MÉTODO PARA EL MANEJO DEL BALANCEO DE CARGA EN SISTEMAS DE CÓMPUTO DISTRIBUIDO DE ALTO DESEMPEÑO

ANDREA MESA MÚNERA

Tesis de Grado presentada como requisito parcial para optar al Título de “Magíster en Ingeniería – Ingeniería de Sistemas”

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS
ESCUELA DE SISTEMAS
MEDELLÍN
2009

MÉTODO PARA EL MANEJO DEL BALANCEO DE CARGA EN SISTEMAS DE CÓMPUTO DISTRIBUIDO DE ALTO DESEMPEÑO

ANDREA MESA MÚNERA

Tesis de Grado presentada como requisito parcial para optar al Título de "Magíster en Ingeniería – Ingeniería de Sistemas"

Director:
Prof. JOHN WILLIAM BRANCH BEDOYA. Ph.D
Escuela de Sistemas

Codirector:
Prof. GUSTAVO ADOLFO PÉREZ ZAPATA. M.Sc
Escuela de Sistemas

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS
ESCUELA DE SISTEMAS
MEDELLÍN
2009

RESUMEN

En la actualidad y basándose en los principios de la computación se han planteado algoritmos secuenciales para tratar de dar solución a variados problemas. A medida que estos se han tornado más y más complejos, las soluciones basadas en este tipo de algoritmos han pasado a ser ineficientes, debido a que su uso implica un consumo de tiempo considerable durante su ejecución.

Para resolver estos problemas complejos nacieron los llamados supercomputadores, los cuales cuentan con arreglos de microprocesadores que trabajan en sincronía empleando procesamiento paralelo.

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

A estos "supercomputadores" se les conoce con el nombre de cluster, el cual no había sido un concepto muy difundido hasta hace poco tiempo.

Un cluster se puede definir como un grupo de múltiples computadores unidos por una red de alta velocidad, de tal forma que el conjunto puede ser visto como una única máquina, pero que por su poder de cómputo resuelve problemas que un solo equipo de escritorio no podría hacer.

Esta tesis mostrará la importancia de la computación paralela y distribuida para la solución de problemas frecuentes de alto desempeño computacional tomando como enfoque principal el estudio de una de sus arquitecturas principales: Cluster. Por medio del manejo de esta arquitectura, en esta investigación se pretenderá proponer un método para el manejo del balanceo de carga en sistemas de cómputo distribuido de alto desempeño realizando estudios previos y exhaustivos para llegar a dicha solución.

PALABRAS CLAVES: cluster, procesamiento paralelo, computación distribuida, computadores, escalabilidad, disponibilidad, fiabilidad, supercomputadores, balanceo de carga, procesos, función de aptitud, procesamiento, memoria, tiempo de respuesta, tiempo de ejecución, rendimiento, tareas.

ABSTRACT

At present and based on the principles of computing have emerged to deal with sequential algorithms to solve various problems. As they got more and more complex, the solutions based on that kind of algorithms have become inefficient, because its use imply a considerable amount of time during its execution.

To solve these complex problems were born the so-called supercomputers, which include arrangements of microprocessors that work in synchrony by using parallel processing.

In the past years, the academic communities of a variety of universities and research centers, have tried to learn how to build their own “supercomputers” by connecting personal computers and developing software to face such problems.

This “supercomputers” are known under the name of cluster, which is a concept not really spread until recently.

A cluster can be defined as a group of multiple computers connected by a high-speed network, so that all can be seen as a single machine, but that its computational power to solve problems a single desktop could not do.

This thesis will show the importance of parallel and distributed computing to solve common problems of high computational performance on the main approach to study one of its major architectures: Cluster. Through the management of this architecture, this research is intended to propose a method for handling load balancing in distributed computer systems high performance and comprehensive studies done to reach that solution.

KEYWORDS: cluster, parallel processing, distributed computing, computers, scalability, availability, reliability, supercomputers, load balancing, process, aptitude function, processing, memory, response time, execution time, performance, tasks.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

AGRADECIMIENTOS

A mis padres, mi hermanita y mi novio por haberme apoyado tanto y haber confiado siempre en mi trabajo. A Dios por darme la oportunidad de estar aquí y de demostrar lo que soy, puedo ser y que tan lejos puedo llegar.

A todas aquellas personas interesadas en el tema y que por sus comentarios y sugerencias aportaron para que la culminación de esta investigación fuera todo un hecho.

A los que directa o indirectamente colaboraron durante el largo proceso de este trabajo:

A John William Branch, mi director, por su apoyo y paciencia.

A muchos de los estudiantes e investigadores del Centro de Investigación y Estudios Avanzados (Cinvestav) del IPN Unidad Guadalajara, porque sin su ayuda llegar al final de este trabajo no hubiese sido posible.

Al Doctor Félix Francisco Ramos Corchado, por ser mi tutor en mi pasantía académica en el Cinvestav Unidad Guadalajara haciendo que mi estadía allí fuera una experiencia linda y amena en la que logré centrar los objetivos reales de mi tesis y pude dar pie a toda mi investigación.

A Esteban Jiménez, Felipe Valencia y Miguel Ángel Sánchez, por la disposición que tuvieron para ayudarme y por compartir sus conocimientos conmigo.

A mi papá (Guillermo) por interesarse más que nadie en mi trabajo, logrando entender el funcionamiento de este y aportando con su valioso conocimiento.

A mi mamá (Rosalba) y mi hermanita (Elizabeth) que siempre estuvieron ahí para apoyarme. Gracias por ser como son.

A mi novio (Hernán Darío Escobar), que al igual que mi papá se metió de lleno en mi tema logrando brindar nuevas y excelentes ideas.

A mis amigos y familiares por darme ánimo y fortaleza para seguir adelante en los momentos más difíciles.

Y a todos aquellos que siempre estuvieron ahí y que olvide mencionar.

Muchísimas gracias a todos. Los quiero mucho

1. INTRODUCCIÓN

Con el transcurrir de los años en la historia de la computación se han presentado diversos problemas de tipo complejo que en el pasado no podían ser resueltos o que simplemente el costo de su solución era sólo alcanzable para algunos. Sin embargo, con la tecnología actual, estos problemas son blanco fácil para los investigadores y con pocos recursos se puede llegar a la solución esperada.

El área de la computación contiene diferentes ramas que ayudan a resolver gran cantidad de problemas. Estas ramas van desde las tecnologías de información hasta un área tan grande y difícil de comprender como la inteligencia artificial, la cual estudia en general el comportamiento humano y procura simularlo en máquinas. La inteligencia artificial es el estudio de las facultades mentales y se ocupa de construir sistemas que permitan exhibir un comportamiento cada vez más inteligente a través del uso de modelos computacionales que conciernen con el diseño y desarrollo de hardware y software destinado a la realización de tareas complejas, que son realizadas actualmente y en forma óptima por los humanos [GIAR, 2007].

Desde sus inicios en el año 1936 la inteligencia artificial ha traído varias consecuencias a nivel de pensamiento humano puesto que lo que se pretende es construir sistemas que permitan exhibir un comportamiento cada vez mas inteligente solucionando los problemas que hoy en día el ser humano resuelve mejor que el computador, con el fin de desarrollar automáticamente aquellas cosas con las que el hombre aún supera a las máquinas [MID, 2006].

No obstante, siempre existirán filosofías escépticas que creen que es imposible lograr una máquina, de verdad inteligente. Consideran que siempre quedará un rasgo de inteligencia humano difícilmente convertible en una serie de algoritmos, llámese éste creatividad o cualquier otra cosa. "Las máquinas podrán resolver problemas, pero nunca podrán plantearse problemas".

Como la mayoría de las áreas, la inteligencia artificial es muy extensa y se divide en varias ramas. En esta investigación se propone realizar un hibrido entre el procesamiento paralelo que nace en la telemática y la computación distribuida que surge de una de las ramas de la inteligencia artificial; la inteligencia artificial distribuida. La computación distribuida ayuda a solucionar dificultades que requieren del uso de gran poder de cómputo resolviéndolo por medio del uso de varias máquinas que no necesariamente contienen software y hardware especializado y que se encuentran unidas y organizadas en un cluster de computadores.

La computación distribuida es un modelo para resolver problemas de computación masiva utilizando un gran número de computadores organizados en cluster incrustados en una infraestructura de telecomunicaciones distribuida. Para solucionar estos problemas se debe

considerar que estos se solucionan bajo una arquitectura que funciona a partir de procesamiento paralelo, el cual no es más que la ejecución simultánea de una misma tarea (dividida y adaptada especialmente) en múltiples procesadores, con el fin de obtener resultados en menor tiempo y de manera más eficiente [PUENTES, 2005] y [LEDESMA, 2007].

La computación paralela y distribuida maneja dos tipos de arquitecturas principales (Cluster y Grid), las cuales hacen posible la solución de problemas frecuentes en alto desempeño computacional. De [PINO, 2002], los Cluster están conformados por una colección de PC's autónomos interconectados trabajando unidos como un solo recurso de computación integrado. Y los Grid son una infraestructura de hardware y software que puede encontrarse distribuida y cuyo objetivo es permitir gestionar y distribuir la potencia de cálculo disponible, de tal forma que los usuarios se beneficien de la potencia de computadores subutilizados que se encuentran dispersos geográficamente [ENGLER, 2004].

Esta investigación se limita a trabajar básicamente con la primera de las arquitecturas (cluster de computadores), esto, con el fin de acotar un poco más la problemática a tratar en el área de la computación paralela y distribuida, debido a su gran extensión. El enfoque principal se encontrará relacionado única y exclusivamente a una de las variables que afectan el rendimiento en los clusters: el balanceo de la carga de trabajo.

De [REGO, 2006], el balanceo de carga trata de distribuir la carga de trabajo de acuerdo a la disponibilidad de procesamiento y a los recursos de cada equipo en un sistema computacional. Esta distribución pretende maximizar la utilización de los recursos, posibilitando el mejor desempeño del sistema.

En la actualidad el problema del balanceo de carga ha sido bastante trabajado, debido al hecho de que las diferentes comunidades requieren disminuir al máximo el tiempo de ejecución de las aplicaciones ejecutadas. Sin embargo, su solución no es siempre trivial. Este es un problema bastante complejo, debido a la dificultad en ocasiones de lograr que las propuestas en las distribuciones de carga de trabajo sean fácilmente escalables, o que se pueda trabajar con equipos que no sean necesariamente similares; en general se recurre a trabajar con cluster homogéneos (máquinas con software o hardware semejantes) y no en heterogéneos (máquinas con software o hardware diferentes).

El objetivo general de esta tesis es entonces, proponer un método para efectuar el manejo del balanceo de carga en sistemas de cómputo distribuido de alto desempeño, empleando técnicas propias de la computación paralela por medio del uso de un cluster heterogéneo.

Y los objetivos específicos de este trabajo son los siguientes:

1. Diferenciar entre los diversos tipos de cluster existentes con el fin de entender su funcionamiento básico para así verificar si es posible combinarlos.

2. Identificar las diferentes formas para balancear la carga entre nodos de un cluster, para escoger la mejor de estas técnicas y analizar el rendimiento en el sistema total.

3. Estudiar y seleccionar un sistema paralelo con el fin de modelarlo, basándose en uno de los modelos de computación y su determinada arquitectura paralela.

4. Proponer un método para el manejo del balanceo de carga en sistemas de cómputo distribuido usando un cluster heterogéneo basado en el tipo de balanceo de carga seleccionado, la arquitectura y el modelo computacional.

5. Validar el método propuesto mediante la implementación de un prototipo con base en experimentaciones pasadas, realizando pruebas de desempeño.

En el marco de esta investigación se tiene como prioridad plantear este método de balanceo de carga siendo éste aporte valioso para la comunidad científica local.

Aunque la parte de implementación es importante para la investigación ésta sólo alcanza un papel secundario para mostrar únicamente los resultados encontrados durante el proceso. La validación fue efectuada en un simulador de tiempo discreto llamado Network Simulator-2 (NS-2), en donde fue implementado el algoritmo de balanceo. Los nodos utilizados que conformaran el cluster son equipos que contarán únicamente con hardware diferente. La versión de NS-2 con la que se trabaja en esta investigación es la versión 2.31, instalada en Linux-SUSE versión 10.1 y ubicada en /usr/local/src/ns-2.31.

El NS-2 se apoya en dos lenguajes de programación para su correcto funcionamiento: OTcl y C++. Como resultado de la simulación se obtiene una gran cantidad de datos matemáticos para los estudios posteriores y trazas específicas que son visualizadas en la herramienta NAM del ns.

Los resultados obtenidos muestran que el método propuesto alcanza todas las expectativas logrando mejorar los tiempos de ejecución en comparación con los métodos con los que fue comparado.

El presente trabajo está organizado de la siguiente manera: En el capítulo 2 se presentará una revisión de los conceptos y términos fundamentales de los cluster de computadores, seguido por el capítulo 3, en el cual se presentará una revisión de tipo conceptual del balanceo carga que incluye definiciones y tipos de balanceo existentes. Para el capítulo 4 se efectuará una revisión del estado del arte, concerniente a algunos trabajos previos aplicativos de cluster de computadores y balanceo de carga. En el capítulo 5 se mostrará la proposición del método de balanceo de carga. Posteriormente en el capítulo 6 se presentarán los experimentos y el análisis de los resultados, seguidos de las conclusiones y trabajo futuro en el capítulo 7 y finalmente las referencias para el capítulo 8.

2. CLUSTER DE COMPUTADORES

La ejecución de tareas complejas siempre ha sido un problema en el mundo, ya que se debe contar con el equipo necesario para la elaboración y solución de estas.

En el pasado gran cantidad de aplicaciones no podían ser ejecutadas debido a la deficiencia de cómputo de las máquinas de esa época. Se hacía necesario el uso de grandes equipos que funcionaran sólo para tal fin, estos equipos eran denominados supercomputadores. Sin embargo este tipo de computadores no era accesible para cualquiera ya que su costo era bastante elevado.

Tras la necesidad de solucionar el problema de complejidad en las tareas y el elevado costo de los equipos que pudieran resolver dichas aplicaciones nacen los cluster.

El concepto de Cluster nació cuando los pioneros de la supercomputación intentaban difundir diferentes procesos entre varios computadores, para luego poder recoger los resultados que dichos procesos debían producir. Con un hardware más barato y fácil de conseguir se pudo perfilar que podrían conseguirse resultados muy parecidos a los obtenidos con aquellas máquinas mucho más costosas [CHIRINOV, 2003].

Según [PINO, 2002], el uso de clusters para desarrollar, depurar y ejecutar aplicaciones paralelas está ganando en popularidad, convirtiéndose en una gran alternativa al uso de arquitecturas más especializadas debido al gran precio de estas. Un factor importante que ha hecho que el uso de clusters sea práctico es la estandarización de numerosas herramientas y utilidades usadas en aplicaciones paralelas.

Basados en [CUETO, 2004], un cluster se puede definir como el trabajo realizado por dos o más computadores que en conjunto se encargan de proveer una determinada solución.
Podría decirse que simula el comportamiento de un sistema MPP (según [ROMO, 2003], el Procesamiento Masivamente Paralelo o por sus siglas en ingles Massively Parallel Processing, es una arquitectura computacional de alto rendimiento), reduciendo considerablemente los costos.

Tiene como finalidad agrupar el poder de cómputo de los nodos implicados para proporcionar una mayor escalabilidad, disponibilidad y fiabilidad [CUETO, 2004] y [PLAZA, 2004].
La escalabilidad es la capacidad de un equipo para hacer frente a volúmenes de trabajo cada vez mayores sin, por ello, dejar de prestar un nivel de rendimiento aceptable. La disponibilidad y la fiabilidad, se encuentran bastante relacionados, aunque difieren ligeramente en algunos aspectos. La disponibilidad es la calidad de estar presente y listo para su uso, mientras que la fiabilidad es la probabilidad de un correcto funcionamiento [PLAZA, 2004].

2.1 CARACTERISTICAS DE LOS CLUSTER

De [CATALÁN, 2006], los cluster deben presentar ciertas características con el fin de que su función de optimizar recursos se cumpla satisfactoriamente:

a. Un cluster consta de 2 o más nodos. Una de las características principales de estas arquitecturas es que exista un medio de comunicación (red) donde los procesos puedan migrar para computarse en diferentes estaciones paralelamente.

b. Los nodos de un cluster están conectados entre sí por al menos un canal de comunicación. Para llevar a cabo su misión los nodos deben estar comunicados entre sí.

c. Los clusters necesitan software de control especializado. Esta no es una característica física, es una característica de programación, la cual involucra toda la parte que tiene que ver con el control interno y la forma en la que funciona el cluster al momento de ejecutarse. Se requiere de algún tipo de software capaz de poder realizar alguna función específica del cluster, tal como: migrar procesos, balancear la carga, etc.
El software de control especializado puede estar a nivel de aplicación o a nivel de sistema.

- **Software a nivel de aplicación:** Se utilizan generalmente bibliotecas de carácter general que permiten la abstracción de un nodo a un sistema conjunto, permitiendo crear aplicaciones en un entorno distribuido de manera lo más abstracta posible. Este tipo de software suele generar elementos de proceso tipo: rutinas, procesos o tareas, que se ejecutan en cada nodo del cluster y se comunican entre sí a través de la red.
- **Software a nivel de sistema:** Suele estar implementado como parte del sistema operativo de cada nodo, o ser la totalidad de éste. Es más crítico y complejo, por otro lado suele resolver problemas de carácter más general que los anteriores y su eficiencia, por norma general, es mayor.

d. Dependiendo del tipo de software, el cluster puede estar más o menos acoplado. Se entiende por acoplamiento del software a la integración que tengan todos los elementos software que existan en cada nodo. Gran parte de la integración del sistema la produce la comunicación entre los nodos, y es por esta razón por la que se define el acoplamiento; otra parte es la que implica cómo de crítico es el software y su capacidad de recuperación ante fallos. Se distinguen tres tipos de acoplamiento:

- **Acoplamiento fuerte:** El software que entra en este grupo es software cuyos elementos se interrelacionan mucho unos con otros y posibilitan la mayoría de las funcionalidades del cluster de manera altamente cooperativa.
- **Acoplamiento medio:** A este grupo pertenece un software que no necesita un conocimiento tan exhaustivo de todos los recursos de otros nodos, pero que sigue usando el software de otros nodos para aplicaciones de muy bajo nivel.
- **Acoplamiento débil:** Generalmente se basan en aplicaciones construidas por bibliotecas preparadas para aplicaciones distribuidas.

e. Todos los elementos del cluster trabajan para cumplir una funcionalidad conjunta, sea esta la que sea. Es la funcionalidad la que caracteriza el sistema.

f. Mejora sobre la disponibilidad. La disponibilidad es la calidad de estar presente y listo para su uso.

g. Mejora del rendimiento. El rendimiento es la medida o cuantificación de la velocidad ó resultado con que se realiza una tarea o proceso

En general la catalogación de los clusters se hace en base a cuatro factores de diseño bastante ortogonales entre sí: acoplamiento, control, homogeneidad y seguridad.

De estos factores en este tema ya se ha visto el que quizás es el más importante, el de acoplamiento.

Por otro lado está el factor de **control del cluster**. El parámetro de control implica el modelo de gestión que propone el cluster. Este modelo de gestión hace referencia a la manera de configurar el cluster y es dependiente del modelo de conexionado o colaboración que surgen entre los nodos. Puede ser de dos tipos:

- **Control centralizado:** Se hace uso de un nodo maestro desde el cual se puede configurar el comportamiento de todo el sistema. Este nodo es un punto crítico del sistema aunque es una ventaja para una mejor gestión del cluster.
- **Control descentralizado:** En un modelo distribuido donde cada nodo debe administrarse y gestionarse. Es propio de sistemas distribuidos, como ventaja tiene que presenta más tolerancia a fallos como sistema global, y como desventajas que la gestión y administración de los equipos requiere más tiempo.

En lo que se refiere a **homogeneidad de un cluster** se ha demostrado que es posible crear sistemas de una sola imagen o heterogéneos con una implementación práctica. En cualquier caso, hay que entender por homogeneidad del cluster a la homogeneidad de los equipos y recursos que conforman a éste. Los clusters heterogéneos son más difíciles de conseguir ya que se necesitan notaciones abstractas de transferencias e interfaces especiales entre los nodos para que éstas se entiendan, por otro lado los clusters homogéneos obtienen más beneficios de estos sistemas y pueden ser implementados directamente a nivel de sistema.

Existen otros muchos factores de diseño que limitan el comportamiento y modelado de un cluster. La imposibilidad de llegar a clusters que paralelicen cualquier proceso se basa en que la mayoría de las aplicaciones hacen uso, en mayor o menor medida, de algoritmos secuenciales no paralelizables.

El último factor a tener en cuenta es la **seguridad de un cluster**, la cual es circunstancial a la hora de la ejecución de cualquier tipo de proceso o tarea, ya que esta hace que la privacidad no sea alterada por cualquier ente o persona mal intencionada.

2.2 CLASIFICACIÓN DE LOS CLUSTER

Después de analizar una a una las características de los cluster, estas dan pie a diferentes tipos de cluster los cuales fueron creados para desempeñar funciones especificas, de acuerdo a los requerimientos y preferencias de quien o quienes los estén implementando. Existen principalmente tres tipos de cluster: cluster de alto rendimiento, cluster de alta disponibilidad y cluster de alta confiabilidad.

- ✓ Alto rendimiento (HP, high performance): Según [CATALÁN, 2006], los clusters de alto rendimiento han sido creados para compartir el recurso más valioso de un computador, el tiempo de proceso. Cualquier operación que necesite altos tiempos de CPU puede ser utilizada en un cluster de alto rendimiento, siempre que se encuentre un algoritmo que sea paralelizable.

  La misión: Mejorar el rendimiento en la obtención de la solución de un problema, en términos del tiempo de respuesta o de su precisión.

  Cualquier cluster que haga que el rendimiento del sistema aumente respecto al de uno de los nodos individuales puede ser considerado cluster HP.

  Problemas que solucionan: Generalmente estos problemas de cómputo suelen estar ligados a cálculos matemáticos, renderizaciones de gráficos, compilación de programas, compresión de datos, descifrado de códigos, rendimiento del sistema operativo, (incluyendo en él, el rendimiento de los recursos de cada nodo). Existen muchos problemas más que se pueden solucionar con clusters HP, donde cada uno aplica de una manera u otra las técnicas necesarias para habilitar la paralelización del problema, su distribución entre los nodos y obtención del resultado.

  Técnicas que utilizan: Las técnicas utilizadas dependen del nivel al que trabaja el cluster.

  - Nivel de aplicación: Los clusters implementados a nivel de aplicación no suelen implementar balanceo de carga. Suelen basar todo su funcionamiento en una política de localización que sitúa las tareas en los diferentes nodos del cluster, y las comunica mediante las librerías abstractas. Resuelven problemas de cualquier tipo de los que se han visto en el apartado anterior, pero se deben diseñar y codificar aplicaciones propias para cada tipo para poderlas utilizar en estos clusters.
  - Nivel de sistema: Estos clusters basan todo su funcionamiento en comunicación y colaboración de los nodos a nivel de sistema operativo, lo que implica generalmente que son clusters de nodos de la misma arquitectura, con ventajas en lo que se refiere al factor de acoplamiento, y que basan su funcionamiento en compartir recursos a cualquier nivel, balanceo de la carga de manera dinámica, funciones de planificación especiales y otros factores que componen el sistema.

  Desventajas: Entre las limitaciones que existen actualmente está la incapacidad de balancear la carga dinámica de las librerías PVM o la incapacidad de openMosix de migrar procesos que hacen uso de memoria compartida. Una técnica que obtiene mayor ventaja es cruzar ambos sistemas: PVM + openMosix. Se obtiene un sistema con un factor de acoplamiento elevado que presta las ventajas de uno y otro, con una pequeña limitación por desventajas de cada uno.

- ✓ Alta disponibilidad (HA, high availability): Según [CATALÁN, 2006], los clusters de alta disponibilidad son bastante ortogonales en lo que se refieren a funcionalidad a un cluster de alto rendimiento. Los clusters de alta disponibilidad pretenden dar servicios 7/24 de cualquier tipo, son clusters donde la principal funcionalidad es estar controlando y actuando para que un servicio o varios se encuentren activos durante el máximo periodo de tiempo posible. En estos casos se puede comprobar como la monitorización de otros es parte de la colaboración entre los nodos del cluster.

  La misión: Los clusters de alta disponibilidad han sido diseñados para la máxima disponibilidad sobre los servicios que presenta el cluster. Este tipo de clusters son la competencia que abarata los sistemas redundantes, de manera que ofrecen una serie de servicios durante el mayor tiempo posible. Para poder dar estos servicios los clusters de este tipo se implementan en base a tres factores:
  - Fiabilidad
  - Disponibilidad
  - Dotación de servicio

  Mediante estos tres tipos de actuaciones y los mecanismos que lo implementan se asegura que un servicio esté el máximo tiempo disponible y que éste funcione de una manera fiable. Respecto al tercer punto, se refiere a la dotación de uno de estos clusters de un servicio que provea a clientes externos.

  Problemas que solucionan: La mayoría de estos problemas están ligados a la necesidad de dar servicio continuo de cualquier tipo a una serie de clientes de manera ininterrumpida. En una construcción real se suelen producir fallos inesperados en las máquinas, estos fallos provocan la aparición de dos eventos en el tiempo: el tiempo en el que el servicio está inactivo y el tiempo de reparación del problema.

  Entre los problemas que solucionan se encuentran: sistemas de información redundante, sistemas tolerantes a fallos, balanceo de carga entre varios servidores, balanceo de conexiones entre varios servidores.

  En general todos estos problemas se ligan en dos fuentes de necesidad de las empresas u organizaciones: tener un servicio disponible y ahorrar económicamente todo lo que sea posible

  Técnicas que utilizan: Como se ha visto en el apartado anterior los servicios y el funcionamiento de los mismos suelen ser de carácter bastante distinto, en cualquier caso, se suelen proponer sistemas desde SSI (Single System Image) que plantean serias dudas en lo que se refiere a localización de un servidor, hasta balanceo de carga o de conexiones. También suelen contener secciones de código que realizan monitorización de carga o monitorización de servicios para activar las acciones necesarias para cuando estos caigan.

- ✓ Alta confiabilidad (HR, high reliability): Según [CATALÁN, 2006], estos clusters tratan de aportar la máxima confiabilidad en un entorno, en la cual se necesite saber que el sistema se va a comportar de una manera determinada. Puede tratarse por ejemplo de sistemas de respuesta a tiempo real.

  Este tipo de clusters son los más difíciles de implementar. No se basan solamente en conceder servicios de alta disponibilidad, sino en ofrecer un entorno de sistema

altamente confiable. Esto implica muchísima sobrecarga en el sistema. Son también clusters muy acoplados.

En los clusters de alta disponibilidad generalmente una vez que el servicio ha caído éste se relanza, y no existe manera de conservar el estado del servidor anterior, más que mediante puntos de parada o checkpoints, pero que en conexiones en tiempo real no suelen ser suficientes. Por otro lado, los clusters confiables tratan de mantener el estado de las aplicaciones, no simplemente de utilizar el último checkpoint del sistema y relanzar el servicio.

Generalmente este tipo de clusters suele ser utilizado para entornos de tipo empresarial y esta funcionalidad solamente puede ser efectuada por hardware especializado. Por el momento no existe ninguno de estos clusters implementados como software. Esto se debe a limitaciones de la latencia de la red, así como a la complejidad de mantener los estados.

Dada la naturaleza asíncrona actual en el campo de los clusters, este tipo de clusters aún será difícil de implementar hasta que no se disminuyan los costos en las técnicas de comunicación.

2.3 VARIABLES QUE AFECTAN EL DESEMPEÑO DE LOS CLUSTER

Los cluster son creados para sustentar alto desempeño sobre un único problema, proveer alta tasa de salida para un conjunto de aplicaciones diferentes (throughput), mantener alta disponibilidad de los nodos y permitir alto acceso a los discos y canales de entrada y salida [MEREDITH, 2003]. Por lo tanto, existen algunas variables que afectan las características esenciales de los cluster, las cuales se encuentran directamente ligadas al rendimiento que un cluster puede presentar o a la forma en que trabaja éste. Se pueden mencionar las siguientes:

- ✓ Migración de procesos: se realiza para equilibrar la carga de trabajo de los nodos de un cluster.
- ✓ Latencia de red: es la cantidad de tiempo que tarda una transferencia de datos de un punto de la red a otro.
- ✓ Balanceo de carga: es el proceso mediante el cual se intenta reducir la diferencia de carga entre pares de nodos, migrando procesos desde los nodos más cargados hacia los menos cargados.
- ✓ Gestión de memoria: esta variable intenta disponer el máximo número de procesos en la RAM del sistema con el fin de evitar el intercambio de procesos.
- ✓ Monitorización: proporciona información sobre el comportamiento de aplicaciones en tres niveles: secuenciales paramétricas, paralelas intracluster y paralelas intercluster.

2.4 ESTRUCTURAS DEL CLUSTER

Según [REGO, 2006], existen 3 formas de tener un cluster organizado estructuralmente, la diferencia básica entre estas estructuras es si el recurso computacional está o no exclusivamente dedicado para el procesamiento del cluster y la localización física de los nodos.

- Pila de PC's: esta es la estructura comúnmente adoptada por comunidades científicas, en el que generalmente el uso del cluster es destinado a aplicaciones de alto desempeño. Básicamente un cluster de esta categoría puede ser descrito como un empilamiento de esclavos que son gerenciados por un computador maestro ó servidor a través de redes. Un esclavo no posee monitor, teclado o mouse, por estos motivos los nodos esclavos pueden ser empilados formando pilas de PC's (PoPC's) [CARNS, 1999]. Todos los cluster configurados y operados por un servidor pueden ser también responsable de dividir las tareas y enviarlas a los esclavos. Así que la tarea es realizada por un esclavo y luego enviada por este al maestro.
- Cluster de estaciones de trabajo: diferente pila de PC's. Aquí todos los nodos son computadores completos, con monitor, teclado, mouse y otros dispositivos. En este caso, los nodos del cluster son utilizados en periodos en que las estaciones se encuentren ociosas, por ejemplo, durante las noches. Los resultados son devueltos a los usuarios.
- Cluster distribuido o Grid: un cluster distribuido, como su nombre lo dice es un sistema paralelo y distribuido, compuesto por una colección de recursos computacionales (clusters) interconectados que se comunican a través de redes no locales (WAN). Estos clusters que componen un grid pueden estar estructurados tanto en pilas de PC's como en cluster de estaciones de trabajo.

3. BALANCEO DE CARGA

De [DORMIDO, 2003], el aspecto del balanceo de carga es bastante importante debido a que en muchas aplicaciones paralelas, como la búsqueda o la optimización, es extraordinariamente difícil predecir el tamaño de las tareas asignadas a cada procesador, de manera que se realice una división de las mismas para que todos mantengan la carga computacional uniforme. Cuando no es uniforme, es decir, hay desbalanceo en la carga, entonces algunos procesadores terminarán permaneciendo inactivos mientras otros todavía están calculando.

Frecuentemente todos los procesadores o algunos de ellos necesitan algún sincronismo durante la ejecución del programa, de forma que si no están todos en la misma situación en todo instante entonces algunos de ellos deberán esperar a que otros terminen.

El estudio del balanceo de carga es muy importante para poder distribuir de una forma equitativa la carga computacional entre todos los procesadores disponibles y con ello conseguir la máxima velocidad de ejecución.

Se considera que el problema a resolver se divide en un número fijo de procesos que pueden ejecutarse en paralelo. Cada proceso realiza una cantidad conocida de trabajo. Además, se supone que los procesos se distribuyen entre las máquinas disponibles sin tener en cuenta el tipo de procesador y su velocidad.

Sin embargo, puede ocurrir que algunos procesadores finalicen sus tareas antes que el resto y queden libres debido a que el trabajo no se haya repartido de una forma equitativa o porque algunos procesadores sean más rápidos que otros, o por ambas situaciones. La situación ideal es que todos los procesadores trabajen de una forma continua sobre las tareas disponibles para conseguir el mínimo tiempo de ejecución.

3.1 TIPOS DE BALANCEO DE CARGA

Según [DORMIDO, 2003], existen dos formas de balanceo de carga: balanceo de carga estático y balanceo de carga dinámico. En el primer caso, la distribución de las tareas se realiza al comienzo de la computación, lo cual permite al maestro (nodo principal dentro del cluster) participar en la computación una vez que haya asignado una fracción del trabajo a cada esclavo (el resto de los nodos de un cluster. Los nodos esclavos obedecen órdenes del nodo maestro). La asignación de tareas se puede realizar de una sola vez o de manera cíclica. El segundo caso, balanceo de carga dinámico es muy útil cuando el número de

tareas es mayor que el número de procesadores disponibles o cuando el número de tareas es desconocido al comienzo de la aplicación. Una importante característica del balanceo de carga dinámico es la capacidad que tiene la aplicación de adaptarse a los posibles cambios del sistema, no sólo a la carga de los procesadores sino también a posibles reconfiguraciones de los recursos del sistema. Debido a esta característica, un cluster puede responder bastante bien cuando se produce el fallo de algún procesador, lo cual simplifica la creación de aplicaciones tolerantes a fallos que sean capaces de sobrevivir cuando se pierde algún esclavo o incluso el maestro.

3.1.1 BALANCEO DE CARGA ESTÁTICO

De [DORMIDO, 2003], el balanceo de carga también es llamado mapeado del problema o planificación del problema. Este tipo de balanceo de carga se trata antes de la ejecución de cualquier proceso.

El balanceo de carga estático tiene serios inconvenientes que lo sitúan en desventaja sobre el balanceo de carga dinámico. Entre ellos cabe destacar los siguientes:

- Es muy difícil estimar de forma precisa el tiempo de ejecución de todas las partes en las que se divide un programa sin ejecutarlas.
- Algunos sistemas pueden tener retardos en las comunicaciones que pueden variar bajo diferentes circunstancias, lo que dificulta incorporar la variable retardo de comunicación en el balance de carga estático.
- A veces los problemas necesitan un número indeterminado de pasos computacionales para alcanzar la solución. Por ejemplo, los algoritmos de búsqueda normalmente atraviesan un grafo buscando la solución, y a priori no se sabe cuántos caminos hay que probar, independientemente de que la programación sea secuencial o paralela.

3.1.2 BALANCEO DE CARGA DINÁMICO

Según [DORMIDO, 2003], este tipo de balanceo de carga se trata durante la ejecución de procesos.

Con el balanceo de carga dinámico todos los inconvenientes que presenta el balanceo de carga estático se tienen en cuenta. Esto es posible porque la división de la carga computacional depende de las tareas que se están ejecutando y no de la estimación del tiempo que pueden tardar en ejecutarse. Aunque el balanceo de carga dinámico lleva consigo una cierta sobrecarga durante la ejecución del programa, resulta una alternativa mucho más eficiente que el balanceo de carga estático.

En el balanceo de carga dinámico, las tareas se reparten entre los procesadores durante la ejecución del programa. Dependiendo de dónde y cómo se almacenen y repartan las tareas el balanceo de carga dinámico se divide en:

- Balanceo de carga dinámico centralizado. Se corresponde con la estructura típica de Maestro/Esclavo.
- Balanceo de carga dinámico distribuido o descentralizado. Se utilizan varios maestros y cada uno controla a un grupo de esclavos.

3.1.2.1 BALANCEO DE CARGA DINÁMICO CENTRALIZADO

El nodo maestro es el que tiene la colección completa de tareas a realizar. Las tareas son enviadas a los nodos esclavos. Cuando un nodo esclavo finaliza una tarea, solicita una nueva al maestro. Esta técnica también se denomina programación por demanda o bolsa de trabajo, y no sólo es aplicable a problemas que tengan tareas de un mismo tamaño.

En problemas con tareas de distintos tamaños es mejor repartir primero aquellas que tengan una mayor carga computacional. Si la tarea más compleja se dejase para el final, las tareas más pequeñas serían realizadas por esclavos que después estarían esperando sin hacer nada hasta que alguno completara la tarea más compleja. También se puede utilizar esta técnica para problemas donde el número de tareas pueda variar durante la ejecución.

En algunas aplicaciones, especialmente en algoritmos de búsqueda, la ejecución de una tarea puede generar nuevas tareas, aunque al final el número de tareas se debe de reducir a cero para alcanzar la finalización del programa. En este contexto se puede utilizar una cola para mantener las tareas pendientes. Si todas las tareas son del mismo tamaño y de la misma importancia o prioridad, una cola FIFO (First In First Out) puede ser más que suficiente, en otro caso debe analizarse la estructura más adecuada [DORMIDO, 2003].

3.1.2.2 BALANCEO DE CARGA DINÁMICO DISTRIBUIDO O DESCENTRALIZADO

En el balanceo de carga descentralizado intervienen varios maestros los cuales tendrán el control de un grupo diferente de esclavos.

Una gran desventaja del balanceo de carga dinámico centralizado es que el nodo maestro únicamente puede repartir una tarea cada vez, y después de que haya enviado las tareas iniciales sólo podrá responder a nuevas peticiones de una en una. Por tanto, se pueden producir colisiones si varios esclavos solicitan peticiones de tareas de manera simultánea. La estructura centralizada únicamente será recomendable si no hay muchos esclavos y las tareas son intensivas computacionalmente. Para tareas de grano fino (tareas con muy poca carga computacional) y muchos esclavos es apropiado distribuir las tareas en más de un sitio [DORMIDO, 2003].

4. ESTADO DEL ARTE

La revisión de la literatura sobre la problemática de los cluster de computadores viene siendo explorada por diferentes comunidades debido a que esta solución ofrece un equilibrio entre el desempeño deseado y el costo del sistema. A continuación se incluirán algunos de los trabajos realizados con cluster de computadores desde sus inicios en el que era conocido como supercomputador hasta la actualidad.

Los supercomputadores toman vida en los años 40 con la creación del ENIAC (Electronica Numeral Integrator and Computer), el cual, según [PERERA, 1999], fue un computador electrónico digital desarrollado por John Mauchly y John Presper Eckert para fines generales a gran escala. En su época fue considerado la máquina más grande el mundo y era controlado a través de un tren de pulsos electrónicos. El ENIAC estaba dividido en 30 unidades autónomas, las cuales eran capaces de operar simultáneamente logrando así la realización de varias tareas y cálculos en paralelo. Este computador nunca pudo funcionar las 24 horas todos los días. Estuvo en funcionamiento hasta 1955 con mejoras y ampliaciones, y se dice que durante su vida operativa efectuó más cálculos matemáticos que los realizados por toda la humanidad anteriormente.

Durante los años 50 el considerado padre de la supercomputación Seymour Cray trabajó en ERA (Engineering Research Associates). Allí, fue uno de los responsables del diseño del UNIVAC 1103 (Universal Automatic Computer), el cual fue el primer computador fabricado para fines comerciales y el primero en utilizar un compilador para traducir lenguaje de programa a lenguaje de máquinas. El UNIVAC 1103 tenía la capacidad de hacer interrupciones [LANZA, 1999].

Basados en [GARCÍA, 2003], en la década de los 60 aparecen las primeras máquinas paralelas, aunque con muy poco éxito comercial. La universidad de Illinois desarrolla el ILLIAC IV, el cual poseía 64 elementos de cálculo, que disponían cada uno de su propia memoria y estaban todos físicamente unidos por medio de una red de interconexión propia del sistema. En la década de los 60 se crea además una auténtica industria informática alrededor de los computadores de la época evolucionados respecto a los primitivos y ello propiciado por el progreso de la electrónica en cuanto al tamaño de los circuitos, su costo de fabricación y su rapidez. La mayor parte de las máquinas paralelas de esta época, máquinas Cyber, se conocían como calculadores vectoriales y poseían un solo procesador.

De [EVIUXA, 2004], a principios de los años 70 la aplicación predominante del computador era el procesamiento de datos administrativos.
Las comunidades de ingenieros y científicos tenían una necesidad apremiante de computadores más potentes. En respuesta a esa necesidad, los diseñadores de máquinas empezaron a trabajar en lo que ahora se conoce como Supercomputadores.

En el año 1972 Seymour Cray fundó Cray Research en Wisconsin, con el compromiso de dedicarse a construir exclusivamente supercomputadores y además de uno en uno por encargo. El primer sistema Cray-1 fue instalado en el laboratorio “Los Alamos” en 1976 y era el único en su diseño ya que incorporaba el primer ejemplo práctico en funcionamiento de procesador vectorial, junto con el procesador escalar más rápido del momento [LANZA, 1999].

En la investigación espacial, la utilización de computadores se convirtió en esencial. La nave Voyager 2, que fue lanzada el 20 de agosto de 1977 con la misión de explorar los planetas exteriores al sistema solar, iba equipada con seis máquinas diferentes, con capacidad de 540 Megas, algo portentoso para la época [EVIUXA, 2004].

A finales de los 70 aparecen los sistemas paralelos multiprocesador, que introducen varias categorías: máquinas vectoriales multiprocesador, multiprocesadores de memoria distribuida y las máquinas asíncronas [GARCÍA, 2003].

Así continúa el desarrollo de los supercomputadores con el transcurrir del tiempo, los cuales fueron adquiriendo costos bastante elevados que no todas las entidades investigativas podían incurrir; por lo tanto se vieron forzados a buscar alternativas más asequibles en cuanto a costos y que ofrecieran los mismos resultados o incluso mejores.

De acuerdo a [LIZÁRRAGA, 2002], en 1994, se integró el primer cluster de computadores en el Centro de Vuelos Espaciales Goddard de la NASA, para resolver problemas computacionales que aparecen en las ciencias de la tierra y el espacio. Los pioneros de este proyecto fueron Thomas Sterling, Donald Becker y otros científicos de la NASA. El cluster de PCs desarrollado tuvo una eficiencia de 70 megaflops (millones de operaciones de punto flotante por segundo). Los investigadores de la NASA le dieron el nombre de Beowulf a este cluster.

En 1996, hubo también otros dos sucesores del proyecto Beowulf de la NASA. Uno de ellos es el proyecto Hyglac desarrollado por investigadores del Instituto Tecnológico de California (CalTech) y el Laboratorio de Propulsión Jet (JPL), y el otro, el proyecto Loki construido en el Laboratorio Nacional de Los Alamos, en Nuevo México. Cada cluster se integró con 16 microprocesadores Intel Pentium Pro y tuvieron un rendimiento sostenido de más de un gigaflop con un costo menor a $50,000 dólares.

Debido al elevado potencial de procesamiento paralelo y distribuido que los cluster ofrecen, muchos grupos de investigación han buscado soluciones que permitan el máximo aprovechamiento de estos [CHIOLA, 1998]. Hoy en día, la existencia de superordenadores que trabajen en tiempo real, se ha convertido en una necesidad [EVIUXA, 2004]. Lo que se busca realmente en la actualidad, es que los nuevos supercomputadores alcancen velocidades cada vez mayores con el fin de que se puedan solucionar fácilmente diversas aplicaciones del mundo real.

En 1999, Hoganson [HOGANSON, 1999] trabaja en un modelo analítico para analizar el desempeño de clusters interconectados, el cual explora diferentes estrategias de localización de procesos en procesadores en un ambiente multitareas. Haciendo uso de heurísticas, el modelo presenta la mejor relación entre tamaño del cluster en aplicaciones paralelas dedicadas a cada segmento del cluster aumentando el paralelismo y reduciendo la sobrecarga de la comunicación. Tres años después, Nguyen y Peirre [NGUYEN, 2001] también crearon un modelo analítico de una experimentación en un simulador de un sistema de archivos paralelos analizando básicamente el factor de la escalabilidad de los cluster. Como resultado, se mostró que el crecimiento de un cluster depende del grado de saturación de redes de comunicación. La saturación de redes de comunicación limita la escalabilidad del cluster.

Andresen y sus compañeros en el 2003 [ANDRESEN, 2003] presentaron un sistema de monitoreo de comunicación en cluster Beowulf. El sistema fue bautizado DISTOP y puede ser utilizado a nivel de proceso con bajo consumo de recursos.

En el 2005 Avresky y Natchev [AVRESKY, 2005] trabajaron en un algoritmo de reconfiguración dinámica de redes para tolerar fallas con múltiples nodos y enlaces, especialmente en redes de alta velocidad con topología arbitraria. En el mismo año Haase y sus compañeros trabajaron en el SDVM un cluster de computadores heterogéneos, que permite operar en cualquier topología de red y con capacidad de expansión [HAASE, 2005].

De [REGO, 2006], la aplicabilidad de los cluster ha crecido con el pasar de los años en áreas como: meteorología, geografía, genética, física cuántica, computación grafica, junto con el número de comunidades científicas y comerciales que los implementaron debido a su facilidad de aumentar las capacidades de procesamiento a un costo relativamente accesible. Sin embargo, conforme los cluster conquistaban nuevas áreas de aplicación se fueron apreciando gran cantidad de dificultades en cuanto a la falta de escalabilidad, la baja disponibilidad y al poco desempeño. Estas dificultades están siendo superadas con el uso de técnicas de tolerancia a fallos, balanceo de carga, optimización de códigos paralelos y bibliotecas, incluso combinaciones entre estas.

Para maximizar el rendimiento de cluster heterogéneos, se han desarrollado diversas técnicas de balanceo de carga, las cuales se pueden utilizar para distribuir la carga dentro de los recursos evitando mayores recursos obsoletos y permitiendo que sea más eficiente la ejecución de tareas.

Teniendo en cuenta que el balanceo de carga es el esfuerzo por mantener todos los procesadores del cluster realizando algún trabajo productivo [GEORGE, 1999], a continuación se hará una revisión de algunos trabajos realizados en esta problemática y su aporte a la sociedad.

En 1997 Decker presenta la herramienta VDS, que distribuye automáticamente los paquetes generados por aplicaciones paralelas y balancea la carga entre los nodos de una red [DECKER, 1997].

El espacio virtual de datos (VDS) es un estándar de la librería C que distribuye automáticamente los paquetes de trabajo generado por aplicaciones paralelas a través del procesamiento de los nodos. VDS es un sistema universal que ofrece mecanismos de balanceo de carga para aplicaciones que incorporan elementos de carga independiente y programación de algoritmos para estos, ya que comprenden limitaciones de precedencia entre sus diferentes tareas. Este trabajo presenta los conceptos de VDS y muestra algunos resultados de desempeño obtenidos para aplicaciones de punto de referencia sintético.

Bohn y Lamont en 1999 investigan técnicas para realizar el balanceo de carga asimétrico en un cluster heterogéneo, mostrando que cuando la diferencia de poder de procesamiento entre los nodos de un cluster es pequeña los beneficios alcanzados con el balanceo asimétrico son pequeños, en caso contrario, se obtiene ganancia de desempeño evitando los procesadores más lentos [BOHN, 1999]. Con supercomputadores comerciales y cluster homogéneos de computadores, se logra el balanceo de carga estático mediante la asignación equitativa de tareas para cada procesador o nodo. Con cluster heterogéneos, el diseño de sistemas tiene la opción de añadir nuevo hardware que es más poderoso que el hardware existente. Cuando esto se hace, la asignación de tareas de forma equitativa para cada procesador da como resultado un buen rendimiento.

La investigación de Bohn y Lamont aborda las técnicas por las que el tamaño de la tarea asignada a un procesador es una buena opción. Así, los procesadores más potentes realizaran mayor cantidad de trabajo que los procesadores menos potentes. Se encontró que cuando el rango de poder de procesamiento es estrecho, con balanceo de carga asimétrico se puede lograr algún beneficio. Pero si el rango de procesamiento es amplio, se logran grandes mejoras en el rendimiento.

Los experimentos realizados muestran hasta un 92% de mejora usando el balanceo de carga asimétrico. Concluyen que el balanceo de carga asimétrico es una buena herramienta que puede ser usada en problemas de descomposición de datos. Sin embargo sólo funciona con no más de 8 nodos, ya que hace que algunos procesadores se queden ociosos.

También en 1999 Bevilacqua, trabaja con cluster heterogéneos. Él propone un método eficiente de balanceo de carga en un cluster de estaciones de trabajo. Básicamente un nodo maestro es el responsable de enviar la carga a cada nodo ocioso una vez que cada uno lo solicite. Cuando el maestro no esté atendiendo los pedidos, este trabaja los datos que almacena. Los resultados experimentales alcanzan una eficiencia superior al 90% [BEVILACQUA, 1999].

En este trabajo se presenta un método de balanceo de carga para aplicaciones de datos paralelas basada en la asignación dinámica de datos.

Es adecuado para los cluster de estaciones de trabajo, aún si la carga cambia dinámicamente. Este método se basa en un modelo modificado de administración de trabajo y logra equilibrar la carga de trabajo maximizando el tiempo de CPU para todos los nodos involucrados. Además, puede reducir el tiempo necesario para la comunicación de la distribución de datos y para la colección de resultados. Este modelo es conseguido a través de un proceso de trabajo ajustado. El gerente (maestro), sostiene todos los trabajos disponibles y satisface las peticiones de prioridad para aquellos nodos esclavos que se encuentran inactivos y desean realizar más trabajo.

Trabaja con un método llamado pool-based, en donde el maestro almacena todo el trabajo disponible y los esclavos inactivos realizarán peticiones de trabajo. En este esquema el maestro usa su tiempo de inactividad para procesar datos y poner las peticiones que llegan en una cola usando la estrategia FIFO (primero en llegar, primero en salir).

El método heurístico en el trabajo de Bevilacqua es simple: cuando el maestro no desempeña ninguna tarea de control, el debe trabajar, pero nunca deberá escoger trabajo de mas que lo sobrecargue. Sigue el paradigma maestro-esclavo, en el cual el maestro es responsable de los esclavos, de la asignación de tareas y de la colección de resultados.
Una desventaja de esta técnica ocurre cuando el maestro se encuentra demasiado ocupado mientras ejecuta tareas, por lo que deja que los nodos inactivos se queden ociosos aún si hay tareas por ser ejecutadas. Trabaja en un cluster heterogéneo, usando entorno multitareas y no se interesa en las mediciones de rendimiento, por lo que la carga externa no es cuantificada exactamente a diferencia de otros estudios que la mantienen constante.

Liao y Chung en ese mismo año (1999) proponen tres métodos de balanceo de carga basado en árboles: The Maximum Cost Spanning Tree Load Balancing - MCSTLB, The Binary Tree Load Balancing - BTLB y The Condensed Binary Tree Load Balancing – CBTLB, para distribuir la carga en los procesadores, con el fin de dar solución a diversos problemas que tienen que ver con la solución adaptativa de programas de aplicación de elementos finitos en multicomputadores con memoria distribuida. Finalmente encuentran que el método que mejor desempeño presenta entre los tres es el CBTLB, el cual construye un árbol binario [LIAO, 1999].

En el modelo de elementos finitos un objeto puede ser visto como un grafo, en el cual hay cierta cantidad de elementos finitos, cada uno de estos elementos está compuesto por un número de nodos. Es ampliamente utilizado para el modelamiento estructural de sistemas físicos.

En el año 2000, Aversa y Bestavros proponen validar la implementación de un prototipo de servidor Web distribuido con el fin de sacar el mejor provecho de balanceo de carga. En este trabajo se obtienen resultados positivos en cuanto a escalabilidad y costo para aplicaciones pequeñas [AVERSA, 2000].

Presentan la evaluación e implementación de un prototipo escalable de servidor web que consiste en el balanceo de carga en un cluster de host que acepta y ejecuta conexiones TCP.

La direccion IP para cada host se obtiene mediante DNS por medio del algoritmo Round Robin, permitiendo que cualquier host reciba peticiones de cualquier cliente.

En el prototipo empleado cada host posee información sobre el resto de los host del sistema. La información de carga se mantiene periódicamente usando mensajes multicast entre los host del cluster.

La parte de comunicación es bastante importante, por el hecho de que cada nodo posee información del resto de los nodos del cluster. El sistema es altamente escalable, factor de gran importancia en la red. No existe ningún nodo maestro encargado de la distribución de la carga, simplemente cada nodo decide si puede o no realizar el trabajo que le fue pedido por parte del cliente.

Para el 2002, el principal objetivo de Kacer y Tvrdík fue el de verificar el adecuado balanceo de carga para procesos cortos usando intensamente el procesador. Como resultado de este estudio se propuso una técnica de ejecución remota de procesos, la cual fue comparada con la estrategia de balanceo adoptada en Mosix. Con las pruebas realizadas, los resultados apuntaron que la técnica propuesta fue superior a la migración de procesos utilizada en Mosix en muchos de los casos y presenta resultados semejantes en los demás [KACER, 2002].

Para cargas bajas funciona mejor Mosix, pero para grandes cargas los resultados de este trabajo son mejores debido a que Mosix presenta mucha sobrecarga por el proceso de migración.

Por la complejidad de los cluster y los diferentes factores que tienen un impacto directo con el desempeño, en este trabajo no se realizó un modelo realista, sólo poseen resultados experimentales. Usan un cluster homogéneo.

Ibrahim y Xinda evalúan el desempeño de sistemas paralelos que usan balanceo de carga dinámico en términos de tiempo de ejecución, velocidad y eficiencia en ejecución de una versión paralela de un algoritmo de búsqueda de profundidades (depth-first), sobre la plataforma MPI [IBRAHIM, 2002].

Trabajan sobre un cluster heterogéneo, el sistema no es escalable, debido a que cuando se aumenta la cantidad de nodos el desempeño del algoritmo disminuye, la máxima cantidad de equipos usados es 8.

En 2003 Choi y sus compañeros proponen una métrica de carga basada en número de tareas las cuales afectan el desempeño del sistema. Los resultados de su trabajo muestran una ganancia de un 11% en el tiempo de ejecución [CHOI, 2003].

En este trabajo se estudia un nuevo método que previene la disminución de la eficiencia en asignaciones de trabajo iniciales empleando un método de eliminación de ganancia de información por estimación. Con este fin, han desarrollado una nueva métrica de carga

llamada “El número eficaz de tareas”, el cual contiene información acerca de ambos sistemas de carga y la utilización del recurso.

Este tipo de balanceo de carga, conocido como un sistema de localización de tareas inicial, requiere de conocimiento del uso de los recursos de las tareas para que se ajuste de una manera adecuada a las capacidades de cada nodo. Desde la distribución inicial requiere que las tareas sean programadas antes de la ejecución (balanceo de carga estático), todos los recursos usados deben ser proporcionados en términos de la predicción. Este acercamiento puede afectar seriamente el tiempo de ejecución cuando la predicción es inadecuada. Los resultados demuestran que el sistema tiene hasta 11% menor tiempo de ejecución que los acercamientos convencionales usando estimaciones históricas basadas en comportamientos estimados.

Trabajan en una red ethernet con un cluster heterogéneo a nivel de hardware donde todos los nodos esclavos son iguales y el único diferente es el maestro. La métrica de desempeño es el tiempo de ejecución.

Chau y Fu proponen otra técnica de balanceo de carga que se basa en arquitecturas hipercubo con el objetivo de reducir el tráfico de mensajes [CHAU, 2003].

En 2003 DrozDowski y Wolniewicz proponen el modelo de carga divisible, el cual puede ser utilizado cuando las aplicaciones permiten dividir la aplicación en partes de tamaño arbitrario. La principal contribución de este trabajo fue la modelación matemática del problema de carga divisible utilizando jerarquías de memoria [DROZDOWSKI, 2003].

En este trabajo se propone un nuevo modelo matemático para el procesamiento distribuido de carga divisible. El modelo es basado en programación lineal y es capaz de representar pieza por pieza funciones de tiempo de procesamiento lineales convexas de la carga asignada.

Para la realización de las pruebas iniciales formulan un modelo de carga divisible como un programa lineal usando un cluster homogéneo con dos (2) procesadores, en donde obtuvieron resultados no muy buenos. Posteriormente trabajaron con un cluster heterogéneo en donde los resultados fueron más desalentadores obteniendo desequilibrio de carga.

Al ver que los resultados no fueron muy buenos formulan un simple algoritmo de multientrega de tareas para dividir la carga de procesamiento y un método para ajustar sus parámetros. La eficiencia del procesamiento de carga divisible por multientrega de tareas resulto ser ventajoso para seleccionar los tamaños de la carga que iba a ser asignada. En este trabajo no se asume el tiempo de respuesta.

Attiya y Hamam con el objetivo de reducir el tiempo de ejecución del propio algoritmo de distribución de carga, proponen en 2004 un algoritmo ejecutado en dos fases, cada una de

estas fases está basada en una heurística diferente: Simulated Annealing y Branch-and-Bound [ATTIYA, 2004].

En este trabajo presentan un modelo matemático para el problema del balanceo de carga. Proponen un algoritmo óptimo y eficiente para la memoria que se compone de dos fases para la asignación de tareas a cada uno de los procesadores que conforman un sistema distribuido heterogéneo, con el fin principal de reducir al máximo el tiempo de procesamiento de los nodos que ejecutan la carga. El algoritmo inicialmente encuentra un valor óptimo aplicando Simulated Annealing (SA) y luego por medio de la técnica Branch-and-Bound (BB) se encuentra una distribución óptima considerando la solución de la técnica SA como la solución inicial. El algoritmo propuesto supera la baja calidad de las soluciones que son obtenidas usando heurísticas.

Usan una topología de red tipo bus, cosa que no es necesariamente buena por el hecho de que se pueden producir colisiones y problemas de tráfico, además del hecho de que si se desea escalar el sistema puede empeorar la parte de comunicación entre nodos. El cluster sólo contiene 4 nodos, lo que realmente es muy poco para mostrar la verdadera efectividad en el método.

Velasco y Ramos en 2004 comparan tres técnicas de localización de peticiones dinámicas para la evaluación del desempeño de la calidad del servicio percibido por el usuario. Ellos trabajan en un simulador llamado SIDE usando un modelo de comportamiento de tráfico web, donde finalmente obtienen una buena técnica dinámica con la que logran balancear la carga en las limitaciones que tiene cada cliente en el momento de acceder a un servidor de búsqueda llegando a obtener buenos tiempos de respuesta [VELASCO, 2004].

En este trabajo comparan tres técnicas: Random technique (RND), Market-based technique (MKT) y Bilateral technique (BIL), en donde las tres técnicas usan el tiempo de respuesta total de las peticiones asignadas y la sobrecarga inducida, implementando técnicas de negociación para estas tres técnicas, con el único fin de que el rendimiento del sistema sea bueno. La técnica RND es la de menor desempeño aunque no produce mucha sobrecarga comparada con la técnica MKT en la cual el desempeño es alto, debido a su mecanismo de actualización de calidad del servicio, sin embargo esto incurre en una alta sobrecarga también. Por lo tanto se concluye que la estrategia BIL es la más efectiva mostrando alto desempeño y bajos costos en comparación con RND y MKT.

Los resultados de este trabajo son comparados con un proceso de Poisson no homogéneo, donde la comparación cualitativa muestra que el mejoramiento de la calidad del servicio percibido por los usuarios se mantiene. En el aspecto cuantitativo, este trabajo muestra tiempos de respuesta mayores que los presentados anteriormente, la razón, es porque el modelo de carga de trabajo del proceso de Poisson muestra baja variabilidad y tiempos de respuesta uniforme.

En 2004 Savvas y Kechadi propusieron un algoritmo llamado PSTS, en el cual la idea principal era dividir recursivamente el cluster en subespacios y encontrar la dimensión que

suministrara el mejor desempeño. La propuesta se mostró eficiente para sistemas que permanecen desbalanceados por largo tiempo [SAVVAS, 2004].

En este estudio es empleado un ambiente de cluster de computadores como plataforma computacional. Proponen un algoritmo de programación de tareas dinámicas, el cual balancea la cantidad de carga en los nodos de un cluster. La técnica es dinámica, sin preferencias y adaptable, y utiliza una mezcla de políticas centralizadas y descentralizadas. Basado en el principio divide y vencerás el algoritmo modela el cluster como hiper-grid y balancea la cantidad de carga de este. Todos los nodos del cluster son casi igualmente cargados, sólo seleccionan por orden el nodo que logra el mejor rendimiento y determinan los puntos críticos en el algoritmo.

El algoritmo tiene la ventaja de ser altamente escalable, ya que trabaja bastante bien cuando el grid tiene grandes dimensiones. Además de que es un sistema paralelo y eficaz.

Se trabaja con la unión de dos cluster, sin embargo ambos son homogéneos. Por lo que sería útil trabajar con cluster heterogéneos. Por otro lado no trabajan con tareas orientadas a los nodos. Proponen trabajar a futuro con redes irregulares donde el problema de enrutamiento pase a ser un factor de gran importancia.

Para el año 2006 Rego propone el uso de algoritmos de balanceo de carga sobre la plataforma LAM/MPI. La validación de su trabajo se hace sobre aplicaciones diferentes, como multiplicaciones matriciales y reconocimiento de secuencias de DNA [REGO, 2006].

Trabajan en un cluster heterogéneo en donde alcanzan una gran ganancia en el desempeño del sistema, esto se debe a que la carga se distribuye de acuerdo a la potencia de cálculo de los nodos.

Como debilidad, se encuentra el factor de la llegada de tareas a gran escala, haciendo que el cluster se sature.

Como no hay forma de saber la cantidad de carga hay problemas al momento de realizar la repartición de esta acorde con la potencia de cálculo de cada nodo, como trabajo futuro esta evaluar la forma en distribuir los procesos sí la carga puede ser variable.

A continuación y en base al estudio realizado en la revisión de la literatura, se puede resumir el estado del arte con los trabajos más representativos en el siguiente mapa conceptual (figura 1):

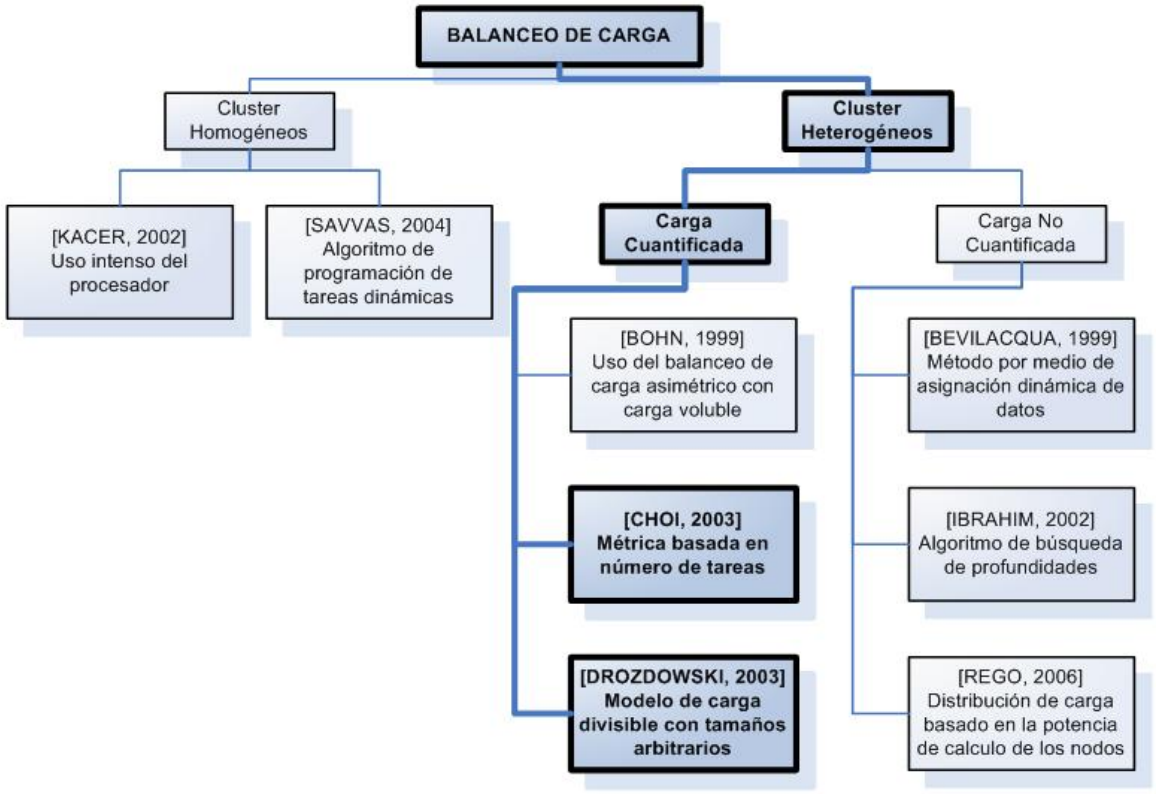


Figura 1. Mapa Conceptual – Resumen Estado del Arte

Donde este trabajo se enfocará en el estudio de cluster heterogéneos cuando se utiliza carga cuantificada.

Los trabajos que fueron más significativos fueron el de Choi y sus compañeros [CHOI, 2003] y el de DrozDowski y Wolniewicz [DROZDOWSKI, 2003], en lo que se encontraron las siguientes fortalezas y debilidades:

Fortalezas

- Uso de cluster heterogéneo
- La métrica de desempeño usada es el tiempo de ejecución
- Uso de carga divisible
- Las tareas son dispuesta por medio de Multi-entrega, esto es, por tandas que contienen varias tareas

Debilidades

- El uso del balanceo de carga estático exige la distribución de tareas antes de la ejecución del programa
- Los recursos usados son por medio de términos de predicción
- El cluster es heterogéneo donde sólo el maestro es diferente a los esclavos
- No se asume el tiempo de respuesta

En base a estas fortalezas y debilidades, el método a proponer en esta investigación tomará en cuenta el uso de un cluster heterogéneo, en donde cada uno de los nodos (maestro y esclavos) sea diferente; se trabajará con el tiempo de ejecución como métrica de desempeño; la carga de trabajo será conocida (no se sabrá cuantas tareas llegarán, pero sí el tamaño de éstas al momento en que lleguen); se atacará el problema del uso del balanceo de carga estático, para trabajar con balanceo de carga dinámico, además será asumido el tiempo de respuesta.

5. METODO DE BALANCEO DE CARGA EN SISTEMAS DE CÓMPUTO DISTRIBUIDO, APLICADO A LA DISTRIBUCIÓN DE LA CARGA CON RESPECTO AL TAMAÑO DE LAS TAREAS Y A LA CAPACIDAD DE LOS NODOS ESCLAVOS

En este trabajo el método que se propone consiste en la distribución de tareas en los nodos esclavos por parte del nodo maestro, el cual recopilará una a una las tareas conforme van llegando a medida que pasa el tiempo. La asignación de las tareas a los nodos esclavos será realizada por el nodo maestro y dependerá del tamaño de las tareas y de una función de aptitud, la cual es calculada por el nodo maestro después de que el nodo esclavo envía ciertos atributos (velocidad de procesamiento, capacidad de memoria y tiempo de respuesta). Después de que el nodo maestro asigna una tarea a cierto nodo esclavo, el esclavo ejecutará la tarea y enviará al nodo maestro una notificación de terminación de tarea, indicándole así al nodo maestro que esta listo para ejecutar una nueva tarea.

En este capítulo se pretenderá dar solución a los primeros cuatro (4) objetivos específicos de esta investigación, ya que el último se presentará en el sexto (6) capítulo.

5.1 SELECCIÓN DEL TIPO DE CLUSTER A MODELAR

En este trabajo se propondrá un método para balancear la carga de trabajo tratando de optimizar al máximo el tiempo total de ejecución. Considerando esto y en base a lo descrito en el capítulo 2, el tipo de cluster seleccionado es el de alto rendimiento (HP, high performance), ya que cuando la palabra cluster es pronunciada, la primera cosa que pasa por la cabeza es alto desempeño. De acuerdo a [REGO, 2006], este tipo de cluster es el más común entre las comunidades científicas, sistemas predictivos, simulaciones y tarifas típicas que exigen alto poder de procesamiento. Su función es sencilla: dado un problema complejo e identificado como paralelizable, un servidor (maestro) debe ser responsable de dividir este problema en numerosas partes para ser procesadas en nodos esclavos (nodos dedicados al procesamiento). Así, una vez que cada nodo esclavo encuentre una solución, este la envía al nodo maestro para que el maestro presente la solución completa del problema.

No se seleccionó un cluster de alta confiabilidad ni de alta disponibilidad debido a que se pretende reducir el tiempo de ejecución de una cierta cantidad de tareas en un cluster heterogéneo a nivel de hardware acogiendo un grupo de tareas y ejecutándolo simultáneamente entre los nodos a pesar de sus diferencias en el menor tiempo posible, en vez de mantener al máximo la seguridad en el entorno mientras se ejecutan las tareas o de garantizar siempre el almacenamiento de tareas a pesar de posibles fallas en alguno de los nodos conectados en el cluster y suministrando servicios siete (7) días a la semana

veinticuatro (24) horas al día, proporcionando a los usuarios confiabilidad en el momento de su acceso, es decir, cuando requieran el servicio les será brindado con un tiempo de respuesta aceptable.

En esta investigación lo que se quiere realmente es utilizar el cluster para procesar gran cantidad de información, en nuestro caso, gran cantidad de tareas. Por lo tanto el cluster que hemos seleccionado esta clasificado como de alto rendimiento ya que se busca que adquiera la funcionalidad de herramienta útil para los investigadores de nuestra sociedad actual en el ámbito científico, para así obtener la mayor cantidad de procesamiento o bien, ejecución de la máxima cantidad de tareas en el menor tiempo posible, para así ofrecer un servicio de gran utilidad y de la forma más optima posible.

5.2 TÉCNICA DE BALANCEO DE CARGA

En esta investigación se propone trabajar con el balanceo de carga dinámico centralizado, el cual según [DORMIDO, 2003], consiste en que la asignación de la carga es efectuada por un único nodo maestro a todos sus esclavos.

El nodo maestro será el encargado de recibir todas las tareas que se deberán ejecutar y tomar la mejor decisión al momento de realizar la repartición de éstas teniendo siempre en cuenta cuál nodo es más o menos fuerte dentro del cluster. En el método que se planteará posteriormente se verá que el nodo maestro no realizará ninguna de las tareas, sólo las distribuirá. De ahí, se descarta todo lo que concierne con el balanceo de carga estático, donde el nodo maestro participa activamente de la ejecución de las tareas en la computación una vez que haya asignado el trabajo a los nodos esclavos realizando este proceso una vez o cíclicamente.

En este método también es descartado el balanceo de carga descentralizado debido a que se plantea que el nodo maestro no se encuentre efectuando ninguna tarea en cuanto a ejecución, dicho nodo sólo será encargado de repartir la carga a sus nodos esclavos. En este tipo de balanceo de carga (descentralizado) se utilizan varios nodos maestros y cada uno controla a un grupo de nodos esclavos diferentes, cosa que en este trabajo sería poco útil, ya que lo que se requiere es el poder de procesamiento de varios esclavos y no el mando de varios maestros.

Según [DORMIDO, 2003] la estructura centralizada únicamente será recomendable si no hay muchos esclavos y las tareas son intensivas computacionalmente. Por lo tanto este es nuestro caso ya que se espera que el método sea eficiente y altamente escalable (aumento progresivo de la cantidad de nodos esclavos con el tiempo).

5.3 MODELOS DE COMPUTACION Y ARQUITECTURAS PARALELAS

En este apartado se presentarán dos mapas conceptuales por medio de los cuales se explicará brevemente el tema de los modelos computacionales y las diferentes arquitecturas paralelas existentes (figuras 2 y 3).

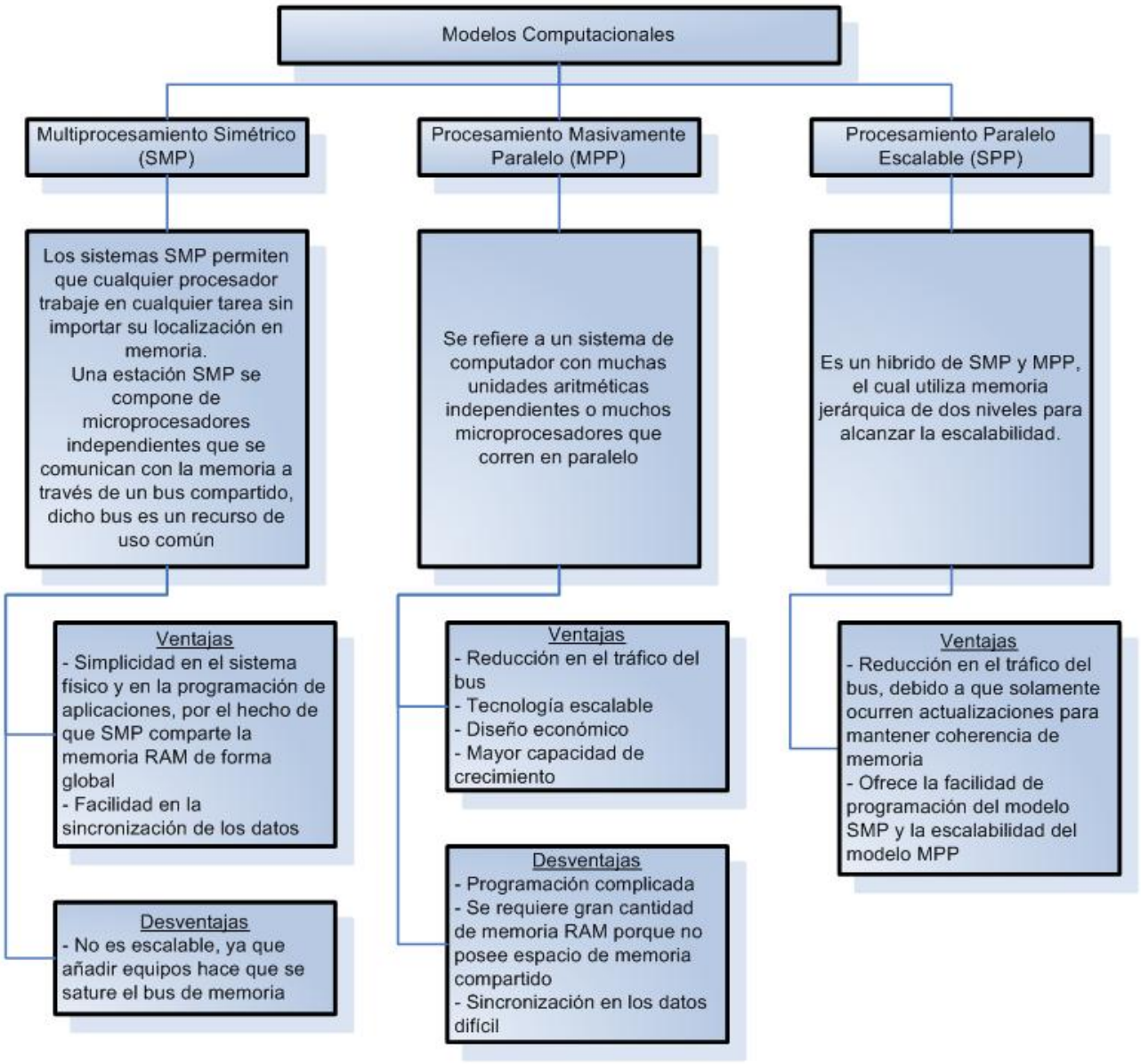
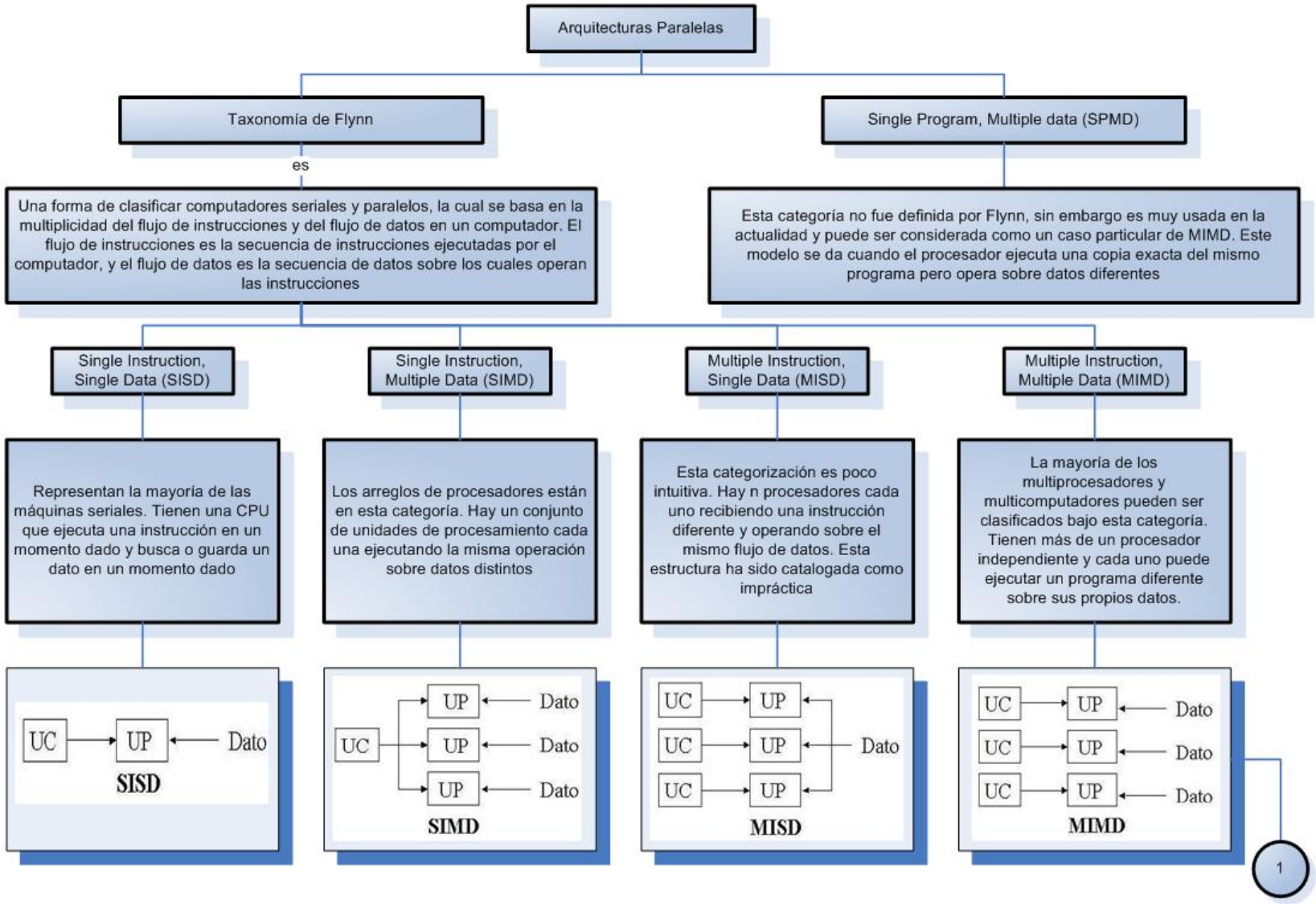


Figura 2. Mapa Conceptual – Modelos Computacionales


Arquitecturas Paralelas
Taxonomía de Flynn
es
Una forma de clasificar computadores seriales y paralelos, la cual se basa en la multiplicidad del flujo de instrucciones y del flujo de datos en un computador. El flujo de instrucciones es la secuencia de instrucciones ejecutadas por el computador, y el flujo de datos es la secuencia de datos sobre los cuales operan las instrucciones
Single Program, Multiple data (SPMD)
Esta categoría no fue definida por Flynn, sin embargo es muy usada en la actualidad y puede ser considerada como un caso particular de MIMD. Este modelo se da cuando el procesador ejecuta una copia exacta del mismo programa pero opera sobre datos diferentes
Single Instruction, Single Data (SISD)
Single Instruction, Multiple Data (SIMD)
Multiple Instruction, Single Data (MISD)
Multiple Instruction, Multiple Data (MIMD)
Representan la mayoría de las máquinas seriales. Tienen una CPU que ejecuta una instrucción en un momento dado y busca o guarda un dato en un momento dado
Los arreglos de procesadores están en esta categoría. Hay un conjunto de unidades de procesamiento cada una ejecutando la misma operación sobre datos distintos
Esta categorización es poco intuitiva. Hay n procesadores cada uno recibiendo una instrucción diferente y operando sobre el mismo flujo de datos. Esta estructura ha sido catalogada como impráctica
La mayoría de los multiprocesadores y multicomputadores pueden ser clasificados bajo esta categoría. Tienen más de un procesador independiente y cada uno puede ejecutar un programa diferente sobre sus propios datos.
UC
UP
Dato
SISD
SIMD
MISD
MIMD
1

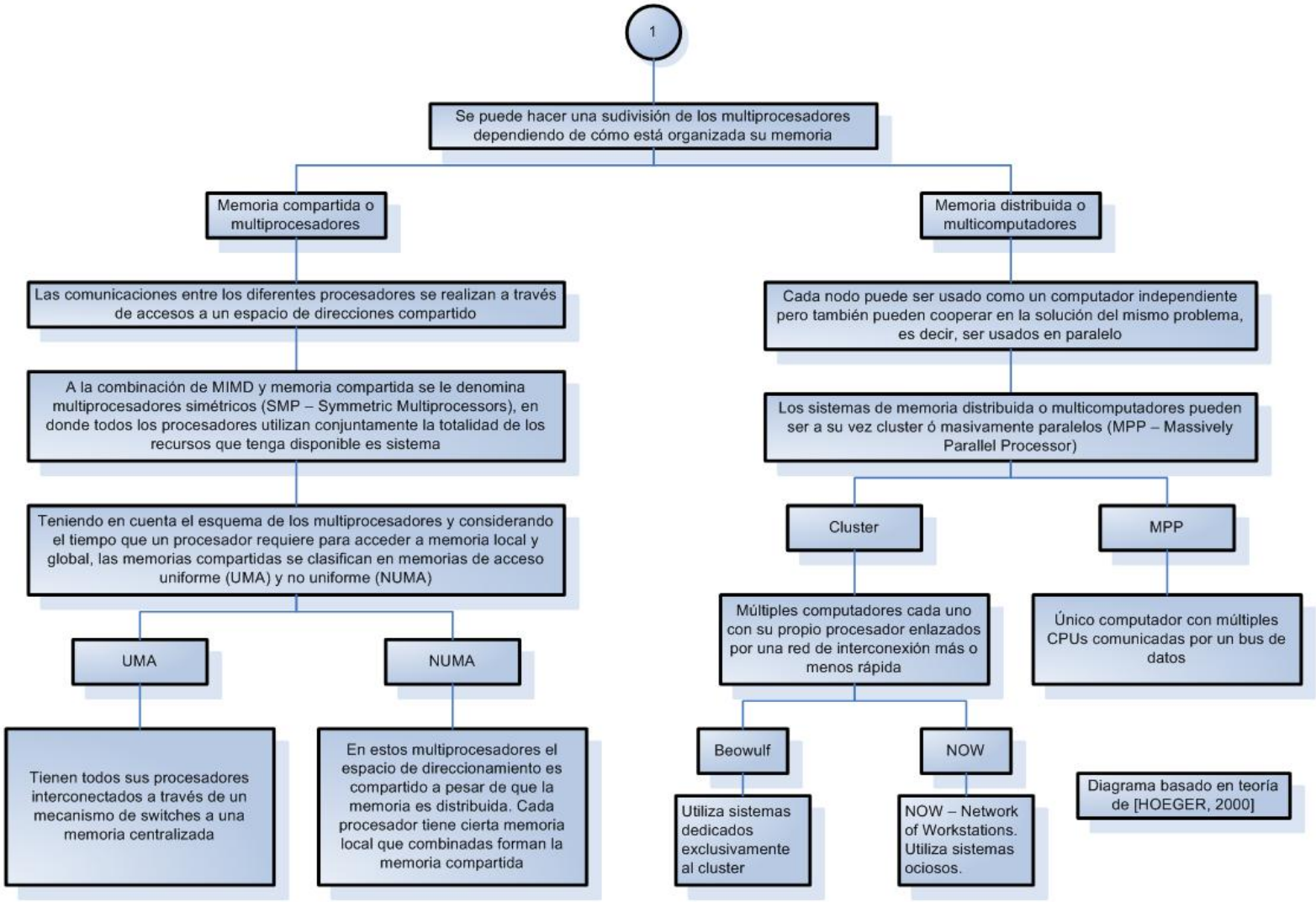


Figura 3. Mapa Conceptual – Arquitecturas Paralelas

Para ampliar un poco más el tema se puede revisar los anexos 9.3.1 y 9.3.3

Luego de haber estudiado las figuras 2 y 3 y todo lo referente al tema se puede concluir que sólo existen dos tipos de arquitecturas que son básicamente paralelas, la arquitectura SIMD (Single Instruction, Multiple Data) y la arquitectura MIMD (Multiple Instruction, Multiple Data), cada una de las cuales posee sus propias virtudes y falencias. Sin embargo en esta investigación se decide adoptar la segunda de estas (MIMD) por el hecho de que en el método a proponer se requiere que cada equipo actúe de forma independiente realizando cada uno un trabajo específico sobre una parte de la tarea completa optimizando así el tiempo de ejecución total y no lo que realiza la arquitectura SIMD que como lo dice [SMERLING, 2001] es cuando cada procesador o equipo ejecuta la misma operación que los demás pero sobre un conjunto particular de datos propios.

Después de haber seleccionado la arquitectura MIMD hay que seleccionar si se desea trabajar con memoria distribuida o compartida.

En esta investigación se opta por trabajar con la memoria distribuida, ya que según [SMERLING, 2001], el concepto de memoria compartida implica la existencia de múltiples procesadores accediendo a un espacio de memoria único y esto en este trabajo no se verá, lo que se realizará realmente es que un nodo se comporte como una estación independiente tal y como lo hace la memoria distribuida representado técnicamente por el paso de mensajes. Los sistemas de memoria distribuida pueden ser de dos tipos: cluster o sistema de procesamiento paralelo MPP. Los clusters como se ha mencionado en capítulos pasados consisten de un conjunto de computadores conectados a una red y los sistemas MPP se encuentran constituidos por varios cientos de procesadores (nodos), los cuales están interconectados mediante una red de conmutación de alta velocidad. Para este trabajo la selección fue trabajar con cluster por el hecho de que se tienen diferentes estaciones de trabajo con capacidades de memoria, un sistema operativo y cualidades de entrada/salida, además de que ofrecen importantes ventajas sobre otros tipos de arquitecturas paralelas.

<u>Ventajas en los cluster</u>

- Cada una de las máquinas de un cluster puede ser un sistema completo utilizable para otros propósitos. Por ejemplo, se puede montar un aula de informática con estaciones que den servicio a los alumnos durante el día y disponer del conjunto durante la noche para realizar cálculos complejos, siendo incluso posible usar parte de la CPU que los alumnos no precisen (“ciclos muertos”) para continuar los cálculos durante el día.
- Los elementos de proceso de un cluster son computadores “normales” y por lo tanto baratos. El hardware de red, es también cada vez más barato. La economía puede alcanzar puntos más altos, teniendo un único monitor, tarjeta de video y teclado para todo el cluster.
- Los cluster escalan bien hasta sistemas muy grandes. No es difícil construir clusters con cientos y hasta miles de computadores.
- Reemplazar un computador defectuoso de un cluster es trivial si se compara con el trabajo necesario para reparar una máquina SMP. Esto permite tener sistemas de

muy alta disponibilidad, siendo posible incluso diseñar un cluster de tal forma que si un nodo falla el resto continúe trabajando.
- Existe mucho soporte software para la programación de clusters. Con el nivel de estandarización actual existe la garantía de que los programas escritos para un cluster funcionarán en cualquier otro con independencia del tipo de procesador de cada nodo.

Sin embargo, los clusters también presentan un conjunto de desventajas:
- Las redes ordinarias no están diseñadas para el procesamiento en paralelo. La latencia es alta y el ancho de banda relativamente bajo si se comparan con los de un sistema SMP. Si el cluster no está aislado del resto de la red, la situación es aún peor.
- Existe muy poco soporte software para tratar un cluster como un sistema único. Por ejemplo, el comando *ps* sólo lista los procesos de un sistema Linux, es decir sólo lista los procesos locales, no los de todo el cluster.

A continuación se presentará en la tabla 1 las principales razones por las que fue escogido trabajar con cluster antes que sistemas SMP y MPP.

Tabla 1. Comparación de características entre SMP, MPP y Cluster
Tomada de [SMERLING, 2001]

Característica	**SMP**	**MPP**	**Cluster**
Número de nodos	2-10 procesadores	2-100	100 o menos
Complejidad por nodo	Media o alta	Baja o media	Media
Comunicación entre nodos	Media o alta	Baja o media	Media
Planificación de las tareas	Una única cola de trabajo	Una única cola de trabajo corriendo en un nodo	Múltiples colas de trabajo pero coordinadas
Soporte para SSI (apariencia de sistema único)	Siempre	Parcialmente	Deseable
Tipo y número de sistema operativo por nodo	Un único sistema operativo monolítico	Numerosos microkernel monolíticos	Numerosas plataformas de sistemas operativos

De [SMERLING, 2001]. Existen dos tipos de clusters, dependiendo de si cada computador del cluster está o no exclusivamente dedicado a él. Si es así se habla de un cluster de clase Beowulf, y si no de una simple red de estaciones de trabajo (NOW, Network Of Workstations).

En este trabajo hubo mayor enfoque en los cluster tipo Beowulf.

Los clusters tipo Beowulf tienen algunas ventajas sobre las redes de estaciones (NOW):

- Al estar todas las CPU al servicio del cluster es más fácil mantener el balanceo de carga, hecho que se destaca en esta investigación. Esto significa lograr que todos los procesadores tengan un grado de ocupación semejante. Además, el único uso de la red es debido a la aplicación paralela que se está ejecutando, lo que permite sincronizar el tráfico y disminuir así la latencia.
- Al ser un entorno más restrictivo, los programas diseñados para NOW no tienen problemas para ejecutarse en Beowulf, pero lo contrario puede no ser cierto.
- En un Beowulf es posible modificar el núcleo para mejorar el procesamiento en paralelo. En Linux es habitual utilizar un módulo que permite la asignación a cada proceso de un identificador global a todo el cluster.
- Un computador de una red ordinaria debe estar configurado para una respuesta interactiva adecuada al usuario pero en un Beowulf se pueden ajustar los parámetros de sistema de acuerdo a la granularidad de la aplicación paralela que se vaya a ejecutar, lo que contribuye a mejorar su rendimiento.
- En cuanto a la exclusividad: cada nodo de un Beowulf se dedica exclusivamente a procesos del supercomputador. La red de estaciones de trabajo, no. En esta última, cada nodo puede ejecutar simultáneamente los procesos correspondientes al NOW y cualquier otro programa mono-procesador. Cuando se quiere usar una red de estaciones NOW se hace desde cualquiera de ellas; cuando se quiere usar un Beowulf se debe hacer a través de un terminal tonto conectado al Beowulf, no siendo posible acceder directamente a la consola de los nodos.
- En cuanto a independencia del nodo: cada nodo de una red de estaciones de trabajo NOW es una estación de trabajo completamente funcional que, con el software apropiado, puede funcionar sin necesidad de participar del sistema distribuido. En un Beowulf, no; cada nodo tiene el hardware mínimo que le permite funcionar como unidad de cálculo. Los nodos carecen de monitores, teclados, disqueteras, ratones; muchas veces de tarjetas de video y hasta de disco duro. Y es frecuente que un Beowulf los nodos no tengan carcasas, y se compongan de una estructura metálica que va a sujetar las fuentes de alimentación y las placas madre.
- Pero si se habla del mecanismo de comunicaciones: en una red de estaciones NOW suele haber un switch central, que es en gran parte el responsable del rendimiento por evitar o no las colisiones. En un Beowulf el mecanismo de comunicaciones es más rudimentario: conexiones placa a placa por cable UTP cruzado con conectores RJ-45.

En la figura 4, se mostrará el resumen de los modelos computacionales y arquitecturas paralelas seleccionados.

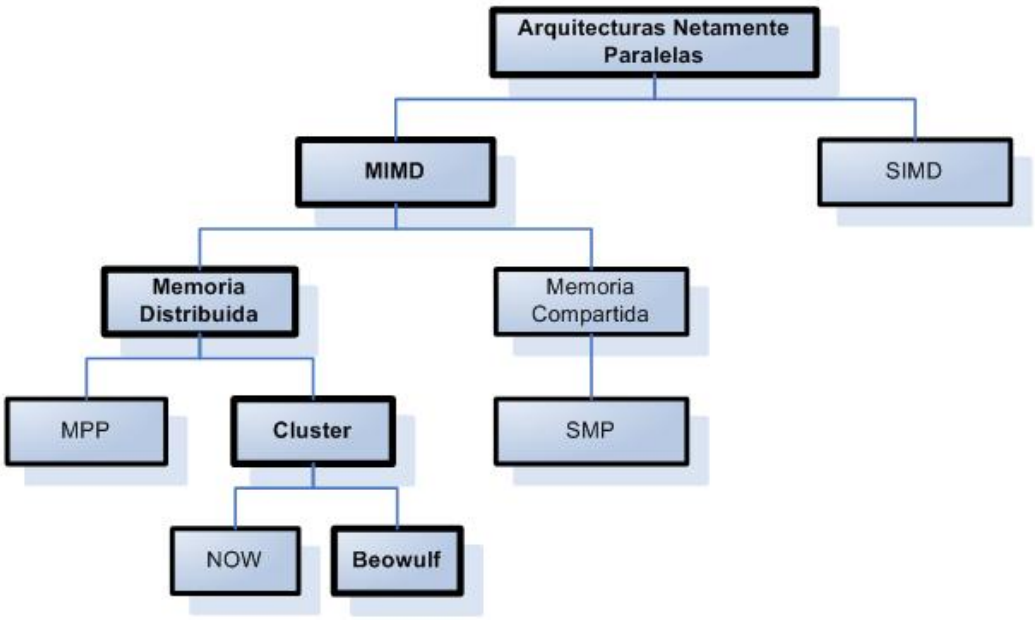


Figura 4. Clasificación de las arquitecturas que son netamente paralelas
Tomada de [DORMIDO, 2003]

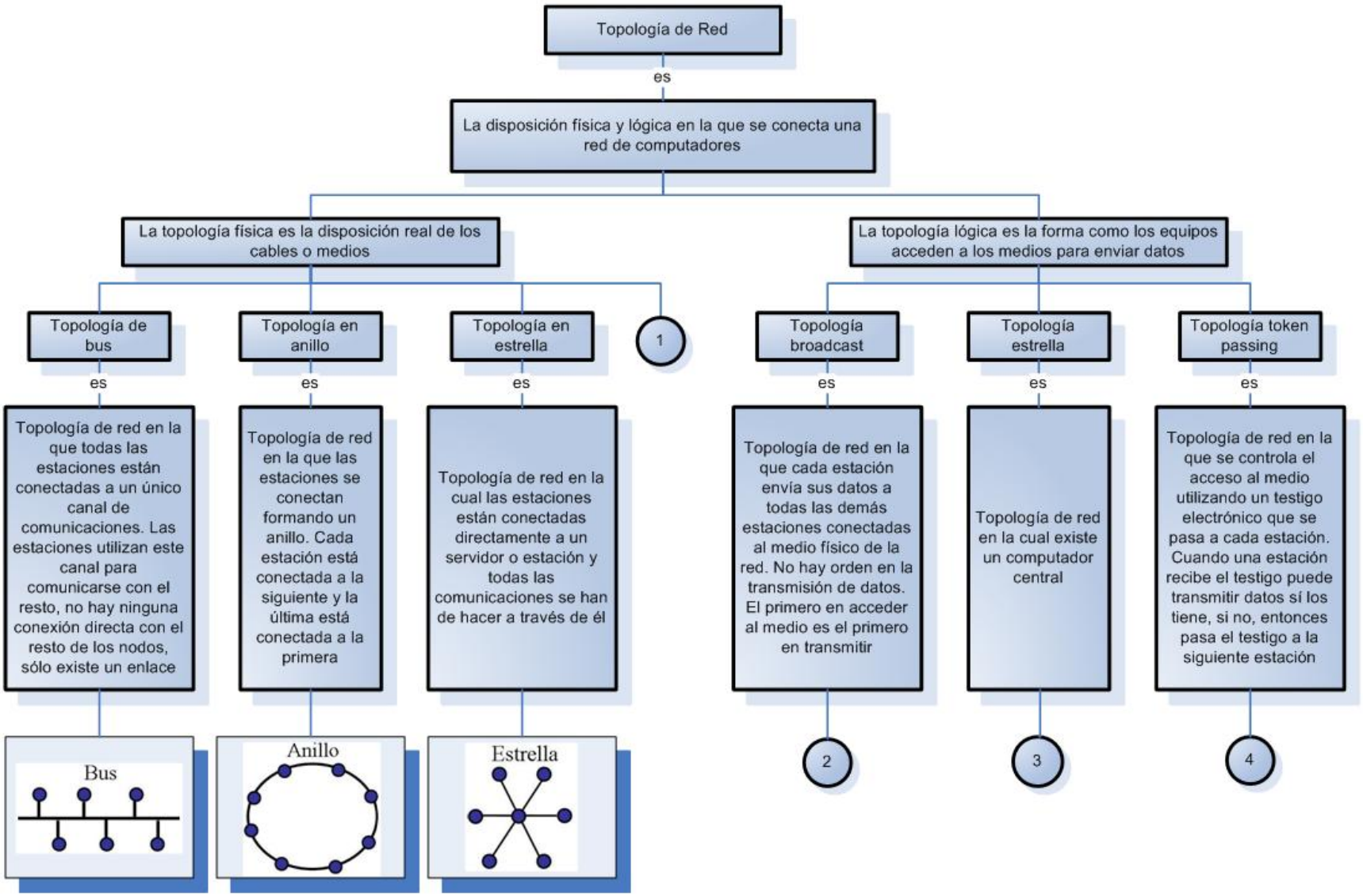
5.4 ARQUITECTURAS EN LAS REDES DE TELECOMUNICACIÓN

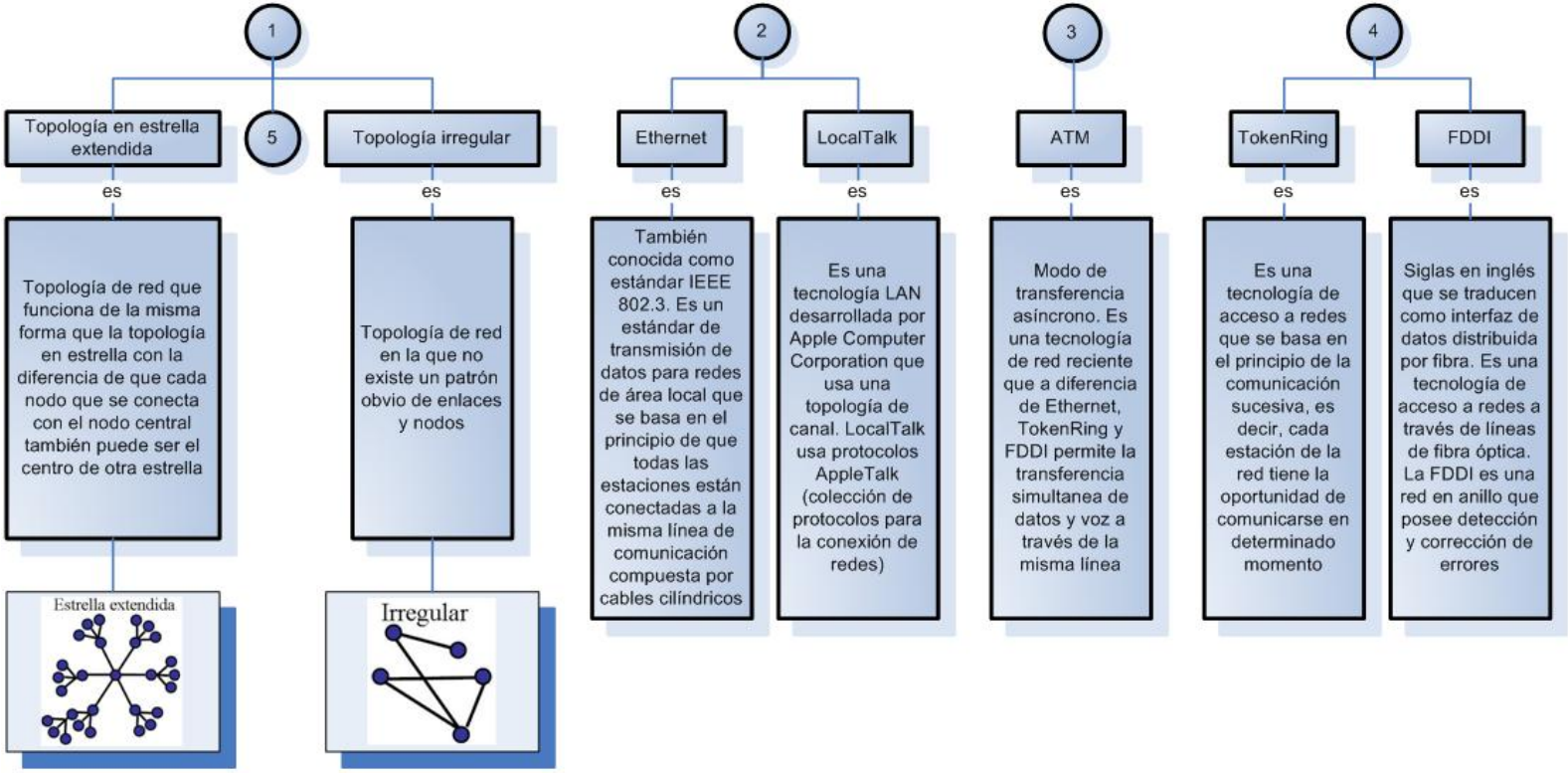
Según [ARAUJO, 2004], las redes de computadores surgieron como una necesidad de interconectar los diferentes hosts de una empresa o institución para poder así compartir recursos y equipos específicos. Pero los diferentes componentes que van a formar una red se pueden interconectar o unir de diferentes formas, siendo la forma elegida un factor fundamental que va a determinar el rendimiento y la funcionalidad de la red. La disposición de los diferentes componentes de una red se conoce con el nombre de topología de la red. La topología idónea para una red concreta va a depender de diferentes factores, como el número de máquinas a interconectar, el tipo de acceso al medio físico que se desee, entre otros.

5.4.1 SELECCIÓN DE LA TOPOLOGÍA DE RED

Antes de realizar la selección de la topología con la que se va a trabajar en esta investigación hay que adentrarse un poco más en este tema de las topologías de red.

A continuación se presentará un mapa conceptual (figura 5) en el que se explicará brevemente este tema, continuando posteriormente con la selección de la arquitectura de telecomunicación más apropiada para el método que se propondrá en el numeral siguiente (numeral 5.5).


Topología de Red
es
La disposición física y lógica en la que se conecta una red de computadores
La topología física es la disposición real de los cables o medios
La topología lógica es la forma como los equipos acceden a los medios para enviar datos
Topología de bus
Topología en anillo
Topología en estrella
1
Topología broadcast
Topología estrella
Topología token passing
es
Topología de red en la que todas las estaciones están conectadas a un único canal de comunicaciones. Las estaciones utilizan este canal para comunicarse con el resto, no hay ninguna conexión directa con el resto de los nodos, sólo existe un enlace
Topología de red en la que las estaciones se conectan formando un anillo. Cada estación está conectada a la siguiente y la última está conectada a la primera
Topología de red en la cual las estaciones están conectadas directamente a un servidor o estación y todas las comunicaciones se han de hacer a través de él
Topología de red en la que cada estación envía sus datos a todas las demás estaciones conectadas al medio físico de la red. No hay orden en la transmisión de datos. El primero en acceder al medio es el primero en transmitir
Topología de red en la cual existe un computador central
Topología de red en la que se controla el acceso al medio utilizando un testigo electrónico que se pasa a cada estación. Cuando una estación recibe el testigo puede transmitir datos sí los tiene, si no, entonces pasa el testigo a la siguiente estación
Bus
Anillo
Estrella
2
3
4


1
Topología en estrella extendida
es
Topología de red que funciona de la misma forma que la topología en estrella con la diferencia de que cada nodo que se conecta con el nodo central también puede ser el centro de otra estrella
Estrella extendida
5
Topología irregular
es
Topología de red en la que no existe un patrón obvio de enlaces y nodos
Irregular
2
Ethernet
es
También conocida como estándar IEEE 802.3. Es un estándar de transmisión de datos para redes de área local que se basa en el principio de que todas las estaciones están conectadas a la misma línea de comunicación compuesta por cables cilíndricos
LocalTalk
es
Es una tecnología LAN desarrollada por Apple Computer Corporation que usa una topología de canal. LocalTalk usa protocolos AppleTalk (colección de protocolos para la conexión de redes)
3
ATM
es
Modo de transferencia asíncrono. Es una tecnología de red reciente que a diferencia de Ethernet, TokenRing y FDDI permite la transferencia simultanea de datos y voz a través de la misma línea
4
TokenRing
es
Es una tecnología de acceso a redes que se basa en el principio de la comunicación sucesiva, es decir, cada estación de la red tiene la oportunidad de comunicarse en determinado momento
FDDI
es
Siglas en inglés que se traducen como interfaz de datos distribuida por fibra. Es una tecnología de acceso a redes a través de líneas de fibra óptica. La FDDI es una red en anillo que posee detección y corrección de errores

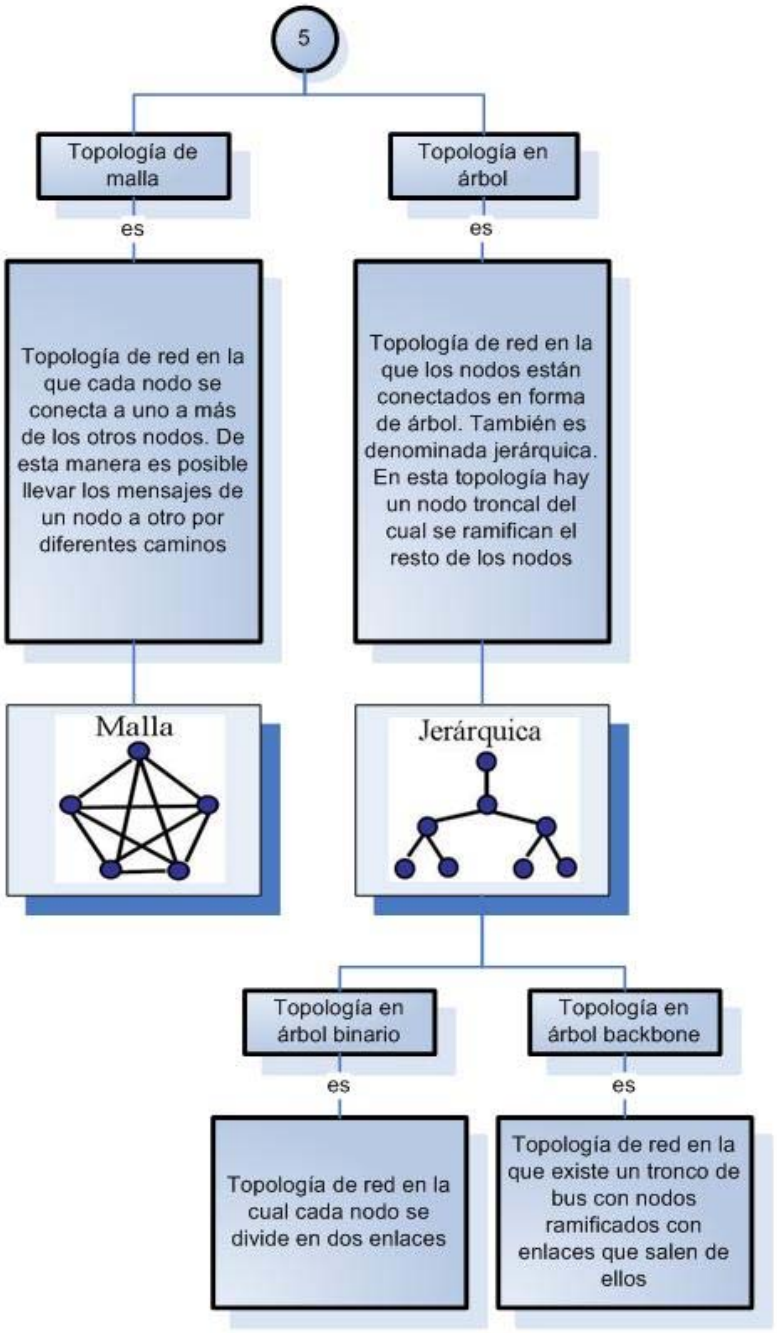


Figura 5. Mapa Conceptual – Topologías de Red

Para ampliar un poco más el tema se puede revisar el anexo 9.1.1

Según lo estudiado anteriormente sobre las topologías de red y al numeral 5.2 de esta investigación en el cual se determina que existirá un único nodo maestro encargado de la repartición de las tareas a sus esclavos, se determinó que la topología para trabajar en esta

investigación será la topología en estrella, ya que en el método que se propondrá existirá un nodo maestro que estará directamente conectado a todos los nodos esclavos y capaz de enviar y recibir mensajes de estos. El factor de comunicación entre los nodos no será tan importante como el factor de la velocidad de procesamiento, por lo que el uso de otras topologías más seguras no será necesario.

Por otro lado, la selección de esta topología en estrella fue escogida por el hecho de que es ideal para configuraciones que necesiten conectar estaciones a un nodo central (maestro) y que este a su vez se encargue de la transmisión de mensajes a los extremos (esclavos). Las terminales conectadas al nodo maestro pueden incluir nodos que no posean muchas capacidades además de ser diferentes en cuanto a estructura física y de comunicaciones lógicas (distintas velocidades de transmisión y distintos medios de transmisión).

Sin embargo en esta investigación al trabajar por medio de un simulador no se esta teniendo en cuenta que el nodo maestro puede ser bastante susceptible a diversos tipos de averías, cosa que en la realidad es bastante factible, además del hecho de que el nodo central deberá soportar mucha actividad por lo que debe ser el nodo más fuerte en el cluster.

5.5 METODO PARA EL MANEJO DEL BALANCEO DE CARGA

El método que se propone a continuación recopila ideas de dos trabajos revisados previamente en el capítulo 4. El trabajo realizado por DrozDowski y Wolniewicz en el que se propone un modelo de carga divisible, el cual se utiliza sólo en aplicaciones que permitan dividirla en tamaños diferentes [DROZDOWSKI, 2003], y el trabajo de Choi, Yu, Kim y otros, en el que el desempeño del sistema tiene mucho que ver con la cantidad de tareas para cada nodo [CHOI, 2003].

En esta investigación el método ha proponer radica en que la distribución de las tareas es realizada por el nodo maestro a sus nodos esclavos. El nodo maestro almacena la cantidad total de tareas que ingresan al cluster con su respectivo tamaño, además de factores importantes para la decisión de asignación como los atributos de cada nodo esclavo (velocidad de procesamiento, capacidad de memoria y tiempo de respuesta), con los cuales el maestro calcula la función de aptitud propia de cada esclavo y el porcentaje de capacidad de trabajo de los esclavos con respecto al resto. Luego de que el nodo maestro asigna una tarea al esclavo, el nodo esclavo realizará la tarea y enviará al nodo maestro una notificación de terminación de tarea y el tiempo que utilizó éste para la ejecución de dicha tarea (tiempo de ejecución). Así el esclavo le hace saber al nodo maestro que esta listo para recibir y realizar una tarea nueva.

La función de aptitud se define como la rapidez asociada a la velocidad de procesamiento, la capacidad de memoria y el tiempo de respuesta o latencia.

El cálculo de la función de aptitud corresponde a la sumatoria entre los atributos mencionados previamente, donde cada uno de estos estará multiplicado por un peso que dependerá de la importancia de este factor en el cluster.

$$FuncionAptitud_{(i)} = a * cp_{(i)} + b * cm_{(i)} + c * tr_{(i)}$$

donde:
$cp_{(i)}$ = velocidad de procesamiento del nodo i
$cm_{(i)}$ = capacidad de memoria del nodo i
$tr_{(i)}$ = tiempo de respuesta o latencia del nodo i
a, b, c = pesos asociados a cada variable y que dependen del tipo de cluster seleccionado

Cada uno de los valores para los atributos y los tamaños de las tareas fue calculado aleatoriamente, por el hecho de que la forma en que debía ser probado el método era por medio de una simulación y se tornaría complicado ingresar datos para cada nodo cuando existiese gran cantidad de estos.
En cuanto a los atributos de los nodos, la velocidad de procesamiento se da en GHz por el hecho de que esta velocidad se define como un tiempo de frecuencia de la señal del reloj; la capacidad de memoria será dada en unidades que abarcan entre los KB y los GB, por el hecho de que consiste de un espacio de almacenamiento en el disco; el tiempo de respuesta o latencia esta dado en segundos y es el tiempo que tarda un nodo esclavo en recibir y dar respuesta a una petición generada por el nodo maestro.
Cuando hablamos del tamaño de las tareas nos estamos refiriendo al tamaño en Byte del programa.

Para hacer el cálculo de la función de aptitud en los nodos es necesario normalizar cada una de las variables, ya que si se trabaja con los valores originales la escala numérica sería muy diferente y la función de aptitud se tornaría altamente variada para los diversos nodos.

En este método se tendrán en cuenta los tamaños de las tareas para la asignación, recopilando estos por medio de series de tandas que se crearan acorde a la cantidad de nodos esclavos.

Descripción del método

1. Se toman dos vectores iniciales: en un vector se almacenan todas las tareas iniciales organizadas de mayor a menor tamaño cada una de estas presenta un ID que identifica que tarea es; el otro vector recopila los nodos esclavos organizados por función de aptitud de mayor a menor.

2. Se calculan los valores de las sumatorias de los tamaños de las tareas (STT) y de las funciones de aptitud de los nodos esclavos (SFA).

$$STT = \sum_{i=1}^{n} TamañoTarea_{(i)}$$
$$SFA = \sum_{i=1}^{m} FuncionAptitud_{(i)}$$

donde:
n es la cantidad total de tareas que se presentan inicialmente y m es la cantidad de nodos del cluster. El nodo 0 es el nodo maestro.

3. Se calcula el porcentaje de carga ($FactorNodo_{(i)}$) que podrá recibir cada uno de los nodos esclavos como sigue a continuación:

$$FactorNodo_{(i)} = \frac{FuncionAptitud_{(i)} * 100}{SFA}$$

donde:
$FactorNodo_{(i)}$ corresponde al porcentaje de la carga de trabajo o tareas del nodo (i) relacionada con su propia capacidad.

4. Se calcula la capacidad que posee cada nodo con respecto a los tamaños de las tareas, con el fin de obtener un rango de tamaños propios para cada nodo de acuerdo a las tareas existentes hasta el momento.

$$CentroIntervaloNodo_{(i)} = STT * FactorNodo_{(i)}$$
$$RangoNodo_{(i)} = CentroIntervaloNodo_{(i)} \pm error$$

el error se precisó en un 10%, lo que equivale a un manejo de eficiencia del 90%.

$$LimiteInferiorNodo_{(i)} = CentroIntervaloNodo_{(i)} - error$$
$$LimiteSuperiorNodo_{(i)} = CentroIntervaloNodo_{(i)} + error$$

5. En este punto, el maestro ya conoce cual es la capacidad de cada nodo esclavo y procede con la distribución de las tareas teniendo en cuenta los cálculos realizados hasta el paso 4 y que las tareas se encuentran organizadas de mayor a menor tamaño y los nodos se encuentran también organizados de mayor a menor pero en cuanto a función de aptitud.

Para iniciar con la asignación se deberá recorrer el vector en donde se encuentran almacenadas las tareas. Para cada nodo esclavo se creará un nuevo vector llamado $Tanda_{(i)}$ que contendrá las tareas que le serán asignadas para ejecutar.
En el nodo maestro se almacenará una variable para cada uno de los nodos esclavos que será un contador (sumatoria de tamaños de tareas) que examinará si la tarea a ser asignada podrá o no estar en la tanda de tareas del nodo respectivo cumpliendo o

no con los requisitos del rango de cada nodo esclavo. Esta variable se conocerá como $SumatoriaTanda_{(i)}$.

A continuación se realiza la asignación de las tareas a los nodos teniendo en cuenta la variable $SumatoriaTanda_{(i)}$ almacenada en el nodo maestro para cada nodo esclavo.

a) Se toma el primer tamaño de tarea que se encuentra en el vector de tareas y se acumula en $SumatoriaTanda_{(i)}$.

$$SumatoriaTanda_{(i)} = SumatoriaTanda_{(i)} + TamañoTarea_{(i)}$$

b) Se examina sí $SumatoriaTanda_{(i)}$ es menor que $LimiteInferiorNodo_{(i)}$. De ser cierto, se asigna esa tarea a la $Tanda_{(i)}$, se elimina esta tarea del vector y se evalúa sí hay aun hay tareas en el vector de tareas, si es así se continua con la siguiente tarea del vector hasta que $SumatoriaTanda_{(i)}$ sea menor que $LimiteSuperiorNodo_{(i)}$, esto implica que la tanda se encuentra lista para ser asignada al nodo y se encuentra dentro del rango admisible por ese nodo.
Se examina si hay más tareas por asignar en el vector de tareas y se continúa con el siguiente nodo realizando el mismo procedimiento.
Sin embargo puede haber casos excepcionales en el que ya se hayan asignado algunas tareas a la tanda y la tarea siguiente a ser asignada sobrepase el $LimiteSuperiorNodo_{(i)}$, en este caso esta tarea no se asigna a dicho nodo y se evalúa para el nodo siguiente, de ahí se puede observar que este nodo aún posee espacio disponible para ejecutar alguna tarea. También puede existir la posibilidad de que la primera tarea del vector de tareas exceda inmediatamente el $LimiteSuperiorNodo_{(i)}$, cuando esto ocurre, se le asigna dicha tarea a este nodo, y se cambia de nodo y tarea.

c) Después de haber asignado las tareas a los nodos esclavos en sus respectivas tandas se debe analizar nuevamente sí aún quedan tareas en el vector de tareas, de ser el caso, estas deberán ser incluidas en una nueva tanda que será destinada a las tareas faltantes y que deberán ser asignadas a las tandas de los nodos esclavos que tengan aún espacio sin exceder en el $LimiteSuperiorNodo_{(i)}$.

d) Sí llegasen a faltar tareas por ser asignadas después de realizar el paso anterior y ninguna entra en el rango de los nodos esclavos entrará una nueva variable a tomar parte de la asignación. Esta será el tiempo de ejecución, que será el tiempo que tarda cada nodo en ejecutar una tarea respectiva. Esta variable es calculada como el espacio (tamaño de la tarea que se esta realizando) sobre la velocidad (función de aptitud del nodo ejecutor de la tarea).

$$TiempoEjecucion_{(i)} = \frac{TamañoTarea_{(i)}}{FuncionAptitud_{(i)}}$$

El procedimiento a seguir es calcular el tiempo de ejecución total acumulado en cada nodo esclavo (TiempoEjecucionTotal$_{(i)}$), tomando la tanda de tareas de cada nodo (Tanda$_{(i)}$) y calculando el tiempo de ejecución para cada una de las tareas que tiene dicho nodo.
Posteriormente se procede a tomar la primera tarea que se encuentra en la tanda de tareas faltantes y a calcular un tiempo de ejecución supuesto (TiempoEjecucionSupuesto$_{(i)}$), para cada nodo con dicha tarea y a acumularlo en el TiempoEjecucionTotal$_{(i)}$ de cada nodo. Luego se procederá con la revisión de cual nodo posee el menor tiempo de ejecución total supuesto (TiempoEjecucionTotalSupuesto$_{(i)}$), aquel que tenga dicho tiempo será el nodo al cual se le asignará en su tanda esta tarea.

$$TiempoEjecucionSupuesto_{(i)} = \frac{TamañoTareaFALTANTE_{(i)}}{FuncionAptitud_{(i)}}$$

$$TiempoEjecucionTotalSupuesto_{(i)} = TiempoEjecucionTotal_{(i)} + TiempoEjecucionSupuesto_{(i)}$$

Esto se hace hasta que no queden tareas en la tanda de tareas faltantes.

e) Finalmente el nodo maestro envía a sus nodos esclavos una por una las tareas respectivas de las tandas que les corresponden para que comiencen a ejecutarlas. Cada vez que un esclavo finalice una tarea propia este envía una notificación de terminación de tarea con el fin de que el maestro envíe la siguiente tarea de su tanda, haciendo esto hasta que culmine de ejecutar todas sus tareas.