECT Y JOIN con diferentes nombres y modalidades.

Para crear las relaciones, modificarlas, eliminarlas, recuperar los datos almacenados en ellas, y para manipularlas en general, necesitamos un lenguaje formal que nos facilite el acceso, de lo contrario nos veríamos obligados a trabajar a bajo nivel, o nivel de máquina.

Este lenguaje debe ser lo suficientemente expresivo para permitirnos llevar a cabo todas estas operaciones, y debe estar basado en formalismos que cumplan con todas las premisas expuestas en los apartados anteriores respecto a reglas de integridad, formas normales, etc.

Pueden agregarse y borrarse registros de un sistema relacional con un mínimo de trabajo de procesamiento porque no necesitan agregarse, borrarse o reorganizarse ningún apuntador.

---

[24] Fundamentos de Base de Datos 4ta Edición, SILBERSCHATZ, KORTH, SUDARSHAN, McGraw Hill, 2001

En un sistema relacional el resultado de cada operación de procesamiento es una nueva tabla de registros, por lo tanto se crean fácilmente nuevos archivos con este propósito.

La ***tabla*** (*table*) es el concepto básico, fundamental del Modelo Relacional, su razón de ser y su característica distintiva. Es la unidad de almacenamiento tanto mínima como máxima, y por lo tanto *única:* no existe dato almacenado en el Modelo Relacional fuera de las tablas. Una tabla está compuesta de dos partes bien distintas: el ***encabezado***, y un conjunto de ***filas*** (*rows*). Cada fila de la tabla representa una relación entre un conjunto de valores. Dado que la tabla es un conjunto de dichas relaciones, hay una fuerte correspondencia entre el concepto de tabla y el concepto matemático de relación.

Una *relación* R definida sobre un conjunto de *dominios* $D_1, D_2, \ldots, D_n$ consta de:

- *Cabecera:* conjunto fijo de pares *atributo: dominio*

  $$\{(A_1 : D_1), (A_2 : D_2), \ldots (A_n : D_n)\}$$

  Donde cada atributo A_j corresponde a un único dominio D_j y todos los A_j son distintos, es decir, no hay dos atributos que se llamen igual. El grado de la relación R es n.

- *Cuerpo:* conjunto variable de *tuplas*. Cada tupla es un conjunto de pares *atributo: valor*:

  $$\{(A_1 : v_{i1}), (A_2 : v_{i2}), \ldots (A_n : v_{in})\}$$

  con $i = 1, 2, \ldots m$, donde m es la cardinalidad de la relación R. En cada par $(A_j : v_{ij})$ se tiene que $v_{ij} \in D_j$.

Cada fila es un conjunto de valores *en estricta correspondencia* con cada elemento (del encabezado; es decir, cada valor en una fila no sólo pertenece al tipo que le corresponde en el encabezado, sino que tiene asociado el nombre correspondiente.

Esto es un poco difícil de entender, porque los conjuntos no están ordenados: si bien no hay un orden intrínseco dentro del encabezado (porque es un conjunto), y no hay orden dentro de la fila (de nuevo, porque es un conjunto), de todas formas cada uno de los elementos del encabezado está en correspondencia *con uno y sólo uno* de los valores de todas las filas de la tabla.

De nuevo, esto es fácil de conceptualizar concentrándonos en la presentación tabular:

Título	Año	Duración
Star Wars	1977	124
Gladiador	2000	124
Vanilla Sky	2003	135
Batman	1998	124

En términos tradicionales una relación se asemeja a un archivo, una tupla a un registro, y un atributo a un campo. Pero estas correspondencias son aproximadas, en el mejor de los casos.

Una relación no debe considerarse como "solo un archivo", sino mas bien como un archivo disciplinado, siendo el resultado de esta disciplina una simplificación considerable de las estructuras de datos con las cuales debe interactuar el usuario, lo cual a su vez simplifica los operadores requeridos para manejar esas estructuras.

Las relaciones tienen las siguientes características:

- Cada relación tiene un nombre y éste es distinto del nombre de todas las demás.
- Los valores de los atributos son atómicos: en cada tupla, cada atributo toma un solo valor. Se dice que las relaciones están *normalizadas*.
- No hay dos atributos que se llamen igual.
- El orden de los atributos no importa: los atributos no están ordenados.
- Cada tupla es distinta de las demás: no hay tuplas duplicadas.

Características principales de los "archivos" relacionales:

- Cada "archivo" contiene solo un tipo de registros
- Los campos no tienen un orden específico, de izquierda a derecha
- Los registros no tienen un orden específico, de arriba hacia abajo
- Cada campo tiene un solo valor
- Los registros poseen un campo identificador único (o combinación de campos) llamado clave primaria

En la parte superior de una relación vemos ***atributos*** de una relación sirven de nombre a las columnas de la relación. Por lo regular, describen el significado de las entradas de la columna situada debajo de ellos.

El nombre de una relación y el conjunto de sus atributos recibe el nombre de ***esquema de la relación***. El esquema de la relación se denota con el nombre de esta seguido de una lista de sus atributos entre paréntesis. Recuerdese que, aunque en los atributos de un esquema de relación son un conjunto y no una lista, para hablar de relaciones debemos especificar a menudo un orden estándar de los atributos. Así, siempre que introduzcamos un esquema de relación con una lista de atributos supondremos que este ordenamiento en el orden estándar cuando despleguemos un renglón o uno de sus renglones.

A los renglones de una relación, si no son el renglón del encabezado que contiene los atributos, se les da el nombre de ***tuplas***. Una tupla tiene un componente para cada uno de los atributos de la relación. Cuando una tupla aparece sola, los atributos no aparecen y, por lo tanto, habrá que dar alguna indicación de la relación a la que pertenece. A veces se piensa que las tuplas representan objetos y que, en cambio, la relación a que pertenece representa su clase, sin embargo, los objetos poseen identidad, no así las tuplas. Las relaciones son conjuntos de tuplas y no es posible que una tupla aparezca más de una vez en una relación.

El modelo relacional exige que los componentes de una tupla sean atómicos, o sea, que pertenezcan a algún tipo elemental como enteros o cadena de caracteres. No se admite que un valor sea una estructura de registro, un conjunto, una lista, un arreglo o cualquier otro tipo que pueda tener sus valores divididos en componentes más pequeños. Cada atributo de una relación se asocia con un ***dominio***, o sea, un tipo elemental. Los componentes de una tupla de la relación deberán tener, en cada componente, un valor que pertenezca al dominio de la columna correspondiente.

Tanto el esquema como las tuplas de una relación **SON CONJUNTOS**, no listas. De ahí que **NO IMPORTE EL ORDEN EN QUE SEAN PRESENTADAS**. Mas aún, **SE PUEDEN REORDENAR LOS ATRIBUTOS DE LA RELACIÓN COMO SE QUIERA, SIN CAMBIAR CON ELLO LA RELACIÓN**.

Cuando se reordena el esquema, debe recordarse que los atributos son encabezados de columna. Así, cuado modificamos el orden de los atributos, también cambiamos el orden de las columnas. Cuando estas se mueven, también los componentes de las tuplas alteran su orden. El resultado es que los componentes de cada tupla se permutan del mismo modo que los atributos.

Año	Título	Duración
2003	Vanilla Sky	135
1977	Star Wars	124
1998	Batman	124
2000	Gladiador	124

Una relación no es estática: las relaciones cambian con el tiempo. Esperamos que los cambios incluyan tuplas de la relación; como la inserción de nuevas tuplas a medida que se agreguen datos a la base de datos, cambios de las tuplas actuales, si obtenemos información revisada o corregida, y quizá la eliminación de las tuplas que se expulsan de la base de datos por alguna razón.

Es menos frecuente que cambie el esquema de una relación. Pero se dan situaciones donde tal vez se quieran agregar o eliminar atributos. Los cambio de esquema; aunque posibles en los sistemas comerciales de bases de datos, son muy costosos porque es necesario reescribir cada una de las tuplas para agregar o suprimir componentes.

Existen dos tipos básicos de formalismos para expresar las consultas sobre las relaciones de una base de datos relacional: el álgebra relacional y el cálculo relacional.

- El *Álgebra Relacional* es una notación algebraica, en la cual las consultas se expresan aplicando operadores especializados a las relaciones.
- El *Cálculo Relacional* es una notación lógica, donde las consultas se expresan formulando algunas restricciones lógicas que las tuplas de la respuesta deban satisfacer.

El *Álgebra Relacional* fue introducida por E.F.Codd en 1972. Consiste en un conjunto de operaciones con las relaciones [25].

- **SELECT (σ):** *extrae tuplas a partir de una relación que satisfagan una restricción dada.* Sea *R* una tabla que contiene un atributo *A*. $\sigma_{A=a}(R) = \{t \square R \square t(A) = a\}$ donde t denota una tupla de *R* y t(A) denota el valor del atributo *A* de la tupla t.
- **PROJECT (π):** *extrae atributos (columnas) específicos de una relación.* Sea R una relación que contiene un atributo X. $\pi_X(R) = \{t(X) \square t \square R\}$, donde t(X) denota el valor del atributo X de la tupla t.
- **PRODUCT (×):** *construye el producto cartesiano de dos relaciones.* Sea R una tabla de rango (arity) k_1 y sea S una tabla con rango (arity) k_2. R × S es el conjunto de las $k_1 + k_2$-tuplas cuyos primeros k_1 componentes forman una tupla en R y cuyos últimos k_2 componentes forman una tupla en S.
- **UNION (□):** *supone la unión de la teoría de conjuntos de dos tablas.* Dadas las tablas R y S (y ambas deben ser del mismo rango), la unión R □ S es el conjunto de las tuplas que están en R S o en las dos.
- **INTERSECT (∩):** *construye la intersección de la teoría de conjuntos de dos tablas.* Dadas las tablas R y S, R □ S es el conjunto de las tuplas que están en R y en S>. De nuevo requiere que R y S tengan el mismo rango.
- **DIFFERENCE (− o □):** *supone el conjunto diferencia de dos tablas.* Sean R y S de nuevo dos tablas con el mismo rango. R - S Es el conjunto de las tuplas que están en R pero no en S.

[25] Base de Datos desde Chen hasta Codd. CRUZ Irene, GOMEZ NIETO Miguel, Ed. AlfaOmega-Rama, 2001

- **JOIN (∏):** *conecta dos tablas por sus atributos comunes.* Sea R una tabla con los atributos A,B y C y sea S una tabla con los atributos C,D y E. Hay un atributo común para ambas relaciones, el atributo C. $R \prod S = \pi_{R.A,R.B,R.C,S.D,S.E}(\sigma_{R.C=S.C}(R \times S))$. ¿Qué estamos haciendo aquí? Primero calculamos el producto cartesiano R × S. Entonces seleccionamos las tuplas cuyos valores para el atributo común C sea igual ($\sigma_{R.C = S.C}$). Ahora tenemos una tabla que contiene el atributo C dos veces y lo corregimos eliminando la columna duplicada.

El *Cálculo Relacional* usa un enfoque diferente al álgebra relacional. No obstante, los dos lenguajes son lógicamente equivalentes. Esto significa que cualquier consulta puede resolverse en el otro.

La solución para toda consulta en el cálculo relacional es una relación que se define por una ***lista resultado*** y una ***sentencia de cualificación***. La lista resultado *define los atributos de la relación solución.* La sentencia de cualificación *es una condición usada para determinar que valores de la base de datos actual van a la relación solución.*

Las operaciones de unión, intersección, diferencia, producto, selección y de proyección pueden ser fácilmente derivadas de las construcciones del cálculo relacional. Debido a que el cálculo no usa el procedimiento del álgebra paso a paso, no se necesita la sentencia de asignación.

Por tanto, la única operación del álgebra relacional para la cual aun no se ha mostrado equivalencia en el cálculo relacional es la reunión. Esta requiere de cuantificadores existenciales. Un cuantificador cuantifica o indica la cantidad de algo.

El cuantificador existencial indica que existen al menos una instancia de algo de un tipo particular. En el cálculo relacional, el cuantificador existencial se usa para indicar que existe una fila de un tipo en particular en una relación.

En un Sistema de Gestión de Base de Datos Relacional pueden existir varios tipos de relaciones, aunque no todos manejan todos los tipos[26].

- ***Relaciones base.*** Son relaciones reales que tienen nombre y forman parte directa de la base de datos almacenada (son autónomas).
- ***Vistas.*** También denominadas relaciones virtuales, son relaciones con nombre y derivadas: se representan mediante su definición en términos de otras relaciones con nombre, no poseen datos almacenados propios.
- ***Instantáneas.*** Son relaciones con nombre y derivadas. Pero a diferencia de las vistas, son reales, no virtuales: están representadas no sólo por su definición en términos de otras relaciones con nombre, sino también por sus propios datos almacenados. Son relaciones de sólo de lectura y se refrescan periódicamente.
- ***Resultados de consultas.*** Son las relaciones resultantes de alguna consulta especificada. Pueden o no tener nombre y no persisten en la base de datos.
- ***Resultados intermedios.*** Son las relaciones que contienen los resultados de las subconsultas. Normalmente no tienen nombre y tampoco persisten en la base de datos.
- ***Resultados temporales.*** Son relaciones con nombre, similares a las relaciones base o a las instantáneas, pero la diferencia es que se destruyen automáticamente en algún momento apropiado.

Ya que en una relación no hay tuplas repetidas, éstas se pueden distinguir unas de otras, es decir, se pueden identificar de modo único. La forma de identificarlas es mediante los valores de sus atributos. Estos valores se determinan **llaves o claves**.

Una ***superllave*** o ***superclave*** es un atributo o un conjunto de atributos que identifican de modo único las tuplas de una relación.

[26] Fundamentos de Sistemas de Base de Datos. ELSMAN Ramón A. Afelssen Wesley, 2002

Una ***llave candidata*** es una superllave en la que ninguno de sus subconjuntos es una superllave de la relación. El atributo o conjunto de atributos K de la relación R es una llave candidata para R si y sólo si satisface las siguientes propiedades:

- ***Unicidad***: nunca hay dos tuplas en la relación R con el mismo valor de K.
- ***Irreductibilidad***: ningún subconjunto de K tiene la propiedad de unicidad, es decir, no se pueden eliminar componentes de K sin destruir la unicidad.

Cuando una llave candidata está formada por más de un atributo, se dice que es una ***llave compuesta***. Una relación puede tener varias llaves candidatas. Para identificar las llaves candidatas de una relación no hay que fijarse en un estado o instancia de la base de datos.

El hecho de que en un momento dado no haya duplicados para un atributo o conjunto de atributos, no garantiza que los duplicados no sean posibles. Sin embargo, la presencia de duplicados en un estado de la base de datos sí es útil para demostrar que cierta combinación de atributos no es una llave candidata. El único modo de identificar las llaves candidatas es conociendo el significado real de los atributos, ya que esto permite saber si es posible que aparezcan duplicados. Sólo usando esta información semántica se puede saber con certeza si un conjunto de atributos forman una clave candidata.

La ***llave primaria*** de una relación es aquella llave candidata que se escoge para identificar sus tuplas de modo único. Ya que una relación no tiene tuplas duplicadas, siempre hay una llave candidata y, por lo tanto, la relación siempre tiene llave primaria. En el peor caso, la llave primaria estará formada por todos los atributos de la relación, pero normalmente habrá un pequeño subconjunto de los atributos que haga esta función.
Las llaves candidatas que no son escogidas como llave primaria son denominadas ***llaves alternativas***.

Una ***llave ajena*** es un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la llave primaria de alguna otra relación (puede ser la misma). Las llaves ajenas representan *relaciones entre datos*

El modelo relacional de datos contempla tres tipos de restricciones:

1. Integridad de la llave. Ningún atributo de una llave candidata puede tomar valores nulos. Lógicamente, los atributos que forman una llave candidata han de tomar siempre valores distintos para cada posible tupla.

2. Integridad de referencia o referencial. Sea T1.a un atributo de la tabla T1 que forma parte de una llave ajena para la tabla T2. Es decir, que en T2 existe un atributo definido con el mismo dominio, aunque no obligatoriamente con igual nombre, y que es parte de su llave primaria. Entonces, T1.a debe ser siempre igual a algún valor ya contenido en el atributo referenciado en la tabla T2, o bien tomar un valor nulo.

3. Otras restricciones de acuerdo con la semántica concreta del problema. Pueden ser sencillas, como la especificación de valores mínimos o máximos que puede tomar un atributo numérico, lista de valores permitidos de un atributo, o más complejas: condiciones sobre valores de los atributos en función de valores de otros atributos de esa u otras tablas.

Un segundo modelo a considerar para su estudio y explicación es el modelo de base de datos distribuida.

3.5 Base de Datos Distribuida (BBD).

Con la amplia prevalecía de los sistemas de base de datos centralizados, los usuarios y los programas de aplicación acceden a la base de datos desde sitios locales, así como desde localidades remotas. En contraste, una base de datos distribuida ***no se almacena completamente en una localidad central***, sino que se ***distribuye en una red de localidades que pueden estar geográficamente separadas y conectadas por enlaces de comunicaciones***. Cada localidad ***tendrá su propia base de datos y está capacitada para acceder a los datos de otras localidades.***

Una Base de Datos Distribuida es, una base de datos construida sobre una red computacional y no por el contrario en una máquina aislada. La información que constituye la base de datos esta almacenada en diferentes sitios en la red, y las aplicaciones que se ejecutan accesan datos en distintos sitios.

Entonces, una Base de Datos Distribuida es

..... *"una colección de datos que pertenecen lógicamente a un sólo sistema, pero se encuentra físicamente esparcido en varios "sitios" de la red"* [27] .....

Hay varias razones para desarrollar y usar un sistema de BDD, entre las cuales se encuentran:

- A menudo, las organizaciones tienen ramales o divisiones en diferentes localidades. Los sistemas distribuidos se puede adecuar de una manera más sencilla a las estructuras de la organización de los usuarios
- Permitir que cada sitio almacene y contenga su propia base de datos facilita el acceso inmediato y eficaz a los datos que se usan con mayor frecuencia. Tales datos podrían usarse en otros sitios, pero con menos frecuencia. Datos similares almacenados en otras localidades también podrían accederse si fuese necesario.
- Las BDD pueden mejorar la fiabilidad. Si las computadoras de un sitio fallan, o si se queda fuera de servicio algún enlace de comunicación, el resto de la red puede seguir funcionando. Los datos duplicados aumentan su confiabilidad. Cuando falla una computadora, se pueden obtener los datos extraídos de otras computadoras. Los usuarios no dependen de la disponibilidad de una sola fuente para sus datos.
- Permitir el control de los datos que se usan con más frecuencia en un sitio puede mejorar el grado de satisfacción de los usuarios con relación al sistema de base de datos.

[27] Principios de Base de Datos Distribuidas 2da. Edición. OZSU, VALDURIEZ, Prentice Hall, 2000

- Se puede obtener un mejor rendimiento que el que se obtiene por un procesamiento centralizado. Los datos pueden colocarse cerca del punto de su utilización, de forma que el tiempo de comunicación sea mas corto. Varias computadoras operando en forma simultánea pueden entregar más volumen de procesamiento que una sola computadora.
- Los sistemas distribuidos pueden variar su tamaño de un modo más sencillo. Se pueden agregar computadoras adicionales a la red conforme aumentan el número de usuarios y su carga de procesamiento. A menudo es más fácil y más barato agregar una nueva computadora más pequeña que actualizar una computadora única y centralizada. Después, si la carga de trabajo se reduce, el tamaño de la red también puede reducirse.

En consecuencia, la llamada "base de datos distribuida" es en realidad una especie de objeto virtual, cuyas partes componentes se almacenan físicamente en varias bases de datos "reales" distintas ubicadas en diferentes sitios. De hecho, es la unión lógica de esas bases de datos. En otras palabras, cada sitio tiene sus propias bases de datos "reales" locales, sus propios usuarios locales, sus propios Sistemas de Gestión de Base de Datos. (SGBD) y programas para la administración de operaciones, y su propio administrador local de comunicación de datos. Esto requiere que las bases de datos locales sean capaces de comunicarse entre si.

Las conexiones de comunicación que proporcionan las capacidades necesarias de transferencia se llaman ***enlaces (links)***. La estructura de enlace brinda la arquitectura básica de un sistema de administración o gestión de bases de datos distribuidas (SGBDD).

El diseño de un sistema de BDD implica la toma de decisiones sobre la ubicación de los programas que accederán a la base de datos y sobre los propios datos que constituyen esta última, a lo largo de los diferentes nodos que configuren una red de computadoras. La ubicación de los programas, no debería de suponer un excesivo problema dado que se puede tener una copia de ellos en cada máquina conectada a la red.

Un SGBDD es un sistema compuesto por varios SGBD's operando en sitios locales y que estan conectados por facilidades de manejo de mensajes. El diccionario de datos del SGBDD incluye la información usual necesaria para la gestión de los datos junto con la información concerniente a cada localidad, a la duplicidad y a la fragmentación de los datos en varias relaciones. En la medida en que se procesan las consultas para recuperar o actualizar datos, el diccionario de datos del SGBDD puede proporcionar la información requerida sobre la localidad y la duplicidad, mientras que asegura que las actualizaciones se propagan a las localidades apropiadas.

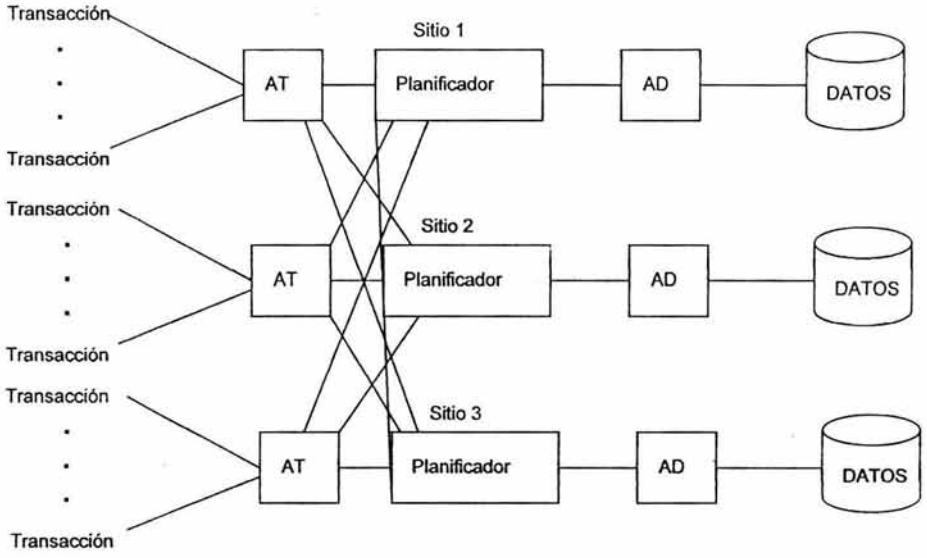
En particular un usuario dado puede realizar operaciones sobre los datos en su propio sitio local exactamente como si ese sitio no participara en absoluto en el sistema distribuido (al menos, ése es uno de los objetivos). Los usuarios interactúan con el SGBDD ejecutando programas que se llaman ***transacciones***. Las transacciones en tales sistemas no están ya restringidas a un solo proceso controlado por un módulo de software, sino que pueden invocar un conjunto de procesos cooperantes, funcionando en varios sitos y controlados por módulos independientes de software.

Cada uno de estos procesos cooperantes en la transacción se llama ***un agente.*** Una transacción que requiere solo un agente se le conoce como ***transacción local***. Una transacción que requiere varios agentes se denomina ***transacción global***.

Un agente dado puede solo acceder a los datos controlados por su software de gestión de datos local. Acceder a los datos de otro sitio requiere de la cooperación de los agentes de esos sitios. Un agente que inicia una transacción se llama *agente iniciador*. El agente iniciador puede demandar la activación de agentes de otros sitios para poder acceder a los datos necesarios. Una vez que estos estan activados, dos o más agentes se pueden comunicar mediante intercambio de mensajes.

Las transacciones acceden a los registros mediante las operaciones *read (lectura)* y *write (escritura)*. Read (x) retorna el valor actual de *x*. Write (*x, nuevo valor*) actualiza el valor actual de x al nuevo valor. Una transacción expide las ordenes read y write al SGBDD y ejecuta las entradas y salidas por la terminal.

Cada sitio participante en el SGBDD típicamente ejecuta uno o más de los siguientes módulos de software: un administrador de transacciones (AT), un administrador de datos (AD) y un planificador.



Interrelaciones en la Arquitectura del SGBDD

Las transacciones se comunican con los AT's, estos con los planificadores; los planificadores se comunican con los AD's, y estos administran los datos.

Cada transacción comunica todas sus lecturas y escrituras a un único AT. Una transacción también expide una operación *begin (empezar)* a su AT cuando comienza su ejecución y luego un *end (finalizar)* cuando esta termina. El AT comunica cada lectura y escritura al planificador. La selección del planificador se determina por un algoritmo de control concurrente, aunque el planificador que se escoge mas a menudo este en el mismo sitio que los datos sobre los que se está operando.

El planificador controla la secuencia en la cual los AD's procesan las ordenes de lectura y escritura y mantiene el control de concurrencia. Cuando un planificador recibe una instrucción de lectura o escritura, el planificador puede procesar la instrucción inmediatamente, demorar el procesamiento guardando la instrucción para su acción posterior, o rechazar la instrucción en el caso de un error de transmisión, violación de acceso o algún problema similar.

El AD ejecuta cada lectura y escritura que recibe. Para una lectura, el AD recorre su base de datos local y retorna el valor solicitado. Para una escritura, el AD modifica la base de datos local y retorna un reconocimiento al planificador, que lo envía de regreso al AT y este lo regresa a la transacción.

Los sistemas de base de datos distribuidas a menudo enlazan sitios bastante dispersos geográficamente (WAN's). Alternativamente, los sistemas de bases de datos distribuidas pueden organizarse para servir en redes de área local (LAN's), donde las estaciones de trabajo de las oficinas son los sitios y hay enlaces entre las estaciones de trabajo. En ambas variantes los sitios pueden enlazarse en varios momentos.

Tradicionalmente se ha clasificado la organización de los sistemas de BDD sobre tres dimensiones:

- **El nivel de compartición**: Presenta tres alternativas:
    - *Inexistencia*, es decir, cada aplicación y sus datos se ejecutan en una computadora con ausencia total de comunicación con otros programas u otros datos.
    - *Se comparten solo los datos y no los programas*, en tal caso, existe una réplica de las aplicaciones en cada máquina y los datos viajan por la red.
    - *Se reparten datos y programas*, dado un programa ubicado en un determinado sitio, este puede solicitar un servicio a otro programa localizado en un segundo lugar, el cual podrá accesar a los datos situados en un tercer emplazamiento

- **Las características de acceso a los datos**: existen dos alternativas: el modo de acceso a los datos que solicitan los usuarios puede ser *estático*, es decir, no cambiará a lo largo del tiempo, o bien *dinámico*, el cual consiste en el número de variaciones que sufre a lo largo de tiempo.

- **El nivel de conocimiento de las características de acceso**: una posibilidad es, evidentemente, que los diseñadores carezcan de información alguna sobre como los usuarios acceden a la base de datos. Lo más práctico sería conocer con detenimiento la forma de acceso de los usuarios.

Las bases de datos distribuidas pueden clasificarse de dos maneras: *bases de datos distribuidas homogéneas y bases de datos distribuidas heterogéneas.*

En las ***bases de datos homogéneas***, todos los sitios tienen idéntico software gestor de bases de datos, son conscientes de la existencia de los demás sitios y acuerdan cooperar en el procesamiento de las solicitudes de los usuarios. En estos sistemas, los sitios locales renuncian a una parte de su autonomía en cuanto a su derecho a modificar los esquemas o el software del sistema gestor de base de datos. Ese software también debe de cooperar con los demás sitios en el intercambio de la información sobre las transacciones para hacer posible el procesamiento de las transacciones entre varios sitios.

En las ***bases de datos heterogéneas*** se presenta el caso de que sitios diferentes puede que utilicen esquemas y software diferente. Puede que unos sitios no sean conscientes de la existencia de los demás y puede que solo proporcionen facilidades limitadas para la cooperación de las transacciones. La diferencia entre los esquemas suele constituir un problema importante para el procesamiento de las consultas, mientras que la diferencia del software supone un inconveniente para el procesamiento de transacciones que tengan acceso a varios sitios.

Para almacenar una relación en una base de datos distribuida existen dos enfoques:

- **Réplica:** el sistema conserva réplicas o copias idénticas de la relación y guarda cada réplica en un sitio diferente. La alternativa a las réplicas es almacenar sólo una copia de la relación.

- **Fragmentación:** el sistema divide la relación en varios fragmentos y guarda cada fragmento en un sitio diferente.

La fragmentación y la réplica pueden combinarse. Las relaciones pueden dividirse en varios fragmentos y puede haber varias réplicas de cada fragmento.

Si la relación se replica, se guarda una copia de dicha relación en dos o más sitios. En el caso mas extremo se tiene una réplica completa, en la que se guarda una copia en cada sitio del sistema. Hay varias ventajas y desventajas de las réplicas:

- Si alguno de los sitios que contiene la relación falla, la relación puede hallarse en otro sitio distinto, por tanto, el sistema puede seguir procesando las consultas que impliquen a la relación pese al fallo del sitio.
- En caso de que la mayoría de los accesos a la relación solo resulten en la lectura de la relación, varios sitios pueden procesar en paralelo las lecturas que impliquen a la relación. Cuando más réplica haya, mayor será la posibilidad que los datos necesarios se hallen en el sitio en que se ejecuta la transacción. Por tanto, la réplica de los datos minimiza el movimiento de los datos entre los sitios.
- El sistema debe asegurar que todas las réplicas de la relación sean constantes, en caso contrario, pueden producirse errores de cómputo.
- Siempre que se actualice la relación, se debe de propagar la actualización a todos los sitios que contienen réplicas. El resultado es una sobrecarga incrementada.

Dado que una relación se corresponde esencialmente con una tabla y la cuestión consiste en dividirla en fragmentos menores, inmediatamente surgen dos alternativas lógicas para llevar al cabo el proceso:

Si la relación se fragmenta, se divide en varios fragmentos. Estos fragmentos contienen suficiente información como para permitir la reconstrucción de la relación original. Hay dos esquemas diferentes de fragmentación de las relaciones: *la fragmentación horizontal y la fragmentación vertical:*

- La ***fragmentación vertical*** divide la relación descomponiendo el esquema de la relación. Dicha fragmentación se basa en los atributos de la relación para efectuar la división.

- En la ***fragmentación horizontal***, la relación se divide en varios subconjuntos. Cada tupla de la relación debe pertenecer como mínimo a uno de los fragmentos, de modo que se pueda reconstruir la relación original, si fuese necesario. La fragmentación horizontal suele utilizarse para conservar las tuplas en los sistemas en que más se utilizan, para minimizar la transferencia de datos.

Estos dos tipos de partición podrían considerarse los fundamentales y los básicos. Sin embargo, existen otras alternativas, Fundamentalmente se habla de la fragmentación híbrida o mixta, cuando el proceso de partición hace uso de los dos tipos anteriores.

- ***La Fragmentación mixta*** puede llevarse a cabo de tres formas diferentes:
  - Partición VH.- Desarrollando primero la fragmentación vertical y, posteriormente, aplicando la fragmentación horizontal sobre los fragmentos verticales.
  - Partición HV.- Aplicando primero una división horizontal para luego, sobre los fragmentos generados, desarrollar una fragmentación horizontal.
  - De forma directa.- considerando la semántica de las transacciones.

Otro enfoque distinto y relativamente nuevo, consiste en aplicar sobre una relación, de forma simultánea y no secuencial, las fragmentaciones horizontal y vertical; en este caso, se genera una rejilla y los fragmentos formaran las celdas de esa rejilla, cada celda será exactamente un fragmento vertical y un fragmento horizontal, lográndose así un grado de fragmentación más alto, mas no por ello, la descomposición resultará más eficiente.

No se debe exigir a los usuario de lo sistemas distribuidos de bases de datos que conozca la ubicación física de los datos ni el modo en que se puede tener acceso a ellos en un sitio local concreto. Esta característica denominada ***transparencia de los datos***, puede adoptar varias formas:

- **Transparenta de la fragmentación**: No se exige a los usuarios que conozcan el modo en que se ha fragmentado la relación.
- **Transparencia de la réplica**: Los usuarios ven cada objeto de datos como lógicamente único. Puede que el sistema distribuido replique los objetos para incrementar el rendimiento del sistema o la disponibilidad de los datos. Los usuarios no deben preocuparse por los objetos que se hayan replicado ni por la ubicación de esas réplicas.
- **Transparencia de la ubicación**: No se exige a los usuarios que conozcan la ubicación física de los datos. El sistema distribuido da bases de datos debe hallar los datos siempre que la transacción del usuario facilite el identificador de los datos.

Los elementos de los datos como las relaciones, las réplicas y los fragmentos deben tener nombres únicos, por lo que todos los nombres tienen que registrarse en un servidor de nombres central. Este ayuda a asegurar que el mismo nombre no se utilice para elementos de datos diferentes. También se puede utilizar el servidor de nombres para ubicar un elemento de datos, dado el nombre del elemento.

Un aspecto importante en el diseño de la distribución es la cantidad de factores que contribuyen a un diseño óptimo. La organización lógica de la base de datos, la localización de las aplicaciones, las características de acceso de las aplicaciones a la base de datos y las características del sistema de cada sitio, tienen una decisiva influencia sobre la distribución. La información necesaria para el diseño de la distribución de la aplicación, la información sobre la red y la información sobre las computadoras entre sí. Las dos últimas son de carácter cuantitativo y servirán, principalmente, para desarrollar el proceso de asignación.

3.6 Software Recomendado.

Varios son los manejadores de bases de datos que existen, ya sea comerciales o de código abierto, cada uno con sus propias características y ventajas, de entre los cuales podemos mencionar los siguientes:

- **MySQL**[28] **:** Es un servidor de bases de datos multiusuario, concretamente, el más rápido en entornos web. SQL es el lenguaje de bases de datos más popular y estandarizado del mundo. MySQL es una implementación cliente/servidor que consiste en un demonio mysqld y varios programas clientes y librerías Licencia GPL a partir de la versión 3.23.19

MySQL es un gestor de bases de datos SQL. Un dato histórico interesante es que IBM empezó a comercializar en 1981 el SQL y desde entonces este producto ha tenido un papel importante en el desarrollo de la bases de datos relacionales. IBM propuso y fue aceptada, una versión de SQL al Instituto de Estándares Nacional Americano (ANSI) y desde entonces es utilizada de forma generalizada en las bases de datos relacionales. En 1983 nació DB2 la más popular (por lo menos en los grandes ordenadores) de las bases de datos de este tipo hasta estos mismos momentos.

[28] Mysql, Edición Especial. DOBOIS Paúl. Prentice Hall, 2002

Esta base de datos es considerada (en su propia documentación así lo reseña) como la más rápida y robusta tanto para volúmenes de datos grandes como pequeños (siempre, claro está, comparada con las de su categoría).

Su principal objetivo de diseño fue la VELOCIDAD. Se sacrificaron algunas características esenciales en sistemas más "serios" con este fin.

Otra característica importante es que consume MUY POCOS RECURSOS, tanto de CPU como de memoria.

Ventajas:

- Mayor rendimiento. Mayor velocidad tanto al conectar con el servidor como al servir selects y demás.
- Mejores utilidades de administración (backup, recuperación de errores, etc).
- Aunque se cuelgue, no suele perder información ni corromper los datos.
- Mejor integración con PHP.
- No hay límites en el tamaño de los registros.
- Mejor control de acceso, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos

- **PostgreSQL**[29]: se coloca en la categoría de las Bases de Datos conocidas como objeto-relacionales. Ha generado algunas características que son propias del mundo de las bases de datos orientadas a objetos. Esto ha llevado a que, algunas Bases de Datos comerciales han incorporado recientemente estas ventajas en las que PostgreSQL fue pionera.

En 1986 *Michael Stonebraker* de Berkeley continuó el desarrollo del código de *Ingres* para crear un sistema de bases de datos objeto-relacionales llamado *Postgres*.

[29] PostgreSQL Práctico, WORSLEY, DRAKE.. This document is distributed under the GNU General Public Licence.

En 1996, debido a un nuevo esfuerzo de código abierto y a la incrementada funcionalidad del software, *Postgres* fue renombrado a *PostgreSQL*, tras un breve periodo como *Postgres95*. El proyecto *PostgreSQL* sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.

PostgreSQL está considerado como la base de datos de código abierto más avanzada del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle

PostgreSQL; como un motor de BD Relacional y Orientada a Objetos que cuenta con todas las características de un motor de BD comercial, incorpora conceptos que permite al usuario extender fácilmente el sistema.

PostgreSQL puede ser integrada al ambiente Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes. Permite desarrollar o migrar aplicaciones desde Access, Visual Basic, Foxpro, Visual Foxpro, C/C++ Visual C/C++, Delphi, etc., para que utilicen a PostgreSQL como servidor de BD; Por lo expuesto PostgreSQL se convierte en la mejor alternativa de soporte para el desarrollo de las aplicaciones de su empresa o institución

Tomando en cuenta esas características y sumando la estabilidad, performance, disponibilidad y la eficiencia de un sistema operativo como **Linux**, muchas empresas e instituciones lo están adoptando como servidor de Bases de Datos

Además cuenta con el servicio de soporte comercial, que refleja la medida de las necesidades de su empresa o institución. La oferta esta orientada a otorgarle la mayor flexibilidad que se adapte a los tiempos y conveniencia de sus procesos productivos, personalizando de esta manera la solución que mas se adapte a sus necesidades

- **SQL SERVER**[30] : Microsoft SQL Server constituye un lanzamiento determinante para los productos de bases de datos de Microsoft, continuando con la base sólida establecida por SQL Server 6.5. Como la mejor base de datos para Windows NT, SQL Server es el SGBD Relacional de elección para una amplia gama de clientes corporativos y Proveedores Independientes de Software (ISVs) que construyen aplicaciones de negocios. Las necesidades y requerimientos de los clientes han llevado a la creación de innovaciones de producto significativas para facilitar la utilización, escalabilidad, confiabilidad y almacenamiento de datos.

Se adapta a las necesidades de la empresa, soportando desde unos pocos usuarios a varios miles. Empresas centralizadas u oficinas distribuidas, replicando cientos de sites Microsoft SQL Server es la mejor base de datos para Windows NT Server. Posee los mejores registros de los benchmarks independientes (TCP) tanto en transacciones totales como en costo por transacción.

Con un completo interfaz gráfico que reduce la complejidad innecesaria de las tareas de administración y gestión de la base de datos, Visual Basic, Visual C++, Visual J++ y muchas otras herramientas son compatibles con Microsoft SQL Server.

Para Datos distribuidos y su replicación [31]:

- Llamadas a procedimientos remotos servidor-a-servidor (procedimientos almacenados remotos).
- Replicación asíncrona o contínua basada en registros, o sincronización planificada de tablas point-in-time.
- Configuración de replicación gráfica y características de gestión.
- Replicación de subscriptores ODBC, incluyendo IBM DB2, ORACLE, SYBASE y Microsoft Access.
- Replicación de tipos de datos Texto e Imagen.

[30] Microsoft SQL Server al Descubierto. BJELETICH, MABLE. Prentice Hall
[31] SQL Server, Manual de Referencia. GOFFMAN Gayle, McGraw Hill, 2001

- **ORACLE**[32]**:** El manejador de Base de datos ORACLE, surgió a final de los años 70 y principio de los años 80. George Koch y su equipo de técnicos fue el primero en desembarcar en el terreno de Oracle en 1982, durante un proceso de evaluación de sistema de gestión de base de datos para una importante aplicación comercial que el equipo estaba diseñando y construyendo. Cuando termino, la evaluación fue descrita en Computer World como el estudio más severo de SGBD que se había hecho. El estudio fue tan riguroso con los vendedores cuyos productos habían estudiado con anterioridad, que la prensa hizo eco de sus palabras en lugares tan distantes como Nueva Zelandia y en publicaciones muy alejadas del campo como el Christian Sciencia Monitor.

Oracle soporta dos tipos de almacenamiento, por carácter (RAW) o por bloques (Files System), generalmente es recomendable que los sean colocados en Raw Divice.

Raw Divice: es un dispositivo de caracteres disponibles en algunos sistemas operativos el cual es asignado directamente a Oracle.

Oracle corre más rápidamente con Raw Divice que con Files System, por varias razones:

1. El I/O (Input/Output) es realizado directamente en el disco por Oracle, independientemente del sistema operativo.
2. El buffer cache del sistema del sistema operativo es dejado a un lado.
3. Los buffers del sistema operativo y de Oracle son independiente entre sí.

Con la intención de evitar la contención de los discos, se debe considerar la instalación de Oracle en dispositivos separados, especialmente si se tienen varios discos, y más esencialmente, si se poseen más de una controladora de disco.

[32] Oracle 8. LONEY Kevin, McGraw Hill, 2002

Oracle posee igual interacción en todas las plataformas (Windows, Unix, Macintosh y Mainframes). Estos porque más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de Sistemas Operativos.

Oracle soporta bases de datos de todos los tamaños, desde severas cantidades de bytes y gigabytes en tamaño.

Oracle provee salvar con seguridad de error lo visto en el monitor y la información de acceso y uso.

Oracle soporta un verdadero ambiente cliente servidor. Este establece un proceso entre bases de datos del servidor y el cliente para la aplicación de programas.

Por sus características y aunado a la ventaja que ofrece al ser "software libre"; se tomó la decisión de seleccionar Mysql para el desarrollo del caso práctico, el cual se presentará más adelante en este trabajo.

4.0 BASES DE DATOS PARALELAS Y PROCESAMIENTO EN PARALELO.

4.1 Introducción.

Desde hace unos años, los sistemas paralelos de base de datos han venido comercializándose con éxito por prácticamente todos los fabricantes de bases de datos, debido a que este tipo de sistemas mejoran la velocidad de procesamiento de entrada y salida mediante la utilización de computadoras y discos en paralelo.

Cada vez son más comunes las maquinas paralelas, lo que hace que cada vez sea más importante el estudio de los sistemas paralelos de bases de datos. La fuerza que ha impulsado a este tipo de sistemas se centra en la demanda de aplicaciones que han de manejar bases de datos extremadamente grandes; del orden de los Terabytes o que tienen que procesar un número enorme de transacciones por segundo. Los sistemas de base de datos centralizados o cliente/servidor no son suficientemente potentes como para poder soportar tales aplicaciones.

El cambio que se ha presentado en relación a este tipo de sistemas se sustenta en las siguientes tendencias:

- En el procesamiento paralelo se realizan muchas operaciones simultáneamente, mientras que en el procesamiento secuencial, los distintos pasos computacionales han de ejecutarse en serie.
- Los requisitos transaccionales de las empresas han aumentado con el uso creciente de las computadoras.
- El crecimiento de Internet ha creado muchos sitios con millones de visitantes, y las cantidades crecientes de los datos recogidos por los visitantes ha producido bases de datos en muchas empresas.
- Las empresas utilizan volúmenes crecientes de datos, para planificar sus actividades y sus tarifas.

- La naturaleza orientada a conjuntos de las consultas de bases de datos se presta de forma natural a la paralelización. Varios sistemas han demostrado la potencia y dimensionalidad del procesamiento paralelo de consultas.
- Al abaratarse los microprocesadores, las máquinas paralelas se han vuelto mucho más accesibles.

Varios son los modelos de computación paralela que han surgido con la idea de cubrir el vacío ente hardware y software que permita su uso. Uno de los más promisorios es el modelo *Bulk Synchronous Parallel (BSP)*, propuesto en 1990.

Tiene como principales objetivos permitir el desarrollo de software escalable e independiente de la arquitectura y proveer un entorno de trabajo simple y práctico para computadoras paralelas de propósito general. Las características claves de BSP son el tratamiento del medio de comunicación como una red abstracta totalmente conectada y un modelo de costo de comunicación y sincronización independiente y explicito.

En los últimos años se ha demostrado la utilidad práctica del modelo BSP en varias aplicaciones de computación paralela; en particular, ha sido ampliamente utilizado en la paralelización eficiente de algoritmos no numéricos y estructuras de datos.

4.2 Procesamiento en Paralelo.

Procesamiento en paralelo es **el método de fraccionar o dividir grandes problemas en pequeñas unidades, tareas o cálculos que pueden ser resueltas en paralelo**. El procesamiento en paralelo ha emergido como una tecnología disponible para la computación moderna. Años atrás se ha testificado una creciente aceptación y la adopción de procesamiento en paralelo, tanto para el cómputo de alto rendimiento científico como para aplicaciones de cómputo de propósito general, esto gracias al desarrollo del procesamiento masificado en paralelo.

En el procesamiento paralelo se realizan muchas operaciones simultáneamente, mientras que en el procesamiento secuencial, los distintos pasos computacionales han de ejecutarse de manera secuencial.

El paralelismo se refiere ***a la reducción del tiempo necesario para recuperar relaciones del disco dividiéndolas en varios discos.*** [33]

4.3 Sistemas de Base de Datos Paralelas.

Los sistemas paralelos de base de datos constan de varios procesadores y varios discos conectados a través de una red de interconexión de alta velocidad. Para medir el rendimiento de los sistemas de base de datos existen 2 medidas principales: la primera es *la productividad (throughput)* que se entiende como **el número de tareas que pueden completarse en un intervalo de tiempo determinado**. La segunda es el *tiempo de respuesta (response time)* que es la **cantidad de tiempo que necesita para completar una única tarea a partir del momento en que se envíe.**[34] Un sistema que procese un gran número de pequeñas transacciones puede mejorar su productividad realizando muchas transacciones en paralelo. Un sistema que procese transacciones más largas puede mejorar tanto su productividad como sus tiempos de respuesta realizando en paralelo cada una de las subtareas de cada transacción. Los sistemas paralelos representan un intento por construir máquinas más rápidas y menos costosas.

Las ganancias en este tipo de SGBD se pueden dar en términos de **velocidad** *(menor tiempo de ejecución para una tarea dada)* y **ampliabilidad** *(capacidad de procesar tareas más largas en el mismo tiempo)*. La ampliabilidad es normalmente el factor más importante para medir la eficiencia de un sistema paralelo de bases de datos.

El objetivo del paralelismo en los sistemas de base de datos suele ser *asegurar que la ejecución del sistema continuará realizándose a una velocidad aceptable, incluso en el caso de que aumente el tamaño de la base de datos o el número de transacciones.*[35] El incremento de la capacidad del sistema mediante el incremento del paralelismo proporciona a una empresa un modo de crecimiento más suave que el de remplazar sistemas centralizados por una máquina más rápida. Sin embargo, cuando se utiliza la ampliabilidad debe atenderse también a los valores de rendimiento absoluto.

---

[33] Algoritmica del paralelismo, RAYNAL Michael, Ed. Omega

[34] Fundamentos de Bases de Batos (4ta. Edición). SILVERSCHATZ, Avi., KORTH, Hank., S, SUNDARSHAN.. McGraw Hill, 2001

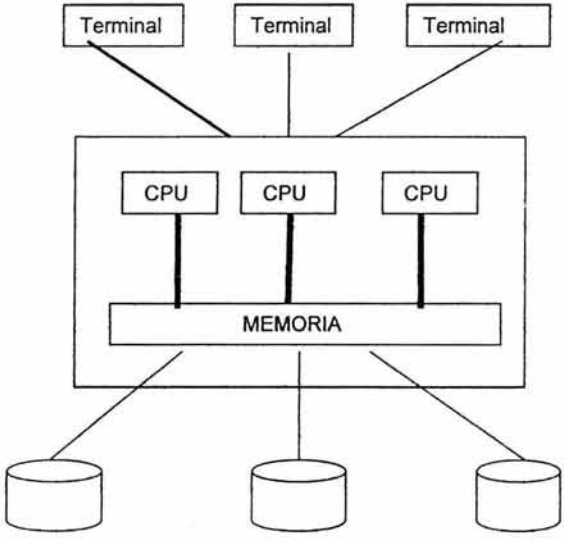
[35] Fundamentos de Bases de Batos (4ta. Edición). SILVERSCHATZ, Avi., KORTH, Hank., S, SUNDARSHAN.. McGraw Hill, 2001

Existen varios modelos de arquitecturas para maquinas paralelas, los más mencionados son:

4.3.1 Memoria Compartida

El primer tipo de sistemas de base de datos paralelas en donde ***todos los procesadores comparten una memoria común***. En este tipo de sistema, una computadora es simultáneamente activada por varios CPU's, los cuales comparten acceso a una memoria y a una interfase de disco comunes.

Este es el tipo de arquitectura paralela que más se acerca al modelo tradicional de CPU con un solo procesador. El desafío de este diseño está en conseguir en N-tiempo un trabajo mejor desempeñado con N-CPU's que el que se puede alcanzar con un servidor con el mismo poder.



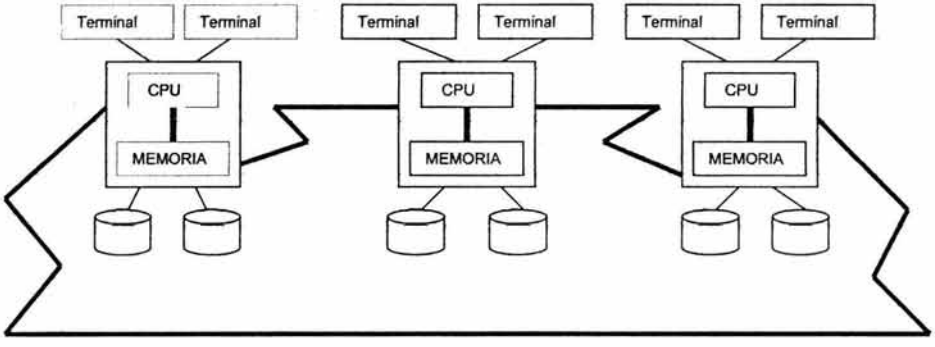
Arquitectura de memoria compartida

Sin embargo, el diseño debe tomar precauciones especiales ya que los diferentes CPU deben de tener el mismo acceso a la memoria común, y algunos de los datos recobrados por un CPU no son modificados por otro que este actuando en paralelo. Debido a que el acceso a memoria usa un mecanismo de alta velocidad, que es difícil de particionar sin perder eficiencia, los problemas de acceso de memoria toman mayor dificultad en la medida que se incrementa el número de computadoras conectadas en paralelo.

4.3.2 Sin Compartimiento:

El segundo tipo de arquitectura, en el cual un conjunto de CPU's actúan en paralelo cada uno con su propia memoria y su propio disco, es decir, ***los procesadores no comparten ni memoria ni disco.***

En este tipo de configuración, los equipos comparten responsabilidades sobre los servicios de la base de datos usualmente fisurados entre sí mismos y desempeñan transacciones y consultas para dividir el trabajo, enviando mensajes sobre una red de alta velocidad.



Arquitectura sin compartimiento

Otras arquitecturas que se mencionan son:

Disco Compartido: Todos los procesadores comparten una disposición de discos común.
Jerárquico: Compartimiento tanto de memoria como de disco.

Dado que los sistemas paralelos de base de datos se utilizan para procesar grandes volúmenes de datos y para procesar consultas de ayuda a las decisiones básicas en dichos datos, su funcionamiento se basa en el almacenamiento de los datos y el procesamiento de las consultas.

4.4 Almacenamiento de los datos.

La forma más frecuente de división de los datos en un entorno de base de datos paralela es **la división horizontal**.

En la división horizontal, las tuplas de las relaciones se dividen o desagrupan entre varios discos, de modo que cada tupla resida en un disco.

Podemos señalar las siguientes estrategas de división:

- **Turno rotatorio**: La relación se explora en cualquier orden y la i-esima tupla se envía al disco numerado D i mod n. El esquema de turno rotatorio asegura una distribución homogénea de las tuplas entre los discos es decir cada disco tiene aproximadamente el mismo numero de tuplas que los demás.

Este esquema se adapta perfectamente a las aplicaciones que desean leer secuencialmente la relación completa para cada consulta a diferencia de división por asociación o por rangos

- **División por asociación**: En esta estrategia de desagrupación; uno o mas atributos del esquema de la relación dada se designan como atributos de la división , se escoge una función de asociación cuyo rango sea { 0,1,.....,n-1} . cada tupla de la relación original se asocia en términos de los atributos de la división, si la función de asociación devuelve i, la tupla se ubica en el disco D i+1.

Este esquema se adapta mejor a las consultas concretas basadas en el atributo de división. Por ejemplo si se divide una relación en términos del atributo numero-teléfono, se puede responder a la consulta "Buscar el registro del empleado con numero teléfono 53100290 aplicado a la función de división por asociación 53100290 y buscando luego en ese disco. Dirigir la consulta a un solo disco ahorra el costo inicial de iniciar una consulta en varios discos y deja a los demás discos libres para realizar otras consultas.

La división por asociación también es útil para las exploraciones secuenciales de la relación completa.

Si la función de asociación es una buena función aleatoria y los atributos de la división forman una clave de la relación, el número de tuplas en cada uno de los discos es aproximadamente el mismo, sin mucha varianza.

Por tanto, el tiempo empleado para explorar la relación es aproximadamente n/1 del tiempo necesario para explorar la relación en un sistema de disco único. Pero este no se adapta bien a búsquedas concretas ni a búsquedas por rangos.

- **División por rangos**: Esta estrategia distribuye rangos contiguos de valores a los atributos a cada disco. Se escoge un atributo de división , A, como vector de división sea { v0,v1,v2,.....v n-2} el vector de división tal que , si i<j entonces v1 <j1 la relación se divide como sigue . Considérese una tupla t tal que [A] =x. Si x < v0 entonces t se ubica en el disco d0. Si x $\geq$ v n-2 entonces t se ubica en el disco D n-1. Si V1 $\leq$ x< v i+1 entonces t se ubica en el disco D i+1.

Por ejemplo la división por rangos con tres discos numerados del 0 al 2 puede asignar tuplas con valores menores que 5 al disco 0, con valores entre 5 y 40 al disco 1, y con valores mayores a 40 al disco 3.

Este esquema se adapta bien a las consultas concretas y de rangos basadas en el atributo división. Para las consultas concretas, se puede consultar el vector de división para encontrar el disco en el que reside la tupla.

Para las consultas de rangos de consulta, el vector de división se consulta para hallar el rango de discos en que pueden residir las tuplas. En ambos casos la búsqueda se limita exactamente a aquellos discos que pudieran tener tuplas de interés.

Una ventaja es que, si solo hay unas pocas tuplas en el rango consultado, la consulta se suele enviar a un disco, en vez de hacerlo a todos. Dado que se pueden utilizar otros discos para responder a otras consultas, la división de rangos da lugar a una mayor productividad de consultas al tiempo de que se mantiene un buen tiempo de respuesta, por otro lado si hay muchas tuplas en el rango consultado, hay que recuperar muchas tuplas de pocos discos, lo cual origina un cuello de botella en esos discos.

4.5 Procesamiento de las consultas.

Para hablar del procesamiento de las consultas en un sistema de base de datos paralelo, se deben de considerar dos conceptos esenciales; **el paralelismo entre consultas** y **el paralelismo en consultas**.

En el paralelismo entre consultas, *se ejecutan en paralelo diferentes consultas entre si o transacciones.* El paralelismo en consultas se refiere *a la ejecución en paralelo de una única consulta en varios procesadores y discos.*

El paralelismo entre consultas es la forma más sencilla de paralelismo que se permite en los sistemas de base de datos, esencialmente en sistemas paralelos de memoria compartida. Los sistemas de base de datos diseñados para sistemas con único procesador pueden utilizarse en arquitecturas paralelas de memoria compartida con pocos cambios o con ninguno, dado que incluso los sistemas secuenciales de base de datos permiten el procesamiento concurrente. Las transacciones que se habían realizado en tiempo compartido en una máquina secuencial se realizan en paralelo en la arquitectura paralela de memoria compartida.

Permitir el paralelismo entre consultas es más común en las arquitecturas de disco compartido y sin compartimiento. Los procesadores tienen que realizar algunas tareas, como los bloqueos y el registro histórico, de forma coordinada, lo cual exige que se intercambien mensajes. Los sistemas con arquitectura paralela también deben asegurar que dos procesadores no actualicen simultáneamente los mismos datos de una manera independiente.

El uso de paralelismo en consultas es importante para acelerar las consultas de ejecución largas. El paralelismo entre consultas no ayuda en esta labor, dado que cada consulta se realiza de manera secuencial, se puede hacer una consulta paralela haciendo paralelas las operaciones que la forman.

4.5.1 Paralelismo en Operaciones.

Dado que las operaciones relacionales tratan con relaciones que contienen grandes conjuntos de tuplas, se pueden paralelizar las operaciones ejecutándolas en paralelo en subconjuntos diferentes de las relaciones. Dado que el número de tuplas en una relación pueden ser grandes, el grado de paralelismo es enorme. Por tanto, el paralelismo en operaciones es natural en los sistemas de base de datos.

Se puede acelerar el procesamiento de consultas haciendo paralela la ejecución de cada una de las operaciones, como pueden ser:

- **La Ordenación en Paralelo:** Si la relación se ha dividido por rangos basándose en los atributos por los que se va a ordenar, se pueden ordenar por separado cada partición y concentrar los resultados para obtener la relación completa ordenada. Dado que las tuplas se hallan divididas en n discos, el tiempo necesario para leer la relación completa se reduce gracias al acceso en paralelo.
- **La Selección:** La cual puede llevarse a cabo en un solo procesador o en cada procesador cuya partición se solape con el rango de valores especificado, dependiendo del valor y el atributo especificado.

- **La Eliminación de Duplicados:** Esta puede llevarse a cabo por ordenación; puede utilizarse cualquiera de las técnicas de ordenación en paralelo, con la optimización para eliminar los datos duplicados durante la ordenación tan pronto como los encuentre.
- **La Proyección.** Se puede llevar a cabo sin eliminación de duplicados, según se leen en paralelo las tuplas del disco.
- **La Agregación**: Puede considerarse una operación. Se puede paralelizar la operación dividiendo a la relación basándose en los atributos de agrupación y procesando luego localmente los valores de agregación en cada procesador.

4.5.2 Paralelismo entre operaciones.

Hay dos formas de paralelismo entre operaciones; el paralelismo de encauzamiento y el paralelismo independiente.

El paralelismo de encauzamiento supone una importante fuente de economía de procesamiento para el procesamiento de consultas de base de datos. En el encauzamiento, las tuplas resultado de una operación las consume una segunda operación, incluso antes de que la primera operación haya producido el conjunto completo de tuplas de su resultado.

La ventaja principal de la ejecución encauzada de las evaluaciones secuenciales es que se puede ejecutar una secuencia de operaciones de este tipo sin escribir en el disco ninguno de los resultados intermedios.

En los sistemas paralelos, el encauzamiento puede utilizarse como fuente de paralelismo, lo cual resulta útil con un número pequeño de procesadores, pero no puede extenderse bien, debido a que las cadenas de cause no suelen lograr la longitud suficiente para proporcionar un alto grado de paralelismo ya que no es posible encauzar los operadores de relación que no producen resultados hasta que se ha tenido acceso a todas las entradas. Por consiguiente; cuando el grado de paralelismo es elevado, la importancia del encauzamiento como fuente del paralelismo es secundaria.

La razón fundamental para utilizar el encauzamiento es que las ejecuciones encauzadas puedan escribir en el disco los resultados intermedios.

El paralelismo independiente sostiene que las operaciones en las expresiones de las consultas que son independientes pueden ejecutarse en paralelo. Al igual que el paralelismo encauzado, el paralelismo independiente no proporciona un alto grado de paralelismo y es menos útil en sistemas con un elevado nivel de paralelismo, aunque resulta más efectivo con un menor grado de paralelismo.

Dado que los sistemas paralelos de base de datos de gran escala se utilizan para almacenar grandes cantidades de datos, principalmente, y para procesar consultas de ayuda a las decisiones basadas en dichos datos, estos temas son los más importantes en los sistemas paralelos.

4.6 Diseño de Sistemas Paralelos de Base de Datos.

Al momento de empezar a diseñar un sistema paralelo, no solo se debe de considerar el volumen de la información y las operaciones que se realizaran con estas; sino también se deben de abordar aspectos tales como lo son: el poder de recuperación frente al fallo de algún nodo, la reorganización interactiva de los datos y los cambios de los esquemas.

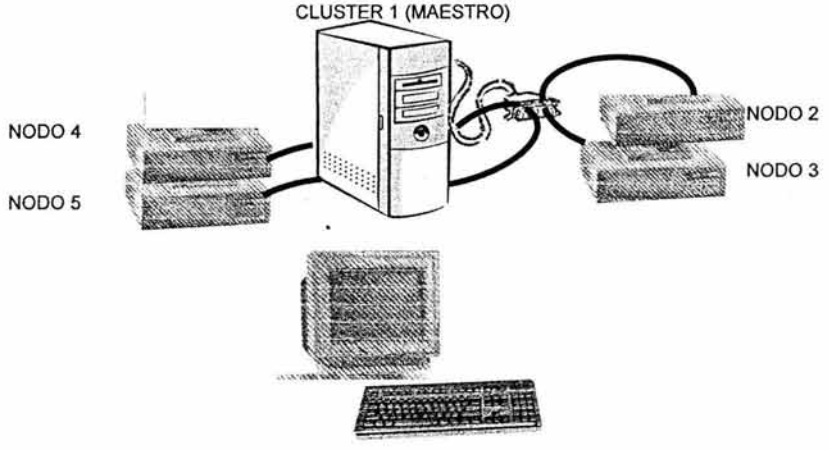
Con un gran número de procesadores y de discos la probabilidad de que al menos un procesador y un solo disco funcionen mal es relativamente mayor que en un sistema con un solo disco y un único procesador. Un sistema paralelo mal diseñado dejará de funcionar si cualquier nodo falla.

En los sistemas paralelos, los datos se replican en al menos dos procesadores; si falla un procesador, se puede seguir teniendo acceso desde los demás procesadores a los datos que guarda. El sistema hace un seguimiento de los procesadores con fallos y distribuye el trabajo entre los que funcionan. Las peticiones de los datos guardados en el emplazamiento con fallo se desvían automáticamente a los emplazamientos que las copias de seguridad que guardan una réplica de los datos.

Cuando se manejan grandes cantidades de información, del orden de Terabytes, las operaciones sencillas, como lo son la creación de índices, los cambios en los esquemas o como añadir una columna a una relación, pueden tardar algo de tiempo, por lo que no es posible que los sistemas de base de datos no estén disponibles mientras se llevan a cabo dichas operaciones. Los sistemas paralelos deben de permitir que tales operaciones se lleven a cabo interactivamente, es decir, mientras el sistema ejecuta otras transacciones.

4.7 Plataforma a Utilizar.

Una vez que se ha expuesto de manera clara las características de la tecnología de clusters y que se mencionaron las propiedades y funciones principales de las bases de datos paralelas, a continuación se describe la plataforma que se ha utilizado para realizar la instalación de una base de datos paralela en un cluster Linux:



Características del Cluster:

Cinco nodos.

- Un nodo maestro
    - Procesador Pentium Cérelon a 400MHz.
    - 64 MB en RAM
    - Tarjeta de red 10/100 3com
    - Switch a 100Mbps

- Cuatro nodos restantes
    - Misma Instalación.
    - Procesador Pentium a 200MHz en cada nodo.
    - 4.0 GB en Disco Duro en cada nodo
    - 32 MB en RAM en cada nodo
    - Tarjeta de red 10/100 3com

Sistema Operativo LINUX MANDRAKE

- Versión 9.1
- KERNEL 2.4.21
- Pingroot 1,2,3,4

Balanceo de procesos por OPENMOSIX
Lenguaje C más API de MYSQL.

4.8 Mosix y Open Mosix

Mosix es una herramienta diseñada para realizar balanceo de cargas en el cluster de forma totalmente transparente de forma tal que los nodos del cluster se comportan como una sola máquina, y de esta manera incrementar el aprovechamiento de cada uno de los nodos.

También cuenta con herramientas para observar lo que esta pasando en cada nodo del cluster, es decir que podemos observar el comportamiento del cluster. Para observar el comportamiento del cluster se utiliza el monitor de carga de Mosix llamado mon.

A diferencia de los clusters Beowulf clásicos, el Mosix se integra en el kernel y una vez configurado es prácticamente transparente y permite ejecutar cualquier software en el cluster.

Por ultimo cabe destacar el balanceo de carga automático entre los nodos, el cual a la larga nos será muy útil.

OpenMosix es una especie de modificación o conjunto de parches del kernel de Linux que permite una distribución automática y transparente de la carga de tareas a lo largo de las máquinas que forma el cluster. Y todo ello sin tener que modificar el código del programa

De Mosix y openMosix podemos mencionar lo siguiente:

- Nace en los 80's, y en 2001 se divide Mosix y openMosix
- OPENMOSIX Es una adición al Kernel de Linux que convierte una red de computadores en un cluster (supercomputador).
- Con OPENMOSIX No se requiere ningún desarrollo adicional.
- Los programas convencionales pueden ejecutarse utilizando la adición al Kernel
- Los nodos pueden incluirse y/o excluirse del cluster de manera transparente

Características de OpenMosix:

- Algoritmo avanzado para balanceo de carga (Adaptativo)
- Migración dinámica y transparente de procesos
- Sistemas de archivos en Cluster
- Transparencia total al usuario y a las aplicaciones

Ventajas:

- No se requieren paquetes extra.
- No son necesarias las modificaciones en el código.

Desventajas:

- Es dependiente del Kernel.
- No migra todos los procesos, llega a presentar limitaciones en el funcionamiento.
- Problemas con memoria compartida entre procesos.

Con openMosix se puede lanzar un proceso en una computadora y ver si se ejecuta en otra, en el seno del cluster. Cada proceso tiene su único nodo raíz (UHN, unique home node) que se corresponde con el que lo ha generado.

De entre las propiedades compartidas entre Mosix y openMosix podemos destacar el mecanismo de migración, en el que puede migrarse cualquiera proceso a cualquier nodo del cluster de forma completamente transparente al proceso migrado.

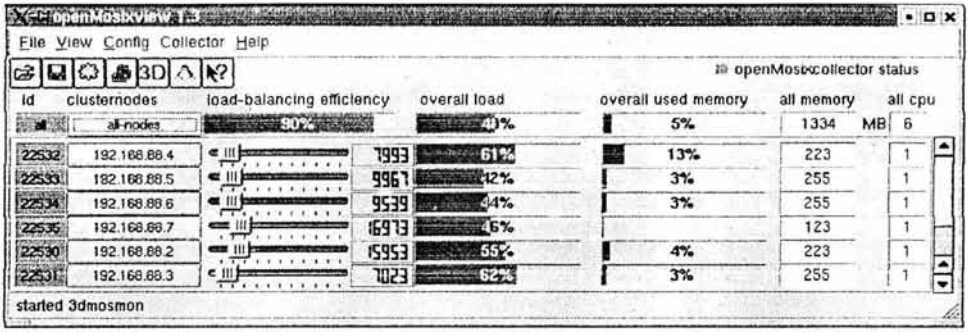
El concepto de migración significa que un proceso se divide en dos partes: la parte del usuario y la del sistema.

La parte, o área, de usuario será movida al nodo remoto mientras el área de sistema espera en la raíz. OpenMosix se encargará de establecer la comunicación entre estos 2 procesos.

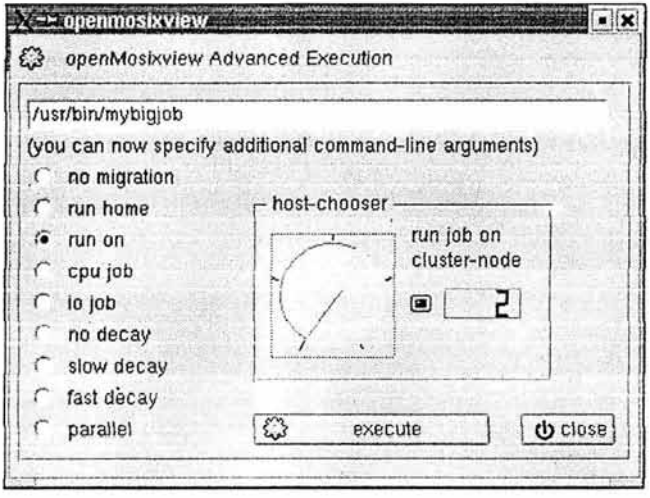
La idea de este modelo es que la distribución de tareas en el cluster la determina openMosix de forma dinámica, conforme se van creando tareas. Cuando un nodo está demasiado cargado, las tareas que se están ejecutando pueden migrar a cualquier otro nodo del cluster. Así desde que se ejecuta una tarea hasta que ésta muere, podrá migrar de un nodo a otro, sin que el proceso sufra mayores cambios.

También hay que señalar según se comenta que openMosix consigue muy bien balancear la carga de programas que lanzan múltiples procesos mientras que no es demasiado efectivo si se trata de distribuir un proceso "gordo".

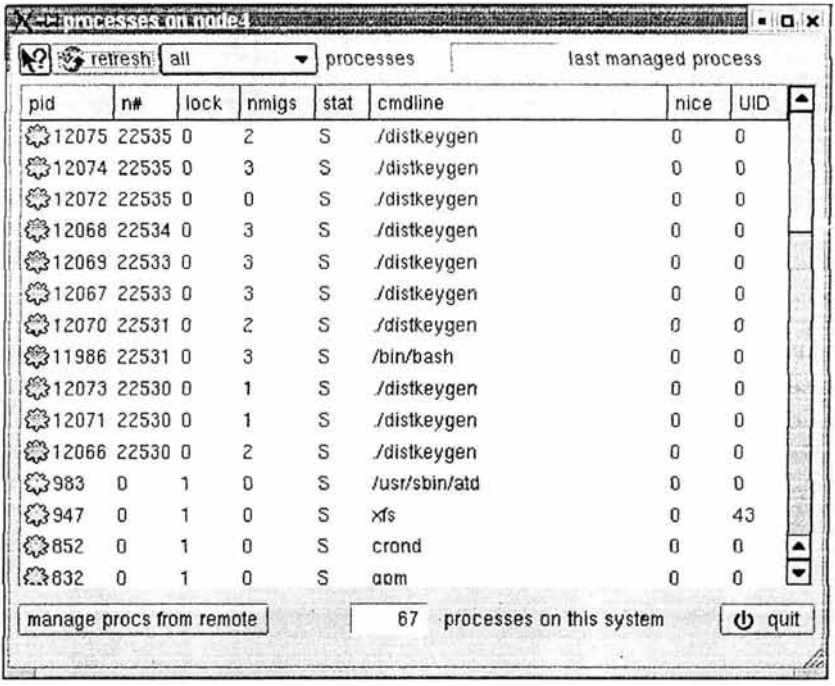
Monitoreo de nodos



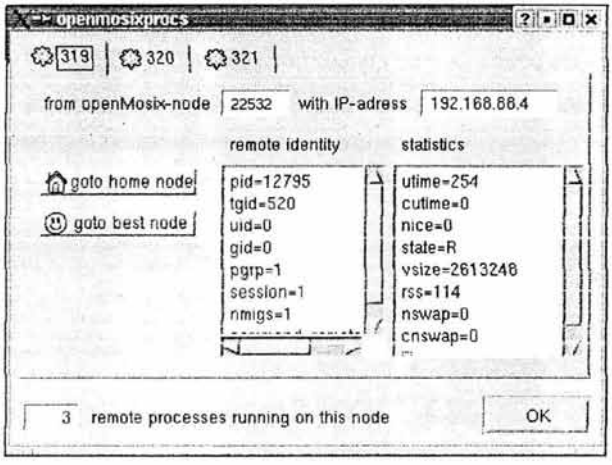
Ejecución avanzada



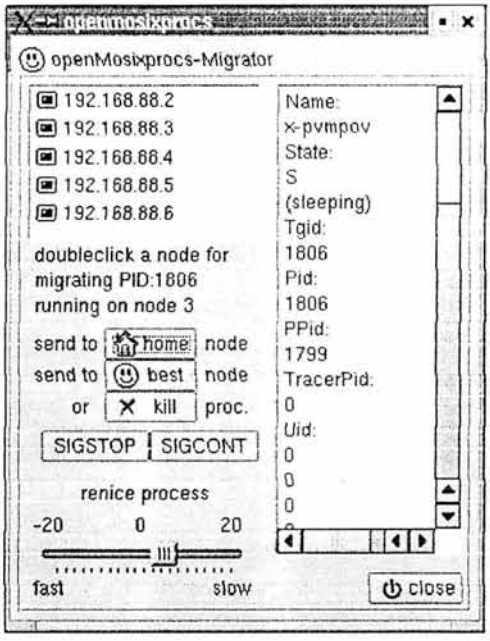
Listado de procesos en un nodo



Detalles de procesos



Migración de procesos



Ya que el cluster ha sido ensamblado y el software instalado, se procede a cargar la base de datos, cuyas características son:

Base de Datos en MYSQL

- Nombre de la Base de Datos: Biblio3
    - Contiene un total de 221,166 registros
    - Nodo 1→ No incluye datos de la base
    - Nodo 3 → Contiene la Base de Datos Completa
    - Nodos 2, 3, 4 y 5 →Se distribuye la base de datos en cuatro tablas, las cuales son colocadas de manera aleatoria en cada nodo.→ Distribución asimétrica horizontal.

Una vez terminado este proceso, se inician una serie de pruebas para demostrar la funcionalidad y eficiencia de esta tecnología.

Con las pruebas llevadas a cabo se muestra como al paralelizar las búsquedas se obtiene una ganancia en tiempo de proceso cuando se incrementa el número de procesadores en el cluster.

Dichas pruebas y los resultados que se obtuvieron serán presentados en la última parte de este trabajo.

Conclusiones.

Para cubrir las necesidades que se van presentando en relación al almacenamiento, gestión, control y manejo de la información; la respuesta siempre ha estado en el desarrollo continuo en el que se encuentran tanto el hardware como el software. Desde el origen del Mainframe, la evolución de la computadora que se presenta de manera constante, tiende a desarrollar más el potencial de procesamiento de los mismos, a tal grado que ahora una simple Computadora Personal funge las tareas de un equipo servidor, con lo cual se reducen de manera significativa los costos, haciendo más factible el desarrollo de proyectos encaminados a mejorar el desempeño empresarial de las compañías. Como resultado de este desarrollo se cuenta ahora con la tecnología de clusters.

Desde el desarrollo de los clusters en 1994, este ha tenido impacto inmediato a nivel costo/beneficio, en comparación con los grandes y costosos equipos de cómputo. Este impacto no solo se refleja en la reducción significativa de los recursos económicos que deben ser invertido para el desarrollo de proyectos de gran importancia, sino también en el desarrollo de proyectos que van desde el supercómputo hasta el almacenamiento de bases de datos de alto rendimiento, así como aplicaciones en las ciencias computacionales y comerciales. Esto es posible gracias a la utilización de software libre, ya que desde la creación del primer cluster Beowulf, se ha utilizado sistema operativo Linux, el software GNU y la estandarización de envío de mensajes vía PVM y MPI, las cuales son de licencia libre y pueden ser utilizadas sin restricción alguna.

Cabe resaltar, que al tratarse de un tema relativamente nuevo, la mayor parte de la información obtenida sobre el tema de clusters para la elaboración de este trabajo, fue obtenida de sitios de Internet, los cuales cuentan con datos muy actualizados acerca de este tópico.

Dentro del concepto de clusters va implícito el concepto de paralelismo, definido como la introducción de varios procesadores para resolver un problema. Esto se ve reflejado durante el desarrollo, funcionamiento y operación de los cluster, ya que este implica la utilización de varios nodos, cada uno con un procesador independiente, los cuales deben de trabajar de manera conjunta para obtener los resultados deseados.

Tanto el software libre como el paralelismo se han convertido en una herramienta fundamental para el desarrollo de la tecnología de clusters

Como dato relevante, gracias al cluster que ha sido montado dentro de las instalaciones de la FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN, se ha podido dar el primer acercamiento hacia esta nueva tecnología , y la UNAM sigue integrando esta en mayores proyectos.

Una de las tareas primordiales en las que interviene un cluster, es la gestión y almacenamiento de las bases de datos. Las bases de datos constituyen una parte integrante y fundamental de los sistemas de información y tienen su razón de ser en la misma existencia de estos sistemas.

La necesidad de conocer y manear la información para el desarrollo y el crecimiento no solo de las grandes corporaciones, sino de la sociedad en general, se ha convertido en una parte significativa para el desarrollo de las bases de datos. Nadie puede, en estos momentos, poner en tela de juicio la importancia de la información. Lamentablemente, aun nos encontramos muy lejos de poder satisfacer las necesidades de información, ya que no solo existen problemas tecnológicos relativos al almacenamiento y acceso a la información, sino que existen también problemas económicos, políticos o administrativos que impiden el desarrollo de sistemas de información eficientes, capaces de atender debidamente las demandas de información.

El uso de grandes cantidades de información, requiere de sistemas informáticos avanzados, capaces de almacenar dicha información para su posterior manejo. Dicha información debe de ser precisa, clara, oportuna, completa y segura; todo esto para poder elaborar estrategias a futuro que permitan el desarrollo de las empresas, instituciones educativas, y de la sociedad en general.

Las bases de datos, entendidas como un conglomerado de datos almacenados en sistemas informáticos los cuales permiten su utilización posterior, brindan el apoyo necesario para poder tomar decisiones a futuro. Los datos, como eje central de las bases de datos deberán de ser actualizados de manera constante, para evitar tomar decisiones erradas o mal fundamentadas.

Existen varios modelos de bases de datos, de entre estos modelos, el más conocido es el modelo de bases de datos relacional, el cual es utilizado en la mayoría de las aplicaciones para el modelo de datos. Esto es debido a que utiliza para su esquematización tablas y filas, con lo cual se refleja de manera más significativa el ordenamiento y relación que guardan los datos.

Al almacenar los datos en dichas tablas, la búsqueda se realiza en las diversas columnas que componen la tabla, facilitando en trabajo de los diseñadores y programadores para el desarrollo de las aplicaciones que interactuaran con los datos. Una tabla está compuesta de dos partes bien distintas: el encabezado, y un conjunto de filas Cada fila de la tabla representa una relación entre un conjunto de valores. A los renglones de una relación, si no son el renglón del encabezado que contiene los atributos, se les da el nombre de tuplas. Tanto el esquema como las tuplas de una relación son conjuntos, no listas. de ahí que no importe el orden en que sean presentadas. más aún, se pueden reordenar los atributos de la relación como se quiera, sin cambiar con ello la relación.

El álgebra relacional y el cálculo relacional, expresan las consultas que se realizan sobre las relaciones en una base de datos relacional. Operaciones como Select, Join y Project Son utilizadas para relacionar renglones, columnas o unir información de varias tablas.

El segundo modelo de bases de datos a considerar es el modelo de base de datos distribuidas, en la cual, los datos de una misma base de datos estan esparcidos en varios sitios. Las razones por las cuales se desarrollan este tipo de bases de datos son diversas, entre las que destaca de descentralización de los datos, lo cual permite un mayor rendimiento en el procesamiento de los datos.

Para su desarrollo, la planeación considera factores como los programas que será utilizados en cada sitio de la red, la redundancia y duplicidad de los datos, y las transacciones que deberá de ejecutar dicho sistema.

Para almacenar una relación en una base de datos distribuida existen dos enfoques, la réplica y la fragmentación. En el primer enfoque el sistema conserva réplicas o copias idénticas de la relación y guarda cada réplica en un sitio diferente. La alternativa a las réplicas es almacenar sólo una copia de la relación. Para el segundo enfoque el sistema divide la relación en varios fragmentos y guarda cada fragmento en un sitio diferente.

No se debe exigir a los usuario de lo sistemas distribuidos de bases de datos que conozca la ubicación física de los datos ni el modo en que se puede tener acceso a ellos en un sitio local concreto, ya que un usuario dado puede realizar operaciones sobre los datos en su propio sitio local exactamente como si ese sitio no participara en absoluto en el sistema distribuido.

El tercer modelo a considerar es el de bases de datos paralelas. Los sistemas paralelos de base de datos mejoran la velocidad de procesamiento de entrada y salida mediante la utilización de computadoras y discos en paralelo, por lo cual se hace cada vez más frecuente que la mayoría de los fabricantes de bases de datos los empiecen a comercializar.

Caso similar al tema de cluster es el desarrollo del tema de base de datos paralelas; la mayor parte de la información fue extraida de una sola bibliografía.

Entendamos a los sistemas paralelos de base de datos como sistemas que constan de varios procesadores y varios discos conectados a través de una red de interconexión de alta velocidad.

Nuevamente Interviene el concepto de paralelismo. El procesamiento en paralelo ha emergido como una tecnología disponible para la computación moderna, ya que en este se realizan muchas operaciones simultáneamente, mientras que en el procesamiento secuencial los distintos pasos computacionales han de ejecutarse paso a paso.

El objetivo del paralelismo en los sistemas de base de datos suele ser asegurar que la ejecución del sistema continuará realizándose a una velocidad aceptable, incluso en el caso de que aumente el tamaño de la base de datos o el número de transacciones

El funcionamiento de los sistemas paralelos de base de datos se basa en el almacenamiento de los datos y el procesamiento de las consultas.

Dado que las operaciones relacionales tratan con relaciones que contienen grandes conjuntos de tuplas, se pueden paralelizar las operaciones ejecutándolas en paralelo en subconjuntos diferentes de las relaciones. Por tanto, el paralelismo en operaciones es natural en los sistemas de base de datos.

Al momento de empezar a diseñar un sistema paralelo, no solo se debe de considerar el volumen de la información y las operaciones que se realizaran con estas; sino también se deben de abordar aspectos tales como lo son el poder de recuperación frente al fallo de algún nodo y la reorganización interactiva de los datos y los cambios de los esquemas.

Cuando se manejan grandes cantidades de información, del orden de Terabytes, las operaciones sencillas, como lo son la creación de índices, los cambios en los esquemas o como añadir una columna a una relación, pueden tardar algo de tiempo, por lo que no es posible que los sistemas de base de datos no estén disponibles mientras se llevan a cabo dichas operaciones.

Los sistemas paralelos deben de permitir que tales operaciones se lleven a cabo interactivamente, es decir, mientras el sistema ejecuta otras transacciones.

Dada la relevancia de los clusters, así como su impacto inmediato en el futuro cercano, todos los relacionados a la Informática deben de adentrarse más en este tema y conocer más a fondo las características y ventajas de esta tecnología.

Con las pruebas llevadas a cabo se muestra como al paralelizar las búsquedas se obtiene una ganancia en tiempo de proceso cuando se incrementa el número de procesadores en el cluster.

Desde luego que aún faltan otra serie de pruebas que ya están en proceso, pero que aún no son reportadas, debido a que no se han concluido.

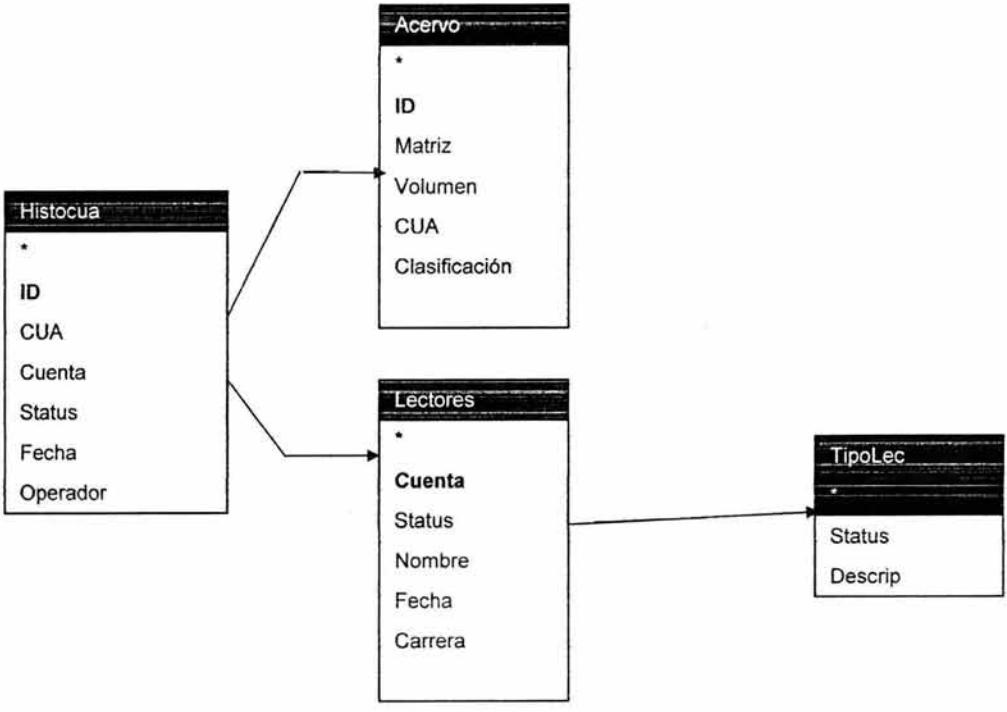
ANEXO

CASO PRÁCTICO. (Instalación de una Base de datos Paralela en un Cluster LINUX).

A. Base de Datos.

Base de Datos en MYSQL

- Biblio1
  - Aprox. 221,000 registros
  - Nodo 1 → No tiene datos de la basa.
  - Nodo 3 → Base completa
  - Nodos 2, 3, 4 y 5 → Distribución asimétrica horizontal→Tablas aleatorias.



B. Diseño.

Se realiza la implementación de un sistema de base de datos no paralela sobre un cluster con sistema operativo LINUX MANDRAKE Versión 9.1, el cual utiliza el paralelismo disponible en las distintas máquinas para realizar el proceso de consulta sobre los datos.

Se implementa una base de datos relacional donde los datos se mantiene distribuidos sobre el cluster y las consultas se procesan en paralelo en cada máquina del cluster.

La plataforma computacional está formada por un grupo de PC's con sistema operativo Linux, conectados entre si mediante un switch de alto desempeño. En cada PC se instala un administrador de base de datos relacionado llamado MySQL el cual tiene una API que proporcionan clases que permiten enviar strings con sentencias SQL desde un programa en C++ al servidor MySQL en el esquema cliente/servidor.

El caso secuencial, es procesar en una sola PC una sentencia SQL, un mensaje (string) entre el proceso cliente (C++) y el proceso servidor (MySQL), y luego MySQL responde con otro mensaje que contiene el resultado de la sentencia SQL ejecutada sobre las tablas almacenadas en la base de datos manejada en MySQL. Aquí, la PC mantiene la base de datos completa.

En el caso paralelo, la base de datos se encuentra distribuida entre las PC's. Las tuplas de las tablas se reparten aleatoriamente entre las distintas máquinas. Un programa C++ corre en cada PC enviando mensajes el proceso servidor MySQL local residente en la PC respectiva. Los mensajes C++/MySQL son las instrucciones que resultan de la implementación en paralelo de la consulta a la base de datos. Esta descomposición depende del modelo de computación paralela adoptado. Cada servidor MySQL esta encargado de administrar el seguimiento correspondiente de la base de datos distribuida. Los programas C++ corriendo en cada PC se comunican entre si según las reglas de la computación paralela.

C. PROGRAMAS.

```
gcc -o siete.exe siete.cpp -g -L/usr/include/mysql -L/usr/lib/mysql -
L/usr/local/lib -lmysqlclient -lstdc++ -lmos

 #include <stdio.h>
//Biblioteca MYSQL
#include "/usr/include/mysql/mysql.h"
// Las siguientes son usadas por las funciones de medicion de tiempo
#include <sys/times.h>
#include <sys/types.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
//Biblioteca C++
#include <iostream>
// Biblioteca Openmosix
#include "/usr/local/include/libmosix.h"
//#include "/usr/include/c++/3.3.1/backward/iostream.h"
/* Primera Prueba de MYSQL */
//includes para la parte de OpenMosix
#include <unistd.h>
#include <dirent.h>
#include <ctype.h>
//include wait para espera del fork
#include <sys/wait.h>

#define clusterdir "/proc/hpc/nodes"

int cpucount() {
DIR *dirhpc;
struct dirent *dir_info;
int cuantos=0;
int cpus=0;
FILE *fp;
char tmpdirname[200];
char tmpfilename[200];
strcpy(tmpdirname, "");
if ((dirhpc=opendir(clusterdir))!=NULL) {
while ((dir_info = readdir(dirhpc))!=NULL) {
strcpy(tmpdirname, dir_info->d_name);
strcpy(tmpfilename, "/proc/hpc/nodes/");
strcat(tmpfilename, tmpdirname);
strcat(tmpfilename, "/cpus");
if (!strchr(tmpdirname, '.')) {
fp=fopen(tmpfilename, "r");
if (fp) {
fscanf(fp, "%d", &cpus);
if(cpus>0) {
cuantos=cuantos+cpus;
}
fclose(fp);
```

```
}
}
}

closedir(dirhpc);
printf("%d\n", cuantos);
return cuantos;
}
}

int conectabase(MYSQL *gestor,const char *host,const char *usuario,const
char *password,const char *base)
{
    /* Inicalizo MySQL*/
   if (!mysql_init(gestor))
   {
       printf("No Puedo Iniciar MySQL");
       exit(1);
  }
   /* Conexion a la la base de datos */
   if (!mysql_real_connect(gestor,host,usuario,password,base,0,NULL,0))
        /*estructura,host,usuario,password,base,puerto,socket,client-flag
*/
   {
        fprintf(stderr,"%d:
%s\n",mysql_errno(gestor),mysql_error(gestor));
        exit(1);
  }

}

int consulta(MYSQL *gestor,const char *cadena,clock_t *finalinterno,int
ciclos)
{
  struct tms tic_finalinterno;
  MYSQL_RES *resultado;
  MYSQL_ROW renglon;
  int i;

for(i=1;i<=ciclos;i++)
{
  if(mysql_query(gestor,cadena))
  {
      fprintf(stderr,"%d: %s\n",mysql_errno(gestor),mysql_error(gestor));
  }
  else
  {
     resultado=mysql_store_result(gestor);
     *finalinterno=times(&tic_finalinterno);
     //Quitar comentarios para poder desplegar en pantalla
     /*while (renglon=mysql_fetch_row(resultado))
     {

printf("%s\t%s\t%s\t%s\n",renglon[0],renglon[1],renglon[2],renglon[3]);
     } */

     mysql_free_result(resultado);
```

```
    }
}
}

int main()
{
    /* variables y estructuras para calculo de tiempos */
    struct tms tic_comienzo, tic_final;
    clock_t comienzo,final,tics_consumidos;
    int CICLOS=1;

    /*variables y estructuras para MySQL*/
    MYSQL mysql1,mysql2,mysql3,mysql4,mysql5,mysqlo;
    MYSQL_RES *resultado;
    MYSQL_ROW renglon;
    double seg_dist_sec[5],seg_remoto,seg_dist_para1[5];

    printf("Conectando las Bases de Datos\n");
    /*Conexion con las bases remotas distribuidas*/
    conectabase(&mysql2,"nodo2","jlgr","jlgr","Biblio1");
    conectabase(&mysql3,"nodo3","jlgr","jlgr","Biblio1");
    conectabase(&mysql4,"nodo4","jlgr","jlgr","Biblio1");
    conectabase(&mysql5,"nodo5","jlgr","jlgr","Biblio1");

 //Consulta a la base remota Completa
    comienzo= times(&tic_comienzo);
    consulta(&mysql3,"SELECT * FROM Acervo WHERE Clasificacion like
'HF%'",&final,CICLOS);

final=times(&tic_final);
tics_consumidos=final-comienzo;
seg_remoto=(double)tics_consumidos/(double)sysconf(_SC_CLK_TCK);

// consulta a las bases remotas distribuidas en 2
  comienzo= times(&tic_comienzo);
  consulta(&mysql3,"SELECT * FROM Acervo2 WHERE Clasificacion like
'HF%'",&final,CICLOS);
  consulta(&mysql4,"SELECT * FROM Acervo2 WHERE Clasificacion like
'HF%'",&final,CICLOS);
  final=times(&tic_final);
  tics_consumidos=final-comienzo;
seg_dist_sec[2]=(double)((double)tics_consumidos/(double)sysconf(_SC_CLK_
TCK));

// consulta a las bases remotas distribuidas en 3
  comienzo= times(&tic_comienzo);
  consulta(&mysql3,"SELECT * FROM Acervo3 WHERE Clasificacion like
'HF%'",&final,CICLOS);
  consulta(&mysql4,"SELECT * FROM Acervo3 WHERE Clasificacion like
'HF%'",&final,CICLOS);
   consulta(&mysql5,"SELECT * FROM Acervo3 WHERE Clasificacion like
'HF%'",&final,CICLOS);
  final=times(&tic_final);
  tics_consumidos=final-comienzo;
seg_dist_sec[3]=(double)((double)tics_consumidos/(double)sysconf(_SC_CLK_
TCK));
```

```
// consulta a las bases remotas distribuidas en 4
  comienzo= times(&tic_comienzo);
  consulta(&mysql2,"SELECT * FROM Acervo4 WHERE Clasificacion like
'HF%'",&final,CICLOS);
  consulta(&mysql3,"SELECT * FROM Acervo4 WHERE Clasificacion like
'HF%'",&final,CICLOS);
  consulta(&mysql4,"SELECT * FROM Acervo4 WHERE Clasificacion like
'HF%'",&final,CICLOS);
   consulta(&mysql5,"SELECT * FROM Acervo4 WHERE Clasificacion like
'HF%'",&final,CICLOS);
  final=times(&tic_final);
  tics_consumidos=final-comienzo;
seg_dist_sec[4]=(double)((double)tics_consumidos/(double)sysconf(_SC_CLK_
TCK));

  //Consulta a las bases de manera remota distribuida paralela
   pid_t procesoX[10];
   int i,condicion;

   //printf("Estoy antes de los Ifs\n");
   //printf(" Mi Id %d\n",getpid());
   //printf("El de mi Pa %d\n\n",getppid());

   // Dividida en 2
   comienzo= times(&tic_comienzo);
   if((procesoX[0]=fork())==0)
   {
     consulta(&mysql3,"SELECT * FROM Acervo2 WHERE Clasificacion like
'HF%'",&final,CICLOS);
     //printf("Estoy en el primer if en el then\n");
     //printf("Mi Id %d\n",getpid());
     //printf("El de mi Pa %d\n\n",getppid());
     exit(EXIT_SUCCESS);
   }
   else
   {
      //printf("Estoy en el else del primer if en el then\n");
      //printf("Mi Id %d\n",getpid());
      //printf("El de mi Pa %d\n\n",getppid());

      if((procesoX[1]=fork())==0)
      {
         consulta(&mysql4,"SELECT * FROM Acervo2 WHERE Clasificacion like
'HF%'",&final,CICLOS);

         //printf(" Mi Id %d\n",getpid());
         //printf("El de mi Pa %d\n\n",getppid());
         exit(EXIT_SUCCESS);
      }
```

```
   }// Primer Else
for (i=0;i<=1;i++)
  waitpid(procesoX[i],&condicion,0);
  final=times(&tic_final);
  //printf("Estoy Afuera de cualquier if\n");
  //printf("Mi Id %d\n",getpid());
  //printf("El de mi Pa %d\n\n",getppid());

  tics_consumidos=final-comienzo;
seg_dist_paral[2]=(double)((double)tics_consumidos/(double)sysconf(_SC_CL
K_TCK));


  // Dividida en 3
  comienzo= times(&tic_comienzo);
   if((procesoX[0]=fork())==0)
   {
     consulta(&mysql3,"SELECT * FROM Acervo3 WHERE Clasificacion like
'HF%'",&final,CICLOS);
     //printf("Estoy en el primer if en el then\n");
     //printf("Mi Id %d\n",getpid());
     //printf("El de mi Pa %d\n\n",getppid());
     exit(EXIT_SUCCESS);
   }
   else
   {
      if((procesoX[1]=fork())==0)
      {
         consulta(&mysql4,"SELECT * FROM Acervo3 WHERE Clasificacion like
'HF%'",&final,CICLOS);
         exit(EXIT_SUCCESS);
      }
      else
      {
          if((procesoX[2]=fork())==0)
         {
         consulta(&mysql5,"SELECT * FROM Acervo3 WHERE Clasificacion like
'HF%'",&final,CICLOS);
        exit(EXIT_SUCCESS);
        }

      }

   }// Primer Else
for (i=0;i<=2;i++)
  waitpid(procesoX[i],&condicion,0);
  final=times(&tic_final);
  //printf("Estoy Afuera de cualquier if\n");
  //printf("Mi Id %d\n",getpid());
  //printf("El de mi Pa %d\n\n",getppid());
```

```
  tics_consumidos=final-comienzo;
seg_dist_paral[3]=(double)((double)tics_consumidos/(double)sysconf(_SC_CL
K_TCK));


// Dividida en 4
comienzo= times(&tic_comienzo);
   if((procesoX[0]=fork())==0)
   {
     consulta(&mysql2,"SELECT * FROM Acervo4 WHERE Clasificacion like
'HF%'",&final,CICLOS);
     //printf("Estoy en el primer if en el then\n");
     //printf("Mi Id %d\n",getpid());
     //printf("El de mi Pa %d\n\n",getppid());
     exit(EXIT_SUCCESS);
   }
   else
   {
      if((procesoX[1]=fork())==0)
      {
         consulta(&mysql3,"SELECT * FROM Acervo4 WHERE Clasificacion like
'HF%'",&final,CICLOS);
         exit(EXIT_SUCCESS);
      }
      else
      {
          if((procesoX[2]=fork())==0)
         {
         consulta(&mysql4,"SELECT * FROM Acervo4 WHERE Clasificacion like
'HF%'",&final,CICLOS);

         //printf(" Mi Id %d\n",getpid());
         //printf("El de mi Pa %d\n\n",getppid());
         exit(EXIT_SUCCESS);
       }
       else
       {
              if((procesoX[3]=fork())==0)
             {
             consulta(&mysql5,"SELECT * FROM Acervo4 WHERE Clasificacion
like    'HF%'",&final,CICLOS);

         //printf(" Mi Id %d\n",getpid());
         //printf("El de mi Pa %d\n\n",getppid());
            exit(EXIT_SUCCESS);
           }


        }

      }
```

```
    |// Primer Else
for (i=0;i<=3;i++)
  waitpid(procesoX[i],&condicion,0);
  final=times(&tic_final);
  //printf("Estoy Afuera de cualquier if\n");
  //printf("Mi Id %d\n",getpid());
  //printf("El de mi Pa %d\n\n",getppid());

  tics_consumidos=final-comienzo;
seg_dist_paral[4]=(double)((double)tics_consumidos/(double)sysconf(_SC_CL
K_TCK));

  //std::cout<<"Ticks Cominezo: "<<comienzo<<std::endl;
  //std::cout<<"Ticks Final: "<<final<<std::endl;
  //std::cout<<"Ticks Utilizados: "<<final-comienzo<<std::endl;
  //std::cout<<"Tick por segundo: "<<sysconf(_SC_CLK_TCK)<<std::endl;
  //printf("\nTicks final %jd",final);
  //printf("\nTicks Utilizados %jd",final-comienzo);
  printf("Numero de Ciclos %d \n",CICLOS);
  printf("Segundos remoto No distriduido (nodo3) tabla completa
%6.2f\n",seg_remoto);
  printf("\n");


   for (i=2;i<=4;i++)
    printf("Segundos remoto distribuido secuencial, tabla dividida en %d
%6.2f\n",i,seg_dist_sec[i]);
    printf("\n");
   for (i=2;i<=4;i++)
  printf("Segundos remoto distriduido paralelo, tabla  dividida en %d
%6.2f\n",i,seg_dist_paral[i]);
    // la funcion sysconf() anterior obtiene el numero de tics por
segundo

  //mysql_close(&mysql);
  mysql_close(&mysql2);
  mysql_close(&mysql3);
  mysql_close(&mysql4);
  mysql_close(&mysql5);
  //std::cout<<"lolo"<<std::endl;

  // Aqui comienzo con openmosix como primera aproximacion
  int processors=0;
  processors=cpucount();
  std::cout<<"El numero de Procesadores disponibles en este Cluster es:"
<<processors<<std::endl;
  pid_t proceso1;
```

```
 /* if(procesol=fork()){
      mysql_query(&mysql,"SELECT * FROM Acervo WHERE Clasificacion like
'QA%'");} */

  //char tempo[200];
  //strcpy(tempo, "migratel");
  //strcat(tempo, procesol);
  //strcat(tempo," 3");
  //system(tempo);

return 0;
}
```

```
gcc -o ocho.exe ocho.cpp -g -L/usr/include/mysql -L/usr/lib/mysql -
L/usr/local/lib -lmysqlclient -lstdc++ -lmos

#include <stdio.h>
//Biblioteca MYSQL
#include "/usr/include/mysql/mysql.h"
// Las siguientes son usadas por las funciones de medicion de tiempo
#include <sys/times.h>
#include <sys/types.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
//Biblioteca C++
#include <iostream>
// Biblioteca Openmosix
#include "/usr/local/include/libmosix.h"
//#include "/usr/include/c++/3.3.1/backward/iostream.h"
/* Primera Prueba de MYSQL */
//includes para la parte de OpenMosix
#include <unistd.h>
#include <dirent.h>
#include <ctype.h>
//include wait para espera del fork
#include <sys/wait.h>

#define clusterdir "/proc/hpc/nodes"

//La busqueda
#define busqueda2 "SELECT
Historico.Cua,Acervo.Clasificacion,TiposLec.Descrip,Lectores.Nombre FROM
((Historico Left Join Acervo ON Historico.Cua=Acervo.Cua)Left join
Lectores ON Historico.Cuenta=Lectores.Cuenta) LEFT JOIN TiposLec ON
Lectores.Status=TiposLec.Status"

#define busqueda3 "SELECT
Historico.Cua,Acervo.Clasificacion,TiposLec.Descrip,Lectores.Nombre,Histo
rico.Fecha FROM ((Historico Left Join Acervo ON
Historico.Cua=Acervo.Cua)Left join Lectores ON
Historico.Cuenta=Lectores.Cuenta) LEFT JOIN TiposLec ON
Lectores.Status=TiposLec.Status WHERE Historico.Cuenta='102295'"

#define busqueda "SELECT
Historico.Cua,Acervo.Clasificacion,TiposLec.Descrip,Lectores.Nombre FROM
((Historico Left Join Acervo ON Historico.Cua=Acervo.Cua)Left join
Lectores ON Historico.Cuenta=Lectores.Cuenta) LEFT JOIN TiposLec ON
Lectores.Status=TiposLec.Status WHERE Lectores.Nombre like'%CARLOS%'"
```

```
int cpucount() {
DIR *dirhpc;
struct dirent *dir_info;
int cuantos=0;
int cpus=0;
FILE *fp;
char tmpdirname[200];
char tmpfilename[200];
strcpy(tmpdirname, "");
if (!dirhpc=opendir(clusterdir))!=NULL) {
while ((dir_info = readdir(dirhpc))!=NULL) {
strcpy(tmpdirname, dir_info->d_name);
strcpy(tmpfilename, "/proc/hpc/nodes/");
strcat(tmpfilename, tmpdirname);
strcat(tmpfilename, "/cpus");
if (!strchr(tmpdirname, '.')) {
fp=fopen(tmpfilename, "r");
if (fp) {
fscanf(fp, "%d", &cpus);
if(cpus>0) {
cuantos=cuantos+cpus;
}
fclose(fp);
}
}
}
closedir(dirhpc);
printf("%d\n", cuantos);
return cuantos;
}
}


int conectabase(MYSQL *gestor,const char *host,const char *usuario,const
char *password,const char *base)
{
    /* Inicalizo MySQL*/
   if (!mysql_init(gestor))
   {
       printf("No Puedo Iniciar MySQL");
       exit(1);
  }
   /* Conexion a la la base de datos */
   if (!mysql_real_connect(gestor,host,usuario,password,base,0,NULL,0))
        /*estructura,host,usuario,password,base,puerto,socket,client-flag
*/
   {
        fprintf(stderr,"%d:
%s\n",mysql_errno(gestor),mysql_error(gestor));
        exit(1);
  }
```

```
}

int consulta(MYSQL *gestor,const char *cadena,clock_t *finalinterno,int
ciclos)
{
  struct tms tic_finalinterno;
  MYSQL_RES *resultado;
  MYSQL_ROW renglon;
  int i,j,Total=0;
  unsigned int Campos;

for(i=1;i<=ciclos;i++)
{
  if(mysql_query(gestor,cadena))
  {
      fprintf(stderr,"%d: %s\n",mysql_errno(gestor),mysql_error(gestor));
  }
  else
  {
     resultado=mysql_store_result(gestor);
     *finalinterno=times(&tic_finalinterno);
     Campos=mysql_num_fields(resultado);
     Total=0;
     while (renglon=mysql_fetch_row(resultado))
     {
       for(j=0;j<Campos;j++)
          printf("%s\t",renglon[j]);
       Total=Total+1;
     }
    printf("\nEncontre %d Registros\n\n",Total);
}

}
 mysql_free_result(resultado);
}

int main()
{
    /* variables y estructuras para calculo de tiempos */
    struct tms tic_comienzo, tic_final;
    clock_t comienzo,final,tics_consumidos;
    int CICLOS=1;

    /*variables y estructuras para MySQL*/
    MYSQL mysql,mysql2,mysql3,mysql4,mysql5,mysql6;
    MYSQL_RES *resultado;
    MYSQL_ROW renglon;
    double seg_dist_sec,seg_remoto,seg_dist_paral;
```

```
    printf("Conectando las Bases de Datos\n");
    /*Conexion con las bases remotas distribuidas*/
    conectabase(&mysql,"nodo2","jlgr","jlgr","Biblio4");
    conectabase(&mysql2,"nodo2","jlgr","jlgr","Biblio3");
    conectabase(&mysql3,"nodo3","jlgr","jlgr","Biblio3");
    conectabase(&mysql4,"nodo4","jlgr","jlgr","Biblio3");
    conectabase(&mysql5,"nodo5","jlgr","jlgr","Biblio3");

 //Consulta a la base remota Completa
    printf("Busqueda Completa Remota ...\n");
    comienzo= times(&tic_comienzo);
    consulta(&mysql,busqueda,&final,CICLOS);

final=times(&tic_final);
tics_consumidos=final-comienzo;
seg_remoto=(double)tics_consumidos/(double)sysconf(_SC_CLK_TCK);

// consulta a las bases remotas distribuidas en 4
  printf("Busqueda Distribuida Secuencial  ...\n");
  comienzo= times(&tic_comienzo);
  printf("Busqueda en nodo2  ...\n");
  consulta(&mysql2,busqueda,&final,CICLOS);
  printf("Busqueda en nodo3  ...\n");
  consulta(&mysql3,busqueda,&final,CICLOS);
  printf("Busqueda en nodo4  ...\n");
  consulta(&mysql4,busqueda,&final,CICLOS);
  printf("Busqueda en nodo5  ...\n");
   consulta(&mysql5,busqueda,&final,CICLOS);
  final=times(&tic_final);
  tics_consumidos=final-comienzo;
seg_dist_sec=(double)((double)tics_consumidos/(double)sysconf(_SC_CLK_TCK
));

  //Consulta a las bases de manera remota distribuida paralela
   pid_t procesoX[10];
   int i,condicion;

   //printf("Estoy antes de los Ifs\n");
   //printf(" Mi Id %d\n",getpid());
   //printf("El de mi Pa %d\n\n",getppid());
```

```
// Dividida en 4
printf("Busqueda Distribuida Paralela  ...\n");
comienzo= times(&tic_comienzo);
   if((procesoX[0]=fork())==0)
   {
     printf("Busqueda en nodo2  ...\n");
     consulta(&mysql2,busqueda,&final,CICLOS);
     //printf("Estoy en el primer if en el then\n");
     //printf("Mi Id %d\n",getpid());
     //printf("El de mi Pa %d\n\n",getppid());
     exit(EXIT_SUCCESS);
   }
   else
   {
      if((procesoX[1]=fork())==0)
      {
        printf("Busqueda en nodo3  ...\n");
         consulta(&mysql3,busqueda,&final,CICLOS);
         exit(EXIT_SUCCESS);
      }
      else
      {
          if((procesoX[2]=fork())==0)
         {
       printf("Busqueda en nodo4  ...\n");
         consulta(&mysql4,busqueda,&final,CICLOS);

         //printf(" Mi Id %d\n",getpid());
         //printf("El de mi Pa %d\n\n",getppid());
         exit(EXIT_SUCCESS);
       }
       else
       {
              if((procesoX[3]=fork())==0)
             {
          printf("Busqueda en nodo5  ...\n");
            consulta(&mysql5,busqueda,&final,CICLOS);

         //printf(" Mi Id %d\n",getpid());
         //printf("El de mi Pa %d\n\n",getppid());
            exit(EXIT_SUCCESS);
           }


        }

      }
```

```
   }// Primer Else
for (i=0;i<=3;i++)
  waitpid(procesoX[i],&condicion,0);
  final=times(&tic_final);
  //printf("Estoy Afuera de cualquier if\n");
  //printf("Mi Id %d\n",getpid());
  //printf("El de mi Pa %d\n\n",getppid());

  tics_consumidos=final-comienzo;
seg_dist_paral=(double)((double)tics_consumidos/(double)sysconf(_SC_CLK_T
CK));

  //std::cout<<"Ticks Cominezo: "<<comienzo<<std::endl;
  //std::cout<<"Ticks Final: "<<final<<std::endl;
  //std::cout<<"Ticks Utilizados: "<<final-comienzo<<std::endl;
  //std::cout<<"Tick por segundo: "<<sysconf(_SC_CLK_TCK)<<std::endl;
  //printf("\nTicks final %jd",final);
  //printf("\nTicks Utilizados %jd",final-comienzo);

  printf("Numero de Ciclos %d \n",CICLOS);
  printf("Segundos remoto No distriduido (nodo3) tabla completa
%6.2f\n",seg_remoto);
  printf("\n");
  printf("Segundos remoto distribuido secuencial, tabla dividida en %d
%6.2f\n",i,seg_dist_sec);
    printf("\n");
   printf("Segundos remoto distriduido paralelo, tabla dividida en %d
%6.2f\n",i,seg_dist_paral);
    // la funcion sysconf() anterior obtiene el numero de tics por
segundo

  mysql_close(&mysql);
  mysql_close(&mysql2);
  mysql_close(&mysql3);
  mysql_close(&mysql4);
  mysql_close(&mysql5);
  //std::cout<<"lolo"<<std::endl;


  // Aqui comienzo con openmosix como primera aproximacion
  int processors=0;
  processors=cpucount();
  std::cout<<"El numero de Procesadores disponibles en este Cluster es:"
<<processors<<std::endl;
  pid_t proceso1;
```

```
 /* if(procesol=fork()){
      mysql_query(&mysql,"SELECT * FROM Acervo WHERE Clasificacion like
'QA%'");} */

  //char tempo[200];
  //strcpy(tempo, "migrate1");
  //strcat(tempo, procesol);
  //strcat(tempo," 3");
  //system(tempo);

return 0;
}
```

```
gcc -o nueve.exe nueve.cpp -g -L/usr/include/mysql -L/usr/lib/mysql -
L/usr/local/lib -lmysqlclient -lstdc++ -lmos

#include <stdio.h>
//Biblioteca MYSQL
#include "/usr/include/mysql/mysql.h"
// Las siguientes son usadas por las funciones de medicion de tiempo
#include <sys/times.h>
#include <sys/types.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
//Biblioteca C++
#include <iostream>
// Biblioteca Openmosix
#include "/usr/local/include/libmosix.h"
//#include "/usr/include/c++/3.3.1/backward/iostream.h"
/* Primera Prueba de MYSQL */
//includes para la parte de OpenMosix
#include <unistd.h>
#include <dirent.h>
#include <ctype.h>
//include wait para espera del fork
#include <sys/wait.h>

#define clusterdir "/proc/hpc/nodes"

//La busqueda
#define busqueda2 "SELECT
Historico.Cua,Acervo.Clasificacion,TiposLec.Descrip,Lectores.Nombre FROM
((Historico Left Join Acervo ON Historico.Cua=Acervo.Cua)Left join
Lectores ON Historico.Cuenta=Lectores.Cuenta) LEFT JOIN TiposLec ON
Lectores.Status=TiposLec.Status"

#define busqueda "SELECT
Historico.Cua,Acervo.Clasificacion,TiposLec.Descrip,Lectores.Nombre,Histo
rico.Fecha FROM ((Historico Left Join Acervo ON
Historico.Cua=Acervo.Cua)Left join Lectores ON
Historico.Cuenta=Lectores.Cuenta) LEFT JOIN TiposLec ON
Lectores.Status=TiposLec.Status WHERE Historico.Cuenta='102295'"

#define busqueda3 "SELECT
Historico.Cua,Acervo.Clasificacion,TiposLec.Descrip,Lectores.Nombre FROM
((Historico Left Join Acervo ON Historico.Cua=Acervo.Cua)Left join
Lectores ON Historico.Cuenta=Lectores.Cuenta) LEFT JOIN TiposLec ON
Lectores.Status=TiposLec.Status WHERE Lectores.Nombre like'%CARLOS%'"
```

```
int cpucount() {
DIR *dirhpc;
struct dirent *dir_info;
int cuantos=0;
int cpus=0;
FILE *fp;
char tmpdirname[200];
char tmpfilename[200];
strcpy(tmpdirname, "");
if ((dirhpc=opendir(clusterdir))!=NULL) {
while ((dir_info = readdir(dirhpc))!=NULL) {
strcpy(tmpdirname, dir_info->d_name);
strcpy(tmpfilename, "/proc/hpc/nodes/");
strcat(tmpfilename, tmpdirname);
strcat(tmpfilename, "/cpus");
if (!strchr(tmpdirname, '.')) {
fp=fopen(tmpfilename, "r");
if (fp) {
fscanf(fp, "%d", &cpus);
if(cpus>0) {
cuantos=cuantos+cpus;
}
fclose(fp);
}
}
}
closedir(dirhpc);
printf("%d\n", cuantos);
return cuantos;
}
}


int conectabase(MYSQL *gestor,const char *host,const char *usuario,const
char *password,const char *base)
{
    /* Inicalizo MySQL*/
   if (!mysql_init(gestor))
   {
       printf("No Puedo Iniciar MySQL");
       exit(1);
  }
   /* Conexion a la la base de datos */
   if (!mysql_real_connect(gestor,host,usuario,password,base,0,NULL,0))
        /*estructura,host,usuario,password,base,puerto,socket,client-flag
*/
   {
        fprintf(stderr,"%d:
%s\n",mysql_errno(gestor),mysql_error(gestor));
        exit(1);
  }
```

```
}


int consulta(MYSQL *gestor,const char *cadena,clock_t *finalinterno,int
ciclos,MYSQL_RES **resultado)
{
  struct tms tic_finalinterno;
  MYSQL_ROW renglon;
  int i,j,Total;
  unsigned int Campos;

for(i=1;i<=ciclos;i++)
{
  if(mysql_query(gestor,cadena))
  {
      fprintf(stderr,"%d: %s\n",mysql_errno(gestor),mysql_error(gestor));
  }
  else
  {
     *resultado=mysql_store_result(gestor);
     *finalinterno=times(&tic_finalinterno);
 }

}
 //mysql_free_result(resultado);
}

int imprime(MYSQL_RES **resultado)
{
   int j,Total;
   unsigned int Campos;
   MYSQL_ROW renglon;

     Campos=mysql_num_fields(*resultado);
     printf("\nEncontre %d Campos\n\n",Campos);

     Total=0;
     while (renglon=mysql_fetch_row(*resultado))
     {
       for(j=0;j<Campos;j++)
          printf("%s\t",renglon[j]);
       Total=Total+1;
     }
    printf("\nEncontre %d Registros\n\n",Total);
    printf("\nEncontre %d Campos\n\n",Campos);

}
```

```
 MYSQL_RES *res_dist_par2;
int main()
{
    /* variables y estructuras para calculo de tiempos */
    struct tms tic_comienzo, tic_final;
    clock_t comienzo,final,tics_consumidos;
    int CICLOS=1;

    /*variables y estructuras para MySQL*/
    MYSQL mysql,mysql2,mysql3,mysql4,mysql5,mysql6;
    MYSQL_RES *resultado,*res_completo,*res_dist_sec1,*res_dist_sec2;
    MYSQL_RES *res_dist_sec3,*res_dist_sec4,*res_dist_par1;
    MYSQL_RES *res_dist_par3, *res_dist_par4;

    MYSQL_ROW renglon;
    double seg_dist_sec,seg_remoto,seg_dist_para1;

    printf("Conectando las Bases de Datos\n");
    /*Conexion con las bases remotas distribuidas*/
    conectabase(&mysql,"nodo2","jlgr","jlgr","Biblio4");
    conectabase(&mysql2,"nodo2","jlgr","jlgr","Biblio3");
    conectabase(&mysql3,"nodo3","jlgr","jlgr","Biblio3");
    conectabase(&mysql4,"nodo4","jlgr","jlgr","Biblio3");
    conectabase(&mysql5,"nodo5","jlgr","jlgr","Biblio3");


 //Consulta a la base remota Completa
    printf("Busqueda Completa Remota ...\n");
    comienzo= times(&tic_comienzo);
    consulta(&mysql,busqueda,&final,CICLOS,&res_completo);

final=times(&tic_final);
tics_consumidos=final-comienzo;
seg_remoto=(double)tics_consumidos/(double)sysconf(_SC_CLK_TCK);

// consulta a las bases remotas distribuidas en 4
  printf("Busqueda Distribuida Secuencial  ...\n");
  comienzo= times(&tic_comienzo);
  printf("Busqueda en nodo2  ...\n");
  consulta(&mysql2,busqueda,&final,CICLOS,&res_dist_sec1);
  printf("Busqueda en nodo3  ...\n");
  consulta(&mysql3,busqueda,&final,CICLOS,&res_dist_sec2);
  printf("Busqueda en nodo4  ...\n");
  consulta(&mysql4,busqueda,&final,CICLOS,&res_dist_sec3);
  printf("Busqueda en nodo5  ...\n");
  consulta(&mysql5,busqueda,&final,CICLOS,&res_dist_sec4);
  final=times(&tic_final);
  tics_consumidos=final-comienzo;
seg_dist_sec=(double)((double)tics_consumidos/(double)sysconf(_SC_CLK_TCK
));
```

```
  //Consulta a las bases de manera remota distribuida paralela

   pid_t procesoX[10];
   int i,condicion;

   //printf("Estoy antes de los Ifs\n");
   //printf(" Mi Id %d\n",getpid());
   //printf("El de mi Pa %d\n\n",getppid());

// Dividida en 4
printf("Busqueda Distribuida Paralela  ...\n");
comienzo= times(&tic_comienzo);
   if((procesoX[0]=fork())==0)
   {
     printf("Busqueda en nodo2  ...\n");
     consulta(&mysql2,busqueda,&final,CICLOS,&res_dist_par1);
     //printf("Estoy en el primer if en el then\n");
     //printf("Mi Id %d\n",getpid());
     //printf("El de mi Pa %d\n\n",getppid());
     imprime(&res_dist_par1);
     exit(EXIT_SUCCESS);


   }
   else
   {
      if((procesoX[1]=fork())==0)
      {
        printf("Busqueda en nodo3  ...\n");
         consulta(&mysql3,busqueda,&final,CICLOS,&res_dist_par2);
         exit(EXIT_SUCCESS);
      }
      else
      {
          if((procesoX[2]=fork())==0)
         {
       printf("Busqueda en nodo4  ...\n");
         consulta(&mysql4,busqueda,&final,CICLOS,&res_dist_par3);

         //printf(" Mi Id %d\n",getpid());
         //printf("El de mi Pa %d\n\n",getppid());
         exit(EXIT_SUCCESS);
       }
       else
       {
              if((procesoX[3]=fork())==0)
             {
          printf("Busqueda en nodo5  ...\n");
            consulta(&mysql5,busqueda,&final,CICLOS,&res_dist_par4);

         //printf(" Mi Id %d\n",getpid());
         //printf("El de mi Pa %d\n\n",getppid());
```

```
            exit(EXIT_SUCCESS);
          }


        }

      }

   }// Primer Else
for (i=0;i<=3;i++)
  waitpid(procesoX[i],&condicion,0);
  final=times(&tic_final);
  //printf("Estoy Afuera de cualquier if\n");
  //printf("Mi Id %d\n",getpid());
  //printf("El de mi Pa %d\n\n",getppid());

  tics_consumidos=final-comienzo;
seg_dist_paral=(double)((double)tics_consumidos/(double)sysconf(_SC_CLK_T
CK));

  printf("Resultados Completos\n");
  imprime(&res_completo);
  printf("Resultados Distribuido Secuencial Nodo2\n");
  imprime(&res_dist_sec1);
  printf("Resultados Distribuido Secuencial Nodo3\n");
  imprime(&res_dist_sec2);
  printf("Resultados Distribuido Secuencial Nodo4\n");
  imprime(&res_dist_sec3);
  printf("Resultados Distribuido Secuencial Nodo5\n");
  imprime(&res_dist_sec4);
  printf("Resultados Distribuido Paralelo Nodo2\n");
  //imprime(&res_dist_par1);
  printf("Resultados Distribuido Paralelo Nodo3\n");
// imprime(&res_dist_par2);
  printf("Resultados Distribuido Paralelo Nodo4\n");
 // imprime(&res_dist_par3);
  printf("Resultados Distribuido Paralelo Nodo5\n");
 // imprime(&res_dist_par4);

  printf("Numero de Ciclos %d \n",CICLOS);
  printf("Segundos remoto No distriduido (nodo3) tabla completa
%6.2f\n",seg_remoto);
  printf("\n");
  printf("Segundos remoto distribuido secuencial, tabla dividida en %d
%6.2f\n",i,seg_dist_sec);
    printf("\n");
   printf("Segundos remoto distriduido paralelo, tabla dividida en %d
%6.2f\n",i,seg_dist_paral);
    // la funcion sysconf() anterior obtiene el numero de tics por
segundo

//Liberando memoria Resultados
  mysql_free_result(res_completo);
```

```
  //Cerrando Bases
  mysql_close(&mysql1);
  mysql_close(&mysql2);
  mysql_close(&mysql3);
  mysql_close(&mysql4);
  mysql_close(&mysql5);
  //std::cout<<"lolo"<<std::endl;

  // Aqui comienzo con openmosix como primera aproximacion
  int processors=0;
  processors=cpucount();
  std::cout<<"El numero de Procesadores disponibles en este Cluster es:"
<<processors<<std::endl;
  pid_t proceso1;

return 0;
}
```

D. Pruebas y Resultados Obtenidos

Una de las primeras tareas como se indicó en el desarrollo de este trabajo es la de probar la plataforma y su funcionamiento. Para la medición de resultados en el Cluster OpenMosix utilizando MySQL, se utilizó la base de datos de la Biblioteca de la FES-Cuautitlán, la cual contiene los registros del acervo total.

La base de datos utiliza únicamente una tabla la cual tiene la siguiente estructura:

Nombre de campo	Tipo de campo	Descripción
Matriz	Char 20	Edición original
Volumen	char 10	No. de volumen
Cua	entero (llave primaria)	Identificación de Ad.
Clasificación	char 20	Clasificación

La base de datos con la tabla antes mencionada tiene un total de 221,166 registros,
Para las pruebas la tabla fue dividida en 2, 3 y 4 segmentos, de manera uniforme sin ordenación, y sin repetición.

Las pruebas consisten en realizar una búsqueda, en este caso se seleccionó el campo Clasificación, debido a que este no tiene ningún orden y se distribuye bien a lo largo de todos los nodos (los campos matriz y Cua por cuestiones cronológicas tienen cierto orden debido a que inician el orden de adquisición de los materiales).

La búsqueda utilizada en la prueba fue todos los registros cuya clasificación inicie con "HF" lo cual arroja un total de 18,613 registros.

Las pruebas se hicieron en un nodo remoto consultando la tabla completa en este, en los nodos remotos con la base distribuida sin repetición, realizando la búsqueda de manera secuencial. Es decir hasta completar la búsqueda en un nodo se inicia en el siguiente, con las particiones mencionadas anteriormente(2,3.4 y 5 nodos).

Finalmente se realiza la búsqueda en los nodos remotos de manera distribuida, pero paralela, mediante instrucciones fork, las cuales crean procesos hijo que son paralelizados de manera automática por OpenMosix.

En la siguiente figura se esquematiza como se desarrolla la creación de los procesos hijos con la instrucción fork.

En todos los casos se hizo la prueba con 1 búsqueda, y utilizando ciclos de 1,2,4,8,16,32 ciclos, de la misma búsqueda.

Como ya se mencionó el cluster cuenta con 5 nodos cuyos nombres de identificación son:

cluster1
nodo2
nodo3
nodo4
nodo5

Las pruebas se ejecutaron desde el nodo cluster1 el cual no tiene datos de la base, utilizando un switch ethernet de 100 Mbps. mediante cable UTP. Todos los nodos tienen tarjetas 10/100 3com.

La base de datos se distribuyó horizontalmente de la siguiente manera:

En todos los casos se utilizo la base de datos llamada Biblio1

Completa:
nodo3 221,166 registros
tabla: Acervo

Dividida en 2:
Nombre de tabla: Acervo2
nodo3 <110,583
nodo4 >=110,583

Dividida en 3:
Nombre de tabla: Acervo3
nodo3 <73,722
nodo4 >=73,722 <147,444
nodo5 >147,444

Dividida en 4:
Nombre de tabla: Acervo4
nodo2 <55,291
nodo3 >=55,291 <110,583
nodo4 >=110,583 <165,874
nodo5 >=165,874

Los resultados de las pruebas son los siguientes:

Resultados obtenidos se miden en segundos

Numero de Ciclos 1
Segundos remoto No distribuido (nodo3) tabla completa 2.68
Segundos remoto distribuido secuencial, tabla dividida en 2 2.46
Segundos remoto distribuido secuencial, tabla dividida en 3 2.64
Segundos remoto distribuido secuencial, tabla dividida en 4 2.81
Segundos remoto distribuido paralelo, tabla dividida en 2 1.21
Segundos remoto distribuido paralelo, tabla dividida en 3 0.73
Segundos remoto distribuido paralelo, tabla dividida en 4 0.71
El numero de Procesadores disponibles en este Cluster es: 5

Numero de Ciclos 2
Segundos remoto No distribuido (nodo3) tabla completa 4.93
Segundos remoto distribuido secuencial, tabla dividida en 2 4.39
Segundos remoto distribuido secuencial, tabla dividida en 3 4.58
Segundos remoto distribuido secuencial, tabla dividida en 4 4.59
Segundos remoto distribuido paralelo, tabla dividida en 2 2.29
Segundos remoto distribuido paralelo, tabla dividida en 3 1.43
Segundos remoto distribuido paralelo, tabla dividida en 4 1.26
El numero de Procesadores disponibles en este Cluster es:5

Numero de Ciclos 4
Segundos remoto No distribuido (nodo3) tabla completa 8.46
Segundos remoto distribuido secuencial, tabla dividida en 2 8.54
Segundos remoto distribuido secuencial, tabla dividida en 3 8.55
Segundos remoto distribuido secuencial, tabla dividida en 4 8.57
Segundos remoto distribuido paralelo, tabla dividida en 2 4.44
Segundos remoto distribuido paralelo, tabla dividida en 3 3.40
Segundos remoto distribuido paralelo, tabla dividida en 4 2.26
El numero de Procesadores disponibles en este Cluster es:5

Numero de Ciclos 8
Segundos remoto No distribuido (nodo3) tabla completa 16.92
Segundos remoto distribuido secuencial, tabla dividida en 2 17.06
Segundos remoto distribuido secuencial, tabla dividida en 3 17.06
Segundos remoto distribuido secuencial, tabla dividida en 4 17.14
Segundos remoto distribuido paralelo, tabla dividida en 2 8.73
Segundos remoto distribuido paralelo, tabla dividida en 3 5.85
Segundos remoto distribuido paralelo, tabla dividida en 4 4.67
El numero de Procesadores disponibles en este Cluster es:5

Numero de Ciclos 16
Segundos remoto No distribuido (nodo3) tabla completa 33.78
Segundos remoto distribuido secuencial, tabla dividida en 2 34.02
Segundos remoto distribuido secuencial, tabla dividida en 3 34.09
Segundos remoto distribuido secuencial, tabla dividida en 4 34.26
Segundos remoto distribuido paralelo, tabla dividida en 2 17.65
Segundos remoto distribuido paralelo, tabla dividida en 3 12.24
Segundos remoto distribuido paralelo, tabla dividida en 4 9.03
El numero de Procesadores disponibles en este Cluster es:5

Numero de Ciclos 32
Segundos remoto No distribuido (nodo3) tabla completa 67.74
Segundos remoto distribuido secuencial, tabla dividida en 2 67.98
Segundos remoto distribuido secuencial, tabla dividida en 3 68.22
Segundos remoto distribuido secuencial, tabla dividida en 4 68.53
Segundos remoto distribuido paralelo, tabla dividida en 2 35.61
Segundos remoto distribuido paralelo, tabla dividida en 3 23.66
Segundos remoto distribuido paralelo, tabla dividida en 4 17.81
El numero de Procesadores disponibles en este Cluster es:5

BIBLIOGRAFIA.

- CLIENTE/SERVIDOR (2da. Edición).
  HARVEY, Dean.
  EDWARS, Jean.
  McGraw Hill. 2001

- ALGORITMIA DEL PARALELISMO.
  RAYNAL, Michael,
  Traducido por: Dr. FERNANDEZ ROSALES, Rodolfo.
  Omega.

- FUNDAMENTOS Y MODELOS DE BASE DE DATOS.
  CASTANO, Ad. De Miguel.
  PIATTINI VELTHUIS, Mario G.
  AlfaOmega-Rama, 1998

- FUNDAMENTOS DE BASES DE DATOS (4ta. Edición).
  SILVERSCHATZ, Avi.
  KORTH, Hank.
  S, SUNDARSHAN.
  McGraw Hill, 2001

- DISEÑO Y ADMINISTRACION DE BASE DE DATOS.
  HANSEN, Gary.
  HANSEN, James.
  Prentice Hall

- INTRODUCCION A LOS SISTEMAS DE BASE DE DATOS
  ULLMAN, Jeffrey.
  WIDOW, Jennifer.
  Pearson, 1999

- FUNDAMENTOS DE SISTEMAS DE BASE DE DATOS (3ra. Edición).
  ELSMAN, Ramón.
  Afelssen Wesley, 2002

- BASES DE DATOS DESDE CHEN HASTA CODD.
  CRUZ, Irene.
  GOMEZ NIETO, Miguel Ángel.
  AlfaOmega-Rama, 2001

- BASES DE DATOS RELACIONALES:
  E. RIVERO, Cornelio.
  Paraninfo, S.A.

- PRICIPLES OF DITRIBUTED DATABSES SYSTEM (2da. Edición).
  OZSU, M. Tamer.
  VALDUDIEZ, Patrick.
  Prentice Hall.

- DATABASES; PRINCIPLES, PROGRAMMING & PERFORMANCE.
  O'NEIL, Patrick.
  O'NEIL, Elizabeth.
  Morgan Kaufmann Publishers, 2001

- MYSQL (Edición Especial.).
  DOBOIS, Paul.
  Prentice Hall, 2001.

- SOL SERVER: MANUAL DE REFERENCIA.
  GOFFMAN; Gayle.
  McGraw Hill, 2001

- MICROSOFT SQL SERVER AL DESCUBIERTO.
  BJELETICH, Sharon.
  MABLE, Greg.
  Prentice Hall.

- ORACLE 8.
  LONEY, Kevin.
  McGraw Hill, 2001

- CURSO DE PROGRAMACION C/C++
  CEBALLOS, Francisco Javier.
  AlfaOmega, 1993

- LINUX, INSTALACION, ADMINISTRACION Y USO DEL SISTEMA.
  BLANCO, Vicente J.
  AlfaOmega, 1997

- TODO SOBRE LINUX.
  WIELSH, Michael.
  Marcombo.

- PARALLEL PROGRAMMING IN OPENMP
  ROHIT, Chandra.
  DARUM, Leonardo.
  KORH, Dave.
  MAYOAN; Dror.
  MCDONALD, Jeff.
  MENON, Ramesh.
  Morgan Kaufmann Publishers, 2001

REFERENCIAS EN INTERNET.

- http://clusters.unam.mx/
- http://clusters.fisica.uson.mx/clusters_de_Linux.htm
- http://clusters.fisica.uson.mx/proyectos/ white_paper/cap3.html
- http://www.bisente.com/documentos/clustering/memoria.html
- http://www.clusters@top500.org
- http://www.gnu.org
- Clusters Computing Theory, MORRISON Richard S. This document is distributed under the GNU General Public Licence
- PostgreSQL Práctico, WORSLEY, DRAKE.. This document is distributed under the GNU General Public Licence