Universidad de la Ciencias Informáticas

Facultad 4

Trabajo de Diploma para optar por el título de

Ingenieros en Ciencias Informáticas

Título: Simulación del proceso de difusión de mezclas de gases en sólidos poli-cristalinos con múltiples estratos.

Autores: Carlos Rodriguez Acosta

Javier Tamayo Lozada

Tutores: Lic. Edisel Navas Conyedo

Dr. Carlos R. González González

La Habana, Cuba

Junio 2009

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____	_____
Javier Tamayo Lozada	Carlos Rodriguez Acosta	Edisel Navas Conyedo
(Autor)	(Autor)	(Tutor)

Datos de Contacto

Licenciado en Física, en la Facultad de Física en la Universidad de la Habana en el año 2003. Profesor de la Universidad de la Ciencias Informáticas (UCI) de la Facultad 4(2005-2007), siendo profesor de Física. En enero del 2008 hasta el momento es J' de Departamento de Ciencias Básicas. Está inscrito en la Maestría en Física en la Facultad de Física de la Universidad de la Habana, con el tema:"Caracterización de las nanopartículas por mediciones óptimas". Ha realizado varias Publicaciones ellas son:

- ✓ Programa"Extensity", software comercial para la determinación de tamaño de partículas usando mediciones de absorción. Revista TECNOLASER 2005.

- ✓ Aplicaciones de Scattering de la luz para la determinación de tamaños de partículas en soluciones de polímeros y liposomas. Revista TECNOLASER 2005.

- ✓ Reducción del Musterscattering para la determinación de tamaño de partículas para la absorción de la luz. Revista TECNOLASER 2007.

- ✓ Framework de Procesamiento y Análisis de Datos UCICIENCIA 2007.

Email: enavas@uci.cu

"Si los jóvenes fallan, todo fallará. Es mi más profunda convicción que la juventud cubana luchará por impedirlo. Creo en ustedes".

Dedicatoria

Javier:

Dedico este Trabajo de Diploma a:

Mi mamá, a mi papá por que fueron los que me dieron esta vida para lograr lo que he alcanzado y que me han apoyado en todo, por ser ellos quienes me ayudaron a lograr este triunfo que marca un lugar muy importante en mi vida.

A mi novia Adriana que le debo mucho por estar desde que nos conocimos siempre a mi lado en los buenos y en los malos momentos, por ser también la que prácticamente me guio para alcanzar esta meta.

A mis amigos Anthony, Dagmar y a mi compañero de tesis Carlos, en general a todos que me han ayudado cuando lo he necesitado.

Carlos:

Dedico este Trabajo de Diploma a:

Mi mamá, por ser lo más grande en mi vida y la única causante de haberme hecho llegar hasta donde he llegado

A mis abuelos Felicia y Emiliano, que aunque no estén junto a nosotros siempre estarán a mi lado.

A mi prima Sussel y su esposo Henry.

A mis amigos, a mi compañero de tesis Javier y a todos que me han ayudado cuando lo he necesitado.

Agradecimientos

A nuestras familias por darnos su confianza.

A todas las personas que de una forma u otra han colaborado con este Trabajo de Diploma.

A la Revolución y a Fidel por habernos dado la oportunidad de estudiar en esta Universidad.

A la UCI por hacer de nosotros lo que somos.

A nuestro tutores por apoyarnos y ayudarnos en todo momento.

A Eutimio por su ayuda incondicional.

A todas aquellas personas que aportaron en nuestra formación.

A todos los que en algún momento preguntaron por nuestra tesis.

Resumen

El proyecto tiene como objetivo principal simular el proceso de difusión de mezclas de gases en sólidos policristalinos, formados por cristales microporosos tales como las zeolitas, incluyendo defectos de superficie y volumétricos, utilizando métodos de Monte Carlo cinéticos, para lo cual se utilizará la programación paralela de un algoritmo secuencial y su corrida en máquinas computadoras personales interconectadas sobre plataforma LINUX (Debian) para la corrida de programas de computación en forma paralela. El programa deberá permitir el estudio de la separación de gases en membranas zeolíticas reales, lo cual no ha sido abordado con anterioridad. En el proyecto participan dos investigadores del ICIMAF de alto nivel y experiencia, y dos profesores de la UCI que serán los tutores. El programa se elaborará en C++ sobre Debian con posibles subrutinas de FORTRAN y deberá estar optimizado para manejar gran cantidad de elementos de información simultáneamente con estricto ahorro de memoria y de tiempo de ejecución. El medio fundamental para la realización de la investigación será un clúster de computadoras individuales existentes en ambas instituciones. El programa será validado mediante estudios de rendimientos, la comparación con otros programas existentes, con datos experimentales tomados de la literatura y por la presentación en consejos departamentales y tribunales de tesis.

Introducción:

La investigación científica de la Zeolita constituye unos de los retos del mundo actual, la búsqueda constante de propiedades químicas y físicas en esta es la base de la mayoría de las investigaciones de la misma. La Zeolita por su composición tiene propiedades que hacen de ella un catalizador en los procesos de separación de gases. Un gas a través de un cristal de Zeolita permite la separación molecular, muy utilizada en la industria para la producción en derivados mineros.

Con el uso de las Tecnologías y las Comunicaciones se puede lograr el desarrollo de un algoritmo que simule el proceso de difusión(Acción y efecto de difundir) de mezclas de gases en sólidos policristalinos, formados por cristales microporosos tales como las zeolitas.

Situación Problemática

Los cálculos que se realizan a la hora de simular el proceso de difusión de mezclas de gases en sólidos poli-cristalinos son demasiado grandes, por lo que se hace necesario implementar un algoritmo que permita realizar este proceso con múltiples estratos.

Problema a Resolver

¿Cómo realizar la simulación del proceso de difusión de mezclas de gases en sólidos poli-cristalinos (Zeolita) con múltiples estratos?

Objeto de Estudio

La Simulación de procesos Físicos que ocurren en la obtención de derivados del petróleo.

Campo de Acción

Proyecto Transmol.

Objetivo General

Realizar la simulación del proceso de difusión de mezclas de gases en sólidos poli-cristalinos (Zeolita) con múltiples estratos.

Objetivos específicos

Diseño e Implementación de un algoritmo “Máster” encargado de realizar la distribución de los estratos y las moléculas.
Diseño e Implementación de un algoritmo “Esclavo” encargado de procesar moléculas y distribuirlas.

Hipótesis

Si implementan algoritmos para la simulación de difusión de mezclas de gases en sólidos poli-cristalinos con múltiples estratos, debe ser posible disminuir los tiempos de cómputo de la simulación.

Tareas Investigativas

1. Revisión y estudio del estado del arte de las tecnologías y herramientas que se utilizan en el desarrollo de la aplicación.
2. Selección de la metodología de análisis y diseño de sistemas informáticos que faciliten la creación y garanticen la calidad de la aplicación.
3. Selección de las herramientas para llevar a cabo la realización del proyecto, así como la elección de la plataforma en la que se desarrollará la aplicación.
4. Estudio de sistemas de simulación de procesos de difusión molecular.
5. Montaje de clúster de alto rendimiento.
6. Diseño de pruebas a los algoritmos de cómputo paralelo (speed up).

Estructura del Contenido

Esta tesis se estructura de tres capítulos, cada uno de los cuales brinda una información detalla del desarrollo de la aplicación, para lograr en su conjunto su entendimiento y aprendizaje.

Capitulo l, Fundamentación teórica.

Capitulo ll, Descripción de los algoritmos.

Capitulo lll, Modelación y descripción de las pruebas a realizar.

Capítulo I: Fundamentación Teórica

1.1 Introducción:

En el presente capítulo se brinda una panorámica general sobre los sistemas informáticos de cómputo paralelo. Las arquitecturas paralelas tienen un notable incremento en la velocidad de procesamiento [1].

Las herramientas y técnicas a utilizar serán descritas y analizadas a fin de obtener una visión más exacta.

La descripción de las definiciones asociadas al dominio del problema también se tiene en cuenta en el presente capítulo, necesario para entender la propuesta de solución.

1.2 Sistemas de Simulación de procesos de difusión molecular vinculados al campo de acción.

1.2.1 Antecedentes Nacionales:

Los antecedentes nacionales de este proyecto se encuentran en los numerosos estudios sobre las zeolitas naturales y sintéticas efectuados en importantes instituciones nacionales a partir de 1976 y que dieron lugar a la ejecución de un Programa Nacional entre 1980 y 1985. Como parte de estos estudios estuvieron las aplicaciones industriales de las zeolitas que tenían que ver con la separación de gases, la refrigeración, la descontaminación ambiental, etc. Por otra parte, se han realizado muchos trabajos referidos a la obtención y aplicaciones del carbón activado. Este tipo de material tiene también notables propiedades adsortivas y entre otras aplicaciones, puede usarse en la separación de gases. En los últimos años se han estado sintetizando también membranas y monolitos de distintos tipos útiles en la separación de gases aprovechando las diferencias en las movilidades moleculares debidas a las diferentes interacciones con el medio poroso. Otros trabajos que resultan antecedentes a las investigaciones que se proponen en el presente proyecto tienen que ver con las tecnologías de separación de gases por los métodos de PSA y VSA empleando zeolitas, en los cuales se han obtenido notables resultados prácticos pero la modelación teórica se ha referido solamente a la simulación de los procesos a escala macro, asumiendo los fenómenos moleculares de forma muy esquemática y limitada.

La inmensa mayoría de los estudios acometidos por grupos cubanos se han dedicado a la caracterización empírica de las sustancias y el ajuste a modelos existentes; solamente algunos investigadores se han dedicado a modelar la estructura de los materiales buscando propiedades predeterminadas. En estos casos se ha hecho uso de los métodos de la Dinámica Molecular [26]. No se tienen antecedentes del empleo en investigaciones desarrolladas en el país de los métodos de Monte Carlo para la simulación de los procesos difusivos intracristalinos de mezclas de gases, aunque sí se han empleado en otros estudios.

1.2.2 Investigaciones Internacionales:

Los trabajos realizados internacionalmente responden a dos vertientes principales. Una de ellas es la modelación de los fenómenos difusivos a partir de la ley de Fick y sus formulaciones en derivadas parciales en condiciones de frontera concretas, cuya expresión más completa son los trabajos sobre la teoría de Maxwell-Stefan realizados por Krishna y colaboradores [27][28][29][30][31][32] La otra vertiente es la de simulación de los movimientos moleculares a través de los métodos de Monte Carlo. Sin embargo, en todos los trabajos publicados se estudian los sistemas bajo hipótesis simplificadoras que restringen notablemente el alcance de la descripción de la difusión. La más coercitiva de las medidas es el tratamiento bidimensional del movimiento, lo cual impide el análisis de la forma de los canales de las zeolitas. Además, se aborda en todos los casos el movimiento en sistemas idealmente periódicos, soslayando así la consideración de defectos estructurales superficiales y volumétricos de distinto tipo que pudieran tener influencia en la permeación de los componentes de las mezclas de gases.

Los estudios de la difusión molecular de mezclas en láminas adsorbentes microporosas se justifican entonces con el objetivo de precisar la respuesta de los sistemas percoladores adsorbentes porosos ante la modificación de los elementos de la celda elemental de los cristales, la distribución y orientación de los cristalitos, y la cantidad, distribución y orientación de los espacios intergranulares y fronteras.

Las ventajas fundamentales de un método de Monte Carlo están claras; puede darse un seguimiento sistemático al estudio del fenómeno de la difusión microscópica de mezclas en los microporos. Sin embargo, es necesaria una comparación cuidadosa entre los resultados teóricos y computacionales por una parte, y las observaciones experimentales por la otra.

1.2.3 En la Universidad de las Ciencias Informáticas:

Las investigaciones y proyectos realizados en la Universidad de las Ciencias Informáticas no han incluido los procesos de difusión molecular, aunque si se ha comenzado el estudio de procesos de la Industria del Petróleo en conjunto con CUPET y PDVSA. Durante el año 2007 se comienza el estudio para la realización de una aplicación que simule y visualice los procesos de difusión molecular en membranas policristalinas de Zeolitas en estratos no perfectos en la facultad 4, liderado por el Doctor Carlos Ricardo.

1.3 Computación Paralela:

[34]La computación paralela es una técnica de programación en la que muchas instrucciones se ejecutan simultáneamente. Se basa en el principio de que los problemas grandes se pueden dividir en partes más pequeñas que pueden resolverse de forma concurrente ("en paralelo"). Existen varios tipos de computación paralela: paralelismo a nivel de instrucción, paralelismo de datos y paralelismo de tareas.

La computación paralela emplea elementos de procesamiento múltiple simultáneamente para resolver un problema. Esto se logra dividiendo el problema en partes independientes de tal manera que cada elemento de procesamiento pueda ejecutar su parte del algoritmo a la misma vez que los demás. Los elementos de procesamiento pueden ser diversos e incluir recursos tales como un único ordenador con muchos procesadores, varios ordenadores en red, hardware especializado o una combinación de los anteriores [35].

Paralelismo a nivel de instrucción

El paralelismo a nivel de instrucciones *(Instruction-Level Parallelism, ILP)* es una familia de técnicas de diseño para procesadores y compiladores orientadas a explotar el paralelismo existente entre instrucciones de lenguaje máquina.

Paralelismo de datos

El paralelismo de datos se basa en dividir los datos que se tienen que procesar. Típicamente los procesos que están usando esos datos son idénticos entre sí y lo único que hacen es dividir la cantidad de información entre los nodos y procesarla en paralelo. Esta técnica es más usada debido a que es más sencillo realizar el paralelismo.

Paralelismo de tareas

El paralelismo de tareas consiste en asignar distintas tareas a cada uno de los procesadores de un sistema de cómputo. En consecuencia, cada procesador efectuará su propia secuencia de operaciones.

Un proceso debe entenderse como un fragmento de código en ejecución que convive con otros fragmentos. El cómputo paralelo ofrece una gran ventaja en cuanto a costos. Sin embargo, su principal beneficio, la escalabilidad (crecer hacia arquitecturas de mayor capacidad), puede ser difícil de alcanzar aún. Esto se debe a que conforme se añaden procesadores, las disputas por los recursos compartidos se intensifican.

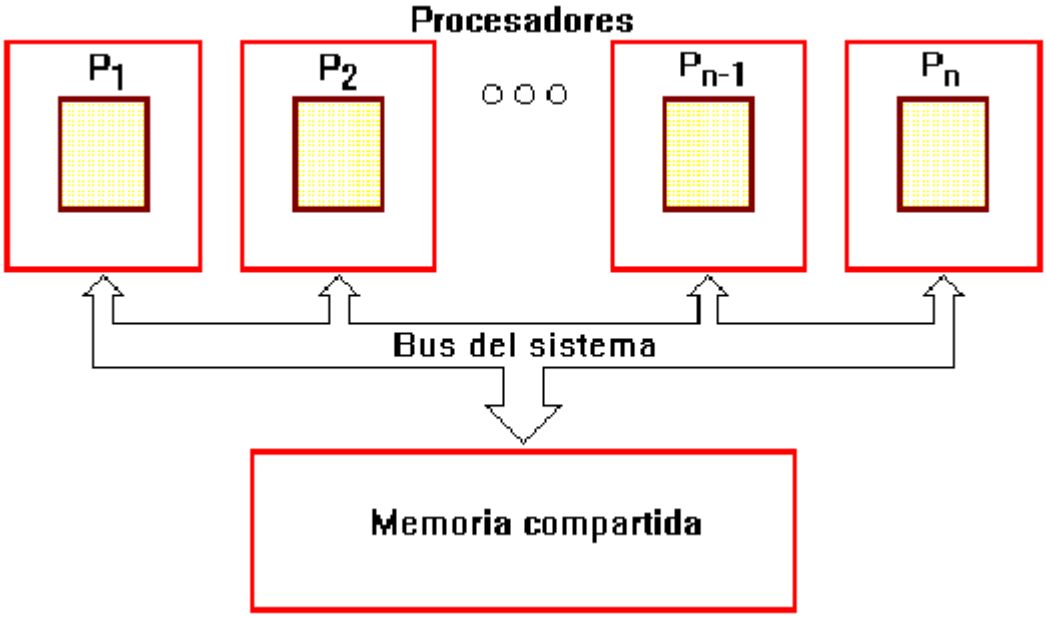
Algunos diseños diferentes de cómputo paralelo enfrentan este problema fundamental:

- Multiprocesamiento simétrico
- Procesamiento masivamente paralelo
- Procesamiento paralelo escalable

Cada diseño tiene sus propias ventajas y desventajas.

1.3.1 Multiprocesamiento Simétrico:

El **Multiprocesamiento simétrico** (**Symmetric Multiprocessing** / **SMP**) tiene un diseño simple pero aún así efectivo. En SMP, múltiples procesadores comparten la memoria RAM y el bus del sistema. Este diseño es también conocido como **estrechamente acoplado** (**Tightly Coupled**), o **compartiendo todo** (**Shared Everything**).

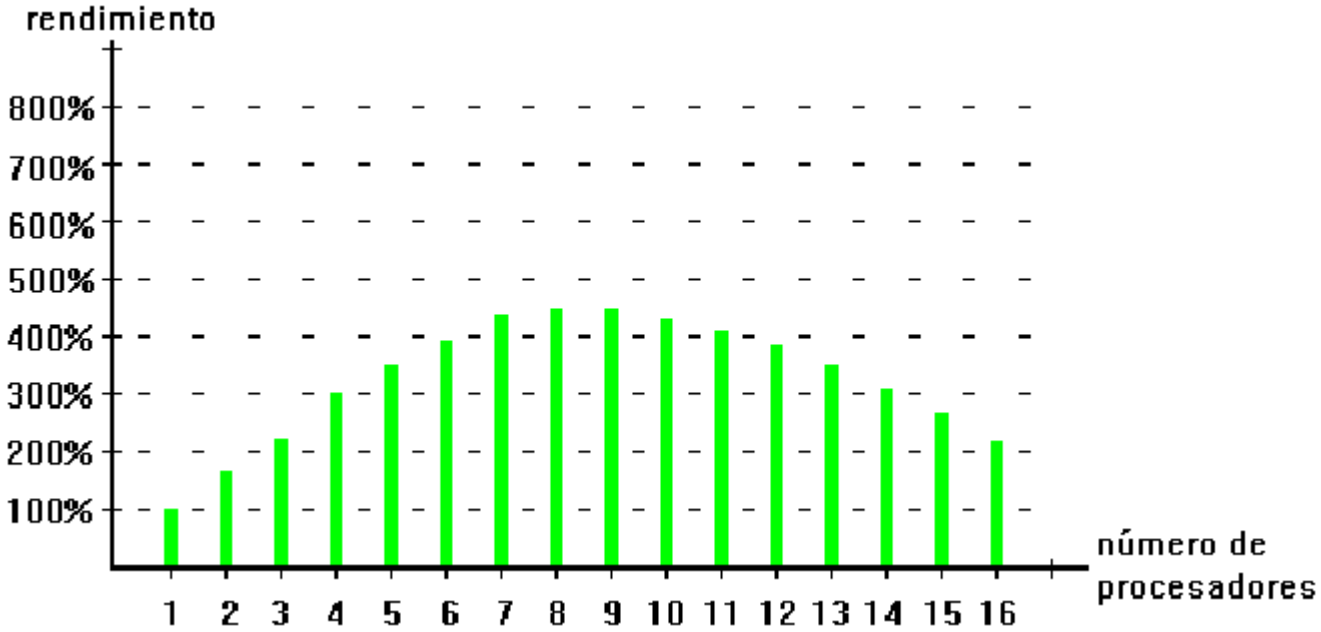


Debido a que SMP comparte globalmente la memoria RAM, tiene solamente un espacio de memoria, lo que simplifica tanto el sistema físico como la programación de aplicaciones. Este

espacio de memoria único permite que un **Sistema Operativo con Multiconexión** (**Multithreaded Operating System**) distribuya las tareas entre varios procesadores, o permite que una aplicación obtenga la memoria que necesita para una simulación compleja. La memoria globalmente compartida también vuelve fácil la sincronización de los datos.

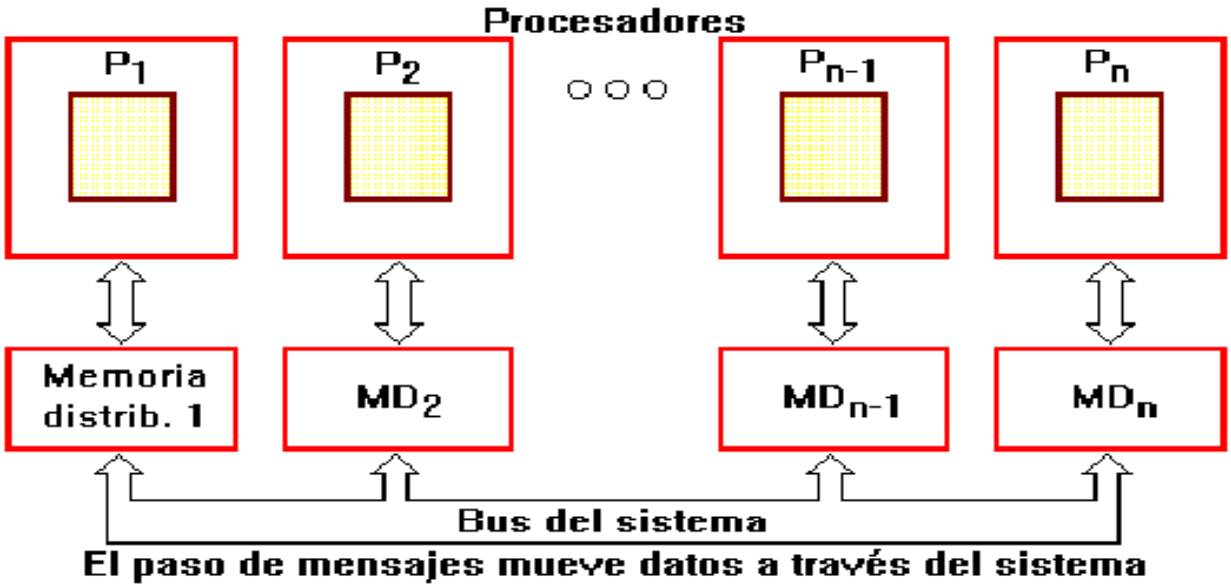
SMP es uno de los diseños de procesamiento paralelo más maduro. Apareció en los supercomputadores **Cray X-MP** y en sistemas similares hace década y media (en 1983).

Sin embargo, esta memoria global contribuye el problema más grande de SMP: conforme se añaden procesadores, el tráfico en el bus de memoria se satura. Al añadir memoria caché a cada procesador se puede reducir algo del tráfico en el bus, pero el bus generalmente se convierte en un cuello de botella al manejarse alrededor de ocho o más procesadores. SMP es considerada una tecnología no escalable.

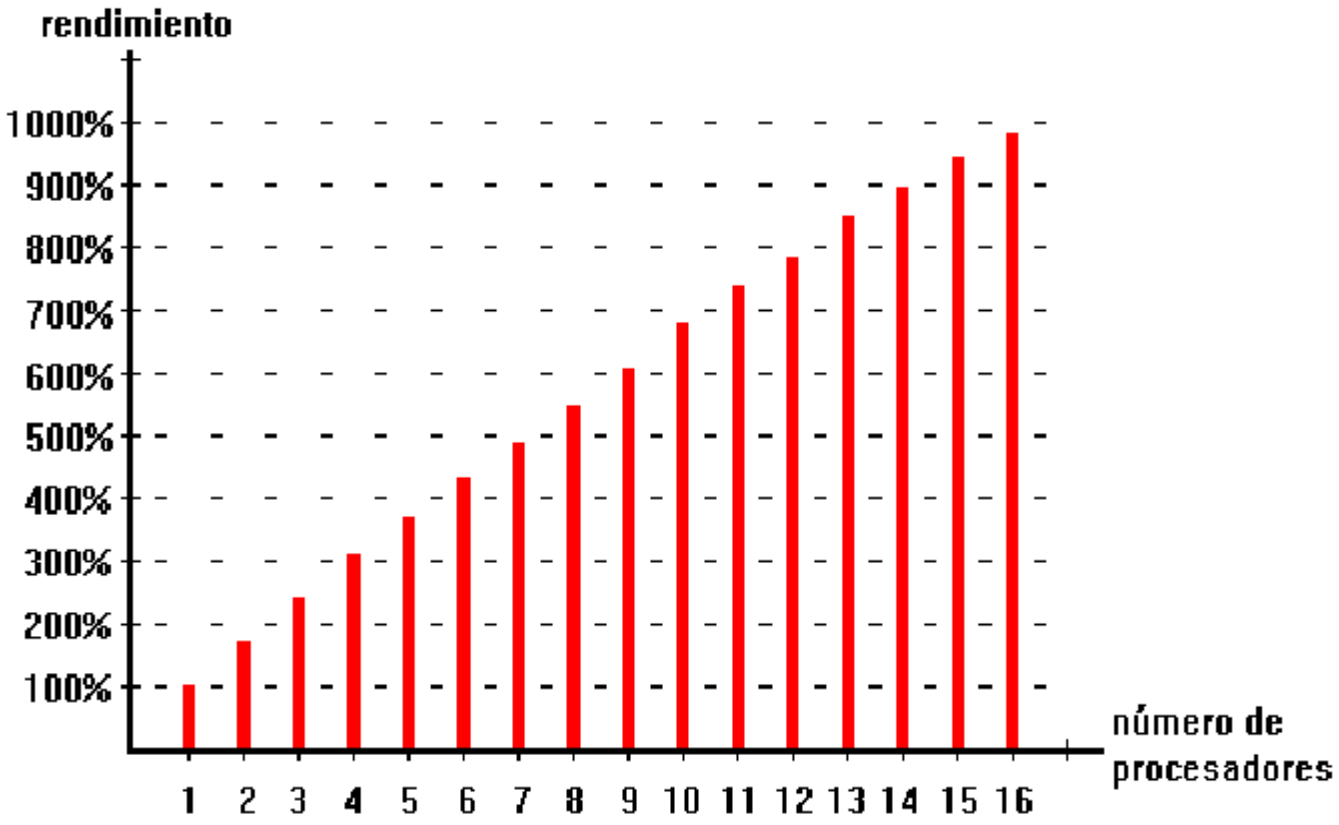


1.3.2 Procesamiento Masivamente Paralelo:

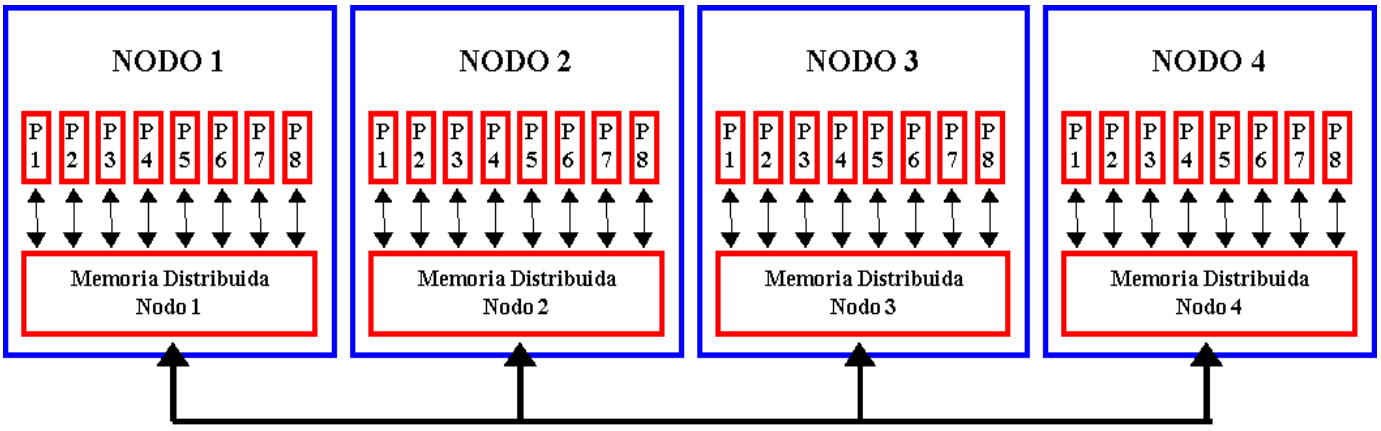
El Procesamiento masivamente paralelo (**Massively parallel processing** / **MPP**) es otro diseño de procesamiento paralelo. Para evitar los cuellos de botella en el bus de memoria, MPP no utiliza memoria compartida. En su lugar, distribuye la memoria RAM entre los procesadores de modo que se asemeja a una red (cada procesador con su memoria distribuida asociada es similar a un computador dentro de una red de procesamiento distribuido). Debido a la distribución dispersa de los recursos RAM, esta arquitectura es también conocida como **dispersamente acoplada** (**loosely coupled**), o **compartiendo nada** (**shared nothing**).



Para tener acceso a la memoria fuera de su propia RAM, los procesadores utilizan un esquema de **paso de mensajes** análogo a los **paquetes de datos** en redes. Este sistema reduce el tráfico del bus, debido a que cada sección de memoria observa únicamente aquellos accesos que le están destinados, en lugar de observar todos los accesos, como ocurre en un sistema SMP. Únicamente cuando un procesador no dispone de la memoria RAM suficiente, utiliza la memoria RAM sobrante de los otros procesadores. Esto permite sistemas MPP de gran tamaño con cientos y aún miles de procesadores. MPP es una tecnología escalable.



El **RS/6000 Scalable Powerparallel System** de IBM (SP2) es un ejemplo de sistema MPP, que presenta una ligera variante respecto al esquema genérico anteriormente planteado. Los procesadores del RS/6000 se agrupan en nodos de 8 procesadores, los que utilizan una única memoria compartida (tecnología SMP). A su vez estos nodos se agrupan entre sí utilizando memoria distribuida para cada nodo (tecnología MPP). De este modo se consigue un diseño más económico y con mayor capacidad de crecimiento.



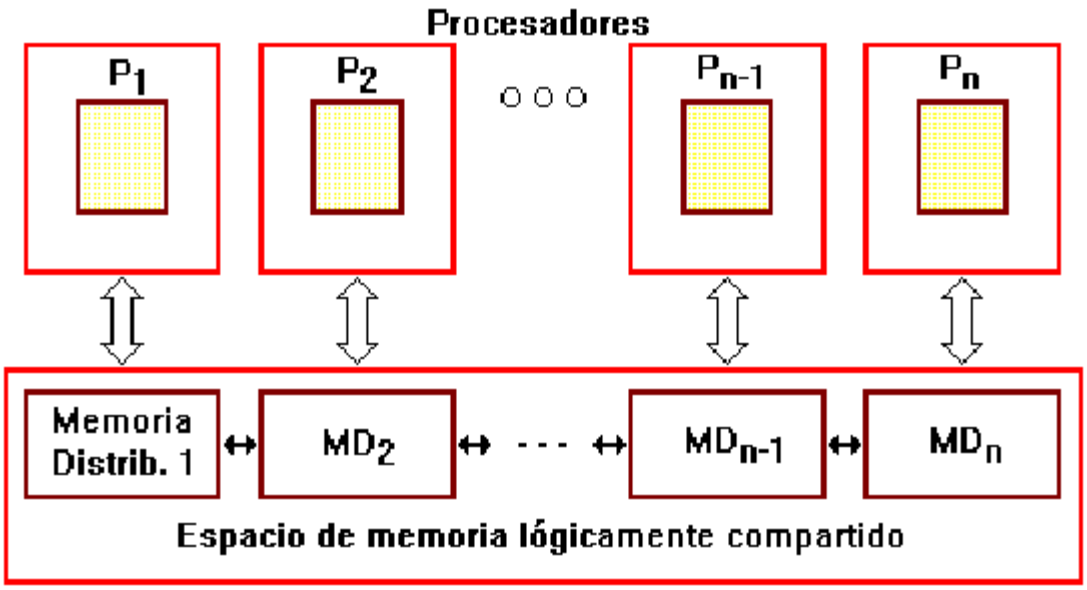
La parte negativa de MPP es que la programación se vuelve difícil, debido a que la memoria se rompe en pequeños espacios separados. Sin la existencia de un espacio de memoria globalmente compartido, correr (y escribir) una aplicación que requiere una gran cantidad de RAM (comparada con la memoria local), puede ser difícil. La sincronización de datos entre tareas ampliamente distribuidas también se vuelve difícil, particularmente si un mensaje debe pasar por muchas fases hasta alcanzar la memoria del procesador destino.

Escribir una aplicación MPP también requiere estar al tanto de la organización de la memoria manejada por el programa. Donde sea necesario, se requieren insertar comandos de paso de mensajes dentro del código del programa. Además de complicar el diseño del programa, tales comandos pueden crear dependencias de hardware en las aplicaciones. Sin embargo, la mayor parte de vendedores de computadores han salvaguardado la portabilidad de las aplicaciones adoptando, sea un mecanismo de dominio público para paso de mensajes conocido como **Máquina virtual paralela** (**parallel virtual machine** / **PVM**), o un estándar en fase de desarrollo llamado **Interfaz de Paso de Mensajes** (**Message Passing Interface** / **MPI**), para implementar el mecanismo de paso de mensajes.

1.3.3 Procesamiento Paralelo Escalable:

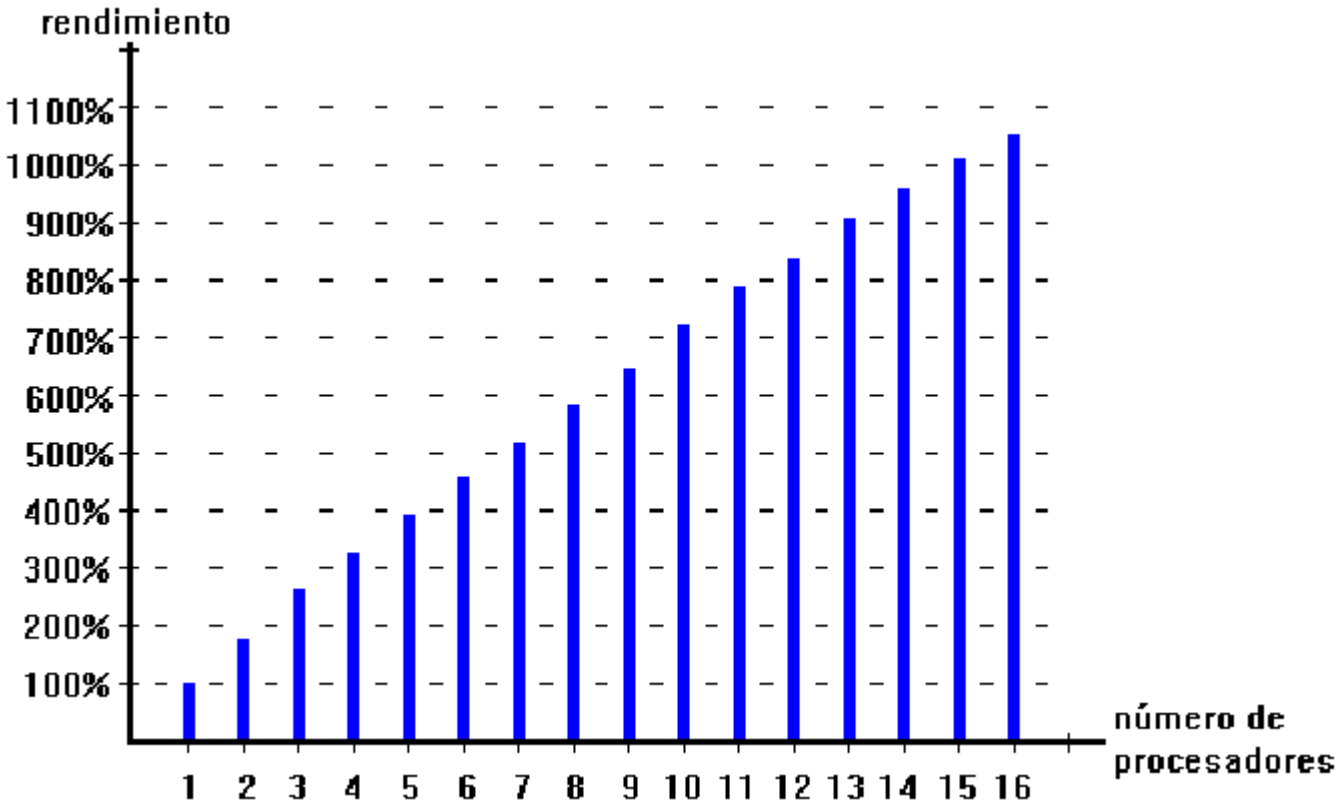
¿Cómo superar las dificultades de SMP y MPP? La última arquitectura paralela, el **Procesamiento paralelo escalable** (**Scalable parallel processing** / **SPP**), es un híbrido de SMP y MPP, que utiliza una memoria jerárquica de dos niveles para alcanzar la escalabilidad. La primera capa de memoria consiste de un nodo que es esencialmente un sistema SMP completo, con múltiples procesadores y su memoria globalmente compartida.

Se construyen sistemas SPP grandes interconectando dos o más nodos a través de la segunda capa de memoria, de modo que esta capa aparece lógicamente, ante los nodos, como una memoria global compartida.



El hardware direcciona los accesos a los bloques de memoria distribuida

La memoria de dos niveles reduce el tráfico de bus debido a que solamente ocurren actualizaciones para mantener coherencia de memoria. Por tanto, SPP ofrece facilidad de programación del modelo SMP, a la vez que provee una escalabilidad similar a la de un diseño MPP.



1.3.4 ¿Como surgió la Computación Paralela?

La Computación Paralela comienza su historia con el interés de compañas productoras de circuitos electrónicos para computadoras con capacidad de procesamiento paralelo. Para el año 1955, IBM desarrolla circuitos aritméticos paralelos binarios unido a una unidad de punto flotante para el IBM 704[15], que aceleraba los procesos de cómputo numérico de manera considerable frente a las soluciones existentes (aritmético-lógicas). Algunas mejoras a este diseño fueron añadidas como la utilización IBM se traza la meta de desarrollar una computadora 100 veces más rápida que las de su época, resultado de un proyecto llamado IBM 7030[16].

De esta manera fueron desarrollándose un conjunto de proyectos que tenían como meta la construcción de superordenadores que basaban su rapidez en la realización de cómputo paralelo pero no existían aún contribuciones importantes en el soporte paralelo del software desarrollado para estos. Un avance notable fue el lanzamiento por Bull[17] en 1958 del _ Gamma 60_ que incluía dentro del flujo de instrucciones el join y el fork , que permitían desarrollar algoritmos para cómputo paralelo brindando el soporte para sistemas multitarea. En ese mismo año, dos empleados de IBM, John Cocke y Daniel Slotnick, plantean el uso del

paralelismo en cálculos numéricos para acelerar los procesos de cómputo lo que constituyó la base para trabajos posteriores.

Para el año 1959 Sperry Rand, entrega la computadora LARC (Livermore Automatic Research Computer) que unido a la STRETCH (IBM 7030) de IBM constituyen las primeras computadoras de gran envergadura en la utilización de los transistores. Utilizando lenguaje ensamblador en vez de lenguaje de máquina, poseía un procesador de entrada y salida independiente que podía operar en paralelo con dos unidades de procesamiento. Esta fue desarrollada para procesar grandes volúmenes de datos en laboratorios de investigación de energía atómica, solo fueron construidas dos.

Luego de estos proyectos se desarrollan avances importantes en el hardware impulsado por el uso de los transistores pero no es hasta el año 1965 cuando General Electric, el MIT y AT&T Bell Laboratories, comienzan el desarrollo de Multics [18], un sistema operativo de memoria compartida, multiprocesamiento y tiempo compartido, que aunque abandonado por Bell en 1969, aportó ideas para el desarrollo de posteriores sistemas operativos. Así desarrolladores de Multics, influenciados por el proyecto se dieron a la tarea de construir el sistema operativo Unix.

Un aporte importante en el área fue el planteamiento del Problema de las Regiones Críticas por Edsger Dijkstra en 1965, donde describe el uso de los semáforos como solución a la manipulación de recursos compartidos (direcciones de memoria). Este aporte constituyó un avance notable para el desarrollo de sistemas operativos multitarea y programas con múltiples hilos y múltiples procesos. En este mismo año James W. Cooley y John W. Tuke describen el algoritmo de la Transformada Rápida de Fourier[19] el cual es considerado un gran consumidor de ciclos de puntos flotantes, el cual tiene gran aplicación en la multiplicación de números enteros grandes, filtrado digital y tratamiento digital de señales resolviendo ecuaciones diferenciales parciales.

La Ley de Amdahl [19], publicada por Gene Amdahl, científico de IBM, estableció en 1967 un marco teórico que describe matemáticamente el aceleramiento que se puede esperar al realizar de manera simultánea, series de tareas en una arquitectura paralela. Este aporte constituye sin dudas la base de la ingeniería del cómputo paralelo en sistemas multiprocesador o de clúster.

El comienzo de la construcción de ordenadores vectoriales fue marcado en 1976 por Cray Research con la Cray-1 que daba un salto revolucionario en la velocidad de procesamiento, dado que esta tecnología se basa en la posibilidad de ejecutar operaciones matemáticas sobre múltiples datos de manera simultánea.

Alrededor del año 1983, los protocolos de comunicación tenían la madurez necesaria para compartir recursos, se establecieron las condiciones para crear sistemas distribuidos. Esto lleva al surgimiento del clúster como otra tendencia en la súper computación, pudiendo unir un conjunto de computadoras (nodos) mediante una red de alta velocidad para que trabajen de forma cooperativa en la realización de tareas complejas. El surgimiento de Internet marcó el desarrollo de nuevas herramientas para minimizar los tiempos de procesamiento y surge la Computación Distribuida como un nuevo modelo para resolver problemas de computación masiva.

En la actualidad, la computación paralela está siendo utilizada en multitud de campos para el desarrollo de aplicaciones y el estudio de problemas que requieren gran capacidad de cómputo, bien por el gran tamaño de los problemas que abordan o por la necesidad de trabajar con problemas en tiempo real. De esta forma, el paralelismo en la actualidad, además de constituir diversas líneas abiertas de intensa labor investigadora, puede encontrarse en infinidad de aplicaciones en campos muy variados, entre los que destacamos:

- Modelado predictivo y simulación: se realiza mediante extensos experimentos de simulación por computador que con frecuencia acarrean computaciones a gran escala para obtener la precisión y el tiempo de respuesta deseado. Entre estos modelados destacamos la previsión meteorológica numérica y la oceanografía.
- El desarrollo industrial también reclama el uso de computadores para progresar en el diseño y automatización de proyectos de ingeniería, la inteligencia artificial y la detección remota de los recursos terrestres. En este campo destacamos: la inteligencia artificial y automatización (procesamiento de imágenes, reconocimiento de patrones, visión por computador, comprensión del habla, deducción automática, robótica inteligente, sistemas expertos por computador, ingeniería del conocimiento, etc.).
- Investigación médica: En el área médica los computadores rápidos son necesarios en tomografía asistida, diseño de corazones artificiales, diagnóstico hepático, estimación de daños cerebrales y estudios de ingeniería genética.

1.3.5 Actualidad

[33]Intel Corporation y Microsoft Corporation se asocian con instituciones académicas para crear dos Centros de Investigación de Computación Paralela (UPCRC), destinados a acelerar adelantos en computación paralela avanzada para consumidores y empresas en los sectores

de las computadoras de escritorio y portátiles. Los nuevos centros de investigación se localizarán en la Universidad de California en Berkeley (UC Berkeley) y en la Universidad de Illinois en Urbana-Champaign (UIUC). Microsoft e Intel han destinado un capital combinado de 20 millones de dólares a los centros de investigación de Berkeley y UIUC en los próximos cinco años. Un adicional de 8 millones de dólares provendrá de UIUC, y UC Berkeley ha solicitado 7 millones de dólares en fondos de un programa con apoyo estatal para igualar las aportaciones de la industria. La investigación se centrará en integrar adelantos a aplicaciones de programación, arquitectura y sistemas operativos paralelos. Ésta es la primera alianza conjunta de investigación entre la industria y universidades en los Estados Unidos enfocada en la computación paralela de vanguardia.

1.4 Clúster:

[14]El concepto de clúster fue inicialmente definido por Digital Equipment Corporation (DEC). Según DEC, un clúster es un grupo de computadoras que están interconectadas y funcionan como una sola unidad de proceso de información.

- ***Escalabilidad****:* capacidad de un equipo de hacer frente a volúmenes de trabajo cada vez mayores, sin dejar por ello de prestar un nivel de rendimiento aceptable.

- ***Disponibilidad***: es la capacidad de estar presente, de estar listo en un determinado momento en el que se quiere hacer uso.

- *Fiabilidad*: es la probabilidad de funcionamiento correcto.

Un clúster puede presentarse como una solución de especial interés sobre todos a nivel de empresas, las cuales pueden aprovecharse de estas especiales características de computación para mantener sus equipos actualizados por un precio bastante más económico que el que les supondría actualizar todos sus equipos informáticos y con unas capacidades de computación que en muchos casos pueden llegar a superar a hardwares de última generación.
Básicamente podríamos distinguir tres tipos de clúster atendiendo al uso que queramos darle:

- ***Fail-over****(Alta-disponibilidad):* Consiste en la conexión de una o varias computadoras conectadas en red utilizándose una conexión *heartbeat* para monitorear cual de sus servicios está en uso, así como la sustitución de una máquina por otra cuando uno de sus servicios haya caído.

- ***Load-balancing*** *(Balanceo de Carga):* Utilizado en los servidores web, el clúster verifica cual de las máquinas de éste posee mayores recursos libres y así, asignarle el trabajo pertinente. Actualmente, los *clúster load-balancing* son también *fail-over* con el extra de balanceo de carga y número de nodos.
- ***High-Performance*** *(Alto-rendimiento):* Clúster destinado al alto rendimiento, capacidad muy alta de proceso para cómputo de grandes volúmenes de datos.
Un clúster de alto rendimiento es un conjunto de computadoras utilizadas como un recurso unificado de procesamiento que comparte una administración común.

Los clústers han evolucionado para apoyar actividades en aplicaciones que van desde supercómputo y software de misiones críticas, servidores Web y comercio electrónico, bases de datos de alto rendimiento.
El cómputo en clústers 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.

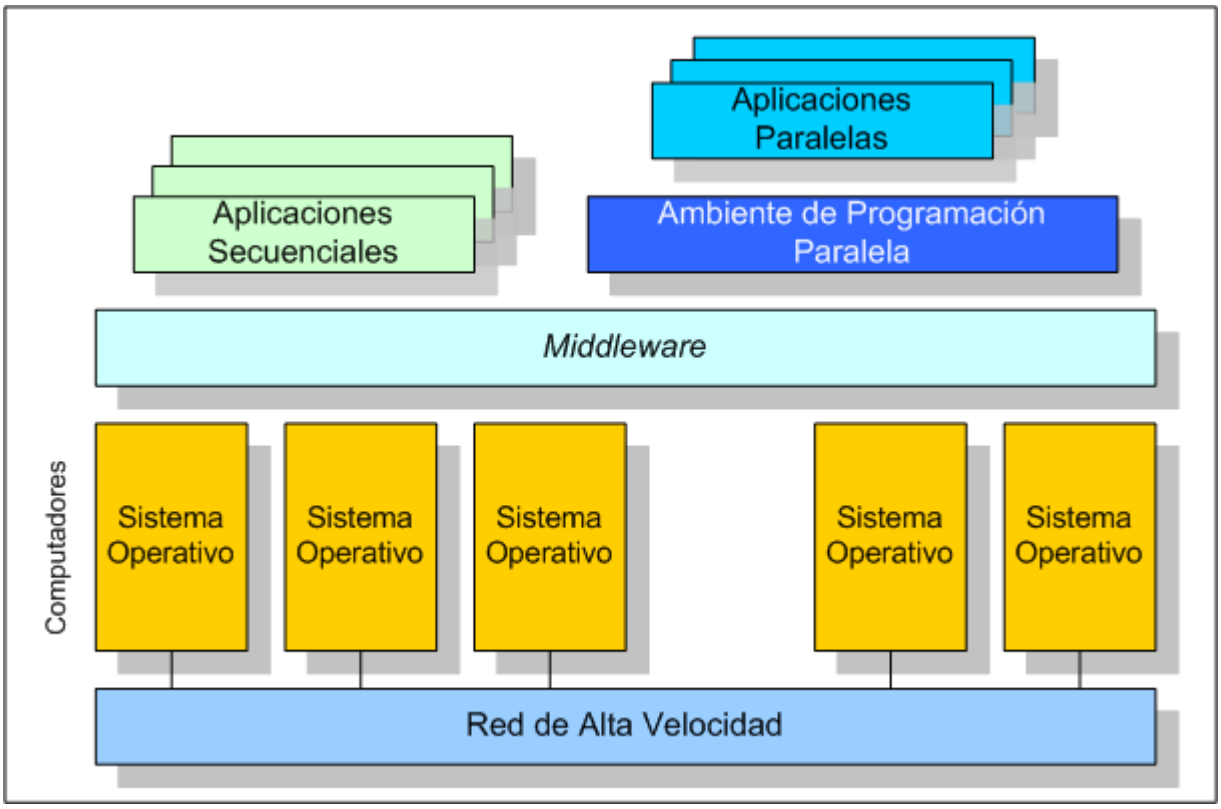
1.4.1 ¿Cómo funciona un clúster?

En su parte central, la tecnología de clústers consta de dos partes. La primera componente, 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 clústers.
La segunda componente es la interconexión de hardware entre las máquinas (nodos) del clúster. 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 clúster intercambian entre si asignación de tareas, actualizaciones de estado y datos del programa. Existe otra interfaz de red que conecta al clúster con el mundo exterior.

1.4.2 Componentes de un *Clúster:*

En general, un *clúster* necesita de varios componentes de software y hardware para poder funcionar:

- Nodos
- Sistemas Operativos
- Conexiones de Red
- Middleware (capa de abstracción entre el usuario y los sistemas operativos)
- Ambientes de Programación Paralela.
- Aplicaciones (pueden ser paralelas o no).



Nodos

Pueden ser simples computadores, sistemas multiprocesador o estaciones de trabajo.

Sistemas Operativos

Debe ser de fácil uso y acceso y permitir además múltiples procesos y usuarios.

Conexiones de Red.

Los nodos de un *clúster* pueden conectarse mediante una simple red Ethernet, o puede utilizar tecnologías especiales de alta velocidad como **Fast Ethernet**, **Gigabit Ethernet**, **Myrinet**, **Infiniband**.

Middleware

El *middlewar*e es un software que generalmente actúa entre el sistema operativo y las aplicaciones con la finalidad de proveer:

- Un interfaz único de acceso al sistema, denominado SSI (*Single System Image*), el cual genera la sensación al usuario de que utiliza un único computador muy potente;
- Herramientas para la optimización y mantenimiento del sistema: migración de procesos, *checkpoint-restart* (detener uno o varios procesos, migrarlos a otro nodo y continuar su funcionamiento), balanceo de carga, tolerancia a fallos, etc.;
- Escalabilidad: debe poder detectar automáticamente nuevos nodos conectados al *clúster* para proceder a su utilización.

Ambiente de Programación Paralela

Los ambientes de programación paralela permiten implementar algoritmos que hagan uso de recursos compartidos: CPU (*Central Processing Unit*), memoria, datos y servicios.

1.4.3 Sistemas clústers implementados:

Existen una gran cantidad de clústers implementados, a continuación hacemos referencia a algunos de ellos.

Beowulf

Fue construido por Donald Becker y Thomas Sterling en 1994. Fue construido con 16 computadores personales con procesadores Intel DX4 de 200 MHz, que estaban conectados a través de un **switch Ethernet**. El rendimiento teórico era de 3.2 GFlops.

Berkeley NOW

El sistema NOW de Berkeley estuvo conformado por 105 estaciones de trabajo Sun Ultra 170, conectadas a través de una red Myrinet. Cada estación de trabajo contenía un microprocesador Ultra1 de 167 MHz, caché de nivel 2 de 512 MB, 128 MB de memoria, dos discos de 2.3 GB, tarjetas de red **Ethernet** y **Myrinet**. En abril de 1997, NOW logró un rendimiento de 10 GFlops.

Google

Durante el año 2003, el clúster Google llegó a estar conformado por más de 15.000 computadores personales. En promedio, una consulta en Google lee cientos de megabytes y consume algunos billones de ciclos del CPU.

Clúster PS2

En el año 2004, en la Universidad de Illinois en Urbana-Champaign, Estados Unidos, se exploró el uso de consolas Play Station 2 (PS2) en cómputo científico y visualización de alta resolución. Se construyó un clúster conformado por 70 PS2; utilizando Sony Linux Kit (basado en Linux Kondora y Linux Red Hat) y MPI.

Clúster X

En la lista "TOP 500" de noviembre de 2004 fue considerado el séptimo sistema más rápido del mundo; sin embargo, para julio de 2005 ocupa la posición catorce. Clúster X fue construido en el Tecnológico de Virginia en el 2003; su instalación fue realizada por estudiantes del Tecnológico. Está constituido por 2200 procesadores Apple G5 de 2.3 GHz. Utiliza dos redes: Infiniband 4x para las comunicaciones entre procesos y Gigabit Ethernet para la administración. Clúster X posee 4 Terabytes de memoria RAM y 176 Terabytes de disco duro, su rendimiento es de 12.25 TFlops.

MareNostrum

En julio de 2004 se creó el Centro de Supercomputación de Barcelona (BSC), de la Universidad Técnica de Cataluña, España. El BSC creó el clúster MareNostrum. En noviembre de 2004 MareNostrum se ubicó en el "TOP 500", como el primer clúster más veloz y el cuarto sistema más rápido del mundo; sin embargo, para julio de 2005 se ubicó en la quinta posición. Está conformado por 3564 procesadores PowerPC970 de 2.2 GHz. Utiliza una red **Myrinet**. Su rendimiento es de 20.53 TFlops.

Thunder

Thunder fue construido por el Laboratorio Nacional Lawrence Livermore de la Universidad de California. Está conformado por 4096 procesadores Intel Itanium2 Tiger4 de 1.4GHz. Utiliza una red basada en tecnología Quadrics. Su rendimiento es de 19.94 TFlops. Se ubicó en la segunda posición del "TOP 500" durante junio de 2004, luego en la quinta posición en noviembre de 2004 y en la lista de julio de 2005 se ubicó en la séptima posición.

ASCI Q

ASCI Q fue construido en el año 2002 por el Laboratorio Nacional Los Álamos, Estados Unidos. Está constituido por 8192 procesadores AlphaServer SC45 de 1.25 GHz. Su rendimiento es de 13.88 TFlops. Se ubicó en la segunda posición del "TOP 500" durante junio y noviembre de

2003, luego en la tercera posición en junio de 2004, en la sexta posición en noviembre de 2004 y en la doceava posición en julio de 2005.

1.4.4 Clúster Beowulf:

Una de las herramientas de más auge en la actualidad son los llamados clúster Beowulf (Anexo1), los cuales presentan diversas capacidades para el cómputo paralelo con un relativo alto rendimiento, siendo muy atractivo para campos como el de la seguridad computacional donde se hace necesario contar con sistemas robustos de gran capacidad de procesamiento.
Son estas características, las que permiten que se seleccione el clúster Beowulf para la solución del problema.

Beowulf no es un paquete de software especial, ni una nueva topología de red ni un núcleo modificado. Beowulf es una tecnología para agrupar computadores basados en el sistema operativo Linux buscando formar un supercomputador virtual paralelo. En 1994 bajo el patrocinio del proyecto ESS del Centro de la Excelencia en Ciencias de los Datos y de la Información del Espacio (CESDIS), Thomas Sterling y Don Becker crearon el primer clúster Beowulf con fines de investigación.

Los principales componentes de un clúster Beowulf son el procesador, memoria principal, red de intercomunicación entre nodos, un sistema de almacenamiento secundario y software que permita la comunicación entre los procesos de diferentes nodos (bibliotecas de envío de mensajes). Se le llama clúster de alto rendimiento ya que se construye con la intención de optimizar los procesos ejecutados tratando de lograr el mayor número de operaciones de punto flotante en el menor tiempo posible [12].
Los procesos que se ejecutan en un clúster corren bajo un entorno de cómputo paralelo. El cómputo paralelo ha sido un punto clave en el desarrollo de nuevos algoritmos para optimizar las capacidades de procesamiento y poder desarrollar cómputo de alto rendimiento.
En el año de 1966 Michael J. Flynn introdujo un esquema de clasificación de computadoras que se basaba en la multiplicidad de instrucciones y el flujo de datos que podía manejar cada tipo de computadoras: SISD (Una instrucción un flujo de datos), MISD (Muchas instrucciones un flujo de datos), SIMD (Una instrucción muchos flujos de datos), MIMD (Muchas instrucciones Muchos datos). Los clústers se encuentran clasificados en el último tipo [13].

El uso del cómputo de alto rendimiento en el área científica se ha propagado de manera tal que se ha generado una rama dedicada al cómputo científico y con esto han surgido nuevas metodologías: experimental, teórica y computacional.

1.4.4.1 ¿Qué es un BeoWulf?

Beowulf es el poema épico más antiguo que se conserva escrito en inglés. Es la historia de un héroe de gran fuerza y valentía que derrotó a un monstruo llamado Grendel.
A pesar de que existen diversas definiciones de lo que es un clúster BeoWulf, a continuación se enuncia una posible definición que trata de armonizar las diferentes opiniones:

[3]Beowulf es una arquitectura multi-computador que puede ser usada para computación paralela. Es un sistema que generalmente consiste en un nodo servidor y uno o varios nodos clientes conectados a través de Ethernet u otro sistema de red. Es un sistema construido mediante hardware corriente, como cualquier PC capaz de ejecutar Linux, tarjetas de red Ethernet y cables. No contiene ningún componente hardware específico y es fácilmente reproducible. Beowulf también usa software corriente, como el sistema operativo Linux, la Parallel Virtual Machine (PVM) y el Message Passing Interface (MPI). El nodo servidor controla el clúster completo y sirve los ficheros a los nodos clientes. De igual forma hace de consola del clúster y es su enlace con el mundo exterior.

Los grandes sistemas Beowulf pueden tener más de un nodo servidor, y también algunos nodos dedicados a tareas específicas, en la mayoría de los casos, los nodos clientes de un sistema Beowulf son mudos, y cuanto más mudos, mejor. Los nodos son configurados y controlados por el nodo servidor, y sólo hacen lo que se les dice que hagan. En las configuraciones donde los clientes no tienen discos, los nodos clientes ni siquiera conocen su IP hasta que el servidor les dice cuál es, generalmente los nodos clientes no tienen teclado ni monitor, y sólo se puede acceder a ellos a través de acceso remoto o por medio de terminales serie. Puede pensarse en un nodo Beowulf como un conjunto CPU + memoria que puede ser conectado al clúster, del mismo modo que una CPU o un módulo de memoria puede ser conectado a una placa base.
A pesar de que hay muchos paquetes de software, como modificaciones del núcleo, bibliotecas para PVM y MPI y herramientas de configuración que hacen más rápida, fácil de configurar y

de usar la arquitectura Beowulf, cualquiera puede construir una máquina de tipo Beowulf usando una distribución normal de Linux sin ningún software adicional, si se poseen dos computadores Linux en red que comparten el sistema de ficheros /home vía NFS(**Network File System**) y se permiten ejecutar shells remotos (**rsh**), entonces podría decirse que se tiene una máquina Beowulf simple de dos nodos.

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

- Clase I. Son sistemas compuestos por computadores cuyos componentes cumplen con la prueba de certificación “Computer Shopper”, lo que significa que sus elementos son de uso común, y pueden ser adquiridos muy fácilmente en cualquier tienda distribuidora.
- Clase II. Son sistemas compuestos por computadores cuyos componentes no pasan la prueba de certificación “Computer Shopper”, lo que significa que sus componentes no son de uso común y por tanto no pueden encontrarse con la misma facilidad que los componentes de sistemas de la clase anterior. Los equipos ubicados en esta categoría pueden presentar un nivel de prestaciones superior al de la Clase I.

Diseño

A la hora de construir un clúster Beowulf es necesario tener en cuenta diversos factores para el diseño del mismo de forma que las decisiones a tomar contribuyan al mejor desenvolvimiento de la máquina según los propósitos para los cuales se implementan. Los diferentes puntos que se deben estudiar en el diseño de un clúster Beowulf son los siguientes:

a. Disco

Existen varios métodos para configurar los medios de almacenamiento en un clúster Beowulf, los cuales difieren en rendimiento, precio y facilidades en la administración.

Clientes sin disco *(Disk-less)*

Los nodos esclavos o clientes no poseen disco duro interno y toman todos los sistemas de archivos a través de la red. Es el nodo maestro el que proporciona a través de NFS los sistemas de archivos para los nodos esclavos.

Instalación Local Completa en los Clientes

Todo el software, tanto el sistema operativo como las aplicaciones, son instaladas en los discos internos de cada nodo cliente. Esta configuración reduce a cero el tráfico NFS para obtener el sistema operativo o cualquier otra aplicación por parte de los nodos esclavos.

Instalación NFS Estándar

Esta configuración es el punto medio de las dos anteriores. El sistema operativo se encuentra en los discos internos de los nodos esclavos y estos obtienen los directorios home de los usuarios y los sistemas de archivos que contienen las aplicaciones, a través de NFS, desde el nodo maestro.

Sistemas de Archivos Distribuidos

Los sistemas de archivos son aquellos que son compartidos por todos los nodos, es decir, cada nodo posee una parte del sistema de archivos lo cual incrementa la velocidad en los accesos a la información debido a la presencia de más de un dispositivo físico para el manejo de los datos. Sin embargo, esta configuración esta en fase experimental y por esta razón no es recomendada.

b**. Memoria**

La selección de la cantidad de memoria depende de dos factores primordialmente, los recursos económicos con que se cuentan y los requerimientos de memoria de las aplicaciones que se ejecutarán en el clúster. La razón principal para contar con una capacidad de memoria razonable es evitar que las aplicaciones necesiten de áreas de *swap* para continuar con su ejecución normal. Intercambiar localidades de memoria hacia el área de *swap* reduce considerablemente el rendimiento de los programas. Se debe tomar en cuenta que en algunas configuraciones del clúster (Disk-less) no es posible contar con particiones destinadas para memoria virtual debido a la ausencia de discos locales, lo cual impone una razón de peso para instalar memoria RAM suficiente.

c. **Procesador**

Los clústers generalmente son construidos con procesadores Alpha o Intel x86. La utilización de otro tipo de procesador es permitido, sin embargo, no se consideran de uso común, ya que se elimina una de las principales características de Beowulf (uso de componentes comunes), la cual permite reemplazar de forma fácil y con bajos costos cualquier componente del sistema.

d. **Simetric MultiProcessor (SMP)**

Las máquinas con más de un procesador son utilizadas comúnmente en clústers Beowulf debido a la gran capacidad de prestaciones que proporcionan. Una de las principales ventajas en la utilización de SMP, además del incremento de poder, es la reducción de la cantidad de tarjetas de red y por supuesto el tráfico en la red.

e. **Red**

La topología de red recomendada es un Bus, debido a la facilidad para proporcionar escalabilidad a la hora de agregar nuevos nodos al clúster. Protocolos como Ethernet, Fast Ethernet, 10/100 Mbps Switched Ethernet, etc, son tecnologías apropiadas para ser utilizadas en Beowulf.

1.4.5 Algunos Conceptos relacionados con Clústers de Pcs:

• **Paralelismo**: El paralelismo permite dividir una tarea en partes que pueden ser ejecutadas independientemente, con lo cual se logra obtener resultados en forma más expedita. El paralelismo se puede implementar a nivel de hardware y a nivel del software.

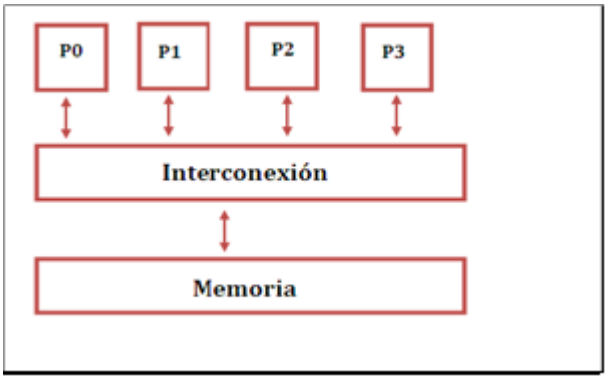
• **Optimización**: Optimizar un código consiste en escribirlo o generarlo (cuando la optimización proviene de un compilador) de forma tal de tome en cuenta las características de la máquina (la arquitectura) y que el número de instrucciones y de bifurcaciones sea el menor posible. Tanto la paralelización como la optimización buscan acortar el tiempo de obtención de la solución de un programa, sin embargo, usan procedimientos distintos.

• **Memoria compartida y memoria distribuida**: Se pueden identificar modelos principales dentro del procesamiento paralelo, donde la diferencia entre uno y otro reside en la forma en que cada procesador accede a su memoria.

- ✓ **Máquinas de Memoria Compartida**

En el primero, llamado modelo de memoria compartida, o conocido también como sistemas multiprocesador, los procesadores que conforman el sistema acceden a una única memoria común a todos, como si fuese un espacio de dirección global. Y la comunicación entre las tareas se hace gracias a operaciones de escritura/lectura sobre esta memoria. Los cambios efectuados en una locación de memoria por un procesador son visibles para el resto de los

procesadores. El espacio de dirección global proporciona una programación de uso fácil desde el punto de vista de la memoria. En el modelo de memoria compartida los procesadores se comunican con la memoria a través de un bus o sistemas de switches (llaves de alta velocidad). Estos le permiten lograr un mayor desempeño en comparación con los sistemas de memoria distribuida. Otra ventaja que presenta este modelo es su uso más eficiente de la memoria ya que hay necesidad de replicación de datos. No obstante, este tipo de arquitectura presenta dos importantes dificultades: el alto costo actual de este tipo de hardware y la escasa portabilidad para migrar un programa codificado en un sistema multicomputador hacia otra plataforma de memoria compartida.

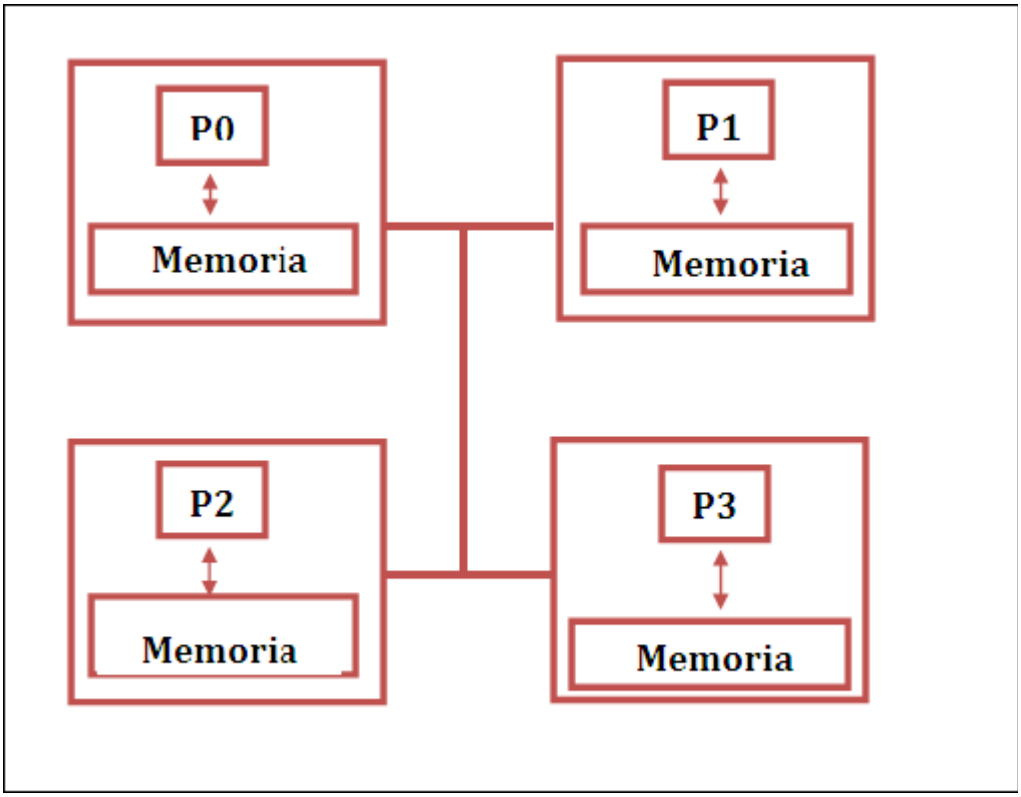


Modelo Máquinas de Memoria Compartida

- ✓ **Máquinas de Memoria Distribuida**

Por el contrario en el modelo de memoria distribuida, conocida también como sistemas multicomputador o clúster de computadoras, cada procesador posee su propia memoria local, lo que significa que no existe concepto alguno de un espacio de dirección global entre todos los procesadores. Como cada procesador tiene su propia memoria local, los cambios que le hace a su memoria local no tienen ningún efecto en la memoria de otros procesadores. Por tal motivo sin un procesador desea acceder a la memoria local de otro procesador, se requiere de un intercambio de mensajes entre ambos mediante una red que los comunica. Los sistemas distribuidos poseen varias ventajas, en primer lugar permiten desarrollar software más portables debido a los estándares existentes en los protocolos de comunicación, además de que cada procesador puede acceder a su propia memoria rápidamente y sin interferencia. Por otra parte los clúster poseen bajo costo y buena escalabilidad dado que usualmente están integrados por computadoras interpersonales conectadas por una red Ethernet. Por estas razones la tendencia actual en procesamiento paralelo apunta hacia el empleo de este tipo de multiprocesador. Este sistema posee también algunas desventajas como son: que el

programador es responsable de mucho de los detalles que asociados con la comunicación de los datos entre los procesadores. Es más difícil distribuir la estructura de datos existente, a la hora de compartir toda la memoria del sistema.



Modelo Máquinas de Memoria Distribuida

Cuando hablamos de paralelismo, y por lo tanto asumimos que tenemos disponibles múltiples procesadores, existen dos paradigmas de programación fundamentales que están basados en la visión que los procesos (o tareas) tienen de la memoria. Cuando la memoria es vista por todos los procesos como un solo bloque y cualquier proceso tiene acceso a cualquier región de la memoria, hablamos de memoria compartida. En este caso la comunicación entre los procesos se hace compartiendo datos que están en la memoria. Cuando los procesadores tienen asociados memorias privadas no accesibles a otros procesadores, se dice que la memoria es distribuida. La comunicación entre los procesos es a través de mensajes. Las dos bibliotecas de pase de mensaje más usadas son: MPI (Message Passing Interface) y PVM (Parallel Virtual Machine). Los clústers de PCs caen en la categoría de máquinas de memoria distribuida.

• **Dependencia**: La dependencia se da cuando cierta parte del código no puede proceder si no se tienen los resultados de otros fragmentos del código.

• **Sincronización**: Sincronizar consiste en poner a la par dos o más procesos o subtareas. Cuando un proceso va a correr un código dependiente de resultados calculados por otro proceso, ejecuta una instrucción de sincronización la cual lo hace esperar por los resultados necesarios. Una vez recibidos los resultados, resume su labor.

• **Latencia**: Se refiere al tiempo que transcurre entre el momento en que se da una solicitud de transferencia de datos y el momento en que la transferencia efectivamente comienza. Esto se debe principalmente a la inicialización de dispositivos y la preparación de los datos. Se da principalmente cuando hay acceso a la memoria, al los discos y a la red.

• **Granularidad**: La granularidad esta relacionada con la cantidad de trabajo que se puede efectuar antes de ser necesario cierto nivel de sincronización debido a las dependencias entre las subtareas. Si el monto del trabajo es considerable, decimos que la granularidad es gruesa. Si es poco hablamos de granularidad fina. En los clústers la comunicación es a través de pase de mensajes. Por lo tanto hay consumo de tiempo para ensamblar el mensaje, enviarlo por la red, recibirlo del otro lado y finalmente desensamblarlo. Este tiempo es mucho mayor que el tiempo que requiere un acceso a memoria en las máquinas de memoria compartida. Los clústers son adecuados cuando el problema presenta una granularidad gruesa. Si la granularidad es fina, el tiempo de comunicación o sincronización predomina haciendo que la solución paralela del problema sea menos eficiente que su solución secuencial. Uno de los problemas principales de los computadores de memoria compartida, es que el acceso a memoria se satura rápidamente a medida que se incrementa el número de procesadores en la máquina, mientras que en las máquinas de memoria distribuida el número de procesadores puede crecer significativamente.

• **Red**: La red esta formada por elementos (interfaces, switches, cables, etc.) que permiten interconectar distintos procesadores, bien sea dentro de una misma caja(como algunos supercomputadores) o en cajas diferentes (como los PCs), para que estos puedan comunicarse entre si. Cuando hablamos de redes hay también un conjunto de conceptos que se deben manejar.

• **Medio de transmisión**: al nivel más bajo la comunicación entre computadores requiere convertir los datos en alguna forma de energía y enviarla a través del medio de transmisión. Por ejemplo, corriente eléctrica para enviar a través de cable (par trenzado), luz a través de fibra óptica y ondas de radio a través del aire.

• **Protocolo**: es un acuerdo o conjunto de reglas que definen el formato, el significado y la manera en que los mensajes son enviados entre computadores.

Entre la información que contiene un mensaje, además de los datos mismos que se quieren comunicar, se encuentra la dirección del destinatario y del remitente.

• **Interface**: es el hardware (una tarjeta que se agrega al PCs) que toma la información empaquetada y la convierte a un formato que puede ser transmitido por el medio físico o medio de transmisión.

• **Latencia**: tiempo que transcurre entre el momento en que un procesador solicita una transferencia de datos y el momento en que efectivamente comienza la transmisión.

• **Ancho de banda**: se define como la tasa a la cual se puede enviar datos entre procesadores y viene expresada en bits por segundo.

• **Switch**: dispositivo electrónico, con varios canales de entrada y de salida, que hace uso de la dirección del destinatario de un mensaje para decidir por que canal enviarlo.

• **Tecnologías de redes**: definen como se usa el medio de transmisión y el tipo de medio. Entre las más conocidas están: ATM (Asynchronous Transfer Mode), Ethernet, Token Ring, FDDI (Fiber Distributed Data Interconnect) y Frame.

• **Relay**. Ethernet es una de las más populares y usa un cable o bus, compartido por todas las máquinas, como medio de transmisión. Cuando un computador quiere enviar un mensaje, chequea si el medio esta desocupado y de estarlo procede con el envío. Si el medio esta siendo usado, espera un tiempo aleatorio antes de volver a tratar. Ethernet viene en tres versiones que difieren en el ancho de banda:

Ethernet original con 10 Mb/s (mega bit por segundo), Fast Ethernet con 100 MB/s y Giga

Ethernet con 1000 Mb/s. Las tres versiones exhiben una latencia menor a los 90 microsegundos (debido al protocolo la latencia no es fija). Existen otras tecnologías con latencias menores pero más costosas.

En los Clústers de PCs tenemos un ambiente de memoria distribuida. Cada PC es propietario de su memoria local. Los PC se comunican mediante el envío de mensajes y por lo general se usan implementaciones de MPI o PVM como bibliotecas de comunicación.
Se debe tener una granularidad gruesa para poder obtener beneficio de la paralelización de un problema. La sincronización de procesos se logra bloqueando un proceso que ejecuta la primitiva receive hasta recibir un mensaje de uno o más procesos con los cuales desea sincronizarse. El medio de trasmisión más usado para conectar clústers es el cable (par trenzado) usando la tecnología Ethernet (Fast o Giga). Como con Ethernet todos los PCs comparten el medio, pares de PCs distintos no pueden comunicarse simultáneamente. Con el fin de permitir las comunicaciones simultáneas y por lo tanto reducir la latencia, los switches ofrecen una alternativa a expensas de una mayor inversión.

1.4.6 Softwares de clúster:

Para facilitar estas tareas de instalación existen los denominados toolkits *(*colección de herramientas integradas que permiten automatizar un conjunto de tareas*),* que guían al usuario en el proceso de instalación. De entre las opciones disponibles, se pueden mencionar algunos que han tenido gran aceptación: OSCAR, NPACI Rocks y openMosix.

OpenMosix

[21]Es un sistema de clúster para Linux que permite a varias máquinas actuar como un único sistema multiprocesador (denominado en inglés SSI). Esto permite que no tengamos que reprogramar nuestras aplicaciones para aprovechen el clúster. Los procesos no saben en qué nodo del clúster se ejecutan, y es el propio OpenMosix el responsable de "engañarlos", y redirigir las llamadas al sistema al nodo del clúster en el que se lanzó el proceso. OpenMosix implementa un algoritmo balanceador que permite repartir de forma óptima la carga, si está el clúster bien calibrado.
Se compone de un parche al kernel, responsable de las migraciones transparentes de procesos, y unas herramientas de área de usuario, necesarias para calibrar y administrar el clúster.

OpenMosix puede migrar cualquier proceso mientras que no haga uso de los segmentos de memoria compartida. Según la calibración, migrará procesos más ligeros, o más pesados.
Actualmente el desarrollo de OpenMosix está parado. Sigue funcionando bien para los kernels2.4; y para el kernel2.6 aún no funcionaba completamente.
Sin embargo, por su estabilidad y robustez aún hay gente que lo emplea, y sigue instalándolo. El único problema real es con las máquinas que tienen hardware no soportado por el kernel2.4.

Rocks

[22]Rocks Clúster (originalmente llamado NPACI Rocks) es una distribución de Linux para clústers de computadores de alto rendimiento. Fue iniciada por la NPACI (National Partnership for Advanced Computational Infrastructure) y la SDSC (San Diego Supercomputer Center) en el 2000. Rocks se basó inicialmente en la distribución Red Hat Linux, sin embargo las versiones más modernas de Rocks están basadas en CentOS (Community Enterprise Operating System), con un instalador anaconda modificado, que simplifica la instalación 'en masa' en muchos computadores. Rocks incluye muchas herramientas (tales como MPI) que no forman parte de CentOS pero son los componentes integrales que hacen un grupo de ordenadores en un clúster.

Rocks realiza una instalación completa del sistema operativo sobre un nodo. Rocks incorpora una versión de la distribución de Red Hat con software adicional específico para clústers. Adicionalmente, Rocks configura correctamente varios servicios. Al instalar Rocks, éste instalará Linux, por lo que no es posible agregar Rocks sobre un servidor existente o usarlo con alguna distribución de Linux diferente.
Es una de las distribuciones más empleadas en el ámbito de clústers, por su facilidad de instalación e incorporación de nuevos nodos. Otra de sus grandes facilidades es que incorpora gran cantidad de software para el mantenimiento y monitorización del clúster, lo que a su vez podría suponer en algunos casos una limitación.

Oscar

[23]Oscar es una colección de software de código abierto para crear un clúster sobre Linux desarrollada por el Grupo de Clústers Abiertos (OCG – Open Clúster Group).
El objetivo primario del grupo de trabajo OS es conseguir que la instalación, la configuración y la administración de un clúster de tamaño modesto, sea fácil. Cualquier cosa necesaria para un clúster – instalación y configuración, mantenimiento, programación [paralela], sistemas de encolamiento, programación de tareas – está incluida en Oscar.

Oscar se instala sobre un computador con una distribución Linux previamente. Oscar realiza la instalación y/o configuración de los paquetes necesarios para el clúster de forma automática.

1.4.7 Hardware de un clúster:

A la hora de construir un clúster Linux es necesario considerar diversos aspectos de diseño para tomar decisiones que contribuyan al mejor desenvolvimiento de la máquina basándose en los requerimientos iníciales. Es recomendable realizar una revisión constante de las tendencias actuales antes de emprender un nuevo proyecto, y así contar con nueva tecnología que satisfaga nuestras expectativas.

La mayoría de los proveedores de PCs acostumbran a vender máquinas con componentes que no son necesarios dentro de un clúster, por ejemplo, tarjetas de video sofisticadas, tarjetas de audio, etc. Con un poco de información extra, se puede obtener el hardware apropiado por un costo mucho más bajo, simplemente evitando la adquisición de elementos innecesarios.

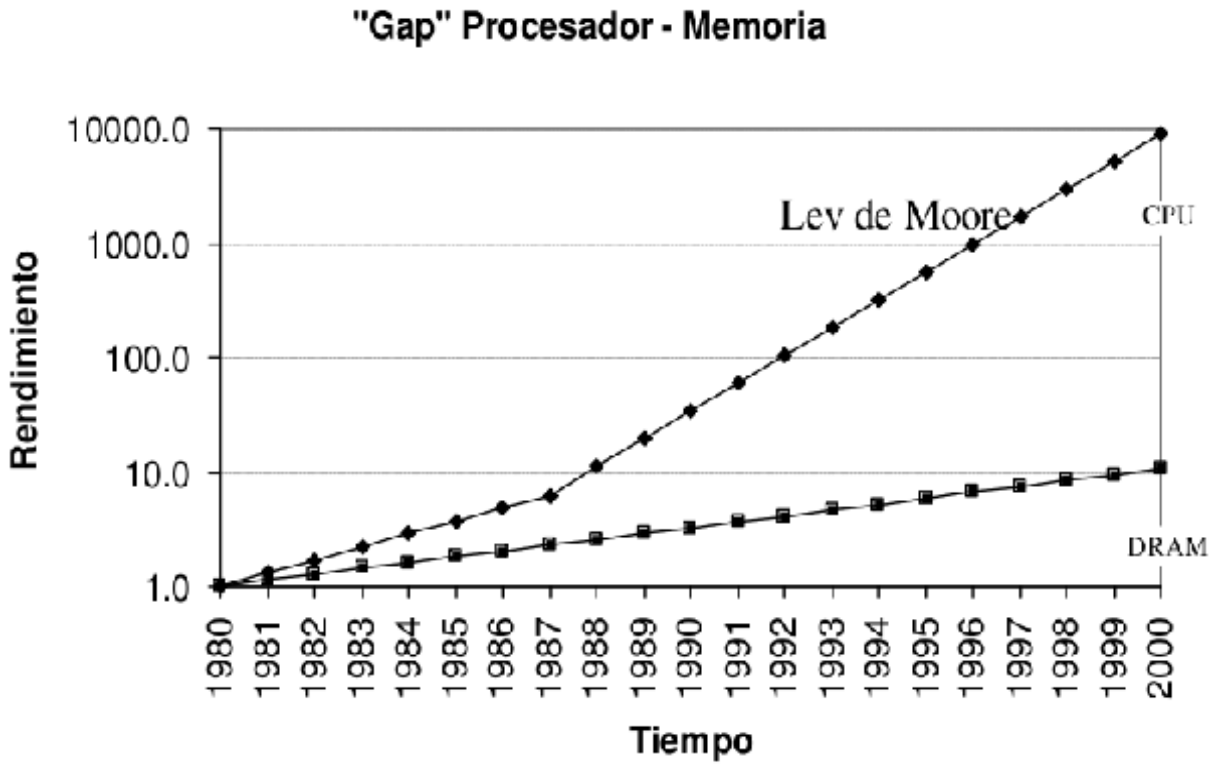
1.4.8 Hardware de los nodos:

Un clúster Linux es una red de nodos, donde cada uno de ellos es un computador personal común. Por esto, los nodos constituyen el elemento principal del clúster, los cuales son responsables de todas las actividades asociadas con la ejecución de los programas de aplicación y de dar soporte al software especializado presente en los clústers. Según la función que cumplen los nodos pueden ser ubicados dentro de las siguientes categorías:

- Ejecución de instrucciones.
- Almacenamiento rápido de información temporal.
- Alta capacidad de almacenamiento de información persistente.
- Comunicación con ambientes externos incluyendo otros nodos.

Uno de los principales inconvenientes impuesto por la tecnología es el llamado processor - memory gap. Esto es, la diferencia de velocidades entre el procesador y la memoria. El rendimiento de los procesadores es duplicado cada 18 meses (ley de Moore), alrededor de un 60 % por año, mientras que la memoria solo mejora un 9 % por año. Por esta razón, no es posible almacenar datos en la memoria tan rápido como el procesador puede manejar esos datos, y por eso a menudo el procesador debe esperar por la memoria. Esta diferencia de velocidades se incrementa un 50 % por año. La siguiente figura muestra la evolución de estos

dos componentes.



"Gap" Procesador - Memoria

Actualmente se puede elegir cada componente de los nodos dentro de una gran variedad, por ejemplo, hay más de una familia de procesadores y dentro de cada familia existe más de una alternativa. Seleccionar la configuración apropiada para los nodos de un clúster puede parecer algo difícil de realizar debido a la gran diversidad de componentes presentes en el mercado. Sin embargo, existe un conjunto de parámetros críticos que caracterizan primordialmente a un nodo.

- Frecuencia de reloj del procesador: Esta es la principal señal dentro del procesador que determina la tasa de procesamiento de instrucciones.
- Rendimiento punto flotante pico: Es la combinación de la frecuencia de reloj y el número de operaciones punto flotante que pueden ser procesadas.
- Tamaño de la memoria cache: Es la capacidad de almacenamiento del buffer de memoria de alta velocidad entre la memoria principal y el procesador.

- Capacidad de la memoria principal: Es la capacidad de almacenamiento del sistema principal de memoria del nodo donde reside el conjunto de datos globales de las aplicaciones.
- Capacidad de disco: Es la capacidad de los dispositivos de almacenamiento secundario.
- Ancho de banda pico de la tarjeta de red: Es el ancho de banda teórico proporcionado por la interfaz de red.

1.4.9 Procesador:

El procesador constituye toda la lógica requerida para la ejecución del conjunto de instrucciones, gestión de la memoria, operaciones enteras y punto flotante, y el manejo de la memoria cache.

Las máquinas con más de un procesador (SMP) son utilizadas comúnmente en clústers debido a la gran capacidad de prestaciones que proporcionan. Sin embargo, la velocidad de los buses de las tarjetas madres no tiene la capacidad necesaria para dar apoyo a arquitecturas SMP, lo que representa un cuello de botella entre los diferentes medios de almacenamiento y el procesador.

1.4.10 Memoria:

La memoria de un computador personal es el sistema de almacenamiento más cercano al procesador. Las características deseables de la memoria son: rapidez, bajo costo y gran capacidad. Desafortunadamente, los componentes disponibles hasta ahora, solo poseen una combinación de cualquiera dos de estas características. Los sistemas de memoria modernos utilizan una jerarquía de componentes implementados con diferentes tecnologías que juntos, y en condiciones favorables, logran obtener las tres características. A pesar de todo esto, la capacidad de almacenamiento de memoria se ha incrementado considerablemente, cuadriplicándose cada tres años aproximadamente, mientras que su costo ha sufrido un constante decremento.

Las memorias constituidas por semiconductores dieron un cambio significativo a la predominancia de los medios de almacenamiento magnéticos de los años 70.

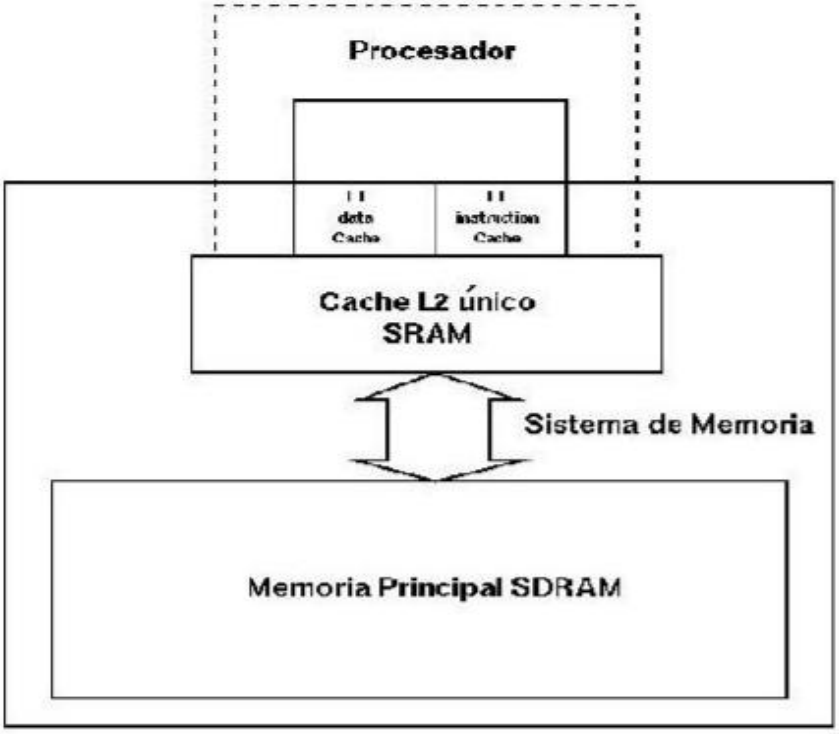
Actualmente hay dos tipos de memoria de semiconductores: memoria estática de acceso aleatorio (SRAM10), la cual se caracteriza por ser muy rápida pero de capacidad moderada, y la memoria dinámica de acceso aleatorio (DRAM) cuya capacidad de almacenamiento es

considerable pero opera de forma más lenta.

La memoria estática es implementada con celdas de bits fabricadas con circuitos flip-flop de transistores múltiples. Estos circuitos activos pueden cambiar su estado y ser accedidos rápidamente, sin embargo, su consumo de energía es significativo. Este tipo de memoria es empleada en aquellas partes del sistema donde se requieren medios de almacenamiento rápidos tales como memorias cache L1 y L2.

La memoria dinámica de celdas de bits es fabricada con capacitores y transistores de puentes simples. Estos capacitores almacenan una carga en forma pasiva y las operaciones de acceso a cada celda consume esta carga, además, el aislamiento de los capacitores no es perfecta y la carga se pierde con el tiempo aunque no sea accedida. Por esto, debe ser restablecida con cierta frecuencia la carga de los condensadores, lo cual implica tiempos de accesos mayores. Dentro de esta categoría podemos encontrar algunas variaciones como memoria dinámica tipo Extended Data Output (EDO DRAM) que proporciona un esquema de buffer interno modificado que mantiene los datos en la salida más tiempo que las DRAM convencionales. El otro tipo de memoria dinámica es la memoria dinámica síncrona, que implementa un modo de cauce por etapas que permite iniciar un segundo ciclo de acceso antes de ser completado el ciclo anterior.

Las diferentes necesidades de velocidad de almacenamiento y las diferentes implementaciones dan origen a una jerarquía de memoria, donde se muestra en la siguiente figura:



Jerarquía de Memoria

1.4.11 Disco:

Toda la información presente en los sistemas de memoria se pierde una vez que el computador se apaga o se reinicia. Es por eso que son necesarios los sistemas de almacenamiento secundarios tales como CD roms, floppies, discos duros, etc. De éstos el único medio realmente necesario es el disco duro ya que el resto de los dispositivos por lo general no se usan en ambientes de cálculo intensivo.

Los discos duros mantienen copia del sistema operativo, programas y datos, así, se cuenta con un medio de almacenamiento para mantener grandes cantidades de información. Existen dos interfaces principales utilizadas para manejar discos duros: IDE y SCSI. Originalmente, las interfaces IDE, de menor rendimiento, tenían predominancia en el mercado de PCs debido al bajo costo en relación a los discos SCSI. Sin embargo, el actual abaratamiento de los costos de las interfaces SCSI ha permitido incorporarlas en las nuevas tarjetas principales, aunque todavía los precios no son comparables con los discos IDE. Por lo general, se recomienda utilizar discos SCSI en situaciones de altas tasas de operaciones de lectura y escritura, por ejemplo, directorios hogares, y utilizar discos IDE en situaciones de bajo número de accesos

como por ejemplo espacios dedicados al sistema.

1.5 Protocolo de Comunicación:

[10] Los protocolos son como reglas de comunicación que permiten el flujo de información entre computadoras distintas que manejan lenguajes distintos, por ejemplo, dos computadores conectados en la misma red pero con protocolos diferentes no podrían comunicarse jamás, para ello, es necesario que ambas "hablen" el mismo idioma, por tal sentido, el protocolo TCP/IP fue creado para las comunicaciones en Internet, para que cualquier computador se conecte a Internet, es necesario que tenga instalado este protocolo de comunicación

1.5.1 Protocolo de control de transmisión/Protocolo de Internet (TCP/IP):

El protocolo TCP/IP (Transmition Control Protocol/Internet Protocol) hace posible enlazar cualquier tipo de computadoras, sin importar el sistema operativo que usen o el fabricante. Este protocolo fue desarrollado originalmente por el ARPA (Advanced Research Projects Agency) del Departamento de Defensa de los Estados Unidos. Actualmente, es posible tener una red mundial llamada Internet usando este protocolo. Este sistema de IP permite a las redes enviar correo electrónico (e-mail), transferencia de archivos (FTP) y tener una interacción con otras computadoras (TELNET) no importando donde estén localizadas, tan solo que sean accesibles a través de Internet.

Arquitectura de Interconexión de Redes en TCP/IP **Características**
• Protocolos de no conexión en el nivel de red. • Conmutación de paquetes entre nodos. • Protocolos de transporte con funciones de seguridad. • Conjunto común de programas de aplicación.

Para entender el funcionamiento de los protocolos TCP/IP debe tenerse en cuenta la arquitectura que ellos proponen para comunicar redes. Tal arquitectura ve como iguales a todas las redes a conectarse, sin tomar en cuenta el tamaño de ellas, ya sean locales o de cobertura amplia. Define que todas las redes que intercambiarán información deben estar conectadas a una misma computadora o equipo de procesamiento (dotados con dispositivos de

comunicación); a tales computadoras se les denominan compuertas, pudiendo recibir otros nombres como enrutadores o puentes.

1.5.2 Protocolo SSH:

[11]SSH™ permite a los usuarios registrarse en sistemas de host remotamente. A diferencia de FTP o Telnet, SSH encripta la sesión de registro imposibilitando que alguien pueda obtener contraseñas no encriptadas.

SSH está diseñado para reemplazar los métodos más viejos y menos seguros para registrarse remotamente en otro sistema a través de la shell de comando, tales como **telnet** o **rsh**. Un programa relacionado, el **scp**, reemplaza otros programas diseñados para copiar archivos entre hosts como **rcp**. Ya que estas aplicaciones antiguas no encriptan contraseñas entre el cliente y el servidor, evite usarlas mientras le sea posible. El uso de métodos seguros para registrarse remotamente a otros sistemas hará disminuir los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto.

1.5.2.1 Características de SSH:

SSH (*S*ecure *SH*ell) es un protocolo para crear conexiones seguras entre dos sistemas usando una arquitectura cliente/servidor.

El protocolo SSH proporciona los siguientes tipos de protección:

- Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor al que se conectó anteriormente.
- El cliente transmite su información de autenticación al servidor usando una encriptación robusta de 128 bits.
- Todos los datos enviados y recibidos durante la conexión se transfieren por medio de encriptación de 128 bits, lo cual los hacen extremamente difícil de descifrar y leer.

Ya que el protocolo SSH encripta todo lo que envía y recibe, se puede usar para asegurar protocolos inseguros. El servidor SSH puede convertirse en un conducto para convertir en seguros los protocolos inseguros mediante el uso de una técnica llamada *reenvío por puerto*, como por ejemplo POP, incrementando la seguridad del sistema en general y de los datos.

Red Hat Linux contiene el paquete general de OpenSSH (openssh), el servidor de OpenSSH (openssh-server) y los paquetes de clientes (openssh-clients). Los paquetes OpenSSH

requieren el paquete OpenSSL (openssl). OpenSSL instala varias bibliotecas criptográficas importantes que ayudan a OpenSSH a proporcionar comunicaciones encriptadas.
Una gran cantidad de programas de cliente y servidor puede usar el protocolo SSH. Muchas aplicaciones SSH cliente están disponibles para casi todos los principales sistemas operativos en uso hoy día.

1.5.2.2 ¿Por qué usar SSH?

Los usuario nefarios tienen a su disposición una variedad de herramientas interceptar y dirigir el tráfico de la red para ganar acceso al sistema. En términos generales, estas amenazas se pueden catalogar del siguiente modo:

- Intercepción de la comunicación entre dos sistemas — En este escenario, existe un tercero en algún lugar de la red entre entidades en comunicación que hace una copia de la información que pasa entre ellas. La parte interceptora puede interceptar y conservar la información, o puede modificar la información y luego enviarla al recipiente al cual estaba destinada.

  Este ataque se puede montar a través del uso de un paquete sniffer — una utilidad de red muy común.
- Personificación de un determinado host — con esta estrategia, un sistema interceptor finge ser el recipiente a quien está destinado un mensaje. Si funciona la estrategia, el cliente no se da cuenta del engaño y continúa la comunicación con el interceptor como si su mensaje hubiese llegado a su destino satisfactoriamente.

Esto se produce con técnicas como el *envenenamiento del DNS o *spoofing de IP.
Ambas técnicas causan que se intercepte información, posiblemente con propósitos hostiles. El resultado puede ser catastrófico.
Si se utiliza SSH para inicios de sesión de shell remota y para copiar archivos, estas amenazas a la seguridad se pueden disminuir notablemente. Esto es porque el cliente SSH y el servidor usan firmas digitales para verificar su identidad. Adicionalmente, toda la comunicación entre los sistemas cliente y servidor es encriptada. No servirán de nada los intentos de falsificar la identidad de cualquiera de los dos lados de la comunicación ya que cada paquete está cifrado por medio de una clave conocida sólo por el sistema local y el remoto.

1.6 Bibliotecas de envío de mensajes

1.6.1 Características:

Parte esencial de un clúster de tipo Beowulf. Contienen instrucciones para intercomunicar procesos que se encuentran en diferentes nodos, algunas de las más comunes son: PVM, NUMA y MPI.

Teniendo en cuenta las características posteriormente descrita sobre estas bibliotecas y el estudio de las mismas, se decide utilizar la PVM, por ser la más apropiada para la solución del problema.

1.6.1.1 PVM:

[6] PVM (Paralel Virtual Machine) es una herramienta diseñada para solucionarnos una gran cantidad de problemas asociados con la programación paralela. Sobre todo, el monetario. Para ello, nos va a crear una nueva abstracción, que es la máquina paralela virtual, empleando los recursos computacionales libres de todas las máquinas de la red que pongamos a disposición de la biblioteca. Es decir, disponemos de todas las ventajas económicas asociadas a la programación distribuida, ya que empleamos los recursos hardware de dicho paradigma; pero programando el conjunto de máquinas como si se tratara de una sola máquina paralela, que es mucho más cómodo.

La **PVM** es el estándar de facto del mundo científico. De hecho, en el área de la Física Computacional, la PVM es una biblioteca ampliamente usada.

La máquina paralela virtual es una máquina que no existe, pero un API apropiado nos permite programar como si existiese. El modelo abstracto que nos permite usar el API de la PVM consiste en una máquina multiprocesador completamente escalable (es decir, que podemos aumentar y disminuir el número de procesadores *en caliente*). Para ello, nos va a ocultar la red que estemos empleando para conectar nuestras máquinas, así como las máquinas de la red y sus características específicas. Este planteamiento tiene numerosas ventajas respecto a emplear un supercomputador, de las cuales, las más destacadas son:

- **Precio.** Así como es mucho más barato un computador paralelo que el computador tradicional equivalente, un conjunto de ordenadores de mediana o baja potencia es muchísimo más barato que el computador paralelo de potencia equivalente. Al igual que

ocurrirá con el caso del computador paralelo, van a existir factores (fundamentalmente, la lentitud de la red frente a la velocidad del bus del computador paralelo) que van a hacer de que sean necesarios más ordenadores de pequeña potencia que los teóricos para igualar el rendimiento. Sin embargo, aun teniendo esto en cuenta, la solución es mucho más barata. Además, al no ser la PVM una solución que necesite de máquinas dedicadas (es decir, el demonio de PVM corre como un proceso más), podemos emplear en el proceso los tiempos muertos de los procesadores de todas las máquinas de nuestra red a las que tengamos acceso. Por ello, si ya tenemos una red Unix montada, el costo de tener un supercomputador paralelo va a ser cero ya disponemos de las máquinas, no tendremos que comprar nada nuevo, y además la biblioteca PVM es software libre, por lo que no hay que pagar para usarla.

- **Disponibilidad.** Todo centro de cálculo tiene un mínimo de una docena de máquinas arrumbadas en una esquina, y que nadie sabe qué hacer exactamente ya con ellas. Con esa docena que hace seis años que ya no corren ni la última versión del Word para Windows, podemos instalar Linux, la PVM y añadirlo al supercomputador paralelo virtual que conforma las máquinas que ya tendríamos en red.
- **Tolerancia a fallos.** Si por cualquier razón falla uno de los ordenadores que conforman nuestra PVM y el programa que la usa está razonablemente bien hecho. Nuestra aplicación puede seguir funcionando sin problemas. En un caso como el nuestro, en el que la aplicación va a estar corriendo durante meses, es crítico que la aplicación sea tolerante a fallos. Siempre hay alguna razón por la que alguna máquina puede fallar, y la aplicación debe continuar haciendo los cálculos con aquel hardware que continúe disponible.
- **Heterogeneidad.** Podemos crear una máquina paralela virtual a partir de ordenadores de cualquier tipo. La PVM nos va a abstraer la topología de la red, la tecnología de la red, la cantidad de memoria de cada máquina, el tipo de procesador y la forma de almacenar los datos. Este último punto es de extrema importancia, ya que el principal problema que tendríamos en los *sockets* era la programación de rutinas de conversión de formato de datos entre todos los ordenadores de la red, puesto que la codificación, tanto de enteros como de flotantes, puede ser distinta. Por último, nos permite incluir en nuestra PVM hasta máquinas paralelas. Una máquina paralela en una PVM se puede comportar tanto como una sola máquina secuencial (caso, por ejemplo, del soporte

SMP de Linux) o, como ocurre en muchas máquinas paralelas, presentarse a la PVM como un conjunto de máquinas secuenciales.

El uso de la PVM tiene muchas ventajas, pero también tiene una gran desventaja: nos podemos olvidar del paralelismo fuertemente acoplado. Si disponemos de una red Ethernet, simplemente la red va a dejar de funcionar para todas las aplicaciones (incluida PVM) de la cantidad de colisiones que se van a producir en caso de que intentemos paralelismo fuertemente acoplado. Si disponemos de una red de tecnología más avanzada; es decir, más cara (como ATM) el problema es menor, pero sigue existiendo.

La segunda desventaja es que la abstracción de la máquina virtual, la independencia del hardware y la independencia de la codificación tienen un coste. La PVM no va a ser tan rápida como son los Sockets. Sin embargo, si el grado de acoplamiento se mantiene lo suficientemente bajo, no es observable esta diferencia.

1.6.1.2 NUMA:

[8] Non-Uniform Memory Access o Non-Uniform Memory Architecture (NUMA) es un diseño de memoria utilizado en multiprocesadores donde la memoria se accede en posiciones relativas de otro procesador o memoria compartida entre procesadores. Bajo NUMA, un procesador puede acceder a su propia memoria local de forma más rápida que a la memoria no local (memoria local de otro procesador o memoria compartida entre procesadores).

Limitar el número de accesos a memoria es la clave de un alto rendimiento en un ordenador moderno. Para los procesadores esto significa el incremento de alta velocidad de la memoria caché y el uso de algoritmos más sofisticados para evitar los errores de caché. Aunque el drástico aumento del tamaño de los sistemas operativos y las aplicaciones que se ejecutan en ellos han abrumado las mejoras del procesamiento de la caché. Los sistemas de Multiprocesamiento hacen que el problema sea peor. Ahora el sistema puede bloquear varios procesadores a la vez, porque solo un procesador puede acceder a la memoria a la vez.

NUMA intenta resolver este problema ofreciendo memoria distribuida para cada procesador, evitando así que afecte al rendimiento del sistema cuando varios procesadores intentan acceder a la misma memoria. Para los problemas de las propagación de datos (comunes en servidores y aplicaciones similares), NUMA puede mejorar el rendimiento utilizando una única

memoria compartida por un factor de aproximadamente el número de procesadores (o separando bancos de memoria).

1.6.1.3 MPI:

[9] Message Passing Interface (MPI) es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores. El paso de mensajes es una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua, de manera similar a como se hace con los semáforos, monitores, etc. Su principal característica es que no precisa de memoria compartida, por lo que es muy importante en la programación para sistemas distribuidos.

Con MPI el número de procesos requeridos se asigna antes de la ejecución del programa, y no se crean procesos adicionales mientras la aplicación se ejecuta. A cada proceso se le asigna una variable que se denomina rank, la cual identifica a cada proceso, en el rango de 0 a p-1, donde p es el número total de procesos. El control de la ejecución del programa se realiza mediante la variable rank; la variable rank permite determinar que proceso ejecuta determinada porción de código. En MPI se define un comunicator como una colección de procesos, los cuales pueden enviar mensajes el uno al otro; el comunicator básico se denomina MPI_COMM_WORLD y se define mediante un macro del lenguaje C. MPI_COMM_WORLD agrupa a todos los procesos activos durante la ejecución de una aplicación. Las llamadas de MPI se dividen en cuatro clases:

1. Llamadas utilizadas para inicializar, administrar y finalizar comunicaciones.
2. Llamadas utilizadas para transferir datos entre un par de procesos.
3. Llamadas para transferir datos entre varios procesos.
4. Llamadas utilizadas para crear tipos de datos definidos por el usuario.

La primera clase de llamadas permiten inicializar la biblioteca de paso de mensajes, identificar el número de procesos (size) y el rango de los procesos (rank). La segunda clase de llamadas incluye operaciones de comunicación punto a punto, para diferentes tipos de actividades de envío y recepción. La tercera clase de llamadas son conocidas como operaciones grupales, que proveen operaciones de comunicaciones entre grupos de procesos. La última clase de llamadas provee flexibilidad en la construcción de estructuras de datos complejos. En MPI, un mensaje está conformado por el cuerpo del mensaje, el cual contiene los datos a ser enviados,

y su envoltura, que indica el proceso fuente y el destino. El cuerpo del mensaje en MPI se conforma por tres piezas de información: buffer, tipo de dato y count. El buffer, es la localidad de memoria donde se encuentran los datos de salida o donde se almacenan los datos de entrada. El tipo de dato, indica el tipo de los datos que se envían en el mensaje. En casos simples, éste es un tipo básico o primitivo, por ejemplo, un número entero, y que en aplicaciones más avanzadas puede ser un tipo de dato construido a través de datos primitivos. Los tipos de datos derivados son análogos a las estructuras de C. El count es un número de secuencia que junto al tipo de datos permiten al usuario agrupar ítems de datos de un mismo tipo en un solo mensaje. MPI estandariza los tipos de datos primitivos, evitando que el programador se preocupe de las diferencias que existen entre ellos, cuando se encuentran en distintas plataformas. La envoltura de un mensaje en MPI típicamente contiene la dirección destino, la dirección de la fuente, y cualquier otra información que se necesite para transmitir y entregar el mensaje. La envoltura de un mensaje en MPI, consta de cuatro partes: la fuente, el destino, el comunicator y una etiqueta. La fuente identifica al proceso transmisor. El destino identifica al proceso receptor. El comunicator especifica el grupo de procesos a los cuales pertenecen la fuente y el destino. La etiqueta (tag) permite clasificar el mensaje. El campo etiqueta es un entero definido por el usuario que puede ser utilizado para distinguir los mensajes que recibe un proceso. Por ejemplo, se tienen dos procesos A y B. El proceso A envía dos mensajes al proceso B, ambos mensajes contienen un dato. Uno de los datos es utilizado para realizar un cálculo, mientras el otro es utilizado para imprimirlo en pantalla. El proceso A utiliza diferentes etiquetas para los mensajes. El proceso B utiliza los valores de etiquetas definidos en el proceso A e identifica que operación deberá realizar con el dato de cada mensaje.

1.6.1.4 PVM como sistema de procesamiento seleccionado comparado con MPI:

[7] Comparación

Features	PVM	MPI
Portability	Yes	Yes
Interoperability	Yes	No
Fault Tolerance	Yes	No
Process Control	Yes	MPI-2: Yes
Resource Control	Dynamic	Static

***Portabilidad**:* la portabilidad alcanzada por MPI significa que un programa escrito para una arquitectura puede ser copiado a una segunda arquitectura, compilado y ejecutado sin ningún tipo de modificación. PVM no solo apoya este nivel de portabilidad, sino que amplía la definición de la portabilidad para incluir la interoperabilidad.

***Interoperabilidad**:* en MPI una aplicación puede ejecutarse en su totalidad, en una única arquitectura y es portátil en ese sentido. Pero un programa PVM puede ser heterogéneamente portado para funcionar cooperativamente a través de un conjunto de arquitecturas diferentes al mismo tiempo. Hay un ejemplo, MPI y PVM difieren en su enfoque de la lengua interoperabilidad. En el PVM, un programa C puede enviar un mensaje que es recibido por un programa de Fortran. En contraste, un programa C no es necesario para comunicarse con un programa de Fortran por el estándar MPI, aunque la ejecución sea en la misma arquitectura. En otras palabras, la decisión de MPI fue no forzar los dos idiomas para interoperar.

***Fault Tolerance**:* la tolerancia a fallos es una cuestión crítica para cualquier científico de aplicaciones informáticas a gran escala, especialmente en las simulaciones de larga duración. Sin fallas detectadas y recuperadas, es poco probable que esas solicitudes se completen. En cualquier caso, debe haber algunos planes bien definidos para identificar fallos en el sistema o las solicitudes y entonces responder automáticamente a los mismos, o, al menos proveer una oportuna notificación al usuario en caso de fracaso.PVM ha apoyado un régimen básico de notificación de fallos. Por ejemplo, bajo el control del usuario, las tareas se pueden registrar con PVM para ser "notificadas" cuando el estado de la máquina virtual cambia o cuando una tarea falla.

***Control de Procesos**:* el proceso de control es la capacidad de iniciar y detener las tareas, para saber qué tareas se están ejecutando y donde se están ejecutando. PVM contiene todas estas capacidades. En cambio, MPI-1 no tiene ningún método para iniciar una aplicación paralela.

***Control de recursos**:* En términos de gestión de los recursos, PVM es intrínsecamente dinámico en la naturaleza. Los recursos informáticos pueden ser añadidos o suprimidos a voluntad, ya sea de un sistema de "consola" o incluso de la aplicación del usuario. Otro aspecto de la dinámica de PVM se refiere a la eficiencia. Las aplicaciones de los usuarios pueden

exponer necesidades computacionales potencialmente cambiantes durante su ejecución. Por lo tanto, la infraestructura de traspaso de mensajes debe proporcionar un control flexible sobre la cantidad de potencia de cálculo que se utiliza. MPI carece de tal dinámica y está específicamente diseñado para ser de carácter estático para mejorar el rendimiento. Es evidente que hay un equilibrio en la flexibilidad y la eficiencia de este margen extra de rendimiento.

Topología de Paso de Mensaje: aunque MPI no tiene un concepto de una máquina virtual, proporciona un mayor nivel de abstracción por encima de los recursos informáticos en función de la topología de paso de mensajes. En MPI, un grupo de tareas se pueden organizar en una topología lógica de interconexión. Por lo tanto, la comunicación entre las tareas tiene lugar sin esa topología. En contraste, PVM no admite este tipo de abstracción, el deja al programador organizar manualmente las tareas en grupos con la organización de la comunicación deseada.

1.7 Lenguajes de programación

Lenguaje de programación C:

C es un lenguaje de propósito general, orientado a la implementación de Sistemas Operativos, concretamente Unix, siendo el lenguaje de programación de sistemas por excelencia. El lenguaje C es muy eficiente en el código que produce, es un lenguaje de alto nivel que permite programar con instrucciones de lenguaje de propósito general.

Se trata de un lenguaje que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel.

El lenguaje C tiene una gran cantidad de ventajas sobre otros lenguajes y constituyen precisamente la razón fundamental de que después de casi dos décadas de uso siga siendo uno de los lenguajes más populares, utilizados en empresas, organizaciones y fábricas de software de todo el mundo [36].

Ventajas y Características de C:

- Alta velocidad de ejecución.

- Una nueva sintaxis para declarar funciones. Una declaración de función puede añadir una descripción de los argumentos de la función. Esta información adicional sirve para que los compiladores detecten más fácilmente lo errores causados por argumentos que no coinciden.
- Asignación de estructuras (registros) y enumeraciones.
- Preprocesador más sofisticado.
- Una nueva definición de la biblioteca que acompaña a C. Entre otras funciones se incluyen: acceso al sistema operativo (por ejemplo, lectura / escritura de archivos), entrada y salida con formato, asignación dinámica de memoria, manejo de cadenas de caracteres. Una colección de cabeceras estándar que proporciona acceso uniforme a las declaraciones de funciones y tipos de datos.
- Es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado no llevado al extremo (permitiendo ciertas licencias rupturistas).

Desventajas de C:

- Carece de instrucciones de entrada/salida, de instrucciones para el manejo de cadenas de caracteres, con lo que este trabajo queda para la librería de rutinas, con la consiguiente pérdida de transportabilidad.
- La excesiva libertad en la escritura de los programas puede llevar a errores en la programación que, por ser correctos sintácticamente no se detectan a simple vista.
- Las precedencias de los operadores convierten a veces las expresiones en pequeños rompecabezas.

Lenguaje de programación C++:

C++ es un lenguaje orientado a objetos. Nació para añadirle cualidades y características de las que carecía. Manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

C++ es un lenguaje de propósito general basado en el C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres,

funciones online, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería[36].

Ventajas:

- Mantiene las ventajas de C.
- Mantiene una alta compatibilidad con C, lo que le permite la reutilización del código existente.
- C++ introduce nuevas palabras clave y operadores para manejo de clases.
- Es un lenguaje de propósito general que se adapta a múltiples situaciones.

Principal desventaja:

- Es un lenguaje muy complejo, ya que programa tanto en bajo como en alto nivel, de ahí que sea muy complejo; además realiza un minucioso tratamiento de errores aumentando aun más su complejidad. Presenta características propias que permite la realización de casi o todo en la programación pero a la vez lo hace muy extenso.

Lenguaje de programación Java:

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es una tarea de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria [36].

Ventajas:

- Es robusto.

- Es distribuido
- De arquitectura neutral.
- Es seguro.
- Es Neutral.

Desventajas:

- Dado que la máquina virtual de java es un intérprete y redunda en una falta de rendimiento con relación a aplicaciones equivalentes escritas en código máquina nativo.
- El poder reducir los problemas de acceso a memoria y liberación automática hacen de java un lenguaje poco apropiado para desarrollar aplicaciones de base como Sistemas Operativos.

Para el desarrollo de la aplicación se decidió utilizar el lenguaje de programación C++ debido a la potencialidad que representa, al ser una evolución del C que es uno de los más difundidos actualmente. El C++ permite el manejo de puntero, excepciones y posee una amplia gama de librerías. El manejo de datos es otra de las características que lo hace muy superior ante otros lenguajes. Además de no tener que poseer una máquina virtual y lograr así un mayor rendimiento con aplicaciones semejantes.

Qt

Qt son un conjunto de librerías multi-plataforma para el desarrollo del esqueleto de aplicaciones GUI, escritas en código C++. Esta completamente orientado a objetos. Con Qt se pueden desarrollar ricas aplicaciones gráficas, incluye soporte de nuevas tecnologías como OpenGL, XML, Bases de Datos, programación para redes, internacionalización, etc. Qt dispone de una amplia gama de herramientas que facilitan la creación de formas, botones y ventanas de diálogo con el uso del ratón. Las aplicaciones creadas con Qt son muy elegantes, se ven y se operan mejor que las aplicaciones nativas. Qt dispone de tres grandes ventajas ante las librerías de ventanas rivales:

- Qt es completamente gratuito para aplicaciones de código abierto.
- Las herramientas, librerías y clases están disponibles para casi todas las plataformas Unix y sus derivados (como Linux, MacOS X y Solaris) como

también para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código y la aplicación se verá y actuará mejor que una aplicación nativa.

Qt tiene una extensa librería con clases y herramientas para la creación de ricas aplicaciones. Estas librerías y clases están bien documentadas, son muy fáciles de usar y tienen una gran herencia de programación orientada a objetos.

Qt Designer: Es una herramienta muy potente que permite diseñar de una forma muy sencilla y rápida ventanas de diálogo con las librerías Qt. Esta herramienta es una aplicación mediante la cual se puede realizar el diseño de aplicaciones GUI de forma gráfica y muy intuitiva.

Qt Creator es un IDE creado por Trolltech, nuevo, ligero y con una plataforma de ambiente de desarrollo integrado (IDE) y diseñada para el desarrollo de aplicaciones con las bibliotecas Qt más rápido y fácil. Los sistemas operativos que soporta son [37]:

- GNU/Linux 2.6.x, para versiones de 32 y 64 bits con Qt 4.x instalado. Además hay una versión para Linux con gcc 3.3.
- Mac OS X 10.4 o superior, requiriendo Qt 4.x
- Windows XP y Vista, requiriendo el compilador MinGW y Qt 4.4.3 para MinGW

Para el desarrollo de la aplicación hemos seleccionado las librerías Qt por la gran potencialidad que representa con el lenguaje c++ y poseer una interfaz gráfica muy eficiente. En el año 2008 Nokia adquiere las acciones sobre Trolltech, cambiándose el nombre de Trolltech por Qt Software. Se espera que en el año 2009 se lance una versión definitiva del Qt Creator. Incluye un paquete de ayuda muy completo, además de incluir un qtdemo que ofrece programas con códigos fuentes que sirven de mucha ayuda.

1.8 Hilos Posix:

[2]Los hilos son similares a procesos. A los hilos, como a los procesos, se les asignan porciones de tiempo por el kernel. En sistemas con un solo procesador el núcleo divide el tiempo asignado a cada hilo para simular la ejecución simultánea de hilos de forma muy similar a como lo divide para los procesos. En sistemas con más de un procesador, los hilos pueden ejecutarse simultáneamente, del mismo modo que dos o más procesos pueden ejecutarse simultáneamente también.

Así que, ¿por qué son los multi-hilos preferibles a múltiples procesos independientes cooperativos? Bien, los hilos comparten la misma ubicación en memoria. Hilos independientes pueden acceder a las mismas variables en memoria. Así pues todos los hilos del programa pueden leer o escribir a los enteros (integers) declarados globalmente. Si alguna vez se ha programado algún código no trivial que use fork (), se reconocerá la importancia de esta herramienta. ¿Por qué? Mientras que fork () permite crear múltiples procesos, también crea el siguiente problema de comunicación: cómo conseguir que múltiples procesos, cada uno con su propio espacio en memoria, se comuniquen. No hay una respuesta simple a este problema. Mientras que hay muchos tipos diferentes de IPC local (comunicación entre procesos), todos ellos sufren de las dos mismas grandes desventajas: Imponen una cierta sobrecarga al núcleo, disminuyendo el rendimiento. En casi todas las situaciones, IPC no es una extensión "natural" del código. Muy a menudo, lo que hace es incrementar la complejidad del código.

Doble decepción: la sobrecarga y la complejidad no son buenas. Si alguna vez se han tenido que hacer modificaciones masivas a alguno de nuestros programas para que soportase IPC, se apreciará realmente la sencilla propuesta de compartir la memoria que los hilos proporcionan. Los hilos POSIX no necesitan hacer complicadas y costosas conferencias, porque resulta que todos nuestros hilos viven en la misma casa. Con una pequeña sincronización, todos los hilos podrán leer y modificar las estructuras de datos de nuestros programas. No tenemos que bombardear los datos a través de un descriptor de fichero o compactarlos en un pequeño espacio de memoria compartida. Únicamente por esta razón debe considerarse el modelo único proceso/multi-hilos en lugar del modelo multi-procesos/mono-hilo.

1.8.1 Ágiles:

Los hilos también son extremadamente ágiles. Comparados con un fork() estándar, los hilos conllevan mucha menos sobrecarga. El núcleo no necesita crear una nueva copia independiente del espacio de memoria del proceso, de los descriptores del proceso, etc. Lo cual ahorra un considerable tiempo de la CPU, haciendo la creación de un nuevo hilo de diez a cien veces más rápida que la creación de un nuevo proceso. A causa de ello, pueden usarse una gran cantidad de hilos sin preocuparse demasiado acerca de la CPU y de la memoria requerida. No tendrán el mismo impacto en la CPU que cuando se realizan con fork(). Lo cual significa que pueden crearse hilos en el programa cada vez que tenga sentido hacerlo.

Por supuesto, al igual que los procesos, los hilos tomarán ventaja con múltiples CPUs. Esto es realmente una gran ventaja si el programa ha sido diseñado para trabajar con una máquina con

varias CPUs (si el programa es de código abierto, seguramente acabará siendo ejecutado en varias de ellas). El rendimiento de ciertos programas en hilos (los que hacen un uso intensivo de la CPU en concreto) escalará linealmente con respecto al número de procesadores por lo menos. Si se está escribiendo un programa que hace un uso intensivo de la CPU, definitivamente se querrán encontrar formas para hacer uso de múltiples hilos en el código. En el momento en que se sea un adepto a escribir código con hilos, también seremos capaces de enfrentarnos a nuevas batallas de programación de forma creativa, sin necesidad de demasiado IPC y de muchas de sus complicaciones. Todos estos beneficios trabajan en conjunto para hacer de la programación multi-hilos algo divertido, rápido y flexible.

1.8.2 Clon:

Clone () es similar a fork (), pero puede hacer muchas cosas que los hilos permiten hacer. Por ejemplo, con __clone() se pueden compartir selectivamente partes del contexto de ejecución del padre (espacio en memoria, descriptores de fichero, etc.) con un nuevo proceso hijo. Esto es muy buena cosa. Pero hay también muchas cosas no tan buenas acerca de __clone(). Como la página del manual de clone indica:

"Las llamadas clone y sys_clone son especificas de Linux y no deberían usarse en aquellos programas que pretendan ser portables. Para programar aplicaciones con hilos (múltiples hilos de control en el mismo espacio de memoria) es mejor usar una biblioteca que implemente la API de hilos POSIX 1003.1c, como la biblioteca LinuxThreads (incluida en glibc2). Vea pthread_create."

Así, mientras que __clone() ofrece muchos de los beneficios de los hilos, no es portable y sólo podrá usarse bajo Linux. Lo cual no significa que no deba usarse en nuestro código. Pero debemos considerar este aspecto cuando pretendamos usar __clone() en nuestro software. Afortunadamente, como se indica en la página del manual de clone, hay una mejor alternativa: los hilos POSIX. Cuando se quiera escribir código multi-hilo portable, código que funcione bajo Solaris, FreeBSD, Linux y otros, los hilos POSIX son el camino a seguir.

1.8.3 Mutex

Las llamadas pthread_mutex_lock() y pthread_mutex_unlock() realizan una función extraordinariamente necesaria en los programas por hilos. Proporcionan el significado de una

mutua exclusión (de ahí su nombre). Dos hilos no pueden mantener el mismo mutex bloqueado al mismo tiempo.

Todos los hilos que se marchen a dormir desde una llamada pthread_mutex_lock() en un mutex previamente bloqueado, serán "puestos en cola" para acceder a dicho mutex.

pthread_mutex_lock() y pthread_mutex_unlock() se usan normalmente para proteger estructuras de datos. Esto es, nos aseguramos de que un solo hilo a la vez puede acceder a una cierta estructura de datos bloqueándola y desbloqueándola. Como puede haberse supuesto, la biblioteca de hilos de POSIX garantizará un bloqueo sin tener que poner al hilo a dormir en absoluto si el hilo trata de bloquear un mutex no bloqueado.

El hilo que tiene el mutex bloqueado tiene acceso a la compleja estructura de datos sin preocuparse de que haya otros hilos pretendiendo confundir al mismo en ese preciso momento. La estructura de datos está efectivamente "congelada" hasta que el mutex se desbloquee. Es como si las llamadas pthread_mutex_lock() y pthread_mutex_unlock() estuvieran "en construcción" hasta que una parte de los datos sea modificada o leída. Las llamadas actúan como avisos para los otros hilos, que se van a dormir mientras se realiza cualquier otra llamada y esperan a que llegue su turno en el bloqueo del mutex. Por supuesto, esto es únicamente cierto de someter cada lectura y escritura a una estructura particular de datos con llamadas a pthread_mutex_lock() y pthread_mutex_unlock().

1.9 Conclusiones:

En este capítulo se realizó un estudio del las tendencias actuales, así como una revisión detallada de lo que ha logrado el mundo en esta materia y en especial Cuba. Además, se explicaron claramente los conceptos, las técnicas, tecnologías y herramientas empleadas, así como otras que servirán de apoyo a la propuesta de solución futura.

Capítulo Il: Descripción de los algoritmos

2.1 Introducción:

En este capitulo se brinda una descripción de los requisitos del algoritmo, así como los algoritmos paralelos utilizados para darle solución al problema. Se brinda especial atención al manejo de la concurrencia y la protección de los datos para lo cual en el modelo se establece una solución que incluye programación concurrente mediante múltiples hilos de ejecución. También se describe la instalación y configuración del clúster, así como su mantenimiento y monitoreo.

2.2 Requisitos funcionales:

Clúster:

_Ejecutar procesos remotos en nodos que son agregados al clúster.

_Para coordinar los procesos se utilice el envío y la recepción de mensajes.

_Se configure un clúster de tipo Beowulf para un mejor rendimiento.

El algoritmo funciona en dos modos (máster y esclavo):

Máster:

RF1-Mandar a ejecutar el proceso de simulación en los nodos esclavos.

RF2-Recibir moléculas de los nodos vecinos.

RF2.1-Ubicar las moléculas en una lista de espera.

RF3-Enviar los parámetros de configuración y los identificadores de los nodos vecinos a cada nodo.

RF3.1-Enviar limite inferior del estrato (zi).

RF3.2-Enviar limite superior del estrato (zp).

RF3.3-Enviar identificador del estrato superior (ids).

RF3.4-Enviar identificador del estrato inferior (idi).

RF4-Enviar moléculas al nodo siguiente.

Esclavo:

RF1-Recibir del máster los parámetros de configuración y los identificadores de los nodos vecinos.

RF1.1-Recibir limite inferior del estrato (zi).

RF1.2-Recibir limite superior del estrato (zp).

RF1.3-Recibir identificador del estrato superior (ids).

RF1.4-Recibir identificador del estrato inferior (idi).

RF2-Recibir moléculas de los nodos anterior o siguiente

RF2.1-Ubicar las moléculas en una lista de espera.

RF3-Ejecutar el proceso de simulación.

RF4-Enviar moléculas al nodo siguiente o al nodo anterior.

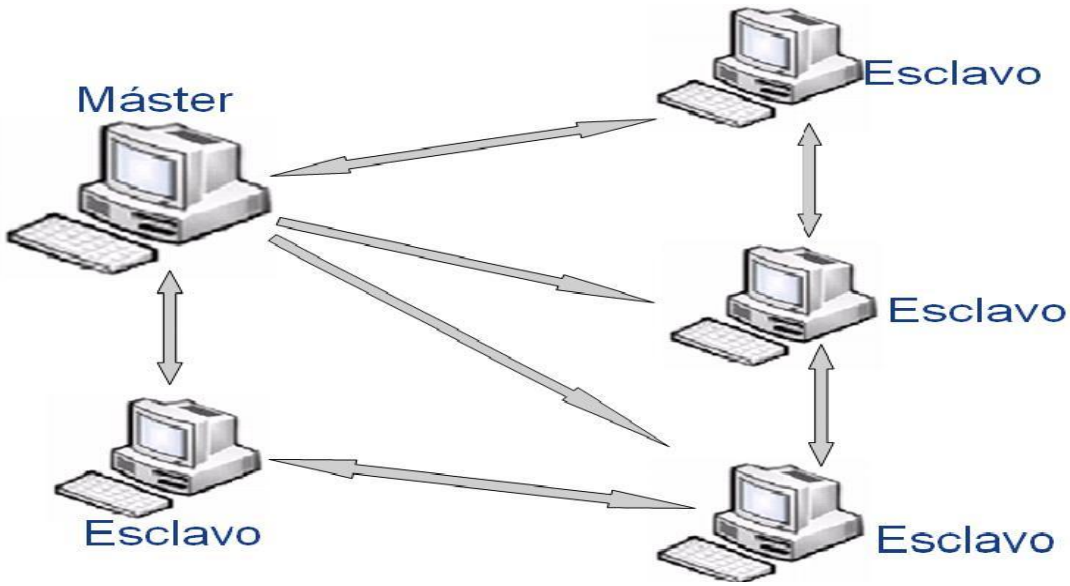
* Enviar y recibir datos entre nodos:

Los datos a enviar son: molécula, límites del estrato, identificadores de los estratos vecinos.

Los datos a recibir son: molécula, límites del estrato, identificadores de los estratos vecinos.

2.3 Descripción General del Proceso:

El proceso inicia cuando el proceso máster manda a ejecutar los procesos esclavos y les envía sus parámetros de configuración, así como los identificadores de sus procesos vecinos. Si una molécula esta lista para abandonar el estrato donde se encuentra, se envían los datos de la misma hacia el proceso superior o inferior y se ubica en la lista del nuevo estrato.



2.3.1 Descripción de la comunicación:

Para darle solución a la comunicación entre procesos, lo primero es levantar los procesos que se deseen mediante el método Levantar_Procesos(), siempre teniendo en cuenta que los procesos deben estar acorde a la cantidad de nodos que posea el clúster en el cual se realiza el cómputo paralelo. Dentro de este método Levantar_Procesos() se envían los parámetros de configuración y los identificadores de los nodos vecinos mediante el método Enviar_Identificadores_Parametros() a cada nodo.

El método Recibir_Molecula() se mantiene en espera que le sea transferida alguna molécula del proceso superior o inferior, este método se encarga además de convertirlo a una Molécula mediante el método Convert_To_Molecula() y ubicarla en una lista de entrada, para posteriormente ser ubicado en la lista general del proceso.

El método Recibir_Identificadores_Parametros(), se encarga de recibir el búfer con los parámetros y los identificadores que le son enviados desde el máster.

La molécula a ser trasladada por el método Enviar_Molecula(), se almacena mediante el método Convert_To_Buf_Molecula() en un búfer, en el cual es enviada al próximo proceso donde será ubicada en una lista de entrada, para posteriormente ser ubicada en la lista general del proceso y eliminada de la lista de salida del proceso donde se encontraba.

2.4 Requerimientos del sistema:

Un nodo tendría que contener, como mínimo un Procesador Pentium 4, superior o igual a 1,6 GHz de velocidad de procesamiento. Los requerimientos de memoria dependen de los requerimientos de datos de la aplicación y del paralelismo, pero un nodo tendría que contener como mínimo Memoria RAM 512 Mb. Entonces los nodos pueden acceder a su partición raíz desde un servidor de archivos a través de la red, normalmente usando el Network File System (NFS). Esta configuración funciona mejor en un entorno con mucho ancho de banda en las conexiones y con un gran rendimiento en el servidor de archivos. Para un mejor rendimiento bajo otras condiciones, cada nodo tendría que tener suficiente espacio en el disco local para el sistema operativo, la memoria virtual y los datos. Cada nodo tendría que tener como mínimo una capacidad de almacenaje de 80Gb de espacio de disco para los componentes del sistema operativo y la memoria virtual. Como mínimo cada nodo tiene que incluir una tarjeta de red Ethernet o Fast Ethernet. Alternativamente, interconexiones de más alto rendimiento, incluyendo la Gigabit Ethernet y la Myrinet, podrían ser usadas en conjunción con CPUs más rápidas. Finalmente, cualquier tarjeta de vídeo, una lectora de disquetes, la caja y la batería,

completan un nodo funcional. Los teclados y los monitores solo se necesitan para la carga y configuración inicial del sistema operativo, a no ser que las máquinas individuales sean usadas interactivamente además de servir como nodos en el sistema paralelo. Todos los componentes hardware escogidos necesitan el soporte de drivers o módulos en Linux.

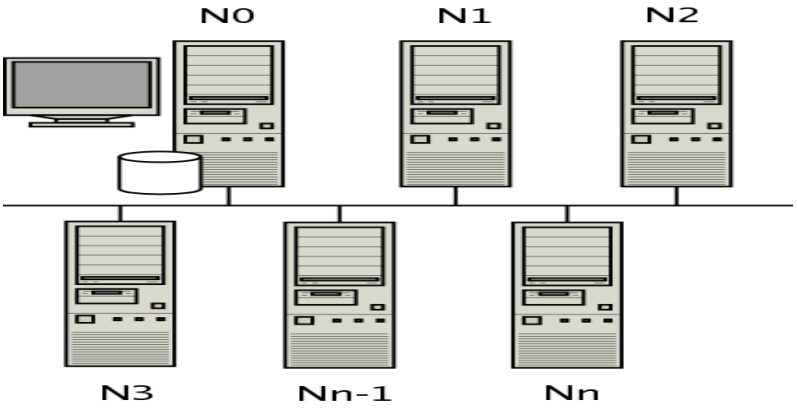
2.5 Instalación y Configuración del Clúster:

2.5.1 Instalación Local Completa en los Clientes:

Todo el software, tanto el sistema operativo como las aplicaciones, son instaladas en los discos internos de cada nodo cliente. Esta configuración reduce a cero el tráfico NFS para obtener el sistema operativo o cualquier otra aplicación por parte de los nodos esclavos.
El sistema operativo a instalar es la distribución Debian de Linux.

2.5.2 Configuración:

La estructura del clúster esta formada por un grupo de computadoras, intercomunicadas entre si por medio de una conexión ya sea de red o por un bus interno. Aquí se presentan los pasos para la configuración del clúster en Linux sobre una red local usando PVM (Parallel Virtual Machine) y el protocolo de seguridad SSH, usando la distribución *Debian* de Linux. El trabajo se realiza sobre la máquina que se desea convertir en MAESTRO del clúster.



Topología utilizada para la interconexión de nodos en el clúster

Cantidad	Equipo	Característica

2	Computadora Haier	2x Intel(R) Pentium(R) D CPU 3.00GHz 3000.38 Mhz 512 Mb RAM 150 GB HD Ethernet 10/100 Mb PCI Debian GNU/Linux 5.0.1
1	Allied Telesis AT 8350GB	48 x Ethernet 10Base-T, Ethernet 100Base-TX Velocidad de Transmisión de Datos 1 Gbps

Descripción del equipo utilizado para la implementación del clúster

Primer Paso

Primero debemos asegurarnos que contamos con un dominio estático de red. Así suponiendo que el IP de nuestra máquina es 192.168.0.X (X puede ser por ejemplo 238), entraremos al fichero de interfaces y convertiremos nuestro dominio a uno de tipo estático. Para esto, usar los siguientes comandos en el shell:

> cd /etc/network //entramos al path del fichero de interfaces
> nano interfaces //accedemos al fichero de interfaces para modificarlo

Este último comando nos presentará el fichero de interfaces, modificarlo y asegurarnos de que quede de la siguiente manera:

```
1.          # iface eth0 inet dhcp
2.             iface eth0 inet static
3.             addres 192.168.0.238
4.             netmask 235.255.255.0
5.             gate way 192.168.0.1
```

Es decir, suponiendo que solo contamos con una tarjeta de red (eth0), lo que hacemos en la línea 1 es comentar nuestro dominio de red mediante dhcp (pues no lo usaremos) y habilitar nuestro dominio estático de red (línea 2). Luego actualizamos nuestra dirección IP, máscara de red, y puerta de enlace. Guardamos los cambios y salimos del archivo interfaces.

Luego, nuevamente en shell tecleamos los comandos:

> ifup eth0 //actualizamos el cambio al dominio estático red
> ifconfig //si deseamos comprobar que todos los cambios se han realizado

Segundo Paso

Ahora debemos configurar los nodos de red del clúster. Es decir, contar con la información de los hosts de aquellas computadoras que queremos que formen parte de nuestro clúster (nodos del clúster). Para esto, usar los siguientes comandos en el shell:

> cd /etc //entramos al fichero etc
> nano hosts //accedemos al fichero de hosts

Este último comando nos presentará el fichero de hosts, aquí colocaremos los hosts de las computadoras que formarán parte de nuestro clúster de la siguiente manera:

```
1.          127.0.0.1             pc01.debian      pc01
2.          192.168.0.231         pc02.debian      pc02
3.          192.168.0.232         pc03.debian      pc03
4.          192.168.0.234         pc04.debian      pc04
```

Por cada IP se coloca el identificador de la computadora (por ejm. pc01.debian) y el identificador con el que se le conocerá en el clúster (por ejm. pc01). En la línea 1 se coloca el IP local de la computadora en la que nos encontramos (computadora MÁSTER). Se pueden colocar todos los hosts que deseemos.

Tercer Paso

Ahora debemos de crear una cuenta de usuario para el acceso al clúster. De esta manera, en shell tecleamos los siguientes comandos:

> adduser *clúster*

El comando adduser sirve para crear un nuevo usuario, el cual, en este caso, tendrá el nombre de *clúster*. Al teclear la tecla INTRO (ENTER) aparecerán solicitudes de información acerca del usuario, siendo la información más importante la contraseña. Entonces, recordar siempre el nombre de usuario y la contraseña, pues son los requisitos necesarios para el acceso al clúster.

Cuarto Paso

Ahora debemos de configurar el protocolo de seguridad SSH para el clúster. La seguridad es sumamente necesaria en todo ámbito de la informática y uno de los protocolos de seguridad implementados para brindar la seguridad de acceso es el protocolo SSH. De esta manera, se le negará el acceso a cualquier computadora con un determinado host que no pertenece a la red de nodos del clúster.

De esta manera, cada vez que una computadora con un determinado host desea acceder, el protocolo SSH se encargará de verificar su autenticidad. Sin embargo, cuando los nodos de nuestra red deseen acceder (para brindar los resultados de su trabajo al máster, por ejemplo), también serán verificados por SSH. Esto puede significar un problema en tiempos de acceso al momento de realizar tareas en el clúster, debido a que además del tiempo ocupado por las tareas de procesamiento, se le sumará el tiempo ocupado en verificación de autenticidad.

Esto puede ser solucionado configurando el protocolo SSH de manera tal que, conociendo a los nodos que pertenecen a la red (gracias al fichero hosts) se les otorgue autorización certificada para su acceso, y se evite su verificación en cada instante. Para esto, los comandos son:

```
//creamos una clave pública mediante el algoritmo RSA
> # ssh-keygen -t rsa
//enviamos la clave pública creada a todos los nodos del clúster (esclavos)
> # cd root /.ssh/
> ssh# scp id_rsa.pub root@pc_esclavo: ~/.ssh/claves.pcmáster
```

En el primer comando creamos una clave pública mediante el algoritmo de cifrado de clave publica RSA. En el segundo comando, se esta enviando dicha clave pública a la computadora esclavo con identificador pc_esclavo, dicha clave será almacenada en el directorio de autenticación de ssh en el archivo claves. SSH sabrá que el computador con identificador pcmáster ha enviado claves de seguridad a pc_esclavo. Hacer esto para todas las computadoras que pertenezcan a la red de nodos del clúster, colocando los identificador pc_esclavo usados en el archivo de hosts y el identificador pcmáster del computador en el que nos encontramos.

Ahora debemos autenticar a los esclavos, hacer:

> ssh# ssh pc_esclavo
pc_esclavo: ~# cd .ssh/
pc_esclavo: ~/ssh# cat claves.pcmáster >> authorized_keys
pc_esclavo: ~/ssh# exit

De esta manera, a la computadora con identificador pc_esclavo le decimos que permita el acceso autorizado de la computadora con identificador pcmáster. Realizar esta tarea también para todas las computadoras que deseemos que sean esclavos de la red de nodos (incluso para la máquina que es nuestro MÁSTER, ya que puede realizar ambas funciones).

Quinto Paso

Finalmente, debemos usar el software PVM para el manejo del clúster. Hasta el paso anterior, lo que logramos es la construcción y configuración de la arquitectura del clúster, sin embargo, ahora nos hace falta un administrador de esa arquitectura. Con este objetivo, usamos el software PVM (Parallel Virtual Machine) , el cual se encargará del manejo del clúster, asignando las tareas que el MÁSTER envíe a sus ESCLAVOS, enviando al MÁSTER los resultados de las tareas que sus ESCLAVOS ejecuten, entre otras.

Para esto debemos tener instalado PVM. Una vez eso, lo único que debemos de hacer es acceder a PVM desde la computadora MÁSTER y agregar todos los identificadores de los esclavos que pertenecen a nuestra red de nodos del clúster de la siguiente manera:

> pvm
pvm > add pc_esclavo

Hacer esto con todos los esclavos. Para salir del entorno de PVM usar:

pvm > exit

Hasta este punto, ya tendremos armado nuestro clúster en nuestra propia red local, el cual

cuenta con la seguridad brindada por el protocolo SSH y administrado por PVM. Ahora solo nos queda aprovechar de la capacidad computacional que nos ofrece un clúster.

2.5.3 Mantenimiento:

El mantenimiento de nuestro clúster no requiere de grandes esfuerzos físicos, ya que el mismo está estructurado inicialmente solo por dos computadores. Tampoco requiere de grandes recursos monetarios por lo anterior descrito y por ser este de Clase l(Sistema compuesto por computadores cuyos componentes cumplen con la prueba de certificación “Computer Shopper”, lo que significa que sus elementos son de uso común, y pueden ser adquiridos muy fácilmente en cualquier tienda distribuidora).

2.5.4 Monitoreo:

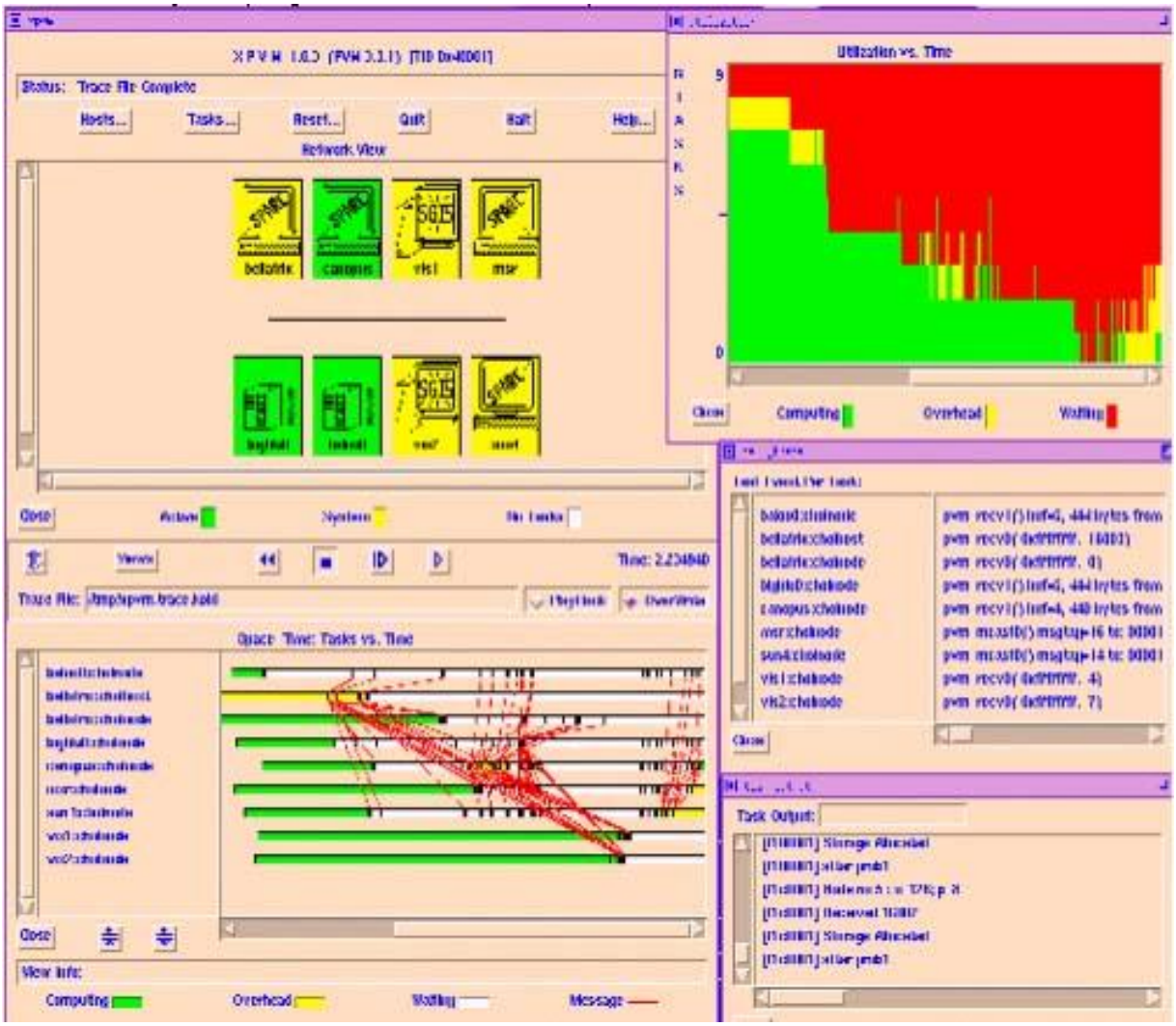
En muchas ocasiones, es muy útil tener una representación gráfica de la configuración de la máquina virtual que se está utilizando, así como una codificación visual de la actividad llevada a cabo en cada host de la máquina virtual, qué mensajes se están enviando, quién los envía y a dónde.

XPVM

[24]La interfaz gráfica de usuario de PVM (XPVM) permite realizar todas estas funciones. XPVM combina las funciones de la consola básica PVM con un monitor de seguimiento de actividades y un debugger en una interfaz tipo X-Windows. XPVM está escrito en C, usando el toolkit TCL/TK.

Para ejecutar XPVM, hay que asegurarse de que el demonio (daemon) no está ya corriendo y que no haya ficheros temporales relacionados con PVM.

La consola se compone de varias vistas de tamaño reconfigurable y una serie de ventanas que son utilizados por XPVM para mostrar mensajes de estado o de ayuda (Status y Help). Por defecto, la consola inicialmente muestra la vista de red (Network View) y la vista de representación temporal de tareas (Space-Time).



El menú Hosts nos permite añadir un nuevo host a la máquina virtual, seleccionado de entre todos los hosts listados en el fichero *.xpvm_hosts*.

En este caso vamos a añadir varios hosts. Cada vez que añadimos uno aparece un nuevo símbolo de host conectado a los símbolos existentes.

A través del menú Tasks-SPAWN pueden lanzarse tareas en cualquiera de los hosts que compone la máquina.

La vista de Representación de Tareas muestra el estado de todas las tareas que se están ejecutando en la máquina virtual en un momento dado. Para que las tareas se muestren, el botón PLAY que se ve en la parte superior de la ventana de visualización de tareas. La visualización puede ser interrumpida o terminada en cualquier momento, utilizando los botones PAUSE y STOP. Una vez detenida la visualización se mover hacia el pasado o el futuro de las tareas utilizando los botones REWIND y FORWARD.

La vista de Representación de Tareas se compone de dos ventanas. La ventana izquierda contiene el nombre del host y el de la tarea ejecutada en el mismo. Las tareas aparecen ordenadas alfabéticamente. El número de tareas mostradas en una ventana puede aumentarse utilizando el botón de compresión de tareas que aparece a la izquierda de los botones anteriormente mencionados.

La ventana derecha muestra, para cada proceso, el estado de dicha tarea en cada momento, así como líneas rojas que emanan de cada proceso y que corresponden a envíos de mensajes entre procesos. El código de colores muestra el estado del proceso, que puede estar ejecutando tareas propias (verde), rutinas PVM (amarillo) o esperando mensajes (blanco).

El usuario puede recabar información detallada sobre un estado determinado o un mensaje, seleccionando con el botón izquierdo un estado u mensaje. Si se selecciona una barra de tarea, se obtiene su estado así como el tiempo de comienzo y fin de la tarea y la última llamada a PVM que se hubiera generado. Si se selecciona una línea de mensaje, la ventana que aparece mostrará el tiempo de envío y recepción, así como el número de bytes enviado y el identificador de mensaje.

La representación de tareas en la ventana derecha puede ampliarse o reducirse utilizando simultáneamente los dos botones del ratón (simula el botón central de un ratón de tres botones) y el botón derecho, respectivamente.

Existe una ventana de salida de tareas, accesible a través del menú VIEWS, que actúa como salida estándar para los procesos.

Finalmente, existe una ventana de utilización de recursos, accesible también a través del menú VIEWS. Esta ventana, que está sincronizada con la ventana de representación de tareas, muestra el número de tareas que están ejecutándose, ejecutando funciones PVM o en espera en cada momento.

2.6 Conclusiones:

En este capítulo quedaron descritos los requisitos del algoritmo, así como la descripción de los mismos. Además, se muestra el proceso de instalación y configuración del clúster, así como el de mantenimiento y monitoreo del mismo, siendo este último, un elemento importante en el desarrollo de algoritmos para procesamiento paralelo, ya que a partir del mismo podemos comprender bajo que circunstancias nuestros algoritmos son más eficientes.

Capítulo lll, Modelación y descripción de las pruebas a realizar.

3.1 Descripción del Problema a Resolver:

Los cálculos que se realizan a la hora de simular el proceso de difusión de mezclas de gases en sólidos poli-cristalinos con múltiples estratos a través de un programa secuencial son demasiado grandes, por lo que se hace necesario implementar un algoritmo de cómputo paralelo para la comunicación entre procesos que se ejecutarán simultáneamente.

Para lograr que este proceso se realice en tiempos óptimos hay que tener en cuenta una serie de parámetros entre los que se destaca la Topología de Red con que se cuenta. Una vez obtenido el algoritmo, se necesita de un **Método** a seguir con el fin de medir que lo que se realizó es realmente óptimo.

3.2 Topología de Red del Centro:

La Red del Centro cuenta con una Topología de Árbol, la cual cuenta un nodo de enlace troncal, generalmente ocupado por un switch, desde el que se ramifican los demás nodos. Desde una visión topológica, la conexión en árbol, es parecida a una serie de redes en estrellas interconectadas salvo en que estas tienen un nodo central. Es una variación de la red en bus, donde la falla de un nodo no implica interrupción en las comunicaciones. Se comparte el mismo canal de comunicaciones. Este Centro posee un Red LAN.

El ancho de banda entre el Switch con los nodos es de 100Mbps. El medio de conexión entre los nodos es por cable Par Trenzado, de tipo UTP de categoría 5, están conectados a puntos de red y estos a un Switch que existe en el laboratorio.

Características de la Fibra Óptica utilizada: medio de transmisión de información analógica o digital, donde las ondas electromagnéticas viajan en el espacio a la velocidad de la luz. Presenta dimensiones más reducidas que los medios preexistentes y el peso del cable de fibras ópticas es muy inferior al de los cables metálicos, redundando en su facilidad de instalación. La fibra óptica presenta un funcionamiento uniforme desde -550 C a +125 C sin degradación de sus características.

Características UTP de Categoría 5: Velocidad de hasta 100 Mbps, con un ancho de banda de 100 MHz, diseñado para soportar voz, video y datos. El UTP tiene un costo accesible y su instalación es fácil. Sus dos alambres de cobre torcidos aislados con plástico PVC, ha demostrado un buen desempeño en las aplicaciones de hoy.

3.3 Casos de Prueba:

Se definirán Casos de Prueba, como **Método** que permita comprobar el rendimiento del sistema.

3.3.1 ¿Qué es un Caso de Prueba?

[25]Los Casos de Pruebas son un conjunto de condiciones o variables bajo las cuáles se determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Agregar que las Pruebas de software son:

- Son un elemento crítico para la garantía de calidad del software.
- Pretenden descubrir errores.

3.3.2 ¿Como diseñar un Caso de Prueba?

Para diseñar un Caso de Prueba hay que tener en cuenta algunas consideraciones:

- Un Caso de Prueba especifica: Componente a probar, Datos de entrada, Estado del componente, Información de contexto, Resultado esperado

Un Caso de Prueba con alta probabilidad de detectar algún error no debe ser redundante, debe ser representativo, no debe ser ni muy simple ni muy complejo. Es por eso que se trata los Casos de Pruebas como una actividad necesaria para comprobar el correcto funcionamiento de un programa o aplicación. Es necesario diseñar un conjunto suficientemente amplio y variado de Casos de Prueba, donde se establezcan de antemano los resultados que debe producir una ejecución correcta de los mismos, para así una vez ejecutados en el programa poder comprobar los resultados obtenidos con las salidas esperadas.

3.3.3 Fases de un Diseño de Casos de Prueba:

Generalmente cuando se diseñan Casos de Prueba para algún programa o aplicación se analizan las siguientes fases:

✓ **Diseño del Proceso.** Determinar los objetivos a analizar: recursos a considerar, factores relevantes en cada recurso, intervalos de análisis para cada factor y recurso. Después, el número de pruebas, para conseguir un análisis fiable y preciso, se entiende que cuanto mayor sea el número de pruebas más completo y fiable será el estudio. Por último establecer las condiciones en las que se deben aplicar los Casos de Prueba.

✓ **Estudio Teórico.** Estudiar de forma teórica las funciones que indican la variación de los recursos en función de los diferentes factores. No se trata de hacer un análisis del código línea por línea, sino de forma más global y abstracta, teniendo en cuenta las estructuras de datos y los algoritmos que intervienen en cada operación.

✓ **Creación de los Casos de Prueba.** Definir un método adecuado para crear los diferentes Casos de Prueba. Asimismo, se deberá definir un método para medir los resultados obtenidos.

✓ **Análisis Estadístico y Contraste Teórico/Experimental.** Se analizarán los resultados obtenidos, sugiriendo las siguientes técnicas: representaciones gráficas (que muestren la información relevante), para ello se usarán las herramientas estadísticas/matemáticas. Además del análisis, se deben mostrar tubularmente los valores obtenidos y usados (es decir, los datos en crudo).

✓ **Predicción y Conclusiones.** El ajuste teórico/experimental puede utilizarse también para realizar previsiones del tiempo y la memoria en tamaños grandes y difíciles de alcanzar. Por último, obtener las conclusiones más relevantes del estudio realizado, en cuanto a: eficiencia obtenida, puntos fuertes y débiles del programa, qué cosas se podrían mejorar, fiabilidad del estudio, resultados del análisis de eficiencia y del contraste teórico/experimental.

3.4 Propuesta de Caso de Prueba:

Teniendo en cuenta la descripción del problema, a continuación se muestra una propuesta de Caso de Prueba, que medirá el tiempo en que demora una molécula en llegar al reservorio lleno dada una cierta cantidad de puntos, así como la memoria promedio usada.

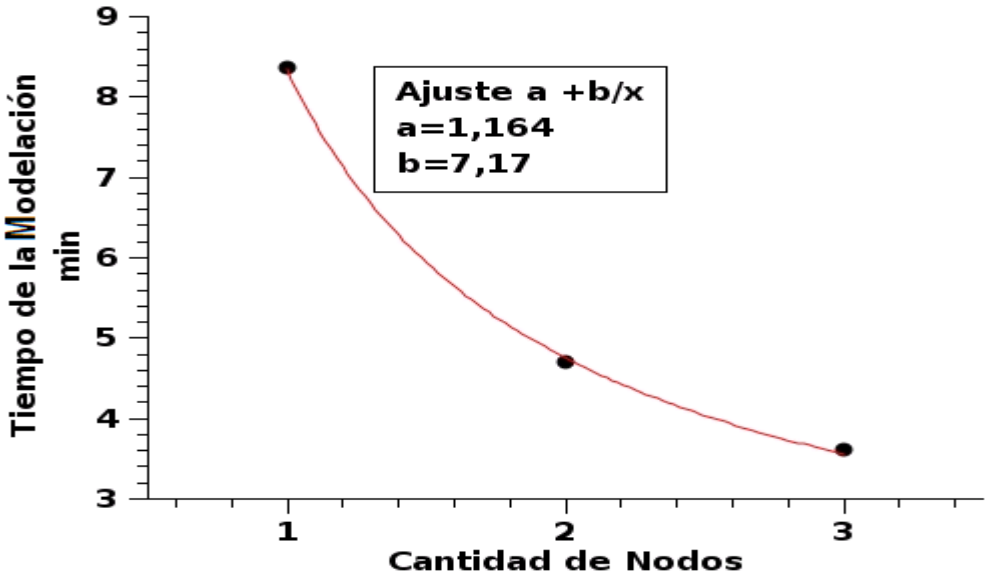
3.5 Propuesta de Solución:

Resultados de la Modelación

Cantidad de Puntos

N **1000**

Modelo	Tiempo (min)	Memoria Promedio Usada (Mb)
Todo el sistema, secuencial	8,35	57,6
2 Estratos	4,59	29,5
3 Estratos	3,6	24,9



Se puede observar que tanto el tiempo de cómputo como la memoria utilizada disminuyen.

El tiempo utilizado disminuye pero según el modelo propuesto tiene un límite inferior originado por el tráfico de la red, en la medida que aumenta la cantidad de estratos la cantidad de datos

analizados en cada uno disminuye pero el tráfico de red aumenta y por tanto la posibilidad de choques.

3.7 Conclusiones:

En este capítulo se modelaron y describieron las pruebas realizadas. Estas pruebas son de una importancia significativa, porque son las que no dan un grado mayor de conocimiento, en cuanto a si el trabajo realizado es factible y cumple con su objetivo principal.

Se realizaron los estudios correspondientes y los resultados fueron plasmados en el mismo, además de una representación gráfica que muestra dichos resultados.

Conclusiones

Las demandas de cómputo de alto rendimiento en las áreas de investigación científica y tecnológica día a día demandan mayores capacidades de procesamiento. El uso de las supercomputadoras es una alternativa para satisfacer las demandas de cómputo de instituciones dedicadas a la investigación, pero el alto costo de una máquina de tal magnitud representa una enorme inversión económica.

Al concluir el presente trabajo queda realizado un estudio sobre los sistemas para cómputo paralelo de datos a partir del cual se decidió realizar la implementación de algoritmos para la comunicación entre procesos específicamente implementados para el proyecto Transmol. El proceso de implementación de estos algoritmos se desarrolló por la herramienta escogida. Por lo anteriormente expuesto se dan por cumplidos los objetivos propuestos al iniciar el trabajo, existiendo una correspondencia entre los objetivos planteados y los resultados obtenidos.

Recomendaciones

Tras de haber alcanzado los objetivos de este trabajo, se recomienda lo siguiente:

- ✓ Comenzar un nuevo ciclo de desarrollo, el cual será el encargado de acoplar al sistema Transmol esta biblioteca encargada de la comunicación entre procesos.
- ✓ Optimizar el clúster para lograr así un mejor funcionamiento del mismo.
- ✓ Que se sigan modelando pruebas, con otras variables y parámetros y se realicen otros tipos de pruebas, para de esta forma lograr una aplicación 100% eficiente.
- ✓ Que este trabajo sea objeto de estudio, para trabajos similares.

Bibliografía

[1]Procesador Intel® Core™ i7 Extreme Edition. Disponible en: http://www.intel.com/espanol/products/processor/corei7ee/index.htm**.** Consultado 22-6-2009.

[2]Hilos Posix. Disponible en: http://www.gentoo.org/doc/es/articles/l-posix1.xml. Consultado 15-2-2009. Consultado 22-6-2009.

[3]Beowulf. Disponible en: http://www.beowulf.org/. Consultado 9-2-2009.

[4]Dorfman, M. and Thayer, R., "*Standards, Guidelines and Examples on System and Software Requirements Engineering*", IEEE Computer Society Press, 1990.

[5]Sitio oficial de investigación sobre clúster en la UNAM. Disponible en: http://clústers.unam.mx. 2003. Consultado 10-2-2009.

[6]PVM Y XPVM. Disponible en: http://www.ibiblio.org/pub/linux/docs/LuCaS/Manuales-LuCAS/doc-clúster-computadoras/doc-clúster-computadoras-html/node43.html. Consultado 9-2-2009.

[7]Parallel Virtual Machine (PVM), by Feng-Chao Yu, Nov. 23, 1999. Disponible en: http://carbon.cudenver.edu/csprojects/csc5809F99/class_projects/feng_chao_yu/full_project/vs.htm. Consultado 20-2-2009.

[8]NUMA. Disponible en: http://oss.sgi.com/projects/numa/. Consultado 10-2-2009.

[9]MPI. Disponible en: http://www.mpi-forum.org/docs/mpi21-report.pdf. Consultado 15-2-2009.

[10]Protocolos de comunicación. Disponible en: http://www.forest.ula.ve/~mana/cursos/redes/protocolos.html. Consultado 10-2-2009.

[11]Manual de referencia de Red Hat Linux. 2003. Disponible en: http://www.linux-cd.com.ar/manuales/rh9.0/rhl-rg-es-9/ch-ssh.html#FTN.AEN16548. Consultado 10-2-2009.

[12]David HM Spector. Building Linux Clústers. O´Really. 2000.

[13]Clúster Computing. Architectures, Operating Systems, Parallel Processing and Programming Languages. Richard S. Morrison. GNU General Public Licensed. 2003.

[14]Concepto de Clúster. Disponible en: http://www.dei.uc.edu.py/tai2003-2/clústering/html/concepto_de_clúster.html. Consultado 15-2-2009.

[15]704 Data Processing System, 2009. Disponible en: http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP704.html. Consultado 15-2-2009.

[16]7030 Data Processing System, 2009. Disponible en: http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP7030.html. Consultado 15-2-2009.

[18]Timeline of Computing History, Computer, vol. 29, no. 10, pp. TL1-TL34, Oct. 1996.

[17]Bull home page, 2009. Disponible en: http://www.bull.com. Consultado 20-2-2009.

[19]Fast Fourier Transform, 2009. Disponible en: http://mathworld.wolfram.com/ FastFourierTransform.html. Consultado 20-2-2009.

[20]Configuración de un Clúster en Linux. Disponible en: http://jc-info.blogspot.com/2009/02/configuracion-de-un-clúster-en-linux.html. Consultado 10-3-2009.

[21]OpenMosix. Disponible en:http://openmosix.sourceforge.net/. Consultado 10-3-2009.

[22]Rocks Clúster. Disponible en:http://www.rocksclústers.org/wordpress/. Consultado 10-3-2009.

[23]OSCAR. Disponible en: http://svn.oscar.openclústergroup.org/trac/oscar. Consultado 10-3-2009.

[24]Introducción XPVM. Disponible en: http://www.netlib.org/utk/icl/xpvm/xpvm.html. Consultado 20-3-2009.

[25]María Elena Alvarez Hymelin. 2008. Estudio de rendimientos de módulos de acceso y procesamiento paralelo a datos. 2008.
Disponible en: http://biblioteca.uci.cu/sbd/biuci/index.html. Consultado 5-6-2009.

[26]Demontis P., Suffritti G.B., Quartieri S., Fois E.S., Gamba G., Molecular dynamics studies on zeolites. II. A simple model for silicates applied to anhidrous natrolite, Zeolites, 7(11),522-527, (1987)

[27]Krishna R. , Standart G.L. A multicomponent film model incorporating a general matrix method of solution to the Maxwell-Stefan equations, AIChE J., 22(2), 383-389(1976)

[28]Krishna R., Predicting transport diffusivities of binary mixtures in zeolites, Chem. Phys. Lett, 355, 483-489 (2002)

[29]Paschek D., Krishna R., Kinetic Monte Carlo simulations of transport diffusivities of binary mixtures in zeolites., Phys. Chem. Chem. Phys, 3, 3185-3191 (2001)

[30]Paschek D., Krishna R., Diffusion of Binary Mixtures in Zeolites: Kinetic Monte Carlo versus Molecular Dynamics Simulations, Langmuir, 17,247-254 (2001).

[31]Krishna R., van Baten J.M., García-Pérez E., and Calero S. Incorporating the Loading Dependence of the Maxwell-Stefan Diffusivity in the Modeling of CH4 and CO2 Permeation Across Zeolite Membranes, Ind. Eng. Chem. Research, 46, 2974-2986 (2007)

[32]Van Baten J.M., Krishna R., Entropy effects in adsorption and diffusion of alkane isomers in mordenite: An investigation using CBMC and MD simulations, Microp. Mesop. Mater., 84, 179-191 (2005)

[33]Intel Corporation y Microsoft Corporation se asocian con instituciones académicas. Disponible en: http://www.intel.fi/espanol/pressroom/releases/2008/0318.htm. Consultado 20-6-2009.

[34]G.S. Almasi and A. Gottlieb. Highly Parallel Computing. Disponible en: http://portal.acm.org/citation.cfm?id=1011116.1011127. Consultado 20-6-2009.

[35]Blaise Barney, Lawrence Livermore National Laboratory. Introduction to Parallel Computing. Disponible en: https://computing.llnl.gov/tutorials/parallel_comp/. Consultado 20-6-2009.

[36]Dynamic Systems Development Method. 2009. Disponible en: http://www.scribd.com/doc/14529506/Dynamic-Systems-Development-Method?autodown=pdf. Consultado 20-3-2009.

[37]Qt Creator: FAQ. Disponible en: http://www.qtsoftware.com/include/qt-4.5-staging-area/depreciated/qt-creator/faq. Consultado 20-3-2009.

Glosario de Términos:

Clúster: conjunto de computadoras las cuales trabajan en conjunto para resolver una tarea.

Conexión heartbeat: Usualmente utilizada para monitorear cuál de todos los servicios está en uso, así como la sustitución de una máquina por otra cuando uno de sus servicios haya caído.

El envenenamiento del DNS: ocurre cuando un intruso entra en el servidor de DNS, apuntando sistemas hacia hosts intencionalmente duplicados.

Encriptar: Es la ciencia que estudia los distintos sistemas de cifrado destinados a ocultar el significado de mensajes a otras partes que no sean el emisor y el receptor de dicha información.

Red Hat: es la compañía responsable de la creación y mantenimiento de una distribución del sistema operativo GNU/Linux que lleva el mismo nombre: Red Hat Enterprise Linux, y de otra más, Fedora. Así mismo, en el mundo del middleware patrocina jboss.org, y distribuye la versión profesional bajo la marca JBoss Enterprise.

Red Hat Linux: Red Hat es una distribución Linux creada por Red Hat(Compañía responsable de la creación y mantenimiento de una distribución del sistema operativo GNU/Linux que lleva el mismo nombre: Red Hat Enterprise Linux, y de otra más, Fedora.), que fue una de las más populares en los entornos de usuarios domésticos.

Rsh: es un programa de consola para ejecutar comandos en ordenadores remotos.

Shell: Es un programa informático, que actúa como interfaz de usuario para comunicar al usuario con el sistema operativo, mediante una ventana que espera órdenes escritas por el usuario.

Software. Programas de computadora, estructuras de datos y su documentación que sirven para hacer efectivo el método 1ógico, procedimiento o control requerido.

speed up: Ganancia de velocidad (proporción establecida entre el tiempo consumido por un algoritmo secuencial y su equivalente paralelo).

SSH: Es un programa para el registro y ejecución de comandos en una computadora remota de forma segura mediante el uso de una comunicación encriptada.

Switch: es un dispositivo analógico de lógica de interconexión de redes de computadores que opera en la capa 2 (nivel de enlace de datos) del modelo OSI. Su función es interconectar dos o más segmentos de red, de manera similar a los puentes (bridges).

TCP/IP: Sistema de protocolos, definidos en la RFC 793, en los que se basa buena parte de la comunicación de Internet. TCP/IP es el estándar de protocolo de comunicaciones requerido por las computadoras que acceden a Internet.

Telnet (**TELecommunication NETwork**)**:** es el nombre de un protocolo de red (y del programa informático que implementa el cliente), que sirve para acceder mediante una red a otra máquina, para manejarla remotamente como si estuviéramos sentados delante de ella.

Anexos:

1 Ejemplos de algunas imágenes de clúster BeoWulf en algunas Empresas del Mundo