Universidad de las Ciencias Informáticas

Facultad 4

TÍTULO: ESTUDIO DE RENDIMIENTO DE MÓDULOS DE ACCESO Y PROCESAMIENTO PARALELO A DATOS.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

AUTOR(ES): MARÍA ELENA ALVAREZ HYMELIN.

TUTOR(ES): LIC.EDISEL NAVAS CONYEDO.

LA HABANA, CUBA

JULIO DE 2008

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 _____.

_____	_____
María Elena Alvarez Hymelin	Edisel Navas Conyedo
(Autora)	(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.

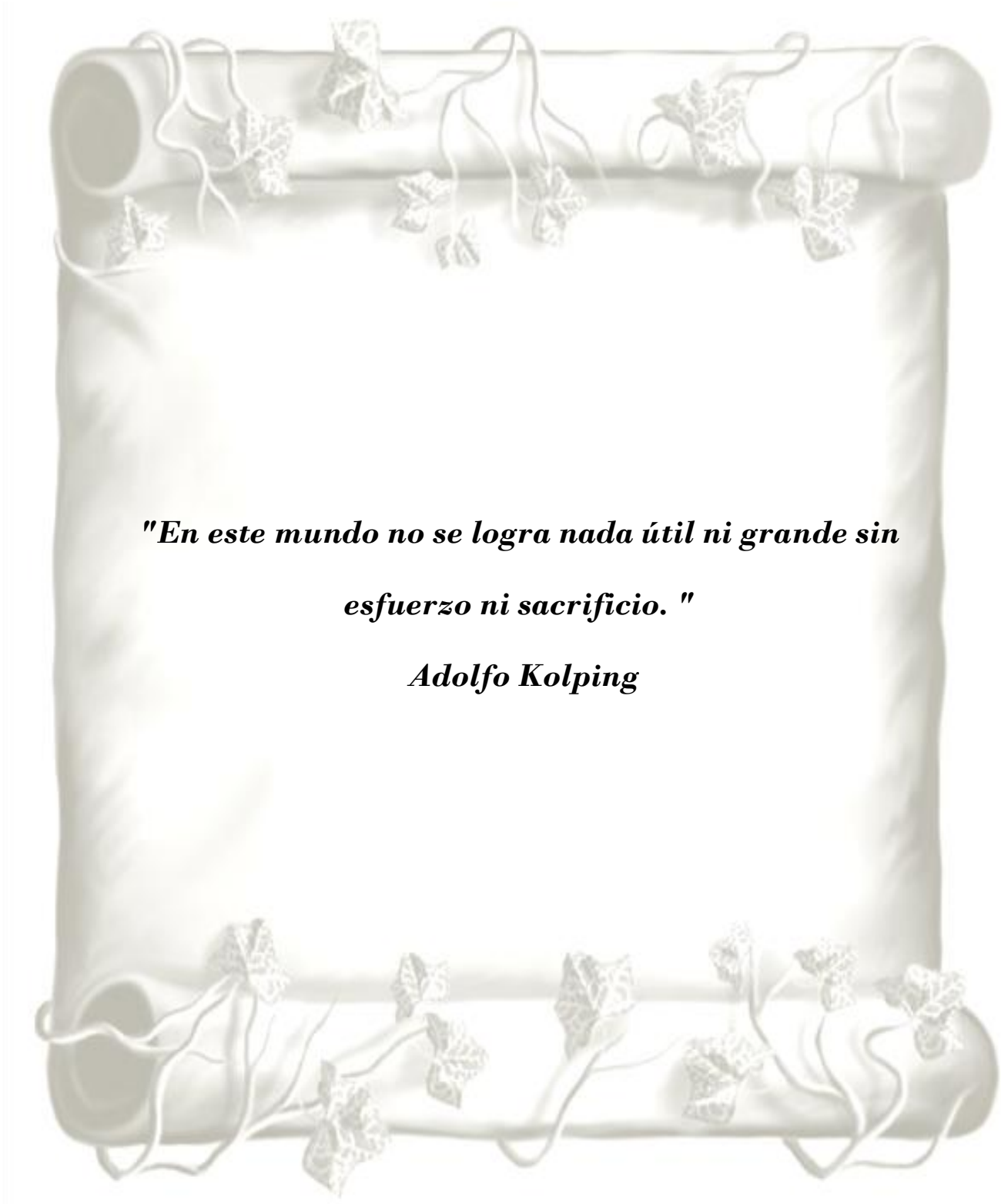
Esta inscrito en la Maestría en Física en la Facultad de Física de la Universidad de la Habana, con el tema:"Caracterización de la s Nanoparticulas por mediciones optimas".

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 depolimeros 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

PENSAMIENTO



"En este mundo no se logra nada útil ni grande sin esfuerzo ni sacrificio. "

Adolfo Kolping

AGRADECIMIENTOS

Agradezco a todas las personas que de una forma u otra han colaborado con este trabajo de Diploma.

A la Revolución por haberme dado la oportunidad de estudiar en esta Universidad.

A la UCI por hacer de mi lo que soy hoy.

A mi tutor por apoyarme y ayudarme en todo momento.

A todas aquellas personas que aportaron en mi formación.

A mis amigos por darme fuerza y su apoyo.

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

A mi familia por darme su confianza.

A mi papá y a mi hermano por ayudarme a sostenerme cada vez que sentía que las cosas eran imposibles.

A mi mamá por ser mi orgullo y mi fuerza en todo este tiempo de estudio y en esta etapa final.

DEDICATORIA

Dedico este trabajo de Diploma a las personas que siempre me han dado la confianza que he necesitado.

A mi familia por su hermosa compañía y estar siempre pendientes de mi.

A mis amigos, los viejos y los nuevos, por estar conmigo en todo este tiempo, porque con ellos he vivido días felices y tristes.

A mi tutor por tener la paciencia necesaria y ayudarme en todo momento.

A mi papá por ser mi orgullo y mi espíritu de lucha.

A mi hermano por ser mi esperanza y mis deseos de vivir.

A mi mamá por ser la persona que siempre he querido ser y a la cual le dedico este trabajo, porque este triunfo es para ella.

A todas las personas que siempre han estado a mi lado.

RESUMEN

Para asegurar y garantizar el correcto funcionamiento de una aplicación o sistema, las pruebas cumplen un papel fundamental, permitiendo de esta forma saber hasta que punto dicha aplicación o sistema puede funcionar correctamente.

En el presente trabajo se realizaron pruebas de rendimiento con el objetivo de saber el funcionamiento de la aplicación en la que se ha estado trabajando, para ello, dicho trabajo se ha dividido en tres capítulos.

El primer capítulo es un estudio del arte donde se abordan de manera general que es la programación paralela, los modelos de programación que existen , la utilización de esta técnica de programación en las diferentes ramas de la economía, conceptos importantes dentro de la programación paralela.

El segundo capítulo está enmarcado en fundamentar teóricamente, qué son los casos de pruebas, y que aspectos y parámetros más importantes hay que tener en cuenta para realizar dichos casos de pruebas.

En el tercer capitulo se propusieron algunas propuestas de casos de pruebas y se realizaron las pruebas de rendimiento, llevando los resultados a tablas y gráficos y mostrándolos también de manera escrita y por último llegar a conclusiones de los datos más relevantes.

PALABRAS CLAVE

Programación paralela, Pruebas de Rendimiento.

Índice de Contenido

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5

1.1 INTRODUCCIÓN AL ANÁLISIS DE PROGRAMACIÓN PARALELA. 5

1.1.1 Paralelismo. 5

1.2 TAXONOMÍA BASADA EN EL ACCESO A LA MEMORIA 7

1.2.1 Clúster 9

1.3 PARALELISMO DE DATOS Y PARALELISMO DE CONTROL. 17

1.4 PRINCIPIOS PARA EL DISEÑO DE ALGORITMOS PARALELOS. 17

1.5 PARADIGMAS DE LA PROGRAMACIÓN PARALELA. 18

1.6 MODELOS DE LA PROGRAMACIÓN PARALELA. 18

1.6.1 Modelo BSP. 19

1.6.2 PVM. 20

1.6.3 MPI 21

1.6.4 Threads/Procesos. 22

1.6.5 PRAM. 22

1.6.6 D-RAM 23

1.7 PRINCIPIOS BÁSICOS PARA REALIZAR UN ANÁLISIS DE RENDIMIENTO EN UN SISTEMA PARALELO. 24

1.7.1 Objetivos del Análisis del Rendimiento en un Sistema Paralelo. 24

1.7.2 Técnicas utilizadas para el Análisis del Rendimiento en un Sistema Paralelo. 24

1.7.3 Cuestiones básicas para llevar a cabo el Análisis de Rendimiento. 25

1.8 FASES DE PROGRAMACIÓN PARALELA. 25

1.9 SITUACIÓN ACTUAL DE LA PROGRAMACIÓN PARALELA EN EL MUNDO. 26

1.9.1 Situación en Cuba. 28

1.10 CONCLUSIONES 28

CAPÍTULO 2: DESCRIPCIÓN DEL PROBLEMA A RESOLVER. 30

2.1 NECESIDAD DE LA UTILIZACIÓN DE TECNOLOGÍAS Y REDES DE COMUNICACIÓN. 30

2.2 DESCRIPCIÓN DEL PROBLEMA A RESOLVER EN EL CENTRO UCIFAR. 30

2.3 Topología de Red del Centro UCIFAR. 31

2.3.1 Análisis y Estudio de la Organización de la Red. 31

2.3.2 Proyección de la Red 33

2.3.3 Simulación de la Red del Centro UCIFAR 35

2.4 CASOS DE PRUEBA. 36

2.4.1 ¿Qué es un Caso de Prueba? 36

2.4.2 ¿Como diseñar un Caso de Prueba? 37

2.4.3 Fases de un Diseño de Casos de Prueba. 37

2.5 CONCLUSIONES 38

CAPITULO 3: MODELACIÓN Y RESULTADOS 39

3.1 PROPUESTA DE CASOS DE PRUEBA. 39

3.1.1 Posibles Casos de Pruebas, a partir de las Condiciones del Centro UCIFAR. 39

- ✓ Caso de Prueba 1 39
- ✓ Caso de Prueba 2 39
- ✓ Caso de Prueba 3 40
- ✓ Caso de Prueba 4 40
- ✓ Caso de Prueba 5 40

3.2 PROPUESTA DE SOLUCIÓN. 41

3.3 RESULTADOS OBTENIDOS. 42

3.3.1 Resultados con la utilización de una sola PC. 42

3.3.2 Resultados para la aplicación de 30 PC's. 44

3.4 Conclusiones 50

CONCLUSIONES 51

RECOMENDACIONES 52

REFERENCIA BIBLIOGRÁFICA 53

BIBLIOGRAFÍA 54

ANEXOS 56

GLOSARIO 59

INTRODUCCIÓN

La industria informática es una de las industrias de mayor crecimiento y es impulsada por los rápidos desarrollos que se producen en las áreas del hardware y del software. Los avances tecnológicos del hardware incluyen el desarrollo de chips y nuevas tecnologías de fabricación, microprocesadores más rápidos y baratos, así como también mayor ancho de banda y menor latencia en las interconexiones de redes. También en el área del software se producen avances rápidos. Software maduro, como sistemas operativos, lenguajes de programación, metodologías de desarrollo y herramientas complementarias están ahora disponibles. (Tschanz, 2001)

Sin embargo, hay muchas áreas dentro de la informática que necesitan de una gran inversión en equipos, y esta inversión rara vez llega o es suficiente. Algunas de estas áreas son la meteorología (una simulación de la atmósfera puede llegar a ser tremendamente pesada), la física de la materia condensada con modelos muy densos (fundamentalmente, cuánticos o, para problemas de dinámica molecular, semiempíricos), el estudio de proteínas, análisis de terremotos, la generación de imágenes por computador con modelos realísticos o el secuenciamiento del genoma humano.

Cada era de la computación comienza con un desarrollo en la arquitectura del hardware, seguida por sistemas de software (particularmente en las áreas de compiladores y sistemas operativos), aplicaciones, y finalmente alcanzando el punto máximo en los ambientes de resolución de problemas. La tecnología detrás del desarrollo de componentes de sistemas computacionales en la era secuencial ha alcanzado su madurez, y desarrollos similares están a punto de producirse en la era paralela. Esto significa que la tecnología de la computación paralela necesita avanzar, ya que aún no está lo suficientemente madura como para ser explotado como una tecnología de disponibilidad masiva. (Tschanz, 2001)

La programación paralela, al ser un modelo de programación que tiene como objetivo fundamental obtener mejores rendimientos, es utilizada ampliamente para resolver problemas complejos. Muchos de estos problemas se presentan en la Inteligencia Artificial, los cuales se caracterizan por manejar grandes cantidades de datos o por poseer una gran complejidad (exponenciales).

En la Universidad de las Ciencias Informáticas (UCI), la programación paralela, es para muchos un tema parcialmente desconocido, pero se ha empezado a trabajar en él, debido a la necesidad existente en algunos proyectos de nuestra universidad de procesar grandes volúmenes de datos, situación que no se puede resolver utilizando la programación secuencial y que mediante la programación paralela

estos problemas pueden ser resueltos de una forma, aunque algo compleja es mucho mas rápida y eficiente.

Se tiene conocimiento que en la UCI existen pocos proyectos que utilizan la programación paralela, y los que lo hacen solo lo hacen para casos específicos, sin embargo en la aplicación que se está trabajando, no solo se utilizan para el caso específico de procesamiento de datos, sino para realizar consultas a bases de datos y otras necesidades como por ejemplo realizar operaciones matemáticas complejas que requieran gran cantidad de tiempo para ejecutarlas.

Se ha comprobado que la Programación Paralela surgió para obtener mayores rendimientos que permitan resolver problemas cada vez más grandes en tiempos de computación aceptables, donde en la programación secuencial seria muy engorrosa trabajar, por lo que la programación paralela ha resuelto grandes problemas científicos, tanto matemáticos, como físicos como en cualquier rama de la sociedad.

Pero no solo esto es lo más importante, en la aplicación se debe utilizar los algoritmos más óptimos para lograr que todos estos problemas sean resueltos y realizar así de esta forma pruebas de rendimiento que den la seguridad que el trabajo que se ha estado realizando sea de calidad.

Para lograr la calidad de este proceso es innegable la utilización e interés en el procesamiento paralelo por muchas razones por ejemplo crecimiento de la potencia de cálculo dada por la evolución tecnológica, transformación y creación de algoritmos que explotan la concurrencia para obtener mejores tiempos de respuesta, la necesidad de tratar sistemas que le brinden datos en tiempo real, entre otros. El límite físico de los algoritmos secuenciales que existen hoy en día para el procesamiento de datos convierte en una salida factible la solución paralela.

Las razones expuestas anteriormente dan lugar a que el **Problema Científico** a solucionar sea: ¿Cómo lograr a través de las pruebas de rendimiento saber hasta que punto es eficiente la utilización de estos algoritmos paralelos a la aplicación existente?

Donde se tiene como **Objeto de Estudio** la: Programación Paralela, Sistemas de gestión de datos y el **Campo de Acción** se enmarca en: Proyectos que necesitan gestión y procesamiento de grandes volúmenes de datos (MINFAR, SOLGE).

Para dar respuesta al problema se define como **Objetivo Principal**: Desarrollar las pruebas de rendimiento necesarias a la aplicación para encontrar la estructura más eficiente para los algoritmos paralelos.

Para dar cumplimiento al objetivo principal se trazaron los siguientes **Objetivos Específicos**:

- ✓ Conocer qué modelo de programación paralela fue utilizado en dicha aplicación.
- ✓ Conocer el término clúster, su utilización y funcionamiento.
- ✓ Conocer como funciona algunos de los algoritmos de programación paralela en la aplicación para el procesamiento y gestión de datos.
- ✓ Mostrar los detalles de las pruebas de rendimiento a la aplicación mediante tablas y gráficos.

Para los que se trazaron la siguientes **Tareas** a cumplir:

- ✓ Seleccionar toda la bibliografía existente en Internet y en otros sitios sobre la programación paralela.
- ✓ Identificar los principales Modelos de Programación Paralela que existen.
- ✓ Realizar estudio práctico de las Librerías PVM/MPI.
- ✓ Elaborar los Test ó Pruebas de Rendimiento.

Dado el objetivo principal se puede plantear la siguiente **Hipótesis**: Si las pruebas de rendimiento que se realizan a la aplicación son satisfactorias entonces dicha aplicación puede utilizarse eficazmente en la gestión y procesamiento de datos.

Para dar cumplimiento a las tareas propuestas se emplean los métodos científicos de la investigación tanto **Teóricos** debido a que permiten estudiar las características y modelos de la programación paralela que no se ven a simple vista y solo pueden ser conocidas por aquellos que la estudian, permitiendo de esta forma obtener la utilización de estos modelos en los diferentes casos que se puedan utilizar, determinando así cada uno de sus algoritmos, su utilización y de esta forma ver como se desarrollan en cada aplicación, como **Empíricos** que describen y explican las características del objeto de estudio.

De los métodos **Teóricos** se emplearon:

El **Histórico**: para analizar los elementos que fueron encontrados en la investigación de nuestro tema sobre la programación paralela, sus modelos y la forma en que trabajan en nuestra aplicación, permitiendo así ver su comportamiento en las diferentes etapas de trabajo, logrando también conocer sobre la historia de la programación paralela.

El **Lógico**: para el estudio histórico de la programación paralela, adquiriendo un conocimiento más profundo del tema. Estos métodos expresan en forma teórica la esencia del tema, explican la historia

de su desarrollo, permitiendo de esta forma unir la investigación realizada del tema con su concepción histórica. Dentro de este método se encuentran el método **Hipotético-Deductivo**: para a partir de una hipótesis y siguiendo reglas lógicas de deducción, llegar a nuevos conocimientos y predicciones, las que posteriormente son sometidas a verificaciones empíricas. El **Sistémico**: para estudiar el objeto mediante la determinación de sus componentes, así como la relación entre ellos que conforma una realidad como totalidad, o sea, no solo se estudia como dar solución al problema, si no cuales son las causas por las cuales se origino.

De ese modo se estudia como se aplica la programación paralela en los diferentes proyectos de nuestra universidad, en cuanto a la gestión y procesamiento de grandes volúmenes de datos, logrando que exista una relación con el medio en que se desarrolla la investigación del tema.

De los métodos **Empíricos** se utilizó:

La Observación: para la percepción planificada y dirigida a estudiar las características de la programación paralela. Esta observación puede ser: **Selectiva**: para precisar que parte del fenómeno se va a observar de acuerdo con el objetivo que se persigue con la investigación, y de **Medición**: para la medición del procedimiento que se realiza con el objetivo de obtener información numérica acerca de los proyectos de la UCI que trabajan con la Programación Paralela y de esta forma ver en cual de ellos se realizan pruebas de rendimiento para ver hasta que punto puede ser eficiente sus algoritmos en cada uno de los casos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

INTRODUCCIÓN

En este capitulo se introducirán aspectos básicos relativos a la programación paralela, incluyendo sus características generales, los modelos que están presente dentro de ella. Así como los principios básicos para realizar un análisis de rendimiento en un sistema paralelo.

1.1 INTRODUCCIÓN AL ANÁLISIS DE PROGRAMACIÓN PARALELA.

La programación paralela es una técnica de programación basada en la ejecución simultánea, bien sea en un mismo ordenador (con uno o varios procesadores) o en un clúster de ordenadores, en cuyo caso se denomina computación_distribuida. Al contrario que en la programación_concurrente, esta técnica enfatiza la verdadera simultaneidad en el tiempo de la ejecución de las tareas. (Bosque, 2007)

Los sistemas con multiprocesador y multicomputadores consiguen un aumento del rendimiento si se utilizan estas técnicas. En los sistemas monoprocesador el beneficio en rendimiento no es tan evidente, ya que la CPU es compartida por múltiples procesos en el tiempo, lo que se denomina multiplexación o multiprogramación. (Bosque, 2007)

El mayor problema de la computación paralela radica en la complejidad de sincronizar unas tareas con otras, ya sea mediante secciones críticas, semáforos o paso de mensajes, para garantizar la exclusión mutua en las zonas del código en las que sea necesario. (Bosque, 2007)

1.1.1 Paralelismo.

Se entiende por arquitectura en paralelo a un conjunto de procesadores interconectados capaces de cooperar en la solución de un problema. El procesamiento paralelo funciona bajo el principio de división del trabajo en subtareas y asignación de estas a distintos elementos computacionales para lograr su ejecución simultanea. Estos deben comunicarse entre si cuando es necesario llevar a cabo ciertas subtareas.

La palabra procesadores, no se refiere solo a los procesadores en su significado más estricto, dígase un procesador SPARC o Pentium IV, sino que puede acoger también a computadoras completas o redes de computadoras, por ejemplo una SUN Workstation o una red de SUN Workstation. El número de procesadores indica (en parte) la capacidad de procesamiento de una máquina paralela. Los

procesadores se identifican con números naturales consecutivos comenzando por 0. El conjunto de estos números será denotado por IND, siendo la cardinalidad de este conjunto (IND), la cantidad de procesadores.

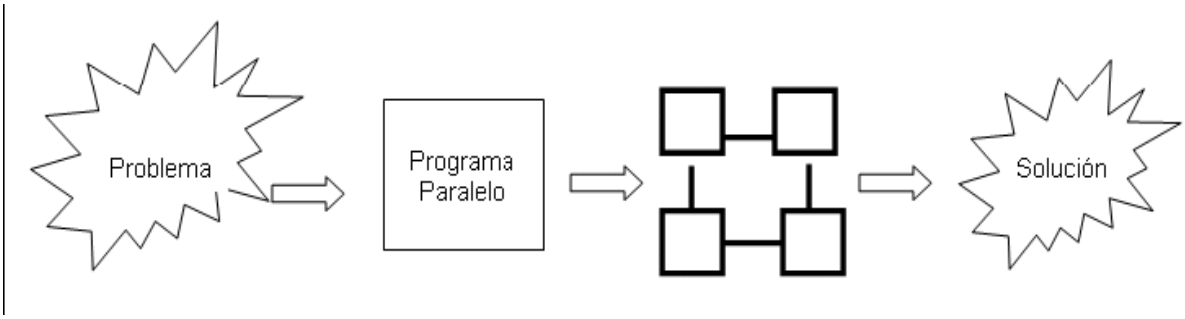


Fig.1 Arquitectura en Paralelo

La comunidad paralela surge como una necesidad de lograr un mayor poder computacional, a partir de los grandes avances de la tecnología y en la que se busca paralelizar como meta en si, el objetivo fundamental es obtener mayores rendimientos que permitan resolver problemas cada vez más grandes en tiempos de computación aceptables. La computación paralela logra acelerar la ejecución de un programa mediante su descomposición en fragmentos que pueden ejecutarse simultáneamente, cada uno en su propia unidad de procesamiento.

Idealmente al paralelizar un problema secuencial utilizando n procesadores, se multiplica por n la velocidad computacional de su ejecución en un solo procesador, sin embargo existen otros factores de índole computacional y físico que hacen que nunca se obtenga un escalado en el rendimiento que sea igual o superior al número de procesadores, estos son la comunicación entre los procesadores que son lentas en comparación con la velocidad de transmisión de datos y cómputo dentro de un procesador y en la práctica los problemas no pueden dividirse perfectamente en partes totalmente independientes y se necesita alguna interacción entre ellas lo que ocasiona una disminución de la velocidad. Sin embargo, el incremento de la velocidad se aproxima al número de procesadores que se integran en el sistema en determinados algoritmos con una paralelización intrínseca muy fuerte.

- ✓ **Arquitecturas paralelas.**

Las principales arquitecturas en las que se puede clasificar actualmente los sistemas paralelos son:

Arreglos de procesadores, también conocidas como maquinas MPP (Massively Parallel Procesing, + 1000 CPU).

Multiprocesadores (SMP).

Multicomputadoras de Memoria Distribuida (cantidad de CPU < = 1000).

Máquinas de memoria hibrida (compartida distribuida).

Clúster (a partir del 2000 aparece el termino Constellation).

El procesamiento paralelo se puede definir, en esencia, como la conjunción de dos o mas procesadores que computan, en forma concurrente, subtareas correspondientes a un problema más grande.

1.2 TAXONOMÍA BASADA EN EL ACCESO A LA MEMORIA

Dentro del procesamiento paralelo se pueden identificar modelos principales, 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.

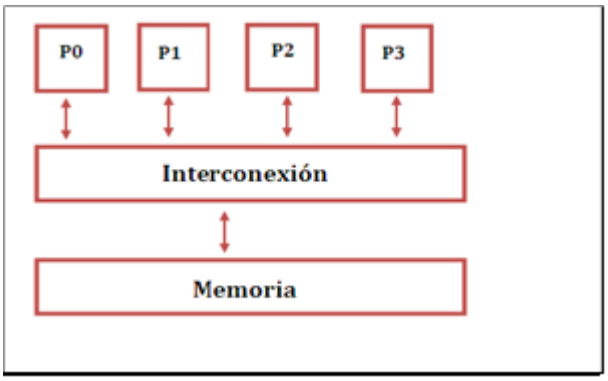


Fig.2 Modelo Máquinas de Memoria Compartida

- ✓ **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.

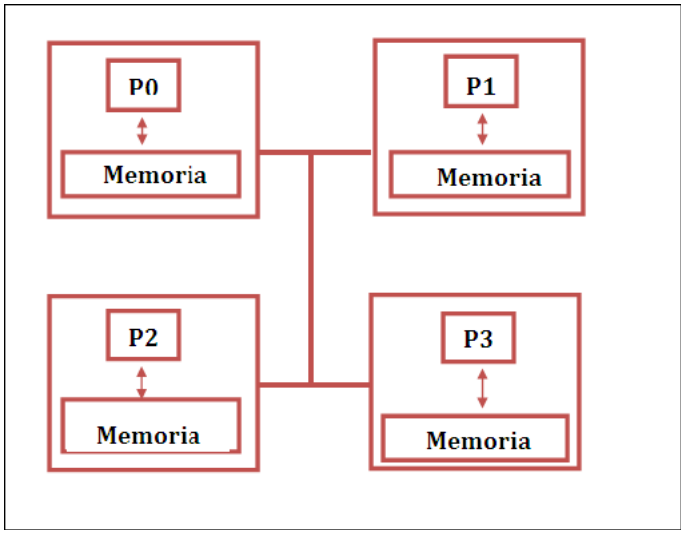


Fig.3 Modelo Máquinas de Memoria Distribuida

1.2.1 Clúster

El término clúster se aplica a un conjunto o conglomerado de computadores, construido utilizando componentes de hardware comunes y en la mayoría de los casos, software libre; los computadores se interconectan mediante alguna tecnología de red. El clúster puede estar conformado por nodos dedicados o por nodos no dedicados.

En un clúster con nodos dedicados, los nodos no disponen de teclado, mouse ni monitor y su uso está exclusivamente dedicado a realizar tareas relacionadas con el clister. Mientras que, en un clúster con nodos no dedicados, los nodos disponen de teclado, mouse y monitor y su uso no está exclusivamente dedicado a realizar tareas relacionadas con el clúster, el clúster hace uso de los ciclos de reloj que el usuario del computador no esta utilizando para realizar sus tareas.

Simplemente, un clúster es un grupo de múltiples computadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único computador, más potente que los comunes de escritorio.

Los computadores del clúster pueden tener, todos, la misma configuración de hardware y sistema operativo (clúster homogéneo), diferente rendimiento pero con arquitecturas y sistemas operativos similares (clúster semi-homogéneo), o tener diferente hardware y sistema operativo (clúster heterogéneo), lo que hace más fácil y económica su construcción.

Para que un clúster funcione como tal, no basta solo con conectar entre sí los computadores, sino que es necesario proveer un sistema de administración del clúster, el cual se encargue de interactuar con el usuario y los procesos que corren en él para optimizar su funcionamiento.

- ✓ **Beneficios de la Tecnología Clúster.**

Las aplicaciones paralelas escalables requieren: buen rendimiento, baja latencia, comunicaciones que dispongan de gran ancho de banda, redes escalables y acceso rápido a archivos. Un clúster puede satisfacer estos requerimientos usando los recursos que tiene asociados a él.

Los clúster ofrecen las siguientes características a un costo relativamente bajo:

- Alto Rendimiento (High Performance).
- Alta Disponibilidad (High Availability).
- Alta Eficiencia (High Throughput).
- Escalabilidad.

La tecnología clúster permite a las organizaciones incrementar su capacidad de procesamiento usando tecnología estándar, tanto en componentes de hardware como de software que pueden adquirirse a un costo relativamente bajo.

- ✓ **Clasificación de los Clúster.**

Los clúster pueden clasificarse en base a sus características. Se pueden tener clusters de alto rendimiento (HPC – High Performance Clusters), clusters de alta disponibilidad (HA – High Availability) o clusters de alta eficiencia (HT – High Throughput).

- High Performance

Son clusters en los cuales se ejecutan tareas que requieren de gran capacidad computacional. El llevar a cabo estas tareas puede comprometer los recursos del clúster por largos periodos de tiempo.

- High Availability

Son clusters cuyo objetivo de diseño es el de proveer disponibilidad y confiabilidad. Estos clusters tratan de brindar la máxima disponibilidad de los servicios que ofrecen. La confiabilidad se provee mediante software que detecta fallos y permite recuperarse frente a los mismos, mientras que en hardware se evita tener un único punto de fallos.

- High Throughput

Son clusters cuyo objetivo de diseño es el ejecutar la mayor cantidad de tareas en el menor.

- ✓ **Componentes de un Clusters.**

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

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, SCI.

Middleware (capa de abstracción entre el usuario y los sistemas operativos): El middleware es un software que generalmente actúa entre el sistema operativo y las aplicaciones con la finalidad de proveer: Una interfaz única 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. Existen diversos tipos de middleware, como por ejemplo: MOSIX, Condor, Open MOSIX, OpenSSI, entre otros.

Ambientes 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.

Aplicaciones (pueden ser paralelas o no)

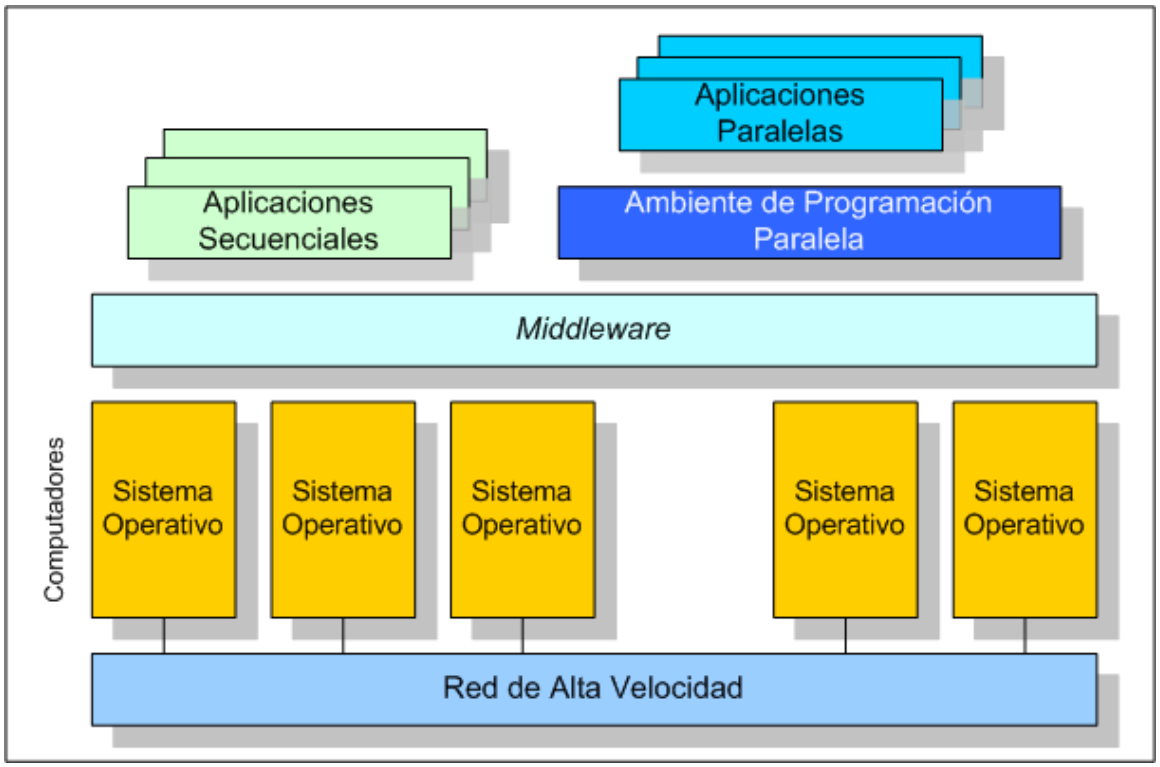


Fig.4 Componentes de un Clúster.

- ✓ **Sistema de Clúster Implementados.**

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. Cluster X posee 4 Terabytes de memoria RAM y 176 Terabytes de disco duro, su rendimiento es de 12.25 TFlops. Se lo conoce también como Terascale.

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.

- ✓ **Conjuntos de Clusters**

Programación Paralela: estos clúster están diseñados y optimizados para correr programas paralelos. En este caso, los programas tienen que ser hechos específicamente para funcionar en forma paralela.

Típicamente estos programas son modelos que requieren realizar gran cantidad de cálculos numéricos. La ventaja de programarlos de esta manera y correrlos en un clúster es que se reducen drásticamente los tiempos de proceso. En el caso de modelos meteorológicos usados para predecir el tiempo es obvia la necesidad de correrlos en tiempo mínimo.

Cuando se programa un modelo en una plataforma multiprocesadores (una máquina con más de un procesador), es necesario usar esquemas de programación paralela. Las bibliotecas (MPI) son las que permiten paralelización de tareas. En el caso de los clúster SCALI, portar programas hechos con bibliotecas MPI es directo gracias al uso de bibliotecas SCAMPI.

Son sumamente importantes los sistemas de interconexión utilizados entre los nodos en este tipo de clusters. SCI es un standard internacional (IEEE1596) para interconexiones de alta velocidad. Dolphin Interconnect Solutions fabrica una variedad de productos SCI, entre los cuales se encuentran las tarjetas adaptadoras SCI-PCI que SCALI utiliza en sus clusters. El performance de estas tarjetas es sorprendente: varios cientos de megabytes por segundo y tiempos de latencia inferiores al microsegundo en la capa SCI llegando a los límites del bus PCI (64bits/66Mhz).

Entre los principales usos de la Programación Paralela se destacan:

- Análisis Estructural
- Simulaciones de mecánica de fluidos
- Modelos genómicos
- Predicción meteorológica
- Estudio de cualquier fenómeno que pueda ser modelado matemáticamente.

Distribución de Procesos: este tipo de clúster tiene como finalidad distribuir los procesos entre los diferentes nodos que componen el clúster. Si un nodo deja de funcionar, los procesos que estaban corriendo en ese nodo, serán redistribuidos entre los nodos que estén en funcionamiento. En este caso es el sistema operativo o software adicional el que se dedica a balancear la carga entre los nodos.

La transferencia de información entre nodos es bastante baja en comparación con los clusters. El sistema utilizado para interconectar los nodos en este tipo de clúster es típicamente una red LAN de alta velocidad.

No es necesario modificar los programas para que puedan correr en este tipo de clusters.

- ✓ **Herramientas de los Clúster**

La operación de clusters requiere de un manejo adecuado de los recursos asociados. Los recursos del clúster deben ser administrados adecuadamente para que el administrador invierta la menor cantidad de tiempo en detectar, investigar y recuperar fallos de hardware y software, y de este modo definir posibles medidas de contingencia y tratar que el sistema esté libre de errores. A su vez, estos pasos permiten la adaptabilidad a los requerimientos y cambios constantes que se presentan en la manipulación de tecnologías clúster, en cuanto se refiere al hardware, software y al uso de ciertos patrones de diseño.

El administrador de un clúster debe tomar en cuenta algunos aspectos, una vez que se ha completado la instalación de los recursos básicos de hardware y software. Estos aspectos incluyen la configuración e instalación de un sistema de archivos universal, la configuración y administración de recursos mediante herramientas implementadas en software; el monitoreo de sus actividades y el registro de cada uno de los eventos generados por la ejecución de cálculos computacionales.

Varios de los sistemas más importantes para la instalación automática de clusters, incluyen herramientas de monitoreo, administración y registro de eventos mediante paquetes de distribución para sistemas Windows y Linux. Entre estos sistemas están OSCAR y Racks NPACI; ambos sistemas permiten el uso de herramientas de software que tienen propósitos específicos tales como:

- Definición y administración de nodos.
- Administración de colas por lotes (Bach Queue Management).
- Administración de recursos: grupos NIS (Network Information Service), cuotas de disco y CPU.
- Administración de servicios de resolución de nombres: DNS (Domain Name System para clusters).
- Registro de usuarios para clusters de dimensiones superiores a los 100 nodos.
- Monitoreo de carga.

La administración de clusters, implica tomar medidas preventivas y planificar tareas. La administración implica los siguientes aspectos: Registro de eventos, Monitoreo o medida del estado de los recursos del clúster, Recuperación ante fallos de hardware, software, incluyendo el sistema de archivos,

Administración del registro de usuarios y grupos de usuarios, de los servicios del clúster (accounting), Planificación de tareas y balanceo de carga.

- ✓ **Tipos de Clúster.**

Debido a la variedad de sistemas de clúster implementados y la utilización de los mismos, se cuentan actualmente con una gran cantidad de clúster, entre los que figuran:

Clúster de servicios: Es el Clúster que garantiza el concepto "Alta disponibilidad" del servicio. Se utiliza en entornos críticos, y su objetivo es dar servicio sin interrupción. Suelen ser Microsoft Clúster Services (MSCS), pero por supuesto también se puede dar clúster de servicios con Linux. Los servicios típicos son: Clúster de disco físico, clúster de compartición de carpetas, DHCP, WINS, servicio de impresión, servicio horario, entre otros.

Clúster de Supercomputación: En el entorno de Supercomputación, todas la máquinas (denominadas nodos), trabajan como una sola, realizando el cálculo entre todos. Es un modelo muy utilizado en centros de cálculo científico de alto rendimiento y universidades. Es un tipo de clúster flexible, se pueden mezclar tipos de nodos P4, Xeon 4 FSB 800Mhz, AMD Opteron Dual Core, Itanium 2, monoprocesadores, multiprocesadores. En este tipo de clusters, existen diferentes distribuciones y módulos como: Beowulf (PVM, MPI), Openmosix.

Normalmente, la comunicación entre nodos se realiza por Gbit Ethernet, pero hay veces que se necesitan más prestaciones y se recurre a redes de baja latencia, del tipo de Myrinet, SCI, o las nuevas redes basadas en PCI-Express Infiniband. Destacar de las redes Infiniband, que disponen de las mejores latencias, y un ancho de banda de hasta 10GB/s.

Granjas de render (Render Farm): En la actualidad, debido a la complejidad y nivel de detalle de los programas 3D, es necesarios gran cantidad de horas en una máquina para poder renderizar nuestros trabajos. En muchas ocasiones es necesario dejar las máquinas funcionando toda la noche para esperar los resultados, con lo que la entrega de los proyectos se retrasa enormemente.

Además de disponer de una gama profesional de Workstation, podemos montar una render farmer o "granja de render", para que los procesos de renderizado no sean interminables. Estas granjas de render normalmente constan de un nodo padre o de gestión, que es una Workstation de altas prestaciones con la que se trabaja, y unidas por red Gigabit ethernet, un clúster de servidores de render, hacia los cuales se migran los procesos, cuya única misión va a ser realizar renderizados. Disponemos de servidores Dual Xeon Woodcrest y Dual Opteron en 1U que empleamos

indiferentemente en función de que plataforma de más rendimiento para la aplicación especifica del cliente, contando con soluciones de hasta 4 procesadores Intel Xeon o AMD Opteron ambos de Doble Core en servidores de 1U. No olvidamos el resto de componentes que necesitan este tipo de sistemas, consiguiendo además de máximas prestaciones, el mínimo costo posible sin descuidar aspectos como el consumo y el espacio.

Los parámetros de los clusters se pueden apreciar en el ANEXO I.

1.3 PARALELISMO DE DATOS Y PARALELISMO DE CONTROL.

Al intentar dividir un problema se identifican dos tipos de paralelismo: el paralelismo de datos, y el paralelismo de control o funcional.

El paralelismo de datos se obtiene al asignar elementos de datos a varios procesadores, cada uno de los cuales realiza la misma operación simultáneamente sobre sus datos. Consiste en una secuencia simple de instrucciones o flujo de instrucciones cada una de las cuales se aplica a diferentes elementos de datos.

El paralelismo de control se logra realizando diferentes operaciones sobre varios conjuntos de datos simultáneamente, o sea, se refiere a la ejecución simultánea de diferentes flujos de instrucciones.

1.4 PRINCIPIOS PARA EL DISEÑO DE ALGORITMOS PARALELOS.

Un algoritmo secuencial es una secuencia de pasos básicos para resolver un problema dado, utilizando la computadora. Semejantemente, un algoritmo paralelo permite solucionar un problema dado usando procesadores múltiples. Sin embargo, especificar un algoritmo paralelo implica más que una simple especificación de pasos. En la práctica, especificar un algoritmo paralelo puede incluir alguno o todos de los siguientes puntos.

- Identificar las partes o porciones del trabajo que pueden ser realizadas de manera concurrente.
- Mapear las piezas concurrentes sobre múltiples procesos que corren en paralelo.
- Distribuir la entrada, salida y datos intermedios asociados al programa.
- Manejar el acceso a datos compartidos por múltiples procesadores.
- Sincronizar los procesadores en las diferentes fases de ejecución del programa paralelo.
- Dividiendo un cómputo en cómputos más pequeños y asignándolo a diferentes procesadores para su ejecución paralela son los dos pasos clave en el diseño de algoritmos paralelos.

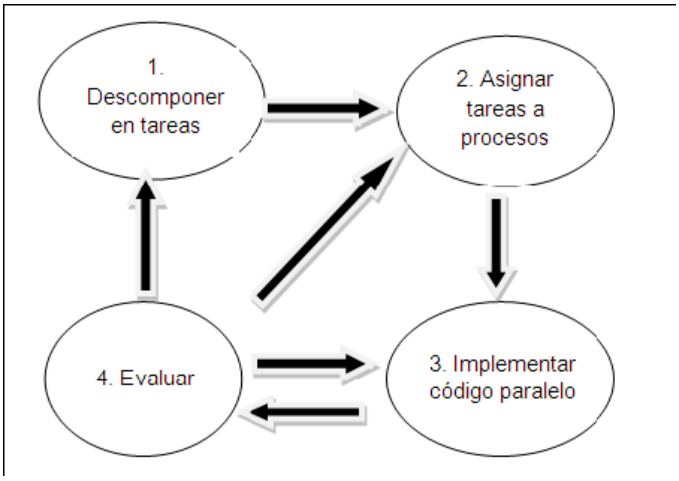


Fig.5 Diseño de Algoritmos Paralelas.

1.5 PARADIGMAS DE LA PROGRAMACIÓN PARALELA.

Existen muchos métodos de programación en cómputos paralelos, estos son:

Paralelismo de Datos: el paralelismo lo determina el particionamiento de los datos.

Paso de Mensajes: el usuario hace llamadas a bibliotecas para explícitamente compartir información entre los procesadores.

Memoria compartida: múltiples procesos comparten un espacio de memoria común.

Operación Remota o memoria: un proceso puede acceder a la memoria de otro proceso sin su participación.

Threads *(hebras, hilos*): un proceso teniendo múltiples trayectorias de ejecución.

Modelo Combinado: compuesto de dos o más de los mencionados anteriormente.

De estos métodos los más usados son el Paralelismo de Datos y el de Paso de Mensajes.

1.6 MODELOS DE LA PROGRAMACIÓN PARALELA.

Todo modelo de computación define la conducta de una máquina teórica. Su objetivo es facilitar el diseño y análisis de algoritmos que, en un amplio rango de arquitecturas, pueden ejecutarse con la eficiencia preestablecida. Un modelo de computación tiene siempre asociado un modelo de

programación, el cual brinda las herramientas necesarias para implementar la metodología propuesta por el modelo de computación. Dichos modelos se relacionan a continuación en la siguiente tabla:

Modelos de Computación	Modelos de programación asociados.
Especificación	CSP (Procesos Secuenciales Comunicantes). Unity.
Programación.	Occam. HPF (High Performance Fortran). GL (Generation Language).
Coste.	PRAM. BSP (Bulk-Synchronous Parallelism)
Arquitectónicos	Pasos de Mensajes. Memoria Virtual Global. SIMD (Una sola instrucción y múltiples datos). SPMD (Un solo programa y múltiples datos). PVM (Máquina Virtual Paralela).
Físicos	Arquitecturas con Memoria Distribuida. Arquitecturas con Memoria Compartida. Clúster de WorkStations

Tabla.1: Modelos de Programación Paralela.

1.6.1 Modelo BSP.

Resulta una propuesta promisoria para establecer los enlaces necesarios entre el hardware y el software paralelos. Los algoritmos programados basados en el modelo BSP, son portables a cualquier

arquitectura paralela. El modelo BSP propone además un mecanismo para determinar la complejidad de los algoritmos paralelos. Este mecanismo, al tener en cuenta detalles como la comunicación y la sincronización, permite hallar una complejidad que caracteriza de manera más adecuada el desempeño real del algoritmo.

Modelo de máquina paralela abstracta denominado Bulk-Synchronous Parallelism (BSP).El modelo BSP propone además un mecanismo para determinar la complejidad de los algoritmos paralelos.

El modelo BSP consta de:

- Un conjunto de pares Procesador-Memoria.
- Un Mecanismo de Comunicación, que permite la transmisión de mensajes entre los procesadores y una unidad de sincronización global, la cual se encarga de la sincronización.

Además hace uso de 3 parámetros, para caracterizar el rendimiento de un algoritmo: el número de pares procesador-memoria, el tiempo requerido para una sincronización global y el tiempo necesario para completar el envío/recibo de a lo sumo una palabra, por cada procesador.

Un algoritmo BSP (se le llamará así a los algoritmos basados en el modelo BSP), se ejecuta como una secuencia de superpasos, los cuales se dividen en dos fases. Cada superpaso viene seguido de una sincronización global.

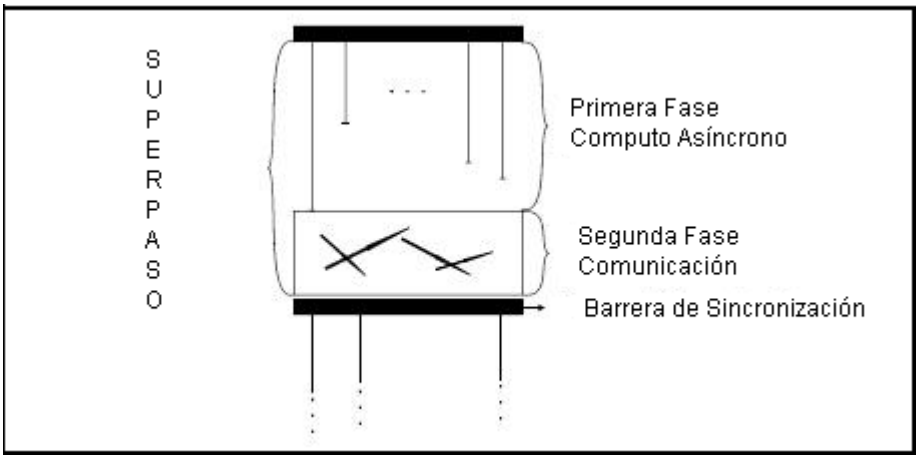


Fig. 6 Fases de un Superpaso de Ejecución BSP

1.6.2 PVM.

(Díaz, 1999)

La Máquina Virtual Paralela (conocida como PVM por sus siglas en inglés de *Parallel Virtual Machine*) es una librería para el cómputo paralelo en un sistema distribuido de computadoras. Está diseñado para permitir que una red de computadoras heterogénea comparta sus recursos de cómputo (como el Procesador y la Memoria RAM) con el fin de aprovechar esto para disminuir el tiempo de ejecución de un programa al distribuir la carga de trabajo en varias computadoras.

1.6.3 MPI

(Alonso, 1997)

MPI (Message Passing Interface): es una especificación para Paso de Mensajes. Constituye la primera librería de Paso de Mensajes estándar y portable. Acabado y publicado en mayo 1994. Actualizado en junio 1995. Ofrece Estandarización, Portabilidad, Buenas Prestaciones, Amplia funcionalidad, Implementaciones libres (mpich, lam).

MPI es una librería de funciones que puede ser usada en programas C, Fortran y C++. Como su propio nombre indica, MPI pretende explotar la existencia de múltiples procesos mediante el modelo de paso de mensajes.

MPI fue desarrollado entre los años 1993 y 1994 por un grupo de investigadores y auspiciado por diferentes empresas y universidades. Actualmente es el estándar más extendido en la programación paralela orientada al paso de mensajes.

- ✓ **Objetivos de MPI.**

Entre sus objetivos fundamentales caben destacar: definir un entorno de programación único que garantice la portabilidad de las aplicaciones paralelas, definir totalmente el interfaz de programación, sin especificar cómo debe ser la implementación del mismo, ofrecer implementaciones de calidad, de dominio público, para favorecer la extensión del estándar, convencer a los fabricantes de computadores paralelos para que ofrezcan versiones de MPI optimizadas para sus máquinas (lo que ya han hecho fabricantes como IBM y Silicon Graphics).

MPI esta diseñado para la implementación de aplicaciones paralelas de gran complejidad debido al hecho que soporta el manejo de topologías y ofrece un conjunto de funciones de comunicación muy potente y flexible. Además permite de forma segura, a combinación de aplicaciones paralelas preexistentes para la creación de aplicaciones paralelas más complejas. Esto se consigue mediante la

creación de distintos ámbitos o universos de comunicación que permiten excluir los mensajes pertenecientes a diferentes partes de una aplicación paralela. (Alonso, enero 1997)

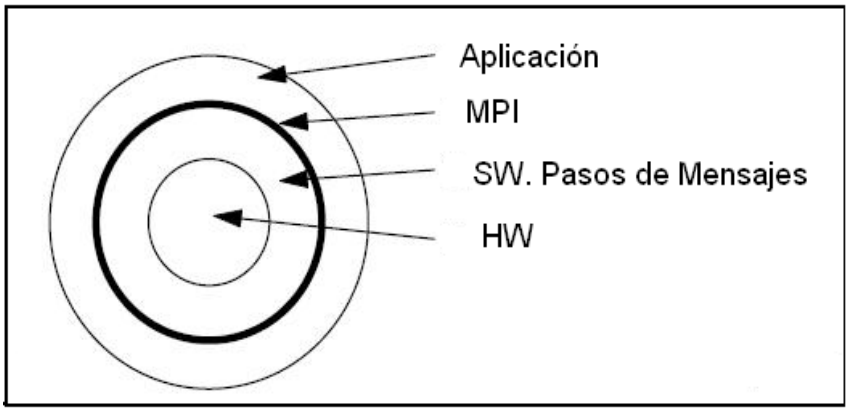


Fig. 7 Ubicación de MPI en el proceso de Programación de Aplicación Paralela.

1.6.4 Threads/Procesos.

Los Threads son la forma más eficiente de paralelización en máquinas de memoria compartida. Son lo que se denominan procesos ligeros, es decir, no son procesos independientes sino hilos de ejecución dentro de un proceso que comparten el mismo espacio de direcciones (datos y código) del proceso padre, pero tienen distintos entornos de ejecución. Su creación y destrucción es muy sencilla a nivel del kernel, y al compartir el espacio de direcciones del proceso los datos son compartidos entre todos los hilos de forma automática. Los principales elementos que lo componen son: dirección de instrucción pila y registro de datos.

Los procesos por su parte están conformados por: Threads, además de espacio de direcciones, descriptores de ficheros.

1.6.5 PRAM.

El Modelo P-RAM (Fortune & Willis, 1978) es una extensión paralela del modelo RAM, que constituye la abstracción del modelo físico de multiprocesador con memoria compartida. Contando con las siguientes características:

- Procesadores RAM convenientemente modificados.
- Memoria común ilimitada.
- Cada procesador puede acceder a cualquier posición de memoria en un ciclo de reloj [O (1)] .

- Cada procesador tiene un índice por el que se le reconoce y se puede referenciar.
- Todos los procesadores ejecutan el mismo programa (SPMD).
- Los códigos pueden particularizarse en función del índice identificador del procesador.
- Todos los procesadores ejecutan una instrucción por ciclo de reloj de forma síncrona (no necesariamente la misma).
- Los datos están en memoria al empezar una ejecución y se devuelven a memoria al acabar.
- Cualquier comunicación entre procesadores se hace a través de la memoria leyendo/escribiendo en las variables adecuadas.
- No existe memoria local ni posibilidad de enviar mensajes de un procesador a otro.
- El coste de ejecutar un algoritmo es el número de ciclos de reloj que requiere su ejecución.

1.6.6 D-RAM

El Modelo D-RAM (Dynamic Random Access Memory) creado en 1990, constituye una abstracción del modelo físico de multiprocesador con memoria distribuida. Entre sus principales características figuran:

- Procesadores RAM convenientemente modificados.
- Cada uno dispone de su propia memoria local con su propio espacio de direcciones.
- No existe memoria común o no se puede usar para intercambiar información.
- Cada procesador tiene un índice por el que se le reconoce y se puede referenciar.
- Todos los procesadores ejecutan el mismo programa (SPMD).
- Los códigos pueden particularizarse en función del índice identificador del procesador.
- Los datos están en memoria local al empezar una ejecución y se devuelven a una memoria local al acabar.
- Cualquier comunicación entre procesadores se hace a través de paso de mensajes.
- Cada procesador dispone de un buffer donde almacenar un mensaje.
- Existen primitivas de comunicación:
- Send (v, i) : Escribe el mensaje almacenado en la variable v en el buffer del procesador Pi.

- Receive (v): Almacena el mensaje que haya en el buffer en la variable v y vacía el buffer.
- Modelo DCM (Directed connected machine): El coste de enviar un mensaje de tamaño n es de la forma: nt+b, siendo b el tiempo de latencia y t el tiempo de envío de una palabra.
- Modelo SCM (Sparse connected machine): El coste de enviar un mensaje de tamaño n entre nodos conectados es de la forma: nt+b, siendo b el tiempo de latencia y t el tiempo de envío de una palabra.
- Pueden existir mecanismos de sincronización.
- El coste de ejecutar un algoritmo es el número de ciclos de reloj que requiere su ejecución más el tiempo de comunicación.

1.7 PRINCIPIOS BÁSICOS PARA REALIZAR UN ANÁLISIS DE RENDIMIENTO EN UN SISTEMA PARALELO.

Análisis de Rendimiento en un Sistema Paralelo es la combinación de medida, interpretación y representación de la velocidad y capacidad de un sistema computacional. Su importancia viene dada por la necesidad de definir buenas técnicas de medida, precisas y reproducibles, que perturben mínimamente el sistema, necesidad de interpretar los datos medidos usando las técnicas estadísticas apropiadas, necesidad de comunicar de forma clara y consistente los resultados de los análisis.

1.7.1 Objetivos del Análisis del Rendimiento en un Sistema Paralelo.

Comparar Alternativas: Disponer de información cuantitativa que permita la selección de la configuración más adecuada de un sistema computacional.

Impacto de una Característica: Determinar cómo influye en el rendimiento el añadido o la eliminación de una determinada característica del sistema.

Afinar el Sistema: Encontrar los valores de un conjunto de parámetros que produzca el mejor rendimiento global del sistema.

Establecer Expectativas: Indicar cuáles son las expectativas más acertadas sobre la capacidad que se puede esperar de un sistema.

Depuración: Determinar los problemas de rendimiento de una cierta aplicación.

1.7.2 Técnicas utilizadas para el Análisis del Rendimiento en un Sistema Paralelo.

Existen varias técnicas para el Análisis de Rendimiento de un Sistema Paralelo entre las que se deben mencionar por considerarse de las más utilizadas se encuentran:

Medida: Medidas reales suelen producir los mejores resultados, pues no hay simplificaciones que les resten credibilidad.

Simulación: Programa que modela ciertas características importantes del sistema, facilitando el cambio de diversos parámetros.

Modelado Analítico: Descripción matemática del sistema, que puede darnos pistas sobre el comportamiento global del sistema.

1.7.3 Cuestiones básicas para llevar a cabo el Análisis de Rendimiento.

- ¿Cómo capturar características de un sistema para analizar su rendimiento?
- ¿Qué atributos de rendimiento deben considerarse?
- ¿Cómo se cuantifica y analiza la escalabilidad de un sistema?
- Otros parámetros de rendimiento que pueden interesar

Métrica	Significado	Unidad
Velocidad de procesamiento	Número de operaciones por unidad de tiempo (segundo)	MIPS, GIPS, Mflop/s Gflop/s, …
Productividad (*throughput*)	Número de procesos ejecutados por unidad de tiempo (segundo)	Procesos/s
Utilización	Relación entre la velocidad de procesamiento real y la velocidad pico	Porcentaje
Relación rendimiento/coste	Relación entre el rendimiento medido según alguna métrica de interés y el coste	(Unidad métrica)/€

Fig. 8 Parámetros de Rendimiento.

1.8 FASES DE PROGRAMACIÓN PARALELA.

(Bosque, 2007)

La Programación Paralela consta de cuatro fases, ellas son:

Descomposición Funcional: Identificar las funciones que deben realizar la aplicación y las relaciones entre ellas.

Partición: Distribución de las funciones en procesos y esquema de comunicación entre procesos, con el objetivo de maximizar el grado de paralelismo y minimizar la comunicación entre procesos.

Localización: Los procesos se asignan a procesadores del sistema, con el fin de ajustar los patrones de comunicación a la topología del sistema.

Escalado: Determina el tamaño óptimo del sistema en función de algún parámetro de entrada.

1.9 SITUACIÓN ACTUAL DE LA PROGRAMACIÓN PARALELA EN EL MUNDO.

En la actualidad existen muchos lenguajes de programación para especificar programas concurrentes y en paralelo, que muy frecuentemente no se acoplan a las aplicaciones ya existentes desarrolladas en lenguajes de programación de uso mas común, como es el C++. Como resultado, han surgido grupos de investigación que buscan la implementación de lenguajes capaces de especificar programas en paralelo y concurrentes, en lenguajes ya existentes, que aprovechen las características actuales de las computadoras. En el caso de la programación orientada a objetos a pesar de hacer más fácil el diseño y desarrollo de software, por proporcionar abstracciones de alto nivel, no proporcionan todas las facilidades necesarias para soportar la Programación Concurrente. A pesar de lo planteado anteriormente existen varias motivaciones para el uso de la Programación Concurrente entre ellas: reducir el tiempo de ejecución, incrementar la tolerancia a fallas y explotar el paralelismo explícito e inherente de ciertas aplicaciones, entre otras. Por lo que se ha puesto especial atención en la programación concurrente entre la comunidad de ciencias de la computación. Donde un programa concurrente especifica dos o más procesos que cooperan para llevar a cabo una tarea. Cada proceso es un programa secuencial que ejecuta una secuencia de instrucciones; los procesos están sincronizados y cooperan entre si comunicándose. Mientras que un programa paralelo ejecuta estos procesos simultáneamente sobre múltiples procesadores, un programa distribuido es un programa concurrente en el cual los procesos se ejecutan sobre computadoras diferentes y se comunican entre si a través de una red. Así, la Programación Concurrente engloba a la Programación Paralela y Distribuida.

Durante las últimas décadas, la computación de altas prestaciones[1] se ha utilizado para resolver los más complejos problemas de diversos campos de la ciencia y el mundo económico. Cuanto mas ha

[1]Engloba la Programación Concurrente compuesta por la Programación Paralela y Distribuida.

evolucionado el hardware y el software orientado a los sistemas paralelos, mayores han sido los problemas que se han conseguido abordar, y se han abiertos las puertas a problemas hasta ahora intratables.

- ✓ **Consecuencias de la utilización de la Computación Paralela.**

El cambio a la Computación Paralela está revolucionando los paradigmas de la programación lineal convencional y generando un renacimiento de las herramientas de desarrollo, nuevos conceptos de programación, nuevos modelos de multitarea y numerosas oportunidades para que los desarrolladores de soluciones y los arquitectos de sistemas innoven y creen aplicaciones que marcan tendencias. Las empresas pueden esperar un futuro en el que el desempeño con consumo eficaz de energía y la preparación para el futuro de las aplicaciones impulsan la innovación, prolongan la vida útil de los productos y ofrecen ventajas competitivas a aquellos que deciden adoptar la tecnología hoy.

La Computación Paralela y/o distribuida trae consigo además implicaciones económicas y sociales importantes. Sus avances se reflejan a menudo en la competición industrial para producir mejores productos, y en algunos campos científicos como biología, química y medicina por el bien social común en la mejor comprensión de mecanismos de enfermedades, de drogas y medicamentos. También, se ha iniciado el camino para grandes hitos futuros como la exploración del conocimiento de los mecanismos del ADN humano, el análisis de la estructura de proteínas y otros campos como la predicción de desastres naturales como terremotos, inundaciones y tsunamis. La situación actual y las posibles mejoras en el cómputo de altas prestaciones nos abren toda una nueva serie de caminos para la ciencia y la economía.

- ✓ **Aplicaciones de la Programación Paralela.**

La Programación Paralela, como ya quedo dicho, se aplica en una amplia gama de ramas científicas entre las que se encuentran las matemáticas, en este caso para resolver problemas complejos como es el caso de la Multiplicación de Matrices. El experimento Grid”5000, que se describe a continuación, fue realizado en Francia y muestra un ejemplo de lo anteriormente planteado, el mismo representó una nueva forma de computación distribuida, para su realización se utilizaron recursos heterogéneos, e implico la utilización de diez laboratorios.

Los elementos que componen Grid”5000 son: Grid y Grid Computing.

GRID: Infraestructura que permite la integración y el uso colectivo de ordenadores de alto rendimiento, redes y bases de datos que son propiedad y están administrados por diferentes instituciones.

GRID COMPUTING: Tecnología innovadora que permite utilizar de forma coordinada todo tipo de recursos que no están sujetos a un control centralizado.

Teniendo como objetivo la construcción de una plataforma que permitiera a los desarrolladores de la comunidad (registrados), validar los distintos niveles de software creados para la puesta en marcha de las tecnologías grid – grid computing. Con el mismo se intentó conseguir:

- Rapidez de cálculo y capacidad de almacenamiento.
- Utilización de red jerárquica de máquinas.
- Permitir al usuario de introducir sus aplicaciones. (Pozuelo)

1.9.1 Situación en Cuba.

Cuba por su parte cuenta ya con algunos avances con la utilización de la Programación Paralela entre estos se destaca:

El primer Clúster hecho en la Universidad Central de las Villas, en dicha institución se investiga diferentes temas de algoritmos paralelos aplicados en la Matemática, Física y Ciencias de la Computación.

La Universidad de la Habana, específicamente en la Facultad de Matemáticas también se ha empezado a trabajar en el empleo de la Programación Paralela para resolver problemas matemáticos complejos.

Por su parte la Universidad de las Ciencias Informáticas, cuenta en la Facultad 6 con un proyecto llamado BioGrid, este proyecto consta con un Clúster Java que permite el cálculo de problemas biológicos, hacer búsqueda de fármacos y se utiliza también en la Minería de datos. Otro ejemplo dentro de esta misma Universidad es proyecto denominado Equipo de Investigación de Procesamiento y Análisis de Datos (EIPAD) realizado en el Centro UCIFAR, este proyecto utiliza las potencialidades del clúster, realizado mediante la Programación Paralela para optimizar tiempos de ejecución y algoritmos, además que es aplicada también en la Minería de Datos, análisis extensivo e intensivo de datos y en sistemas de Toma de Decisiones.

1.10 CONCLUSIONES

En este capítulo se trató de abordar lo más profundo posible, conceptos que no se tenían claros y que necesitaban ser plasmados para lograr entender qué es la programación paralela y porqué se utiliza en

el procesamiento de grandes volúmenes de datos y en otras ramas de la economía, en fin se ha abordado algunos temas fundamentales que se deben conocer para lograr un estudio completo del presente trabajo de Diploma y que sirva de base para lograr realizar otros contenidos plasmados en dicho trabajo.

CAPÍTULO 2: DESCRIPCIÓN DEL PROBLEMA A RESOLVER.

Fundamentación Teórica de los Casos de Prueba.

INTRODUCCIÓN

Este capitulo se enmarca en la descripción del problema existente ha resolver, problema presentado por el Centro UCIFAR, donde se hará un bosquejo de las características de dicho Centro. Se presenta como solución la utilización de Casos de Prueba, por lo que se hace necesario realizar de forma general la fundamentación teórica de cómo realizar dichos Casos de Pruebas y así dar lugar al empleo de las mismas.

2.1 NECESIDAD DE LA UTILIZACIÓN DE TECNOLOGÍAS Y REDES DE COMUNICACIÓN.

Actualmente las personas dependen de la tecnología que está a su alcance, esta contribuye a hacerles la vida más simple, ayudando a aumentar la fiabilidad y efectividad para realizar sus tareas. De aquí la necesidad de las instituciones de aplicaciones que le aporten eficiencia y rapidez para así lograr un aumento en la productividad y calidad de sus productos.

En este marco cabe destacar el papel importante que juegan las redes de comunicación, las cuales están constituidas por equipos interconectados mediante uno o varios medios de transmisión, posibilitando un contacto estrecho entre las personas, debido a que si una institución está conectada a una red nadie esta lejos de nadie, permitiendo la utilización de recursos a la misma vez y el contacto con personas de otras instituciones. La clasificación y caracterización de las redes de computadoras se puede realizar desde diferentes puntos de vista, siendo el más tradicional el del tamaño.

2.2 DESCRIPCIÓN DEL PROBLEMA A RESOLVER EN EL CENTRO UCIFAR.

El Centro UCIFAR es un Centro de Desarrollo de Software del Ministerio de la Fuerzas Armadas Revolucionarias (MINFAR), donde se desarrollan soluciones informáticas de control, análisis y sistemas de Recursos de las FAR. Entre dichas soluciones informáticas se encuentra la realizada por el proyecto EIPAD[2] para el Procesamiento y Análisis de Datos.

[2] Equipo de Investigación de Procesamiento y Análisis de Datos.

Considerando que este proyecto cuenta con una Base de Datos Extensa[3], realizada en el Sistema Gestor de Base de Datos Postgre SQL el cual tiene las siguientes características: servidor de base de dato relacional orientado a objeto de software libre, permite una alta concurrencia y amplia variedad de tipos nativos, se decidió prestar especial atención al Módulo de Diagonalización de Matrices dentro de este proyecto por representar uno de los casos extremos donde el procesamiento de datos es extenso y complejo, en el cual se utilizó la Programación Paralela con el fin de optimizar esta solución informática, y que sirva de base para optimizar otras.

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 cuenta este Centro.

Una vez obtenida la aplicación del Módulo Diagonalización de Matrices, se necesita de un **Método** a seguir con el fin de medir que lo que se realizó es realmente óptimo.

2.3 Topología de Red del Centro UCIFAR.

La Red del Centro UCIFAR 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.

2.3.1 Análisis y Estudio de la Organización de la Red.

El Centro UCIFAR cuenta con un Nodo Central, dos Switch: uno de Capa 2 y otro de Capa 3, entre ellos hay un Firewall, el Switch de Capa 3 es el encargado de todos los laboratorios con que cuenta el Centro.

Switch de capa 2	Switch de capa 3
• Equivalentes a los bridges multipuertos. • Baja latencia y alto rendimiento.	• Combinación de la funcionalidad de los switch capa 2 y de las características de los routers.

[3] Contiene más de un millón de datos a procesar.

• Segmentar la red en dominios de colisión por puertos y dominios de broadcast. • Entrega de tráfico en base a dirección MAC.	• Alto rendimiento. • Entrega trafico basado en direcciones IP (cuando enruta la primera vez) y direcciones MAC (cuando conmuta). • Por ahora la mayoría soporta IP(algunos también IPX)

Tabla. 2 Resumen de algunas de las características más relevantes de los Switching.

La red de este Centro es una Red LAN, figurando entre sus principales características:

- Se expande en un área relativamente pequeña, se encuentran dentro de una edificación.
- Conformada por un número de 330 computadoras, las cuales se conectan entre sí por fibra óptica y cable UTP.
- Los nodos[4] que conforman esta red pueden ser máquinas (PC´s) que cuentan con su propio CPU[5], disco duro y software y tienen la capacidad de conectarse a la red en un momento dado; o pueden ser PC´s sin CPU o disco duro y son llamadas "terminales tontas", las cuales tienen que estar conectadas a la red para su funcionamiento.
- Capaz de transmitir datos a velocidades muy rápidas, algunas inclusive más rápido que por línea telefónica; pero las distancias son limitadas.
- Cuenta con cuatro servidores, encargados de llevar el control de la red: Servicios web, Bases de Datos, Proxy, DNS.

Servidores Web y DNS: Estos servidores de un modo muy concreto permiten la comunicación entre los usuarios a través de la red, la transferencia rápida de documentos, ficheros y el acceso a un

[4] Estaciones de trabajo.

[5] Procesador de datos.

conjunto de páginas WEB que ya conforman parte de los servicios que brinda la Intranet[6], de manera que esté creado un sistema de acceso multiusuario que incorpora en sí sistemas de búsqueda de información en bibliotecas, archivos, guías telefónicas, etc.

Servidor de Base de Datos: Ejecuta un sistema gestor de base de datos (DBMS, Database Management System) cliente-servidor. Las estaciones de trabajo que actúan como clientes, pueden enviar peticiones al servidor por la red y luego este responde. Las estaciones de trabajo clientes manejan la presentación de los datos e interactúan con los usuarios mientras el servidor realiza las operaciones más duras tales como ordenamiento, indexación y distribución de datos para los usuarios. Su utilización reduce la probabilidad de corrupción de la información, siendo fácil de mantener.

Servidor Proxy: Permite que varios ordenadores conectados a una misma red local puedan compartir un mismo acceso a Internet o conexión de manera simultánea. El proxy dará servicio a todos los ordenadores de su red local, siéndole indiferente el Sistema Operativo que estas tengan instalado. Esto es posible debido al protocolo TCP/IP[7].

2.3.2 Proyección de la Red

Dicho centro cuanta con 9 laboratorios de producción, donde hay una cantidad de 330 pc's, como ya se había dicho anteriormente, todas distribuidas en los mismos. Estas PC's son:

Pentium IV: Poseen 2 CPU de 3GHz., con una capacidad de 512MB de Memoria RAM y 1GB y 160 GB de Disco Duro.

Hanell y Asus: También son llamadas Clientes Ligeros estos no son mas que máquinas poco potentes, sin Disco Duro, sin Floppy Disc, equipo sin CD Rom, con la funcionalidad de un equipo de escritorio, Sistema Operativo en firmware[8]. Las cuales no almacenan las aplicaciones, estas son suministradas por un servidor de red.

---

[6] Sitio Central de la Universidad de Ciencias Informáticas.

[7] Protocolo de Control de Transmisión que hace posible los servicios Telnet, FTP, EMAIL entre computadoras de una misma red.

[8] Programa de ordenador que se inserta en un dispositivo de hardware, por ejemplo, un microcontrolador el firmware se encuentra entre el hardware y el software. Al igual que el software, es un programa que es ejecutado por una computadora.

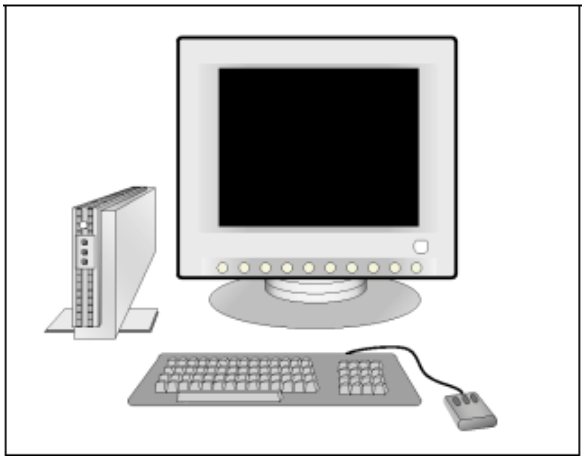


Fig. 9 Clientes Ligeros.

La tabla que se muestra a continuación representa la distribución de las PC´s en los 9 laboratorios del Centro UCIFAR.

Laboratorios	**Cantidad de Pc's**
1	27
2	35
3	67
4	30
5	25
6	43
7	39+15Hanell+1Asus en total 55.
8	35
9	1+ otras pc que todavía no están en uso.

Tabla. 3 Distribución de PC´s en el Centro UCIFAR.

2.3.3 Simulación de la Red del Centro UCIFAR

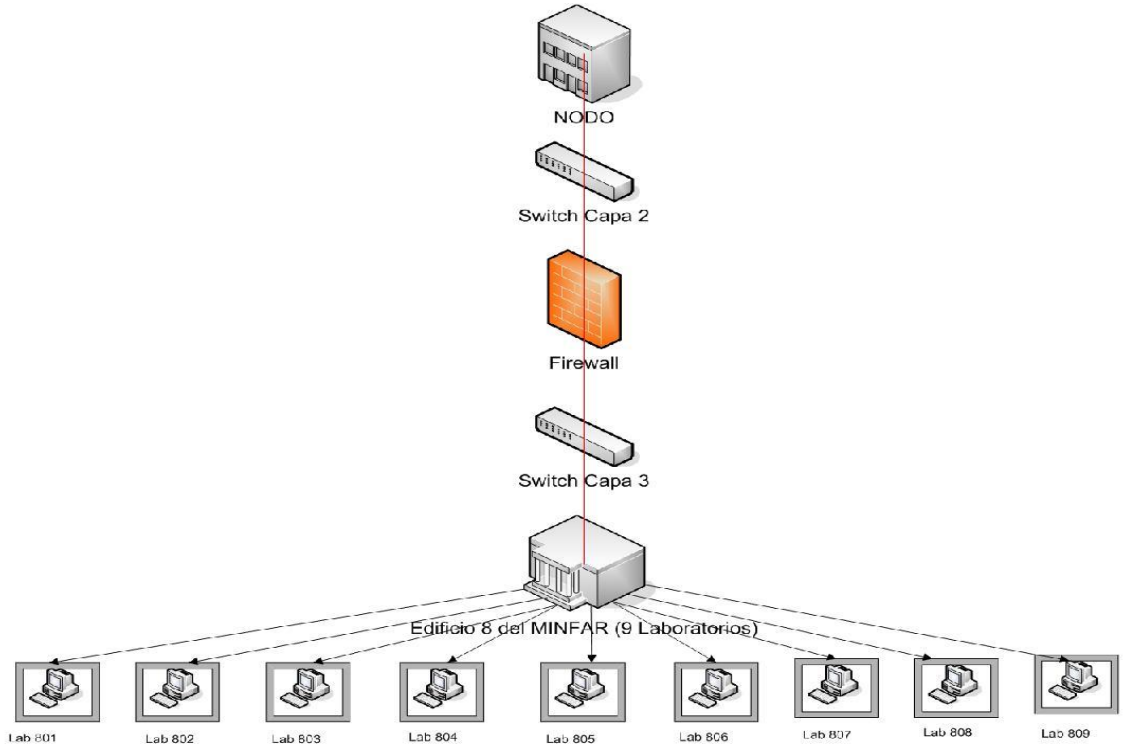


Fig. 10 Modelo de Red General del Centro UCIFAR.

La figura anterior muestra una simulación de la red del Centro UCIFAR, donde cabe destacar que entre las PC´s de los nueve laboratorios hay un Ancho de Banda[9] de un 1GB, mientras que entre los Switch de cada laboratorio con las pc's es de 100Mbps. Utilizándose como medio de conexión entre el Nodo Central, el Switch de Capa 2, Firewall, Switch de Capa 3 una Fibra Óptica.

Se realizó el Modelo de Red de uno de los laboratorios del Edificio 8 del Centro UCIFAR porque todos los demás cumplen las mismas condiciones solo cambia la cantidad de cubículos de dichos laboratorios y la cantidad de PC's, donde el medio de conexión entre las PC´s de los laboratorios 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 cada laboratorio.

[9] Numero de datos que se pueden comunicar por segundo.

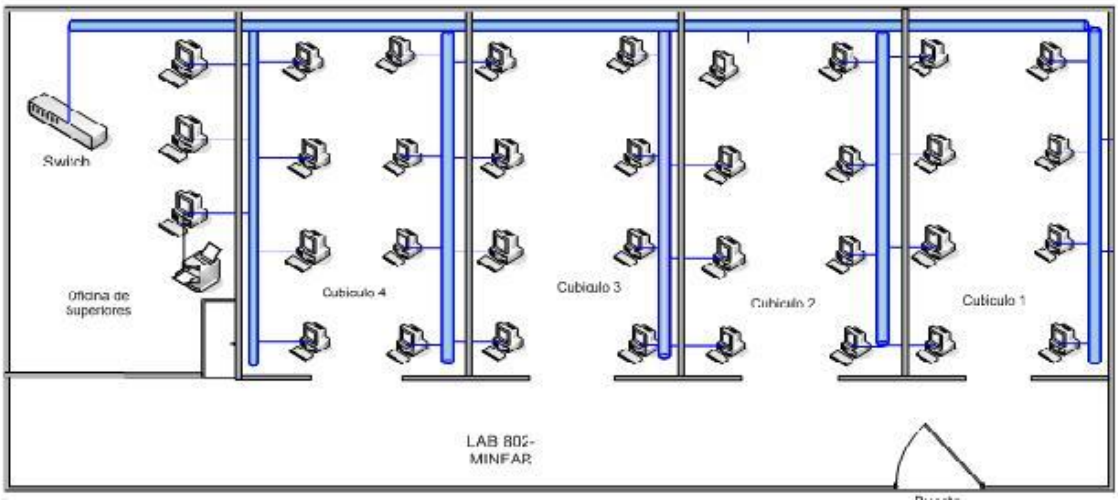


Fig.11 Modelación de Red de uno de los laboratorios del Centro UCIFAR

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 +125C 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.

2.4 CASOS DE PRUEBA.

A partir de la utilización de la Programación Paralela en el módulo Diagonalización de Matrices, surge la necesidad de conocer el tiempo mínimo que se puede obtener en el procesamiento de una cantidad de información. Con el fin de comprobar que el rendimiento alcanzado, para que dicha información sea procesada lo más eficaz y rápido posible, es óptimo. Para lo que se definirán Casos de Prueba, como **Método** que permita realizar lo anteriormente planteado.

2.4.1 ¿Qué es un Caso de Prueba?

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.

2.4.2 ¿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.

2.4.3 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.

2.5 CONCLUSIONES

A partir de la situación presentada por el proyecto el proyecto EIPAD[10] para el Procesamiento y Análisis de Datos, y especialmente por el Módulo Diagonalización de Matrices que se realiza en el Centro UCIFAR, donde se utiliza la Programación Paralela en aras de optimizar el tiempo de ejecución del procesamiento extenso de datos, se hace necesario la utilización de algún método que demuestre que lo realizado cumplió con su principal objetivo. Dando lugar a la utilización de Casos de Prueba para comprobar el rendimiento alcanzado.

[10] Equipo de Investigación de Procesamiento y Análisis de Datos.

CAPITULO 3: MODELACIÓN Y RESULTADOS

INTRODUCCIÓN

A partir de las bases creadas en el Capítulo anterior, en el presente capítulo se mostrarán posibles propuestas de Casos de Pruebas a realizar, donde se visualizarán los datos o información obtenida a través de estas pruebas en gráficas, logrando así un mejor entendimiento y para de esta manera llegar a conclusiones.

3.1 PROPUESTA DE CASOS DE PRUEBA.

Teniendo en cuenta la descripción del problema que presenta el Centro UCIFAR, a continuación se muestran una serie de propuestas de Casos de Prueba, que medirán el menor tiempo en que pueden ser procesados una determinada cantidad de datos, tomándose como base la cantidad de recursos utilizados para el procesamiento de los mismos.

3.1.1 Posibles Casos de Pruebas, a partir de las Condiciones del Centro UCIFAR.

- ✓ **Caso de Prueba 1**

En este 1er Caso de Prueba se medirán los parámetros: Ancho de Banda de la Red contra la Cantidad de Información, contra la Capacidad de las PC's. El cual se realiza con el fin de saber cuál es la mayor cantidad de información que se podrá enviar de manera satisfactoria. Es decir hasta que punto puede el ancho de banda permitir que se trasmita la mayor cantidad de información en el tiempo más rápido posible, permitiendo conocer cuando se puede aumentar o disminuir la cantidad de información a enviar.

Cabe destacar que resulta de gran importancia medir la manera en que puede influir el ancho de banda para la transmisión de cierta cantidad de información, teniendo en cuenta que el tamaño de la información es grande y que necesita de un ancho de banda considerable, sin dejar de considerar la capacidad de las PC's a la que se le va a enviar dicha información. Donde todas la PC's presentan la misma capacidad y el ancho de banda es de 100Mbps.

- ✓ **Caso de Prueba 2**

Este Caso de Prueba se encarga de medir: La Cantidad de Información a Procesar, contra la Velocidad de las PC's, contra la Cantidad de PC's y la Cantidad de Memoria RAM total a utilizar. Con el objetivo de medir como influiría la cantidad de información a procesar, teniendo en cuenta la

velocidad de las PC's, de manera que se logre la menor utilización de recursos posibles: Cantidad de PC's y por consecuencia Memoria RAM, tratando siempre que la cantidad de información a procesar sea lo mayor posible y en un menor tiempo. Este Caso de Prueba el parámetro que más influencia tiene en los resultados alcanzados es la velocidad de las PC's, la cual no varía.

- ✓ **Caso de Prueba 3**

El presente Caso de Prueba medirá: Ancho de Banda contra la Cantidad de Datos a Procesar, contra la Velocidad de las PC's y Cantidad de Procesadores. En este caso se tiene como fin principal llegar al conocimiento del tiempo mínimo de procesamiento de datos según el número de Procesadores a utilizar, en este caso 2, y la Cantidad de datos a Procesar, manteniéndose constante la velocidad de las PC's y el Ancho de Banda.

- ✓ **Caso de Prueba 4**

En este Caso de Prueba se medirá: Cantidad de Información, contra la Cantidad de Memoria RAM total, contra la Cantidad de Procesadores total a utilizar. Se realiza con el objetivo de poder saber el tiempo más factible que se puede obtener, teniendo en cuenta algunas consideraciones:

1-Que la cantidad de información, no varíe y que se defina la cantidad de memoria RAM total a utilizar y la cantidad de procesadores.
2-Que la cantidad de información aumente y se utilice la misma cantidad de memoria RAM y procesadores a utilizar.
3-Que la cantidad de información, no varíe y que se aumente la cantidad de memoria RAM total y disminuya la cantidad de procesadores a utilizar o que disminuya la cantidad de memoria RAM total y aumente la cantidad de procesadores a utilizar.

- ✓ **Caso de Prueba 5**

El presente Caso de Prueba tiene en cuenta el caso en que se utilice una Base de Datos, donde resulta importante el tiempo de cada consulta a dicha Base de Datos, en consideración con la cantidad de información que esta abarque.

Realizándose con la intención de medir el tiempo mínimo en que puede realizarse una consulta, probándose además la Cantidad máxima de Información que soportará esta Base de Datos con el fin de lograr que todas esta consultas se realicen en un tiempo satisfactorio.

3.2 PROPUESTA DE SOLUCIÓN.

Después de realizar varias propuestas de Casos de Prueba se decidió utilizar, para el problema planteado por el Centro UCIFAR, dicho anteriormente y que se basa en lograr demostrar la optimización de tiempo en el procesamiento de datos para el módulo Diagonalización de Matrices tras la utilización de la Programación Paralela, el **Caso de Prueba 3** por considerarse el más completo por medir parámetros de gran importancia como son: Cantidad de Procesadores, Ancho de Banda, Velocidad de PC's y Cantidad de Datos a Procesar, los cuales se le pasarán como parámetros a una Aplicación la cual cuenta con dos procesos fundamentales: Uno distribuir de forma proporcionada la cantidad de información a cada una de las PC´s y el otro de recibir la información y procesarla. Cabe destacar que el Ancho de Banda es de 100 Mbps, y que cada una de las PC's utilizan 2 Procesadores (Pentium IV utilizando dos procesadores (Dual-Core) de 3GHz).

Para llegar a la obtención de los resultados de este Caso de Prueba se hace necesario establecer un mecanismo que permita que todas las PC's de este módulo logren procesar la mayor cantidad de datos al mismo tiempo. Para lograr esto se propone la utilización de un Clúster de estaciones de Trabajo, el mismo estará compuesto por 30 PC´s que cuentan con Sistema Operativo Linux con el fin de lograr la compatibilidad con la Aplicación en cuestión, conectadas con una Red Ethernet.

Realizándose una **Primera Prueba** donde una sola máquina realiza todo el Procesamiento de Datos, lo que sería equivalente a un procesamiento secuencial.

Para luego realizar como **Segunda Prueba** el Procesamiento en máquinas por separadas, es decir todas al mismo tiempo, para de esta forma reducir significativamente el tiempo total de procesamiento, ya que cada computadora aporta su capacidad de cómputo al proceso.

3.3 RESULTADOS OBTENIDOS.

A partir de la aplicación de las dos pruebas descritas en el epígrafe anterior se obtienen una serie de resultados, se realizarán gráficos con el objetivo de ganar en el entendimiento de dichos resultados. Los gráficos mencionados mostrarán hasta que punto fue factible la optimización lograda con la aplicación de la Programación Paralela en el módulo Diagonalización de Matrices, permitiendo sacar una serie de conclusiones como es el caso del tiempo mínimo en que puede ser procesada la mayor cantidad de información.

3.3.1 Resultados con la utilización de una sola PC.

Decir que el tiempo que se dará aquí es en segundos. Resaltar nuevamente que la PC en que se realizo esta prueba es Pentium IV con procesador de tipo Dual Core, memoria RAM de 1 GB y una capacidad de disco duro de 160 GB.

En la siguiente tabla se mostrará el tiempo que se demora esta PC para procesar la información para las distintas dimensiones de la Matriz, datos que serán graficados.

MATRICES DE ELEMENTOS	USER (TIEMPO EN SEGUNDOS)	REAL(TIEMPO EN SEGUNDOS)	SYS(TIEMPO EN SEGUNDOS)
M(10*100)^2	0.128	0.128	0.000
M(15*10)^2	1.124	1.131	0.000
M(20*100)^2	1.644	1.647	0.004
M(25*100)^2	11.601	11.602	0.004
M(30*100)^2	3.844	3.845	0.000
M(35*100)^2	7.892	7.895	0.004
M(40*100)^2	16.721	16.841	0.010

M(45*100)^2	124.572	124.584	0.004
M(50*100)^2	63.268	63.360	0.036
M(55*100)^2	44.779	44.786	0.000
M(60*100)^2	15.17.983	1518.208	0.076
M(65*100)^2	690.655	690.699	0.028
M(70*100)^2	20199.086	20198.784	0.128
M(75*100)^2	2608.819	2609.352	0.216
M(80*100)^2	1410.148	1410.142	0.008
M(90*100)^2	691.155	691.159	0.008
M(95*100)^2	994.106	994.116	0.008
M(100*100)^2	1829.150	1829.175	0.028

Tabla.4 Resultados para una pc.

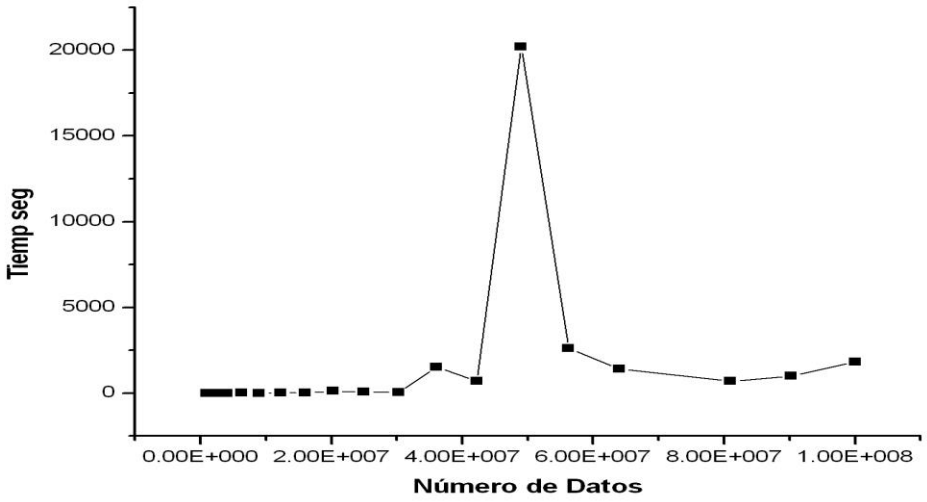


Fig. 12 Gráfica donde se muestran los resultados de una pc.

La gráfica anterior muestra que el tiempo de ejecución para matrices pequeñas es también pequeño. Aunque el punto más alto que se observa en la gráfica no es para el caso extremo, esto se debe a que ocurrió una fluctuación[11] . Llegándose a la conclusión que entre mayor sea la dimensión de la matriz mayor será el tiempo de ejecución.

3.3.2 Resultados para la aplicación de 30 PC's.

Al igual que en caso anterior los tiempos que se mostrarán a continuación es en segundos, en este punto se consideraran los resultados obtenidos del tiempo de ejecución pero en este caso con la utilización de 30 PC's correspondiendo a la segunda prueba realizada.

[11] Es cuando la matriz no tiene diagonal preponderante y por esta razón los tiempos se disparan.

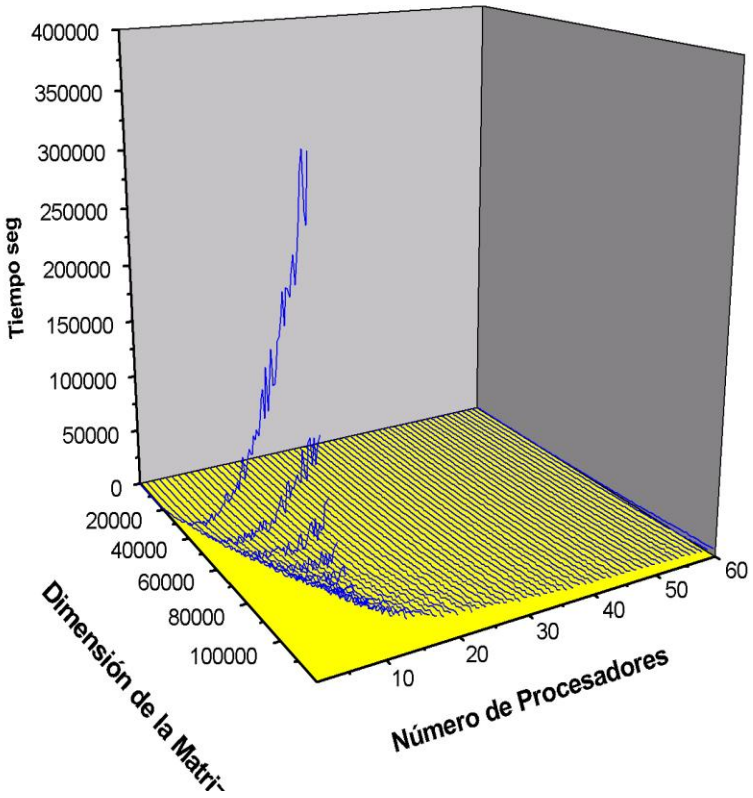


Fig.13 Gráfico de la Prueba Realizada

Después de mostrados los datos en la gráfica anterior, se hizo un análisis para obtener la representación gráfica que para cada dimensión de la matriz, poder buscar el tiempo mínimo de ejecución en dependencia de la cantidad de procesadores.

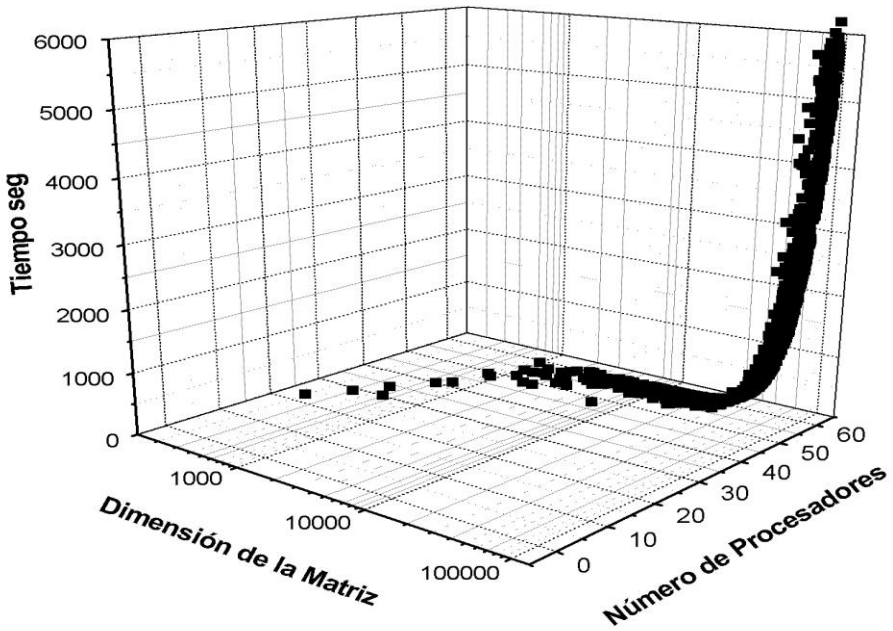


Fig.14 Gráfico de tiempo mínimo de ejecución.

En dicha gráfica se puede observar por ejemplo, para una matriz de 100000 datos, utilizando 60 procesadores su tiempo mínimo puede ser de 6000s.

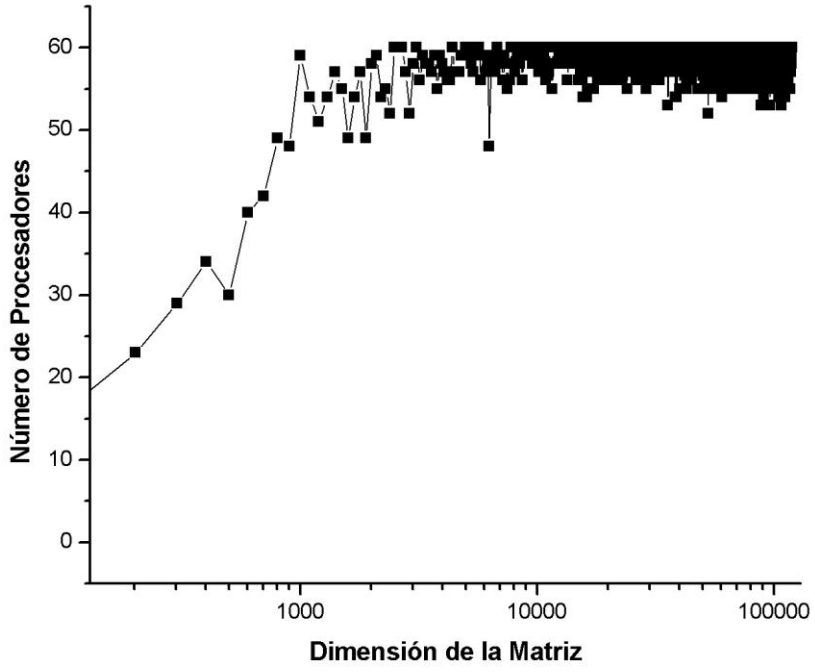


Fig. 15 Gráfico de Datos de Número Procesadores vs Dimensión de Matriz.

La gráfica anterior corresponde a la siguiente situación: ¿Cuántos procesadores hará falta para el procesamiento de una matriz según su dimensión?

En esta gráfica se puede observar que a partir de la matriz de 10000 elementos el problema puede ser resuelto con una cantidad de 60 procesadores.

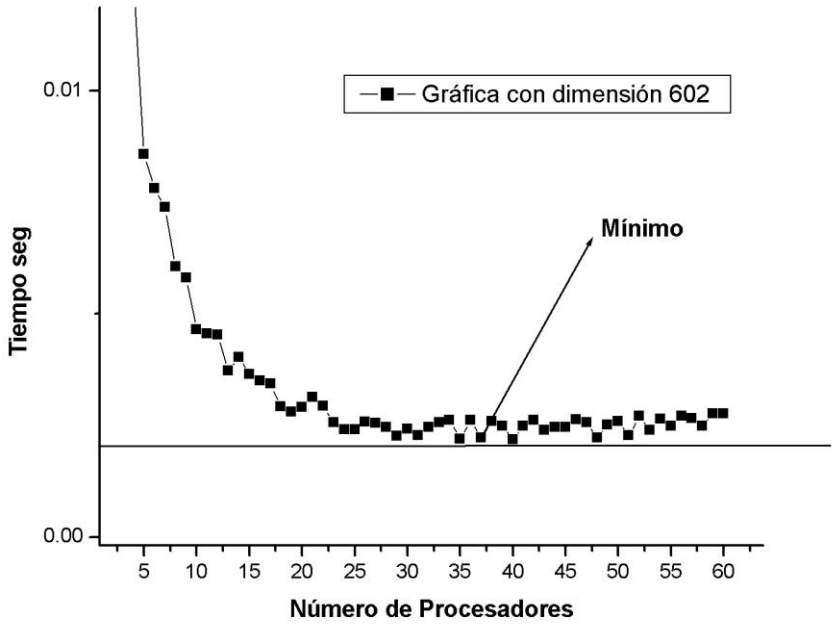


Fig.16 Gráfico con Matriz de dimensión 602.

Se realizó un estudio para un caso especifico se trata de una matriz de 602 elementos, donde se muestra en la gráfica anterior, con cuántos procesadores será el tiempo mínimo de ejecución, En dicha gráfica se pudo constatar que para esta situación se lograría el tiempo mínimo utilizando 35 procesadores.

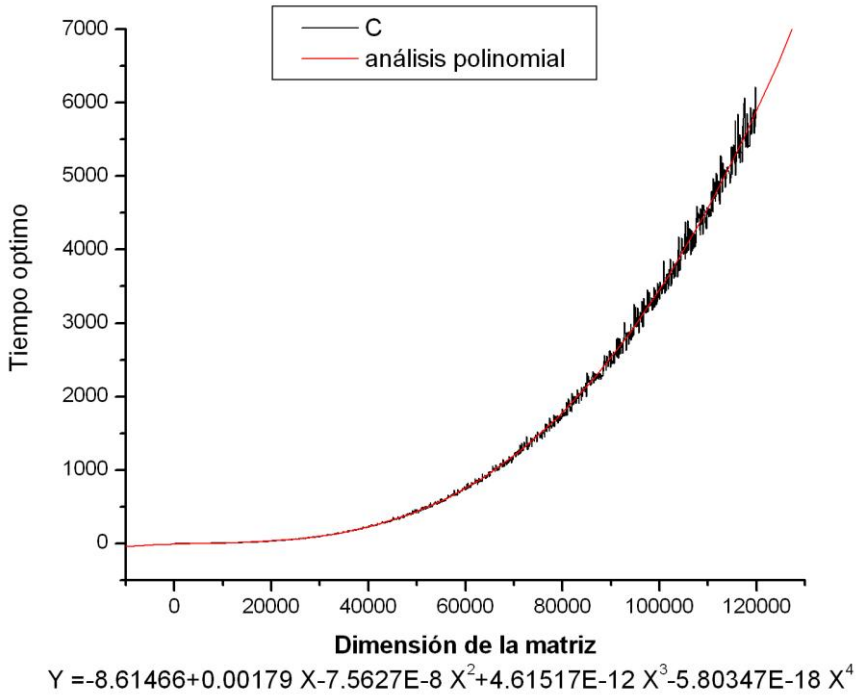


Fig.17 Gráfica Predictiva.

Ya por último la gráfica que presentamos anteriormente representa una gráfica de tipo predictiva[12], esta gráfica pude ser utilizada en caso de que cualquier persona quiera saber que tiempo pueda demorarse el procesamiento de una cierta cantidad de datos. Dicha gráfica demuestra que mientras mayor es la dimensión de la matriz mayor va a ser su tiempo de ejecución, aunque estamos hablando de tiempos pequeños, ya que la Programación Paralela, permite reducir los mismos, esto se debe a que el tráfico de datos en la red aumenta y por esa razón el tiempo es mayor.

Es importante destacar en este estudio que con todas las pruebas realizadas nunca se llegó a saturar[13] la red, a pesar de la gran cantidad información que se logró procesar ,esto se logró gracias a la

[12] Puede predecir el tiempo de procesar una matriz de n elementos.

[13] Llenar, ocupar completamente o utilizar una cosa hasta el límite de su capacidad.

arquitectura Raid1 [14], esta arquitectura permite que cada pc tenga su base de datos interna ,todas estas pc´s conectadas entre si, en forma de anillo y que la toma de un elemento de la base de datos se realice a lo sumo en un tiempo de 2ns[15], permite además que las operaciones de lectura y escritura sean locales y que la de lectura sea muy rápida al igual que la de escritura aunque en esta última cuando se hace una operación de escritura esta se replica en todas las bases de datos y que sobre esa región que se escribe no se puede leer hasta que no esté replicado en todas las PC´s, aunque todo esto se demora un tiempo a la sumo de (n* 5ns), donde n es la cantidad de pc que existen, se puede decir que es rapidísimo, y por eso podemos asegurar que para poder saturar la red, tendría que ser un cantidad de información que todavía no se ha podido predecir o comprobar.

3.4 Conclusiones

En este capitulo se propuso algunas propuestas de casos de pruebas, para la realización de las mismas con el objetivo de conocer hasta que punto puede ser eficiente la estructura de los diferentes algoritmos paralelos implementadas a la aplicación y de esta forma saber hasta que punto se pudo utilizar, sin tener problemas ni contratiempos.

Se realizó los estudios correspondientes y los resultados fueron plasmados en el mismo, además de representaciones gráficas que muestren dichos resultados.

Se realizó las observaciones de cada resultado, haciendo un análisis del tiempo en cada uno de los casos

Así como un estudio predictivo para lograr saber en un momento determinado cuanto se puede demorar en procesar más información de lo propuesto en el estudio realizado.

.

[14] Sistema de Replicación de Datos, aumenta la velocidad de lectura y escritura.

[15] Nanosegundos.

CONCLUSIONES

El modelo de programación paralela es un tema de investigación y una realidad cada vez más expandida en la industria. Permite resolver problemas cada vez más complejos y grandes en diversas áreas.

Las arquitecturas paralelas son alimentadas por diferentes avances tecnológicos como el incremento de poder de procesamiento (por ejemplo la creación de equipos multiprocesadores). La creación de redes con menores latencias (tiempo necesario para que un paquete de información viaje desde la fuente hasta un destino) y mayores anchos de banda (capacidad en la transmisión de datos).

Consideramos como principal conclusión de este trabajo, que los métodos de optimización basados en la programación paralela presentan importantes ventajas frente a otros métodos secuenciales, debido principalmente a la posibilidad de ejecución de diversos algoritmos en diferentes máquinas y a la capacidad de éstas de comunicarse, esto se puedo saber a través de las pruebas realizadas, que nos dieron la posibilidad de mostrar datos concretos y de esta forma, llegar a conclusiones de los datos obtenidos más relevantes.

RECOMENDACIONES

Se recomienda las siguientes actividades:

- ✓ Que se siga realizando más pruebas de rendimiento, 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.
- ✓ Que sea documento de consulta, pues en él se muestran conceptos importantes.
- ✓ Que sea un documento de otras investigaciones sobre lo que se pueda lograr en el tema de la Programación Paralela en la UCI.
- ✓ Que los análisis predictivos mostrados sirvan de base para nuevos estudios y diseños de sistemas de procesamiento.

REFERENCIA BIBLIOGRÁFICA

Alonso, José Miguel. 1997. Programación de aplicaciones paralelascon MPI (Message Passing Interface). España : s.n., 1997. Disponible en:www.sc.ehu.es/acwmialj/edumat/mpi.pdf.

Bosque, Jose Luis. 2007. 2007. Disponible en:dac.escet.urjc.es/docencia/LAAC/LAAC_Tema4.pdf.

Cruz De los Santos De la William, Altamirano Robles Leopoldo, González García Rubén A y Diaz, Gilberto. 1999. PVM. [En línea] 13 de 9 de 1999. Disponible en :www.cecalc.ula.ve/documentacion/tutoriales/beowulf/node53.html.

González Sánchez Alejandro. 2005. Programación Paralelas Prácticas. Mexico : s.n., 2005. Disponible en:webdiis.unizar.es/asignaturas/PPAR/ppar-prac.pdf.

Pozuelo, Carmela. Programacion Paralela:Multiplicacion de Matrices Grid´5000.

Tschanz Raúl, Smerling Leonardo. 2001. Procesamiento paralelo: qué tener en cuenta para aprovecharlo. Conceptos y alternativas en Linux. 2001.

BIBLIOGRAFÍA

Alonso, José Miguel. 1997. Programación de aplicaciones paralelascon MPI (Message Passing Interface). España : s.n., 1997. Disponible en:www.sc.ehu.es/acwmialj/edumat/mpi.pdf.

Bosque, Jose Luis. 2007. 2007. Disponible en:dac.escet.urjc.es/docencia/LAAC/LAAC_Tema4.pdf.

Cambio radical: El procesamiento multi-core ofrece innovadoras posibilidades a las empresas. 2006. s.l.: Intel, Revista 2006, 2006. Disponible en: http://www.intel.com/espanol/technology/magazine/index.htm..

Cluser::Definiciones.[Online] Disponible en:clusterfie.epn.edu.ec/clusters/Definiciones/definiciones.html **Cruz De los Santos De la William, Altamirano Robles Leopoldo, González García Rubén A y González Sánchez Alejandro. 2005.** Programación Paralelas Prácticas. Mexico : s.n., 2005. Disponible en:webdiis.unizar.es/asignaturas/PPAR/ppar-prac.pdf.

Diaz, Gilberto. 1999. PVM. [Online] 9 13, 1999. Disponible en :www.cecalc.ula.ve/documentacion/tutoriales/beowulf/node53.html.

García, José M. UN ENTORNO AVANZADO PARA LA SIMULACION DE MULTIPROCESADORES. España : s.n. Disponible en: www.ditec.um.es/~jmgarcia/papers/QUITO.pdf.

Gutierrez, Claudio. 1999. El paradigma computacional aplicado al estudio de la vida. [Online] 1999. Disponible en:www.claudiogutierrez.com/Paradigma_estudio_vida.html.

Gutiérrez, Claudio. 1999. El paradigma computacional aplicado al estudio de la vida. [Online] 1999. Disponible en:www.claudiogutierrez.com/Paradigma_estudio_vida.html..

Hoffman Forrest y Hargrove William. Computación Paralela con Linux. [Online] Disponible en: www.acm.org/crossroads/espanol/xrds6-1/parallel.html.

Lacarta, Magallon Juan Antonio. Programación Paralela Prácticas. Disponible en:webdiis.unizar.es/asignaturas/PPAR/ppar-prac.pdf.

Marin, Mauricio. 2000. Un modelo de predicción de desempeño para base de datos relacionales sobre BSP. [Online] agosto 11, 2000. Disponible en: kataix.umag.cl/~mmarin/topinf/parbd/paper.html..

2007. Middleware y Programación paralela de clusters. [Online] marzo 20, 2007. Disponible en: dac.escet.urjc.es/docencia/LAAC/LAAC_Tema7.pdf..

Pozuelo, Carmela. Programacion Paralela:Multiplicacion de Matrices Grid´5000.

Programación paralela y concurrente en *C++.* Disponible en:www.dacb.ujat.mx/publicaciones/revista_dacb/v4_n2/v4n2a3.pdf.

Quammen Cory. Introducción a la Programación en ambientes con Memoria-Compartida y Memoria-Distribuída. [Online] Disponible en: www.acm.org/crossroads/espanol/xrds8-3/programming.html.

Tschanz Raúl, Smerling Leonardo. 2001. Procesamiento paralelo: qué tener en cuenta para aprovecharlo. Conceptos y alternativas en Linux. 2001.

Weinbach Natalia L, Tucato Mariano, García Alejandro. Programación en Paralelo Utilizando un Modelo de Sistemas Multi-agente*.* Disponible en:cs.uns.edu.ar/~nlw/files/cacic05parallelsma.pdf.

ANEXOS

ANEXO I Parámetros de los clusters

Parámetros de clúster

Dirección IP principal del clúster

Este parámetro especifica la dirección IP principal del clúster en notación decimal con punto estándar de Internet (por ejemplo, w.x.y.z). Se trata de una dirección IP virtual y debe establecerse de manera idéntica para todos los hosts del clúster. Esta dirección IP se utiliza para dirigir el clúster como un todo y debe ser la dirección IP correspondiente al nombre Internet completo que se especifique para el clúster.

Máscara de subred

Este parámetro indica la máscara de subred para la dirección IP especificada. La máscara se escribe en notación decimal con punto estándar de Internet (por ejemplo, 255.255.255.0).

Nombre Internet completo del clúster

Este parámetro especifica un nombre Internet completo para el clúster de Equilibrio de la carga en la red (por ejemplo, cluster.microsoft.com). Este nombre se utiliza para el clúster como un todo y debe ser el mismo para todos los hosts del clúster. Cuando se aplica varios nombres de alias para el clúster, se debe escribir el nombre principal. En cualquier caso, este nombre debe poder resolverse en la dirección IP principal del clúster a través del servidor DNS o el archivo Hosts.

Dirección de red del clúster.

Este parámetro especifica la dirección de red (dirección MAC) para el adaptador de red que se va a utilizar para controlar el tráfico del cliente al clúster. Equilibrio de la carga en la red genera automáticamente la dirección de red en función de la dirección IP principal del clúster. Equilibrio de la carga en la red utiliza una dirección administrada de manera local, que también es una dirección MAC de multidifusión si se ha habilitado la compatibilidad con multidifusión, en caso de que la compatibilidad con multidifusión este deshabilitada (lo que causará que el hot revierta al modo de unidifusión), Equilibrio de carga en la red indicará automáticamente al controlador que pertenezca al adaptador del

clúster que suplante la única dirección de red integrada del adaptador y que cambie su dirección MAC por la dirección MAC del clúster. Ésta es la dirección utilizada en todos los hosts del clúster.

Compatibilidad con multidifusión

Este parámetro especifica si debe utilizarse o no una dirección MAC de multidifusión para las operaciones del clúster. En este caso, Equilibrio de la carga en la red convierte la dirección MAC del clúster pertenezca al adaptador del clúster en una dirección de multidifusión. También garantiza que la dirección IP principal del clúster se resuelva en esta dirección de multidifusión como parte del protocolo ARP. Al mismo tiempo, el adaptador puede utilizar ahora su dirección MAC integrada original, que se deshabilitó en el modo de unidifusión.

En el caso que los clientes de Equilibrio de la carga en la red tengan acceso a un clúster a través de un enrutador, cuando el clúster se haya configurado para operar en modo de multidifusión, hay que asegurarse que el enrutador cumple con los siguientes requisitos:

Que este acepte una respuesta ARP que tiene una dirección MAC en la estructura ARP, pero parece llegar desde una estación con otra dirección MAC, tal como indica el encabezado de Ethernet.

En el modo de multidifusión, acepta una respuesta ARP que tiene una dirección MAC de multidifusión en la estructura ARP.

Esto permite que el enrutador asigne la dirección IP principal del clúster y otras direcciones multitarjeta a la dirección MAC correspondiente. Si el enrutador no cumple estos requisitos, también puede crear una entrada ARP estática en el enrutador. Los enrutadores Cisco requieren una entrada ARP estática porque no admiten la resolución de direcciones IP de unidifusión en direcciones MAC de multidifusión.

Contraseña remota.

Este parámetro especifica la contraseña que se va a utilizar para restringir el acceso al clúster desde equipos de la red remotos. La contraseña consta de una cadena de caracteres alfanuméricos. Debe escribirse en este parámetro y una segunda vez para confirmarla en el parámetro Confirmas contraseña.

La contraseña de control remoto no se utiliza para restringir las operaciones de control de un host del clúster.

Confirmar contraseña.

Este parámetro especifica la contraseña escrita en el campo Contraseña remota Se utiliza para confirmar que se ha escrito correctamente esta contraseña. Al borrar ambos campos se deshabilita el uso de una contraseña de control remoto.

Control remoto.

Este parámetro especifica si se habilitan las operaciones de control remoto

Si se habilita el control remoto para el clúster de Equilibrio de la carga en la red, que está deshabilitado de manera predeterminada, es muy importante, por razones de seguridad, utilizar un servidor de seguridad para los puertos de control UDP de Equilibrio de la carga en la red (los puertos que reciben los comandos de control remoto), a fin de protegerlos de intromisiones externas. De manera predeterminada, estos puertos son el 1717 y el 2504 en la dirección IP del clúster.

GLOSARIO

Red LAN: Las redes de área local (local area networks) llevan mensajes a velocidades relativamente grandes entre computadores conectados a un único medio de comunicaciones: un cable de par trenzado. Un cable coaxial o una fibra óptica

Memoria RAM: La memoria principal o RAM (Random Access Memory, Memoria de Acceso Aleatorio) es donde el computador guarda los datos que está utilizando en el momento presente. El almacenamiento es considerado temporal porque los datos y programas permanecen en ella mientras que la computadora este encendida o no sea reiniciada.

Parámetro: En Ciencias de la computación, un parámetro o argumento es una variable que puede ser recibida por una subrutina.

Concurrencia: En computación, la concurrencia es la propiedad de los sistemas que permiten que múltiples procesos sean ejecutados al mismo tiempo, y que potencialmente puedan interactuar entre sí.

Switch: Switch (en castellano "conmutador") es un dispositivo electrónico de interconexión de redes de ordenadores que opera en la capa 2 (nivel de enlace de datos) del modelo OSI (Open Systems Interconnection). Un conmutador interconecta dos o más segmentos de red, funcionando de manera similar a los puentes (bridges), pasando datos de un segmento a otro, de acuerdo con la dirección MAC de destino de los datagramas en la red.

Firewall: Es un sistema o grupo de sistemas que impone una política de seguridad entre la organización de red privada y el Internet. Es un mecanismo para restringir acceso entre la Internet y la red corporativa interna. Típicamente se instala un firewall en un punto estratégico donde (o redes) se conectan a la Internet.

Latencia: Es el intervalo de tiempo que ocurre entre la ejecución de la operación de envío y en instante en que los datos comienzan a estar disponibles en el destino.

Plástico PVC: El Policloruro de Vinilo, plástico llamado PVC, es una combinación química de carbono, hidrógeno y cloro. El PVC es un material termoplástico, es decir, que bajo la acción del calor se

reblandece, y puede así moldearse fácilmente; al enfriarse recupera la consistencia inicial y conserva la nueva forma.

Portabilidad: Multiprocesadores, multicomputadores, redes, heterogéneos.

Estandarización: Realización de un proceso de forma única.

Eficiencia: Denota el uso efectivo de los recursos de cómputo. Da idea de la porción de tiempo que los procesadores se dedican a trabajo útil, o sea, una media de cuan eficiente se han utilizado los procesadores.