Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Computació d'Altes Prestacions

Aplicaciones Single Program Multiple Data (SPMD) en Ambientes Distribuidos

Memoria del trabajo de investigación del "Máster en Computación de Altas Prestaciones", realizada por Ronal Muresano, bajo la dirección de Dr. Emilio Luque Presentada en la Escuela Técnica Superior de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

Año 2008

Trabajo de investigación

Máster en Computación de Altas Prestaciones

Curso 2007-08

Título: **Aplicaciones Single Program Multiple Data (SPMD) en Ambientes Distribuidos**

Autor: Ronal Muresano
Director: Dr. Emilio Luque Fadón

Departamento Arquitectura de Computadores y Sistemas Operativos
Escuela Técnica Superior de Ingeniería (ETSE)
Universidad Autónoma de Barcelona

Firmado

Autor Director

Agradecimientos

Comienzo agradeciendo a Dios, que siempre me guía para tomar las mejores decisiones.

A mi Esposa, por su apoyo y comprensión en todo momento

A ese ser querido que viene en camino y pronto alegrará nuestras vidas

A mis padres, por el apoyo que me han dado desde la distancia.

A mi Director Emilio Luque, por ser siempre un buen consejero.

A Dolores Rexachs, por ayudarme en todo momento y por hacerme entender que a partir de los errores, se obtiene el mejor aprendizaje.

A mis Compañeros, Manuel, Gonzalo, Álvaro, Leonardo, Genaro, Guna y todos los que participaron en el desarrollo de ésta investigación.

***Resumen**:* Un reto al ejecutar las aplicaciones en un cluster es lograr mejorar las prestaciones utilizando los recursos de manera eficiente, y este reto es mayor al utilizar un ambiente distribuido. Teniendo en cuenta este reto, se proponen un conjunto de reglas para realizar el cómputo en cada uno de los nodos, basado en el análisis de cómputo y comunicaciones de las aplicaciones, se analiza un esquema de mapping de celdas y un método para planificar el orden de ejecución, tomando en consideración la ejecución por prioridad, donde las celdas de fronteras tienen una mayor prioridad con respecto a las celdas internas. En la experimentación se muestra el solapamiento del computo interno con las comunicaciones de las celdas fronteras, obteniendo resultados donde el *Speedup* aumenta y los niveles de eficiencia se mantienen por encima de un 85%, finalmente se obtiene ganancias de los tiempos de ejecución, concluyendo que si se puede diseñar un esquemas de solapamiento que permita que la ejecución de las aplicaciones SPMD en un cluster se hagan de forma eficiente.

Palabras Clave: *Multicluster, Cluster*, *SPMD*, Solapamiento, Eficiencia.

Resum: Un repte a l'executar les aplicacions en un cluster és assolir millorar les prestacions utilitzant els recursos de manera eficient, i aquest repte és major a l'utilitzar un ambient distribuït. Tenint en compte aquest repte, es proposen un conjunt de regles per a realitzar el còmput en cadascun dels nodes, basat en l'anàlisi de còmput i comunicacions de les aplicacions, s'analitza un esquema de mapping de cel·les i un mètode per a planificar l'ordre d'execució, prenent en consideració l'execució per prioritat, on les cel·les de fronteres tenen una major prioritat pel que fa a les cel·les internes. En l'experimentació es mostra el solapamiento del computo intern amb les comunicacions de les cel·les fronteres, obtenint resultats on el Speedup augmenta i els nivells d'eficiència es mantenen per sobre d'un 85%, finalment s'obté guanys dels temps d'execució, concloent que si es pot dissenyar un esquemes de solapamiento que permeti que l'execució de les aplicacions SPMD en un cluster es facin de forma eficient.

Paraules Clau**:** *Multicluster, Cluster, SPMD, Solapamiento, Eficiència.*

Abstract: A challenge to execute some applications in a cluster, is to achieve better performance using resources efficiently. This challenge is greater when is using a distributed environment. Whereas this challenge, this investigation propose a set of rules to make computing in each nodes, based on an analysis of computing and communications inside of the applications. It analyzes an outline of mapping cell and a method for planning the execution order in the group of cell, taking into consideration the execution priority, where border cells have a higher priority than internal cells. In the experiment shows the overlap between border communications cells and internal cells, where the results increases the speedup and the efficiency levels remain above 85%. Finally obtained profits of execution times, concluding that if it can design an overlapping schemes that allow the execution of applications SPMD in a cluster become an efficient manner.

Key Words: *Multicluster, Cluster, SPMD, Overlaps, Efficiency*

Índice General

Índice Figuras

Índice Tablas

Capitulo 1

1.1 Introducción

El creciente desarrollo que día a día están teniendo las aplicaciones de alto cómputo y los avances científicos, han llevado al mundo del paralelismo a estar en constante ascenso. Los sistemas paralelos, necesitan un conjunto de herramientas que utilizan elementos o recursos de sistemas para controlar y mejorar los tiempos de ejecución de las aplicaciones ejecutadas, permitiendo así que los tiempos de respuesta de las aplicaciones de cómputo mejoren y por ende se obtengan resultados de manera más rápida.

Las aplicaciones paralelas, además de permitir que la ejecución se realice en menores tiempos, buscan que los procesos sean ejecutados con disponibilidad de recursos y con elevado performance de cómputo (*High Performance Computing, HPC)*, con el objetivo de obtener un rendimiento considerables en los tiempos de ejecución de la aplicación. Un aspecto en el cómputo paralelo es, observar la eficiencia del uso de los recursos, debido a que debe existir una relación de velocidad computacional *Speedup* y eficiencia cuando se ejecuta cómputo paralelo.

Para lograr el paralelismo en los desarrollos computacionales se pueden utilizar varias estrategias dependiendo de dónde se necesite la paralelización de la aplicación, es decir; si se requiere a nivel del nodo o en el conjunto global de los nodos donde se ejecuta la aplicación, la correcta selección de la paralelización permite mejoras fundamentales en los tiempos de ejecución de la aplicación.

La selección correcta de paralelización depende de la cantidad de recursos disponibles para la ejecución de la aplicación. Actualmente, las aplicaciones paralelas pueden ser ejecutadas en clusters, en centros de supercomputing, en grid o en unión de clusters distribuidos geográficamente (multicluster), todo depende de la disponibilidad y costes de cada elemento de cómputo que se desea utilizar. La relación de coste/ beneficio de cómputo debe tomarse en

consideración al ejecutar o diseñar una aplicación paralela y cuando se desea seleccionar dónde se ejecutará la misma.

Para algunas aplicaciones paralelas el uso de un cluster puede quedarse corto a nivel de cómputo, más aún si la aplicación es muy compleja. En estos casos los tiempos de ejecución no serán los deseados por los diseñadores de la aplicación, por ésta razón se deben buscar alternativas como los centros de *supercomputing* (e.j. *Marenostrum*), sin embargo, al buscar un centro de éste estilo, se debe tomar en consideración que el cómputo tiene un coste elevado, y debe ser ejecutado de acuerdo a la prioridad que se le asigne, por lo que para algunos casos no es una solución viable de ejecución.

Por otra parte, existe otra solución de ejecución, la cual es utilizando centros de cómputo ubicados en zona geográficamente distribuidas (*multicluster*), que no están siendo usado en todo momento. Un ejemplo son lo clusters que integran los laboratorios de una universidad; esta integración de cluster permite formar una arquitectura computacional, menos costosa que un centro de súper computación, pero que permite tener poder computacional para que se ejecuten las aplicaciones.

Sin embargo, el programador debe asumir que la ejecución de las aplicaciones en entornos multicluster, es compleja debido a la heterogeneidad que se puede presentar en el entorno, no sólo a nivel de la arquitectura de cada uno de los cluster remotos, sino también por el tráfico de los enlaces de comunicación, debido a que utiliza como canal de comunicación la Internet. Los enlaces tienen diferentes tiempos de latencias de comunicación, y esto conlleva a que se deban gestionar las comunicaciones que pueden existir en la ejecución de la aplicación.

En un *multicluster*, se pueden tener varios tipos de comunicaciones Intercluster e Intracluster, las cuales tienen velocidades de comunicación distintas, la primera es manejada a nivel de red de área local (*LAN*) en el cluster y la segunda a nivel de área extensa (*WAN*) y es administrada por la interconexión de los clusters involucrados en la ejecución de la aplicación (figura 1.1).

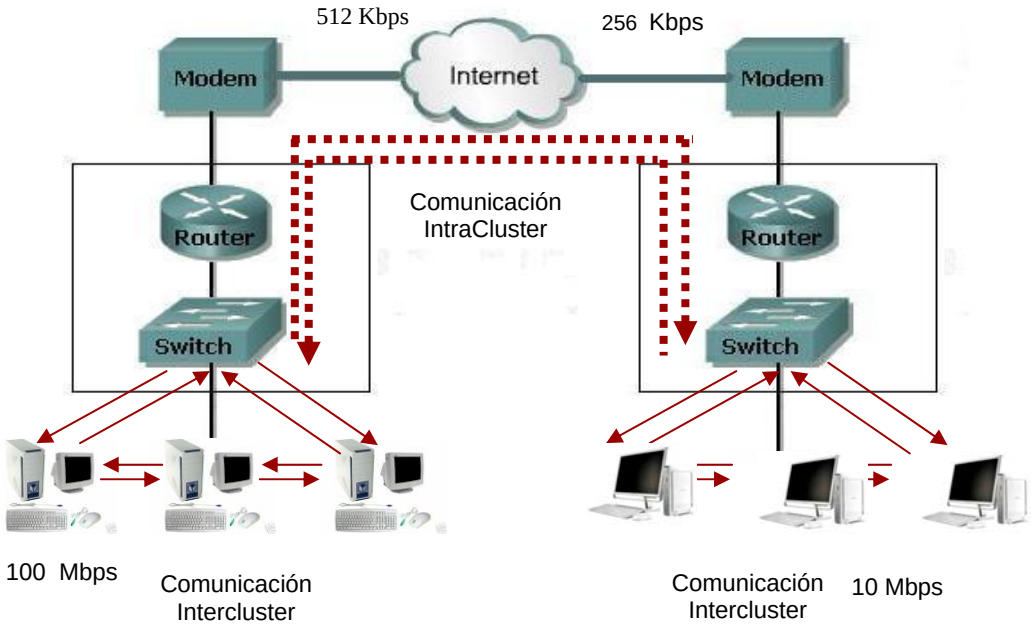


Figura 1.1 Ambiente *Multicluster*

Las latencias de los enlaces de comunicaciones en el ambiente *multicluster*, motiva que las aplicaciones con altos niveles de comunicación entre los procesos, se tengan que diseñar con estrategias de forma controlar estas latencias en los enlaces, lo anterior para que las comunicaciones influyan lo menos posible en los tiempos totales de ejecución de la aplicación. Los problemas de una adecuada distribución de carga, *mapping*, planificación y velocidad de cómputo que se presentan en los *clusters*, se incrementan en un entorno multicluster y se deben controlar de manera que se mejore la performance de la aplicación.

Al colocar una aplicación en un entorno multicluster, se enfrentan problemas por los retardos de las comunicaciones, por lo que se debe orientar a la búsqueda de una relación de cómputo y comunicación, que permita disminuir los tiempos de ejecución, y en realidad justifique la utilización de gran cantidad de recursos para la realización del cómputo de la aplicación, por lo que en el diseño de una aplicación orientada a un paradigma de programación (*Master Worker, SPMD, Pipeline*, entre otros), se deban realizar determinando los patrones de comunicación que incluye cada paradigma, de forma de controlar el aspecto de las comunicaciones en el entorno *multicluster*.

Los paradigmas establecen un modelo de ejecución para los clusters o *multicluster*, lo que origina que la aplicación deba ser rediseñada de acuerdo al paradigma y a la arquitectura que la ejecuta para obtener el máximo beneficio a la hora de paralelizarla; por esto, las aplicaciones son adaptadas a cambios para que puedan ser ejecutadas de manera eficiente en ambientes de cómputo paralelo, tomando en consideración que no se puede variar el objetivo inicial de la aplicación secuencial. Este paralelismo permite que las aplicaciones sean subdivididas en trozos, de los cuales, cada uno de ellos es asignado a un recurso dentro del cluster para que lo ejecute.

Existen estudios para entornos *multicluster*s utilizando el paradigma *Master Worker* (Argollo, 2006), donde se observa la utilización de recursos del entorno, utilizando el paradigma bajo una arquitectura *Master/Worker* jerárquica donde se tiene un nodo *master*, y los *clusters* remotos que trabajan con un nodo *SubMaster* y los demás nodos como *Worker.* Todos los cluster están interconectados a través de un *"comunicator manager"* que permite gestionar las comunicaciones a través de la *WAN.* A nivel de cluster, en el paradigma *Master/Worker*, el nodo Master asigna la carga de trabajo o procesos a los nodos *Workers*, éstos computan y devuelven los resultados al nodo *Master* y así sucesivamente hasta culminar la ejecución de la aplicación.

Otro paradigma que es utilizado en las aplicaciones paralelas es el *Single Program Multiple Data (SPMD*), que a diferencia del *Master Worker*, ejecuta un único programa en todos los nodos y distribuye celdas de datos de la aplicación para ser ejecutada, comunicando los resultados con los nodos vecinos en cada iteración de la ejecución de la aplicación, esto conduce a una dependencia de los datos con los procesos vecinos y alto nivel de comunicación de información (Figura 1.1).

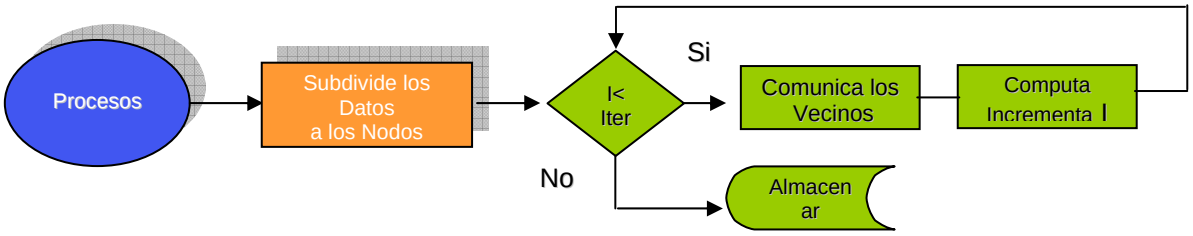


Figura 1.1 Diagrama de Flujo del paradigma SPMD.

En el paradigma *SPMD*, es importante considerar que existe una comunicación con los elementos vecinos, lo que origina un volumen de comunicaciones, que aunado a las latencias de los enlaces de comunicación pueden generar retardos en la ejecución de la aplicación en un cluster, si no esta bien planificado.

Estos problemas de dependencias de datos de las aplicaciones *SPMD*, y la necesidad de velocidad de cómputo de las aplicaciones, originan que se busquen centros para computar la aplicación, entonces haciendo una integración entre los *multicluster*, que pueden ejecutar mayor cantidad de cómputo. Para las aplicaciones *SPMD,* surge un planteamiento: ¿se puede ejecutar aplicaciones *SPMD* en ambientes *Multicluster* de forma eficiente?, la respuesta de ésta interrogante depende de cómo se controlen los elementos que integran la aplicación.

Cuando se quiere asignar la aplicación *SPMD* en el *multicluster*, se deben considerar que existen 3 tipos de comunicaciones, Internodo, Intercluster e Intracluster, por lo que se incrementa el problema de las comunicaciones y por ende se debe mantener una relación muy estrecha entre el cómputo y la comunicación.

Para lograr una ejecución eficiente de las aplicaciones *SPMD* se debe realizar un estudio que permita mejorar los tiempos de comunicación, buscando una relación de cómputo y comunicación, que permita hacer un patrón o modelo de las comunicaciones Internodo, Intercluster e Intracluster, esto para poder determinar la relación que es necesaria entre las mismas para aumentar la velocidad computacional (*Speedup)* en la ejecución de la aplicación, manteniendo un umbral de eficiencia predefinido.

1.2 Objetivos de la Investigación.

En los entornos *multicluster*, la heterogeneidad de los sistemas de cómputo y de comunicaciones, generan dificultades a la hora de hacer ejecuciones de las aplicaciones y más aún dependen del paradigma con el que esté desarrollada; en el caso del *SPMD*, éste presenta alto volumen de comunicaciones y junto con las

latencias de los enlaces, debe ser resueltos para que la ejecución no se vea afectada por este problema de comunicaciones y así ser una alternativa viable para la ejecución de aplicaciones a la hora de disminuir los tiempos de ejecución de la aplicación en el *multicluster*.

Al plantear la pregunta de ¿se puede ejecutar una aplicación *SPMD* en un ambiente *multicluster*?, la cual fue descrita anteriormente, se trae una serie de desafíos para controlar las comunicaciones, por lo tanto delimitado el enfoque inicial del problema planteado, se trabajará en un entorno de un cluster de cómputo homogéneos, para visualizar si las comunicaciones pueden ser controladas en un ambiente con características iguales tanto a nivel de cómputo como de comunicaciones.

Es decir, derivando de la pregunta inicial, podríamos hacer una nueva pregunta, de forma de delimitar la investigación, y se platea como ¿se puede ejecutar una aplicación *SPMD* en un cluster de forma eficiente?, al dar respuesta a esta pregunta permitirá como base para empezar a extrapolar la idea pero en un ambiente *multicluster*.

Para contestar la pregunta en un *cluster*, se deben plantear unos objetivos en mira de cumplir el objetivo general de la presente investigación, estos objetivos son:

- Analizar el comportamiento de la aplicación *SPMD* en un nodo.

- Presentar una mejora de la velocidad computacional (*Speedup*) de la aplicación manteniendo un umbral de eficiencia.

- Establecer un modelo de mapping para repartir la cantidad de celdas a cada nodo, tomando en consideración el número de nodos, manteniendo la relación cómputo- comunicación.

- Planificar el orden de ejecución de las celdas de una aplicación *SPMD* en un *cluster* de cómputo (*Schedulling*).

Una vez cumplidos éstos objetivos, se podrá determinar si se puede hacer una ejecución eficiente de una aplicación *SPMD* en un cluster de cómputo homogéneo.

1.3 Organización del Trabajo

El trabajo está compuesto por un conjunto de capítulos que detallarán cada una de las partes de la presente investigación, en el capítulo 2, se definirá la arquitectura y las bases teóricas de las aplicaciones que emplean el paradigma *SPMD* y la relación con los objetivos planteados. Asimismo, se detallarán los desarrollos de trabajos que permitirán establecer los puntos de comparación de los experimentos a desarrollar.

En el capítulo 3, se describirá el problema y los elementos a aplicar para obtener el cumplimiento de los objetivos planteados para esta memoria.

Continuando la investigación, en el capítulo 4, se presentará un evaluación experimental para demostrar el modelo que permite mejorar la velocidad computaciones, manteniendo los niveles de eficiencia por encima de un 80%, presentando las mejoras que pueden traer en los entornos de las aplicaciones y los paradigma, así como las avances que servirán como fundamento para la culminación de toda la investigación, a partir de los resultados obtenidos se plantearán los trabajos orientados a cumplir el desarrollo general de la investigación planteada en el capítulo 1.

Finalmente, se presentarán las conclusiones y recomendaciones generales de la utilización del paradigma de programación en el entorno planteado en la presente investigación.

Capitulo 2

2.1 Introducción

El cómputo de un computador está ligado al tiempo que éste requiere para ejecutar un conjunto de instrucciones básicas y al número de instrucciones concurrentes que el sistema de procesamiento puede realizar. Las aplicaciones se incrementan de acuerdo a los avances a nivel de arquitectura y tecnológico que tenga el sistema de cómputo, de forma tal que permite que la aplicación se subdivida para que se pueda procesar en paralelo, con el objetivo de obtener mejores prestaciones que la versión serie de la aplicación.

El procesamiento paralelo, consiste en incluir un conjunto de técnicas para proporcionar simultaneidad al procesamiento de información, con el objetivo de disminuir los tiempos de ejecución del entorno. Según Foster (1994), define el procesamiento paralelo como "…el conjunto de procesadores que están activos para trabajar cooperativamente para solventar un problema computacional… [pág. 3]", ésta definición permite describir que las aplicaciones que utilizan el procesamiento paralelo tienen concurrencia, por lo que se distribuye el trabajo a cada nodo de cómputo y éste trabajo debe ser procesado para conseguir un menor tiempo de ejecución de la aplicación.

Para aprovechar al máximo el poder de procesamiento de un computador con arquitectura paralela, es necesario utilizar técnicas de programación paralela, debido a que se deben definir los procesos que el computador ejecutará en forma concurrente. La programación paralela, permite que las aplicaciones se pueden dividir en varias tareas pequeñas que se pueden ejecutar en paralelo, ejemplo de esto son las aplicaciones de estudios ambientales tales como transferencias de calor, climatología, presión atmosférica, entre otros.

El objetivo del procesamiento paralelo, es mejorar la aceleración computacional y los tiempos de ejecución de la aplicación, sin embargo, se debe medir la escalabilidad, la relación de velocidad y el coste de hacer una ejecución. En una aplicación paralela, puede ser difícil de alcanzar un ejecución rápida con un nivel de eficiencia, motivado a la disputa de los recursos que son compartidos

(red, memoria, dispositivos de almacenamiento, entre otros), presentando conflictos con la arquitectura del sistema de cómputo, originando que la aplicación tenga problemas a nivel computacional y a la hora del buen aprovechamiento de los recursos.

Para obtener una ejecución que mejore las prestaciones significativamente, se debe tener en cuenta la administración de los recursos y la arquitectura en la cual se esta ejecutando la aplicación. Es por esto, que si se utiliza un entorno de ejecución donde incluye cluster geográficamente distribuidos o *multicluster*, y se insertan aplicaciones que se comportan con cierto patrón específico como el *SPMD*, por lo que se deben considerar los problemas que incluyen una ejecución en éste tipo de entornos, las cuales fueron explicadas en el capitulo anterior.

Al describir la unión del *multicluster* y de las aplicaciones *SPMD*, se debe considerar los estudios de ambos conceptos por separados de modos de comenzar a describir el entorno de trabajo que se desea estudiar. La figura 2.1, muestra la integración de un concepto de paradigma *SPMD*, e integrándolo con una arquitectura surge la pregunta planteada.

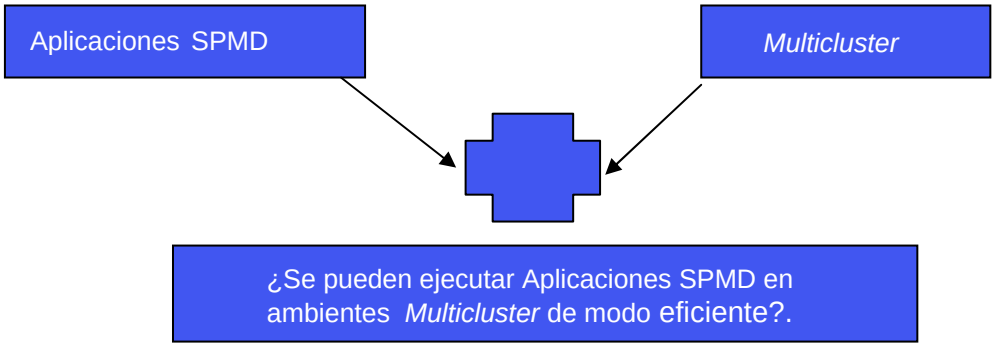


Figura 2.1 Unión de Tecnologías SPMD y *Multicluster*

Por lo tanto, se realizara una descripción de los paradigmas existente y se enfocará el estudio en las aplicaciones SPMD, luego se describirá la arquitectura de *multicluster* y de los cluster, para culminar con los modelos de programación paralela y la evaluación computacional.

2.2 Paradigmas de Programación Paralela.

Los paradigmas de programación son patrones que permiten desarrollar programas de entornos paralelos. Cada uno de los paradigmas es un modelo de algoritmo que tiene forma de distribución de datos, modelo de comunicación y que al final cumple con el objetivo de paralelizar una aplicación.

La selección de los paradigmas de programación se hace de acuerdo al comportamiento que tenga la aplicación y de acuerdo a la disponibilidad de recursos de computación paralelos, por otra parte se debe establecer el tipo de paralelismo que se le desea aplicar al diseño del problema. En la actualidad, casi todas las aplicaciones están identificadas a un paradigma de programación paralela, Divide y Vencerás, Pipeline, Master / Worker y SPMD (Single Program Multiple Data) Buyya (1999).

2.2.1 Paradigma Divide y Vencerás

Los problemas de la aplicación son subdivididos en varios subproblemas, el cual permite que cada una de estas subdivisiones pueda ser resuelta independientemente, antes de finalizar la ejecución se integran todos los resultados para dar el resultado final de la aplicación (figura 2.2). El proceso de partición y de recombinación hace uso de paralelismo, por lo que estas operaciones requieren de comunicaciones. Sin embargo los subproblemas son independientes, por lo que no es necesaria la comunicación entre los procesos o subproblemas de la aplicación. En este paradigma se realizan tres operaciones, dividir, computar y unir, por lo que la estructura del programa es un árbol, siendo los subproblemas las hojas que ejecutarán los procesos o Threads.

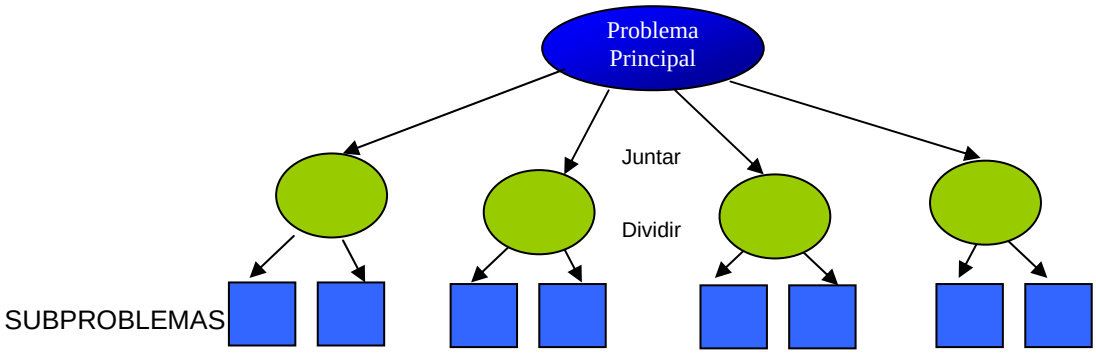


Figura 2.2 Paradigma Divide y Vencerás

2.2.2 Paradigma Pipeline.

El paralelismo en este tipo de paradigma, esta basado en una descomposición funcional del problema, por lo que se deben identificar las tareas del algoritmo que son capaces de ser ejecutadas paralelamente, lo que origina que cada proceso ejecute una porción del algoritmo general. Esta técnica permite aumentar la capacidad de instrucciones por procesador, debido a que las porciones o actividades de programa, pueden ejecutarse de forma solapada (Moreno, 2005).

El pipeline es uno de los algoritmos funcionales más simples, y se puede detallar en la figura 2.3, donde cada proceso ejecuta una etapa del pipeline y es responsable de una tarea individual. Los patrones de comunicación son simples y pueden hacerse de forma tanto sincronía como asíncrona, la eficiencia de ejecución de éste paradigma es dependiente del balanceo de carga a lo largo de las etapas de pipeline, es muy utilizado para algoritmos de procesamiento de imágenes.

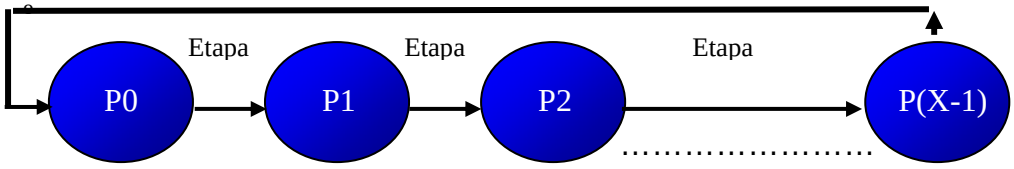


Figura 2.3 Paradigma Pipeline

2.2.3 Paradigma Master/Worker

En un paradigma que esta compuesto por dos entidades el master y los workers. El master es el encargado de dividir y distribuir las tareas de la aplicación, y los workers son entidades de trabajo que reciben la información la procesan y envían los resultados al master. En el caso de los workers, reciben una tarea a través de mensajes del master, por lo que la comunicación es solo a través de estas 2 entidades (Argollo, 2006), (Foster,1994), (Wilkinson,1999).

Este paradigma puede utilizar un esquema de balanceo de carga estático o dinámico, en el caso de utilizar un esquema estático, el master puede subdividir las tareas enviar a los worker y puede computar una parte, al final de la ejecución

todas las asignaciones de los workers han sido devuelta y se le adiciona lo que ha podido procesar el master.

En un esquema de balanceo dinámico, el cual es útil cuando se tiene una aplicación donde el número de tareas es mayor al número de procesadores que la ejecutan, o si se desconoce el número total de tareas, esto motivado a la habilidad de la aplicación de adaptarse a los cambios de heterogeneidad de los sistemas que la estén ejecutando (Ver Figura 2.4).

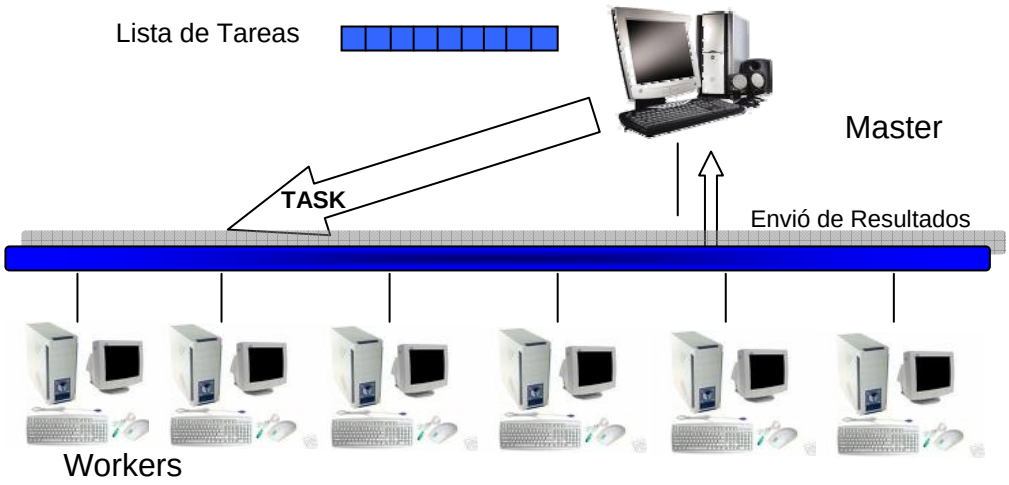


Figura 2.4 Paradigma Master Worker

2.2.4 Paradigma SPMD (Single Program Multiple Data).

El paradigma *SPMD*, esta compuesto por dos características que se pueden estudiar por separados, la primera es que es (SP) o *Single Program* lo que determina que se puede asignar solamente un programa o proceso, a cada nodo que ejecute la aplicación (Figura 2.4).

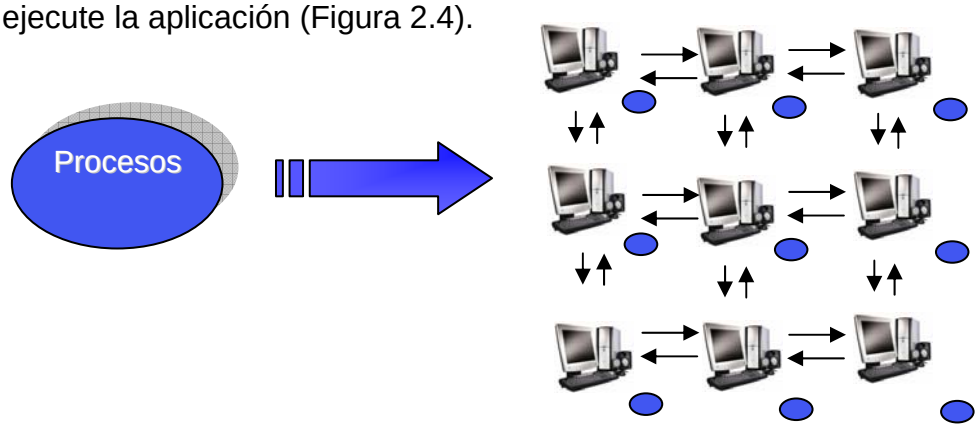


Figura 2.4 Asignación de Procesos a los Nodos en SPMD.

En el caso de MD, que es de *Multiple Data*, se refiere a la cantidad de celdas que pueden asignarse a cada uno de los nodos, para que sean computadas (Figura 2.5), el conjunto de celdas asignar por nodo depende de la aplicación, de la cantidad de celdas y de la estrategia de subdivisión que se tenga para la asignación de celdas.

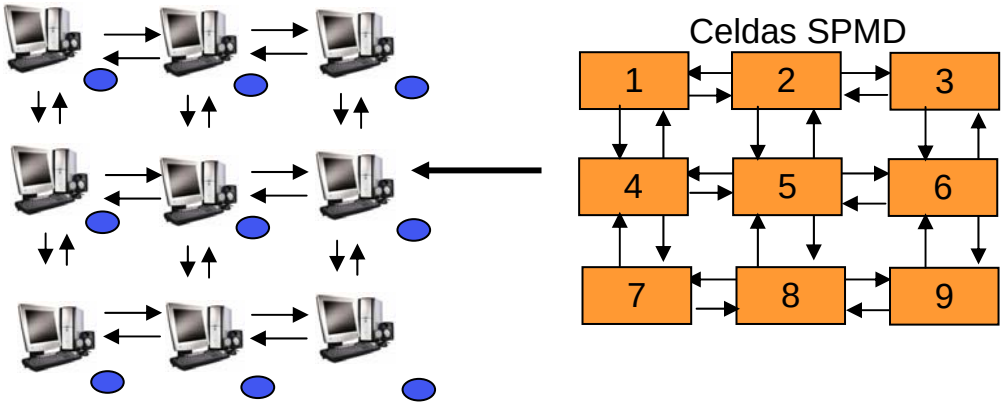


Figura 2.5 Asignación de Celdas a los Nodos en SPMD.

La combinación del *SPMD*, permite entonces asignar un conjunto de celdas de cómputo a cada uno de los nodos, de los cuales tienen un proceso igual de ejecución. El proceso de ejecución es repetitivo y depende del número de iteraciones que se definan en la aplicación, asimismo, incluyen un patrón de comunicación de intercambio, debido a que para hacer el cómputo se necesita la información de las celdas vecinas, por lo que si esta agrupadas en nodos, tendrán internamente acceso a memoria común y entre los nodos a través de paso de mensajes. La figura 2.6, muestra el comportamiento de una aplicación SPMD mediante un diagrama de flujo.

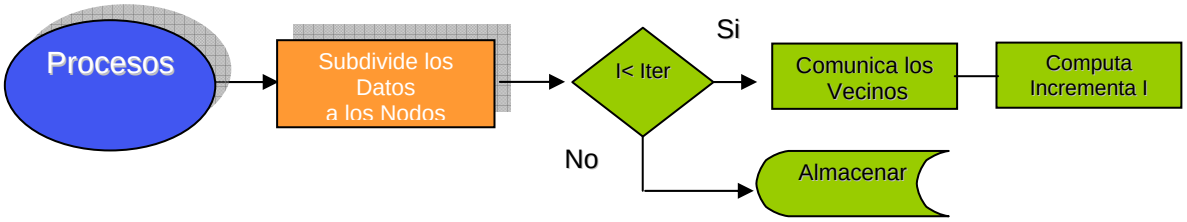


Figura 2.6 Diagrama de Flujo de aplicaciones *SPMD.*

Al describir la forma de asignación y ejecución de las aplicaciones *SPMD*, se puede decir que es uno de los paradigmas más utilizados para hacer estructuración de los entornos de programación paralela, y su funcionamiento es

que cada proceso ejecuta el mismo programa pero con diferentes conjunto de datos.

La comunicación de éste entorno, es que cada vecino por iteraciones comunica la información de las celdas, por lo que la misma es proporcional al volumen o conjunto de celdas, en algunos casos es necesaria la sincronización luego de cada iteración por motivos de desbalanceo.

La alta cantidad de comunicación en éste paradigma es controlada, debido a que siempre se comunican las celdas con sus mismo vecinos en las iteraciones y es predecible el orden de comunicación de los mensajes. Los elementos de cómputos pueden ser generados por cada uno de los nodos que realizarán la ejecución o pueden obtener la información en cualquier sistema de almacenamiento distribuido. Silva y Buyya (1999), describen que Las aplicaciones *SPMD* pueden ejecutarse sin mayor inconveniente en un cluster, si las celdas están correctamente distribuidas en cada uno de los nodos (*Mapping*)) si el entorno es homogéneo. En caso de tener un entorno heterogéneo se debe soportar por algunos esquemas de balanceo de cargas (*Load Balancing*), para adaptar la distribución de carga de acuerdo al entrono de cómputo.

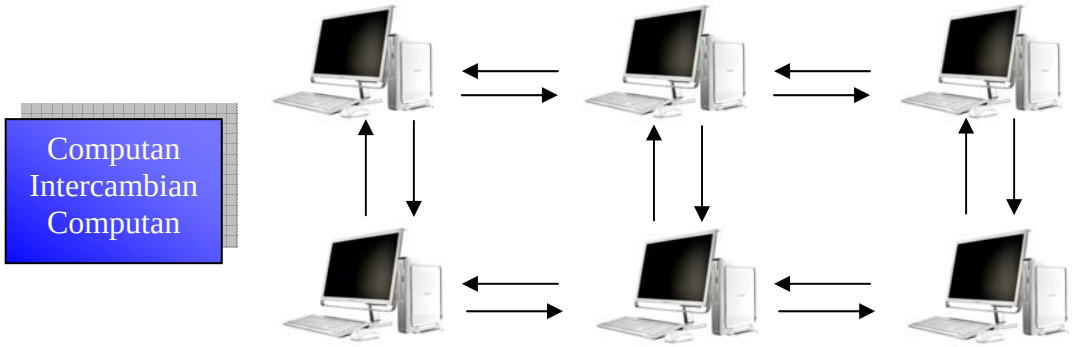


Figura 2.7 Paradigma *SPMD* (*Single Program Multiple Data*)

Otro elemento clave en esta aplicación, es la forma de manejo de ejecución de las tareas (Scheduling), este elemento permite que los procesos se ejecuten en el orden correcto de manera que la información a computar, se complete correctamente, este paradigma es altamente sensitivo a la pérdida de procesos,

por lo que la pérdida de algún proceso puede generar alguna inestabilidad en el sistema.

2.3 Entorno *Multicluster*.

El incremento de la complejidad de las aplicaciones de cálculo científico, ha originado que los sistemas basados en clusters de datos, deban integrarse en una plataforma de colaboración de clusters, permitiendo que las aplicaciones que necesitan mucho tiempo de cómputo se puedan ejecutar en menores tiempos. Este incremento de complejidad computacional, hace que se busquen estrategias para mejorar la administración de los recursos computacionales que se disponen. La disponibilidad de recursos remotos, o clusters ubicados en zonas geográficas distribuidas, permiten que se integren de modo de hacer un cluster de cómputo más complejo y con mayor capacidad de ejecución.

Un *multicluster*s, es un conjunto de *cluster* interconectados equidistantemente y pueden ejecutar un conjunto de trabajos, presentando a los usuarios una interfase de programación simple que permite configurar y utilizar como un entorno integrado Argollo (2006), sin embargo un *multiclusters*, es un ambiente heterogéneo tanto a nivel de cómputo como a nivel de comunicaciones, esto trae como consecuencia que las aplicaciones deban considerar las altas latencias de las redes *WAN* (*Wide Area Network*) que son usadas para interconectar los clusters.

El problema de la heterogeneidad de cómputo, conlleva a que la granularidad de los procesos a ejecutar por el entorno sea variable, por lo que se debe hacer subdivisión de los procesos a la hora de realizar una ejecución, con la intención de mantener balanceado el sistema, una inadecuada granularidad puede desmejorar las prestaciones de la aplicación lo que origina un problema en el performance de la aplicación (Ver Figura 2.7).

Además, un ambiente de *multicluster* incluye otro aspecto relevante, como lo es la comunicación de los clusters, debido a que los enlaces de comunicación *WAN* son muy lentos en comparación a los enlaces de las *LAN* (*Local Area Network*), estas diferencias en las latencias, originan que las aplicaciones deban

tener un alto control con el sincronismo y el nivel de comunicaciones que debe existir en el entorno. Estas consideraciones iniciales son importantes a la hora de configurar un entorno de trabajo que utiliza como plataforma de comunicaciones la Internet.

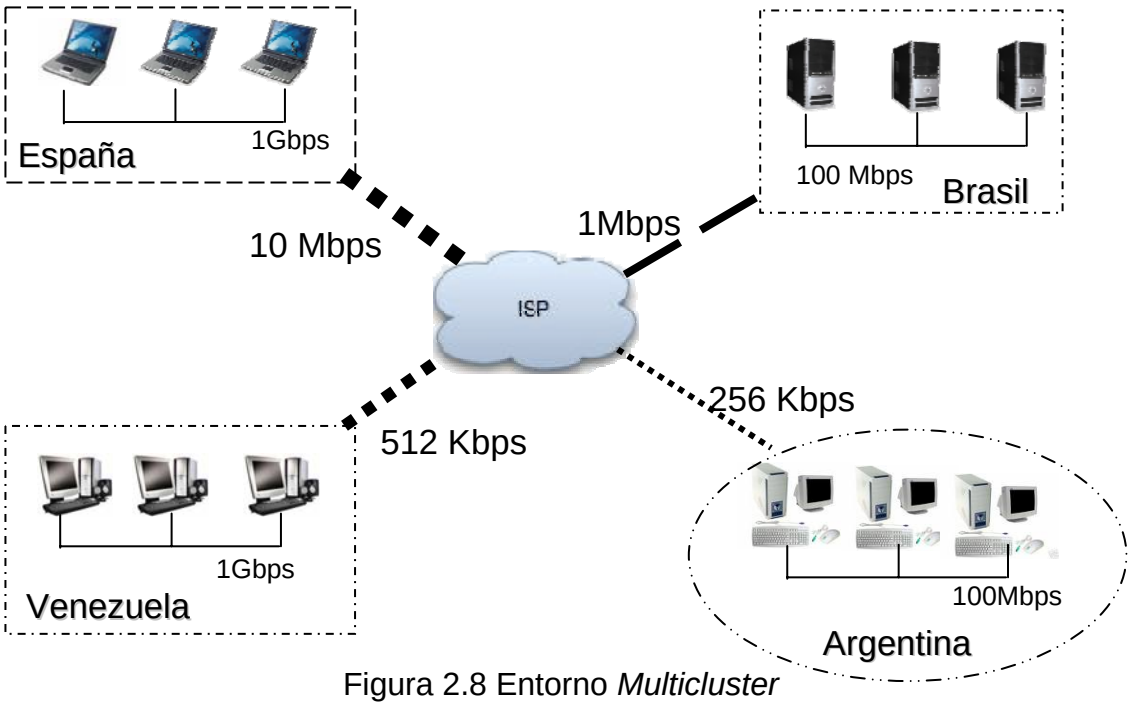


Figura 2.8 Entorno *Multicluster*

Las comunicaciones en un entorno *multicluster*, están compuestas por tres tipos y dependiendo de las aplicaciones pueden usar una combinación o todas las comunicaciones, la comunicaciones se estructuran en niveles de red o Internodo, *InterCluster,* e *Intraclusters,* cada comunicación utiliza un esquema y anchos de bandas variables por lo que las aplicaciones que se ejecutan en estos entornos deben mantener gran control con éste punto Barreto(2000). Este número de comunicaciones origina que las aplicaciones tengan una estructura de mapping de los procesos a ejecutar y el orden de prioridad de ejecución (*Scheduling*).

Por otra parte, el mantener la eficiencia en éste tipo de entornos, depende de la administración de las variaciones de los enlaces de interconexión, por lo tanto una arquitectura *multicluster* a parte de ser un entorno de trabajo que permita generar mayor cantidad de cómputo, debe ser concebido como una alternativa que permite mantener alta robustez de cómputo y ser escalable en el tiempo, lo anterior, con el objetivo de que sea un modelo de implementación para las

aplicaciones paralelas de alto cálculo. Los desafíos de las nuevas tecnologías han permitido implementaciones de éstos entornos mejorando los niveles de eficiencias de las aplicaciones (Barreto, 2000) y (Plat1999)), utilizando como paradigma de programación las aplicaciones desarrolladas en el paradigma de programación *Master Worker (*Argollo, 2006).

2.4 Single Program Multiple Data en un Entorno *Multicluster*.

En las aplicaciones *SPMD*, se mantiene un volumen de comunicaciones con las celdas vecinas, un conjunto de celdas pueden ser asignadas a cada nodo, y las mismas deben seguir un patrón de comunicaciones, por lo que la información debe ser comunicada a las celdas vecinas que en algunos casos se encuentran en otros nodos.

Esa comunicación, hace que en un entorno heterogéneo como el *multicluster*, se deba controlar las comunicaciones que se pueden presentar, en el caso interno de nodo o Intranodo, la comunicación de las celdas es por memoria común, caso que difiere cuando se utiliza las Intercluster y las Intracluster donde se utiliza el paso de mensaje.

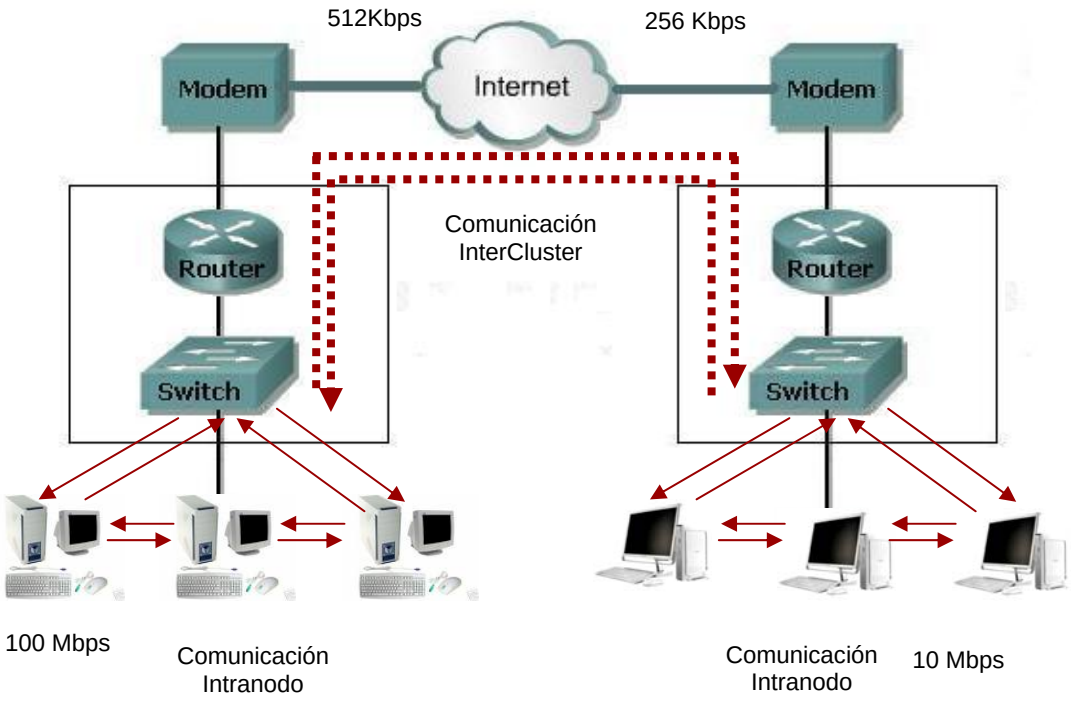


Figura 2.9 Comunicaciones de aplicaciones *SPMD* en Entorno *Multicluster*

Cada comunicación con una velocidad distinta lo que hace un entorno totalmente heterogéneo, y donde se deben controlar las comunicaciones para tener una ejecución donde se obtenga buenas prestaciones. Considerando las variaciones de los anchos de bandas de los canales de comunicación.

La figura 2.9, muestra las comunicaciones entre los clusters remotos, y las que se tienen internas en cada cluster, el control de este tipo de comunicaciones, origina que se deban administrar de acuerdo a la velocidad de transmisión del enlace. Una ejecución de una aplicación *SPMD* en una arquitectura de *multicluster*, hay que tomar en consideración que las aplicaciones son acopladas, por lo que se debe buscar una estrategia para solapar el cómputo y la comunicación, de manera de que las comunicaciones no afecten la ejecución de la aplicación.

Al controlar el conjunto de comunicaciones, se puede seguir con la heterogeneidad de cómputo entre los cluster, la cual determina el número de celdas que se pueden ejecutar, cuando se incluye las celdas, se realiza manejando la teoría de descomposición de Foster (1994), el cual estable que las aplicaciones se pueden subdividir en subconjuntos, y los mismo asignarlos a los nodos.

Inicialmente, para encontrar una relación de cómputo y comunicación que permita controlar las comunicaciones de las aplicaciones *SPMD*, se debe descomponer el problema, comenzando a estudiar el comportamiento en un cluster, en el caso de un cluster se mantienen dos tipos de comunicaciones, tanto a nivel de Intranodo, como Intracluster, por lo que se debe controlar las variaciones de velocidad de las comunicaciones.

2.5 Clusters de Cómputo.

Buyya (1999) , establece que un cluster de cómputo, es un tipo de sistema de procesamiento paralelo o distribuido, compuesto por un conjunto de nodos que ejecutan una serie de aplicaciones de forma conjunta, asimismo, está formando

por una fuente de recursos integrados de computación con el objetivo de ejecutar aplicaciones en menores tiempos de cómputo.

Estos cluster permiten ejecutar una serie de aplicaciones que aumentan el rendimiento y el performance de la aplicación. Los nodos que integran un cluster, puede ser configurados con arquitecturas de uno o varios procesadores (Multicore).

El concepto de cluster de cómputo tomó auge en los años 90, cuando se comienza a disponer de microprocesadores con alto rendimiento de cómputo, incremento de la velocidad de las redes, y estandarización de la computación distribuida a través de librerías de comunicaciones de nodos como MPI (*Message Passing Interface*), PVM (*Parallel Virtual Machine*), entre otros. Adicionalmente, los clusters de cómputo fueron impulsados por las grandes deficiencias que tenían los Sistemas Multiprocesadores Simétricos (SMP) Buyya (1999).

La definición de clusters, tiene diferentes connotaciones, o son clasificados de acuerdo al grupo de investigación o empresa, y de acuerdo al uso para el cual está diseñado. Así, un grupo involucrado en computación científica, que requiere alto rendimiento en la ejecución de la aplicación y con gran cantidad de recursos necesita un cluster *HPC* (*High Performance Computing*), y tendrá una imagen diferente del entrono paralelo a la que tiene un grupo de alta disponibilidad que requiere un cluster *HA* (*High Availability*) que deben ser entornos que funcione ante cualquier tipo de fallas; también, existe otro grupo de clusters que son los utilizados para ejecutar tareas independientes en paralelo o *HT* (*High Throughput*)[8].

Por otra parte, dentro de la definición de clusters, se componen elementos que integrados conforman un sistema de cómputo paralelo, primero un conjunto de computadores (nodos) de altas prestaciones, los cuales pueden estar funcionando con sistemas monoprocesadores, o de multiprocesamiento (*Dual Core, QuadCore*), si se usa entorno multiprocesador entonces permite que el cluster pueda tener un paralelismo a nivel de los nodos, el cual no es diseñado a nivel de programación sino a nivel de la arquitectura del dispositivo. Asimismo, otro

elemento, es el sistema de interconexión de los nodos, que deben ser redes de alta velocidad de comunicaciones, esto, para permitir que la información de transferencia de cómputo de los nodos pueda viajar de manera rápida y confiable sin generar congestión en los enlaces de comunicaciones (Buyya 1999).

Otro elemento para la arquitectura del cluster, es la administración de los enlaces de interconexión, de los cuales incluye el manejo de los protocolos de comunicaciones y servicios de alta velocidad. El *Middleware* o *software* de conectividad del cluster, esta compuesto por dos subniveles de software, en el primero se tiene una imagen única (*SSI: Single System Image*) que permite a los usuarios acceso a todos los recursos del sistemas, y por otra parte el subnivel de disponibilidad del sistema, que permite los puntos de chequeo (*Chekpoints*), los servicios de recuperación de fallos, y el soporte a la tolerancia de fallos. Por último, la integración está compuesta por librerías de comunicaciones y herramientas de computación paralela (Wilkinson 1999). (Ver imagen 2.10)

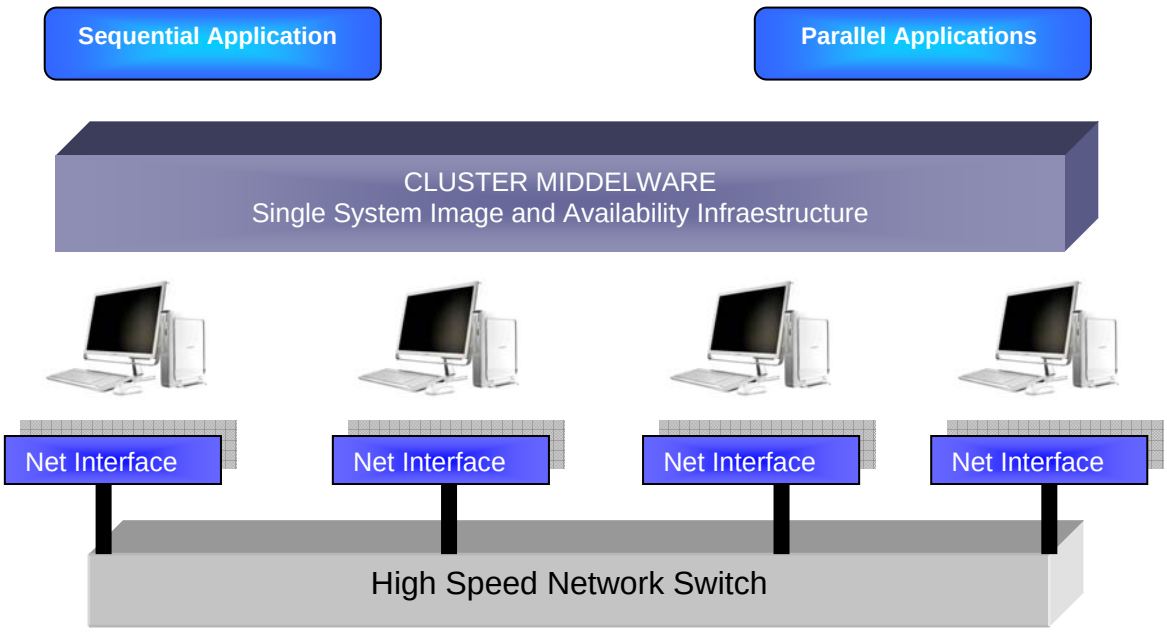


Figura 2.10 Arquitectura de Cluster

2.6 Programación Paralela.

Un programa está compuesto por un conjunto de tareas, que se ejecutan secuencialmente cuando se utiliza un entorno serial; sin embargo, el aumento de

la complejidad de las aplicaciones ha generado la búsqueda de elementos de programación para la ejecución en entornos paralelos; por lo que un programa en la actualidad puede ser visto como un conjunto de procesos que pueden ejecutarse de forma secuencial o paralela.

En un entorno de ejecución secuencial, cada tarea es ejecutada por el procesador, sin que otra actividad o tarea sea ejecutada por la aplicación. En este caso, el algoritmo se evalúa de acuerdo a los términos de tiempos de ejecución, el cual está influenciado por el conjunto de datos de entrada que se tengan para el procesamiento en el sistema.

A diferencia, en los procesos ejecutados en paralelo, se establece una ejecución simultánea de varias actividades, con el objetivo de obtener resultados de ejecución más rápidos a los obtenidos en los entornos seriales. Las tareas ejecutadas, necesitan tener un conjunto de comunicaciones para que puedan mantener un sincronismo o para intercambiar datos necesarios para la ejecución de la misma, con el fin de obtener un objetivo en común el cual es la culminación exitosa de la aplicación.

Los sistemas de cómputo paralelos, utilizan elementos dentro de la arquitectura de los nodos para mejorar la velocidad computacional de la aplicación. La memoria, velocidad del procesador, interconexión de los procesadores, y la forma como se distribuyen las tareas para la ejecución en el nodo, son elementos que permiten que las prestaciones se vean mejoradas para la solución del problema computacional.

La programación paralela se utiliza para generar un grado de simultaneidad a la aplicación que se va a ejecutar, cada componente del algoritmo que se pueda subdividir en un fragmento de ejecución independiente, se puede paralelizar en una actividad siempre y cuando no sean fragmentos dependientes unos de otros, por lo que cada segmento puede ser ejecutado en procesadores diferentes, aumentando la velocidad de ejecución de la aplicación.

Sin embargo, en una ejecución paralela es importante mantener controladas las comunicaciones entre los nodos y más si se utiliza un paradigma como el

SPMD, debido a que ésta es la forma de intercambio de información, por lo que se deben mantener los niveles de sincronismo para que la aplicación paralela pueda culminar con éxito, una mala sincronización puede generar que los procesadores más rápidos tengan que esperar a los más lentos, comprometiendo la eficiencia de la aplicación. La figura 2.11 muestra un entorno paralelo, donde existe un conjunto de comunicaciones entre los nodos para realizar la ejecución de la aplicación.

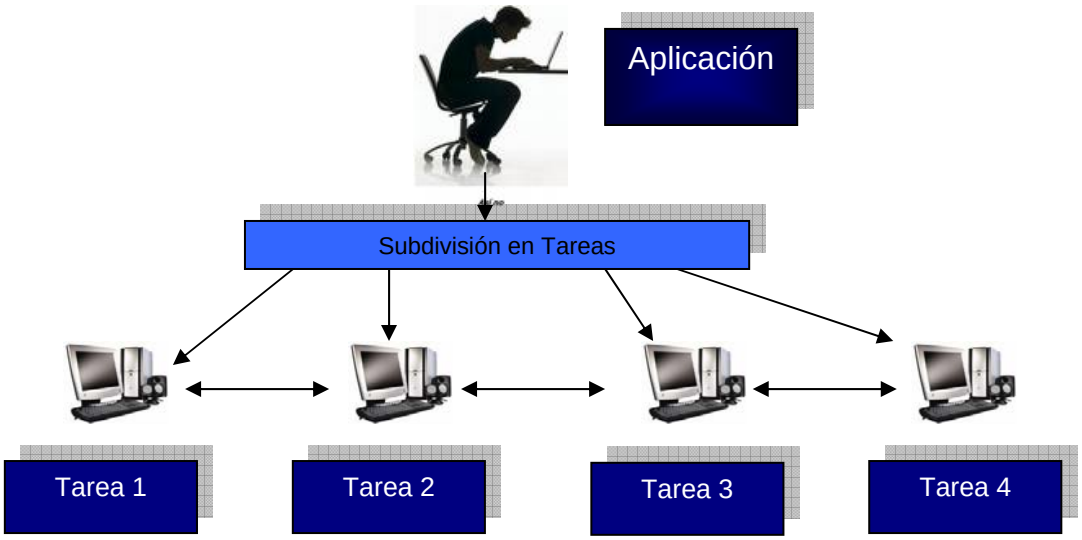


Figura 2.11 Paralelización de una Aplicación

2.6.1 Modelos de Programación Paralela.

El punto fundamental de la aplicación es la correcta subdivisión de tareas a ejecutar; al ejecutar la aplicación más de una tarea, las mismas pueden ser ejecutadas asignándolas en paralelo. Las tareas distribuidas entre los nodos de cómputo, pueden implementarse en forma de procesos o hilos (*pthread*) (Castro, 2007).

Cuando se crea un proceso en la aplicación, se establece un nuevo código, un apuntador de instrucción y la pila. Cuando se crea un proceso se realiza una llamada al sistema operativo, esto crea un nuevo proceso el cual es denominado proceso hijo y tiene las mismas características del proceso padre, con la diferencia que tiene un número de identificador único y diferente. En relación al manejo de los datos, el proceso hijo mantiene una copia de las variables de

entorno del proceso padre, lo que permite manipular los datos independientemente (Goldt, 1995).

Por otra parte, un Hilo o *thread,* tiene generalmente la misma función que un proceso, ejecutar un conjunto de instrucciones; sin embargo, la creación de *thread* difiere a la creación de un proceso, debido a que los *thread*, generan un puntero de instrucción y una pila por cada *thread* desarrollado (Figura 2.12). El nuevo apuntador de instrucción indica otra secuencia de ejecución dentro del mismo código los cuales comparten la misma memoria o área de datos.

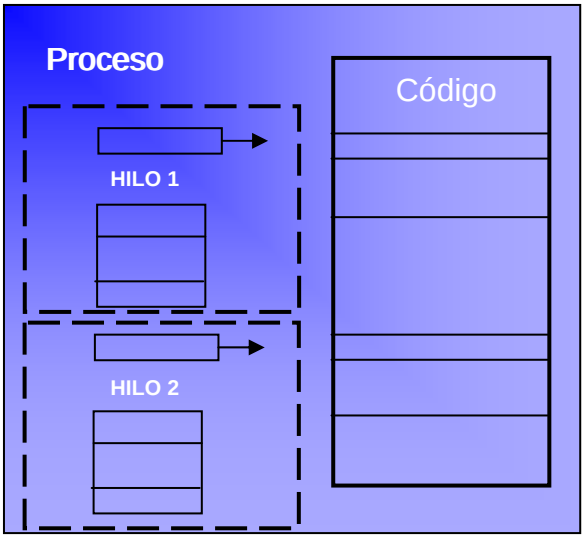


Figura 2.12 Creación de Hilos o *threads*

Los Threads son generalmente independientes, lo que permite que tenga suficiente información del estado de la ejecución, además interactúan entre ellos por mecanismo de comunicación dados por el entorno. Los threads, comparten recursos de sistema de forma directa, y es mucho más rápido cambiar un *thread* a otro en un proceso, a diferencia del cambio de un proceso a otro, el cual no es inmediato. Esto, permite que en el caso de las aplicaciones *SPMD*, se pueda solapar simultáneamente el cómputo con la comunicación.

- **Programación Paralela de Memoria Compartida.**

En el sistema de programación de memoria compartida (*Shared Memory*), varias tareas pueden ejecutar utilizando un mismo espacio de área compartida de la

memoria. Foster (1994), establece que, en los sistemas con memoria central o compartida, cada procesador puede acceder a los espacios reservados, para utilizarlos en la aplicación de comandos de lectura y escritura. Los procesos se pueden comunicar a través de esta área reservada siempre y cuando se haya establecido un sector de la memoria como compartido.

Las modificaciones que se realicen en los espacios compartidos, afectan a todos los procesos que tengan acceso al mismo, el espacio en memoria para proceso tiene que ser declarado previamente. Al hacer una creación o subdivisión de procesos o tareas, el sistema operativo se encarga de la comunicación de ellos a través de entornos de variables, la comunicación resulta sencilla sin embargo se deben tomar ciertos aspectos de sincronismo. La sincronización se necesita en el caso de que varios procesos o threads, quieran hacer acceso de lecturas y escrituras sobre un espacio de datos compartidos.

La memoria convencional puede direccionar un proceso a través de su espacio de direcciones virtuales, por lo que se debe direccionar a los espacios compartidos, para no provocar violaciones de segmentos. Cuando se crean los procesos en una aplicación se realiza una duplicación de todos los elementos, no obstante, para cada proceso nuevo se hace una reserva de memoria, que es inaccesible por otro proceso a menos que éste espacio sea definido como un espacio compartido. Las direcciones de las variables de esta zona son virtuales, y es el módulo de gestión de la memoria el que se encarga de traducirlas a direcciones físicas.

Es importante acotar que el *kernel* no autogestiona los segmentos de la memoria compartida, es necesario asignar cada bloque o segmento de memoria a un conjunto de sincronización, de forma que se eviten problemas de colisiones de datos entre los procesos, es decir que no exista una modificación de un segmento de memoria mientras se realiza una lectura de datos.

Las ventajas de uso de memoria compartida es que permite generar espacios globales de direcciones, además de permitir compartir los datos entre los

procesos, origina que las tareas sean ejecutadas más rápidas y uniforme debido a la proximidad de las memorias a los *CPUs* (*Central Process Unit*) .

- **Programación Paralela de Paso de Mensajes.**

En los desarrollos de aplicaciones paralelas, es necesario intercambios de información y datos entre los procesos, por lo que el modelo de paso de mensajes permite controlar las comunicaciones entre los nodos o procesos que integran la aplicación. En el envío de información, el proceso fuente inicia la comunicación enviando un mensaje de datos utilizando una interfaz de programación de paso de mensajes, y el proceso receptor recibe la información la cual esta etiquetada con el identificador del proceso que recibe la información.

Lo anterior es confirmado por Foster (1994), quien establece que la comunicación de una aplicación se puede dar entre uno o mas procesos, comunicándose llamando rutinas incluidas en librerías de comunicación que permiten el envío y recepción de mensajes hacia otros procesos.

Los mensajes dentro de las aplicaciones pueden ser establecidos de acuerdo al tipo de sincronismo que se le desee incluir al desarrollo paralelo, los mensajes pueden ser sincrónicos, asíncronos, bloqueantes, no bloqueantes, en el caso de ser un mensaje sincrónico, se necesita mantener un acuerdo entre el proceso fuente y destino, en este caso los mensajes se envían y no continúa la ejecución de la aplicación hasta que el nodo receptor confirme la llegada del mensaje, lo cual puede ser detallado en la siguiente figura.

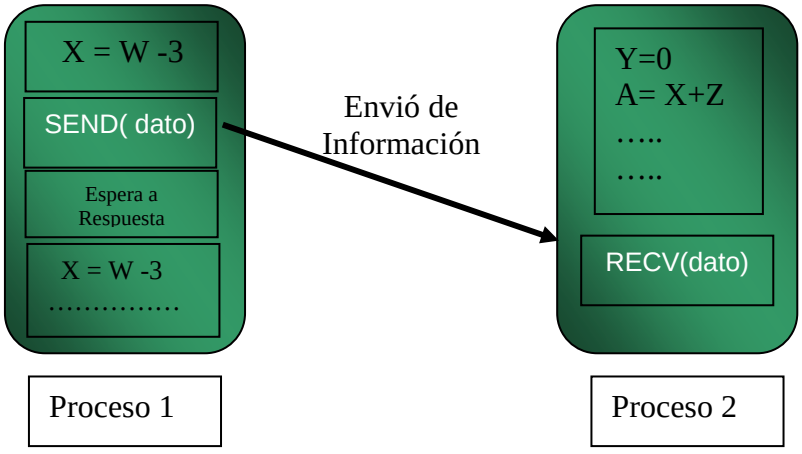


Figura 2.13 Programación de Paso de Mensajes

Al contrario en un esquema de comunicación asíncrono, los procesos no deben establecer un acuerdo de transmisión, por lo que el proceso fuente hace un envío de la información, y continúa con el procesamiento de información sin hacer espera de la recepción del mensaje.

En el paso de mensajes, los procesos tienen sus propios elementos y no son modificados por el mensaje; no obstante, el orden en el que se envían los mensajes y los destinos a la que se transmiten, es un elemento en la ejecución de la aplicación, debido a que un dato erróneo puede afectar la integridad de la ejecución de la aplicación.

La interfaz utilizada para el desarrollo de ésta investigación de la comunicación por paso de mensaje, es conocida como *MPI* (*Message Passing Interface*), y es un protocolo que permite comunicación entre los procesos, y ha sido establecido como un estándar que incluye un conjunto de herramientas y rutinas que permiten que los desarrollos paralelos puedan ser escritos en lenguajes de medio nivel como el C, C++, entre otros; la librerías de *MPI*, han sido ampliamente utilizadas en todas las arquitecturas de memoria distribuida (Ver figura 2.14).

Debido a que el número de procesos en un cómputo de *MPI* es normalmente fijo, se puede enfatizar en el uso de los mecanismos para comunicar datos entre procesos. Los procesos pueden utilizar operaciones de comunicación punto a punto para mandar mensajes de un proceso a otro, estas operaciones pueden ser usadas para implementar comunicaciones locales y no estructuradas.

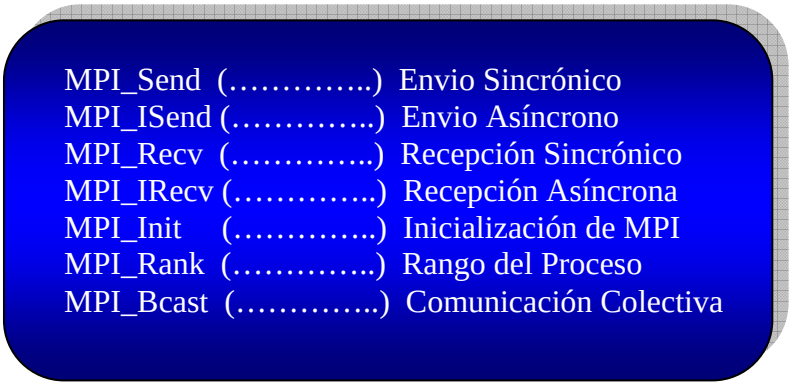


Figura 2.14 Algunas Rutinas de *MPI (Message pssing Interface)*

Además, se puede establecer que un grupo de procesos puede llamar colectivamente operaciones de comunicación para realizar tareas globales tales como *broadcast*, entre otros. Una de las características de *MPI* es que permite la programación modular, el cual establece un mecanismo llamado comunicador, aprueba la definición de módulos que generan un encapsulamiento dentro de las estructuras de comunicación internas.

2.6 Evaluación del Cómputo Paralelo.

En las aplicaciones de entorno paralelo se deben evaluar los parámetros que permiten determinar si la ejecución de la aplicación ha sido exitosa o no, en estos casos no es solo los tiempos de ejecución con la que se ejecutan, si no además elemento que permitan mantener niveles de eficiencia y aceleración computacional.

Las aplicaciones paralelas pueden generar problemas de cómputo, a tal punto que la ganancia en los tiempos de ejecución no sean tan notorias y por ende la ejecución no sea exitosa, no obstante, se tienen formas de hacer evaluaciones de las aplicaciones de acuerdo a métricas de rendimiento, tiempo de ejecución aceleración computacional (*Speedup*), eficiencia, y parámetros como la granularidad, balanceo de carga y escalabilidad de la aplicación.

2.6.1 Tiempo de Ejecución.

El tiempo de ejecución de un programa paralelo, es determinado por el lapso de tiempo que un procesador empieza a ejecutar hasta que el último de los procesadores completa la ejecución. Otro elemento, a considerar en el tiempo de ejecución son los tiempos de comunicación, sin embargo, se logra buscar que los tiempos generados por las comunicaciones se puedan solapar de modo de mantener el mayor tiempo el procesador ocupado.

$$Tiempo\ Ejecución = Tiempo\ Cómputo\ ,\ Tiempo\ de\ Comunicación$$

2.6.2 Aceleración Computacional (*Speedup*).

Wilkinson (1999), establece que uno de los aspectos de mayor interés cuando se diseñan las aplicaciones paralelas, es el valor de la velocidad para resolver un problema o aplicación en condiciones normales de ejecución. Para lograr esto, se considera la mejor versión secuencial de la aplicación, ejecutada sobre un entorno monoprocesador y se debe comparar con la versión paralela para determinar una medida relativa de performance y se define como:

$$Speedup = \frac{Tiempo\ Serial}{Tiempo\ Paralelo}$$

Existen varios factores que pueden sobrecargar la versión paralela y que pueden limitar este factor de aceleración, entre ellos están:

- Periodos en los que los procesadores están ociosos en la aplicación.
- Cálculos adicionales que están en la versión paralela y no en la secuencial.
- El tiempo de comunicación para enviar los mensajes.

El Speedup tiene un límite el cual es definido por la ley de Amdahl, ésta ley determina que independientemente del número de procesadores, existe una velocidad máxima, esto motivado a que existe parte del programa que solamente pueden ser ejecutadas por un procesador f e idealmente los restantes procesos se ejecutan concurrentemente $\frac{1-f}{P}$. El cómputo para P procesadores es definido por $f \times T1 + \frac{1-f}{P} \times T1$, donde *T1* es el tiempo serial, por lo tanto el máximo Speedup esta expresado por:

$$Speed\ Up = \frac{T1}{f \times T1 + \frac{1-f}{P} \times T1} = \frac{P}{1 + \frac{P-1}{f}} = \frac{1}{f + \frac{1-f}{P}}$$

2.6.3 Eficiencia.

La buena administración de los recursos de cómputo es un elemento fundamental a la hora de escribir aplicaciones para entornos paralelos, se debe buscar una

relación entre la aceleración computacional y el porcentaje de tiempo empleado por un recurso efectivo, esto determina si la ejecución ha sido eficiente. La relación es definida como:

$$\textbf{Eficiencia} = \left(\frac{Tiempo\ Serial}{Tiempo\ Paralelo \times Número\ de\ Nodos} \right) \times 100\%$$

Este factor determina cuántos nodos son necesarios para la ejecución de la aplicación de forma que se ejecute de manera eficiente; este valor es expresado en valor porcentual.

2.6.4 Granularidad

Una de los parametros a nivel computacional, es determinar la granularidad, la misma se determina dividiendo el tiempo de cómputo de la aplicación entre el tiempo de comunicación, el tamaño de un proceso se describe por su granularidad y esta es definida por la siguiente razón:

$$Granularidad = \frac{Tiempo\ de\ Computo}{Tiempo\ de\ Comunicación}$$

Un proceso con granularidad gruesa, está conformado por un gran conjunto de instrucciones secuenciales, al contrario de una granularidad fina que tiene un conjunto pequeño de instrucciones. La idea de la granularidad es que el volumen se mantenga una relación del cómputo con respecto a las comunicaciones, sin embargo, con una granularidad mal definida, puede que se reduzca el número de procesos concurrentes, lo que conlleva a que exista menor paralelismo en la aplicación. En este caso, para aplicaciones con alto nivel de comunicaciones se busca que el cociente de la división para la granularidad sea pequeño, al contrario se determina un valor grande cuando se ejecutan aplicaciones con pocas comunicaciones (Foster, 1994).

2.6.5 Balanceo de Carga.

Existen factores que afectan el valor de aceleración de la aplicación. Los problemas de balanceo de carga originan que algunos nodos ejecuten mayor cantidad de cómputo y por ende retrasen la aplicación completa, es importante hacer una correcta distribución de los procesos a ejecutar, de manera que los otros procesos no tengan un tiempo de espera que perjudique los resultados de la ejecución.

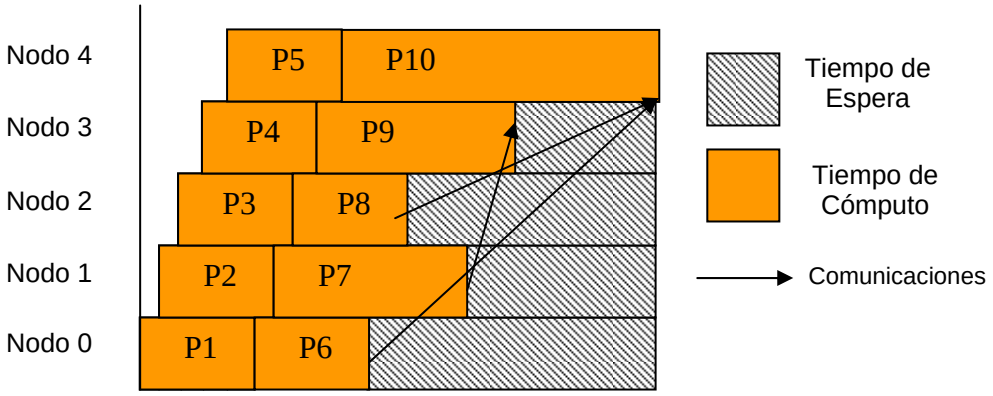


Figura 2.15 Problemas de Balanceo de Carga

2.6.6 Escalabilidad de la Aplicación.

El “performance” de la aplicación, depende del tamaño del problema y del número de procesadores que integran el sistema o entorno en que se este ejecutando la aplicación Wilkinson (1999). Una arquitectura que permite el incremento del número de procesadores para mejorar el rendimiento de la aplicación, se establece como una escalabilidad a nivel de arquitectura. Además, el término de escalabilidad se puede usar a nivel de software, el cual indica que un algoritmo puede trabajar con mayor cantidad de datos, incrementándose los pasos computacionales en una razón importante, esto se conoce como escalabilidad algorítmica. La escalabilidad siempre depende del tamaño del problema.

2.7 Aproximaciones Actuales.

El desarrollo de la investigación, está orientado a la mejora de la eficiencia de las aplicaciones *SPMD* en entornos distribuidos, por lo que los puntos de estudios que hacen referencia en esta investigación están orientados a mejorar las

prestaciones de las aplicaciones *SPMD*, de acuerdo a los puntos de mapping de procesos, scheduling, escalabilidad de aplicaciones y eficiencia, los cuales son puntos focales a los que se están orientando las investigaciones de la comunidad científica en miras de lograr que las aplicaciones SPMD se puedan colocar en entornos heterogéneos y en ambientes *Multicluster*s, para lograr los mejores beneficios del entorno.

En relación a la mejora de las prestaciones de aplicaciones, existen estudios, que permiten medir los tiempos de ejecución de las aplicaciones SPMD; en el caso de manejo de clusters con colas se considera una investigación desarrollada por Cremonesi (2002), el cual propone un modelo que permite la definición de Speedup en aplicaciones con relativa influencia en el procesador, la investigación mencionada, permite modelar aplicaciones paralelas en un cluster, tomando en consideración la entrada / salida y las aplicaciones con alto nivel de comunicaciones.

Además, en los sistemas paralelos que se utilizan, se caracterizan por ser entornos homogéneos o heterogéneos y con aplicaciones que ejecutan una tarea por nodo. En relación a las comunicaciones, las mismas pueden ser alcanzadas utilizando memoria compartida o paso de mensajes, y las cuales pueden ser síncronas o asíncronas.

Por otra parte Oliveira 2005, estudia diferentes técnicas para mantener el balanceo de carga en las aplicaciones SPMD en ambientes grid, las cuales son fuertemente afectadas por éste problema; evalúa el performance de aplicaciones científicas SPMD, ejecutadas sobre un ambiente heterogéneo, para esto ha implementando varias estrategias de balanceo de carga, el aporte de la investigación fue la comparación con varios modelos de balanceo sobre condiciones de ambientes dinámicos, propone la forma de seleccionar el mejor método a la hora de utilizar aplicaciones SPMD sobre ambientes *Grid*.

El Proyecto Prophet**,** es una referencia a la hora de estudiar elementos de scheduling en las aplicaciones SPMD, es un sistema que hace planificación de tareas sobre aplicaciones SPMD en estaciones de trabajos. Este sistema utiliza

una fuente de información que permite seleccionar el tipo de estaciones de trabajos, y el número de tareas asignadas (Weissman 1999).

El sistema está integrado con el Mentat desarrollado por la universidad de Virginia. Una importante característica del Mentat, es que permite controlar la heterogeneidad que existe en varios elementos, como hardware, sistema operativo, sistema de ficheros, y la heterogeneidad de la red, elemento clave a la hora de utilizar aplicaciones SPMD con alto volumen de comunicaciones, además es un sistema diseñado para direccionar las arquitecturas independiente de los programas paralelos. El uso de prophet, ha demostrado buenos resultados a la hora de planificar aplicaciones SPMD, no obstante el modelo ha sido diseñado sin considerar las comunicaciones, por lo que siempre las incluye como un sistema de comunicaciones síncronas.

Por otra parte, las investigaciones anteriores descartan los altos volúmenes de comunicaciones, por lo que Plaat (2001), establece un estudio sobre la degradación de ejecución para un conjunto de aplicaciones, obteniendo diferencias de las latencias y ancho de bandas en redes con comunicaciones intercluster e intracluster en el desarrollo de las aplicaciones, esto permite generar un modelo de descripción de las aplicaciones en estructuras multinivel.

Asimismo, Ball 1998, demostró que una simple optimización de los estructuras multiniveles de las redes, permite obtener alto nivel de performance en aplicaciones que originalmente han sido escritas para un cluster y son ejecutadas en un *multicluster*. Algunos autores Aumage (2002) y Lee (2003) defienden que las comunicaciones de los *Middleware* permiten proveer a las aplicaciones informaciones sobre los estados de las colas de comunicación, esto origina que se tenga mayor control y se pueda mejorar el sincronismo de las aplicaciones.

Con el objetivo de alcanzar altos niveles de *Speedup* y eficiencia en ambientes de cómputo paralelo es necesario estudiar el comportamiento de las comunicaciones para desarrollar un esquema de la topología y por ende del performance de ejecución de la aplicación, las aplicaciones son trabajadas en

Multinivel de comunicaciones donde incluye comunicaciones Internodo, Intercluster e Intracluster.

El propósito de éste trabajo es modelar es comprobar si se puede realizar una ejecución eficiente en ambientes de clusters, lo que permitirá ser un punto de partida para futuras investigaciones en el área de *Multicluster*, una vez concluido, se podrán establecer premisas generales que se desarrollarán en otros estudios y que estarán orientados a conseguir el objetivo que ha derivado este primer paso de la investigación.

Capitulo 3 Aplicaciones SPMD

3.1 Entorno de Trabajo para las aplicaciones SPMD

En el desarrollo de la programación paralela, en la mayoría de los casos se ejecutan aplicaciones donde todos los procesadores corren el mismo programa, pero con diferentes conjuntos de datos. Uno de los modelos que usa este esquema es el *Single Program Multiple Data* (SPMD), por lo que el programador escribe un único programa y luego es ejecutado en cada uno de los nodos que integran el conjunto paralelo de ejecución.

La ventaja de este tipo de aplicaciones es que no es necesario escribir un código por procesador, por lo que existe simplicidad de programación desde el punto de vista de los programadores. No obstante, las aplicaciones SPMD se componen de un alto nivel de comunicaciones, debido a que cada nodo calcula un conjunto de celdas que luego debe intercambiar los datos con los nodos o celdas vecinas, en la figura 3.1 se observa cómo intercambia la información la celda P5 en una aplicación SPMD con los 4 vecinos.

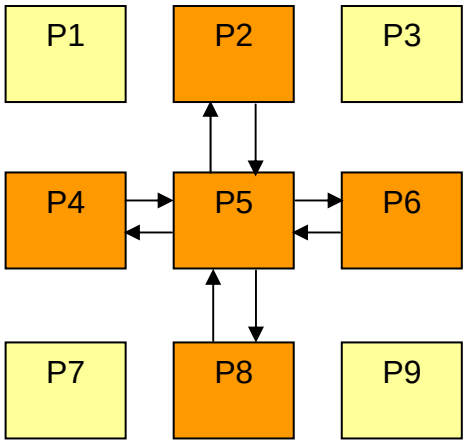


Figura 3.1 Comunicación de una Celda en una aplicación SPMD

La información suministrada por cada celda, es enviada en cada iteración de la ejecución de la aplicación, por lo que cada una para ser ejecutada depende de la información de los vecinos para comenzar los cálculos establecidos en la aplicación; entonces, una aplicación SPMD estará compuesta por un conjunto de

iteraciones donde se repite la siguiente secuencia, envío de información a los vecinos, cómputo de información y envío de los resultados; todos estos pasos se estarán repitiendo por el tiempo que se establezca que dure la ejecución del programa.

Uno de los problemas de estas aplicaciones es el volumen de comunicaciones que se incluyen en este tipo de desarrollo, más si la cantidad de información a ser enviada por los enlaces es muy grande, esto motivado a que los nodos no pueden ejecutar una siguiente iteración hasta recibir la información de los vecinos, los cuales pueden ser 2, 3, 4 o 6 dependiendo del tipo de problemática que se este estudiando en la aplicación SPMD.

La relación de tamaño del mensaje y tiempo de comunicación es un factor variable, debido a que depende del tipo de aplicación a la que se le aplique el paradigma SPMD, esto por las latencias que incluyen las redes y por las tasas de ocupación de los canales de comunicación (volumen de tráfico en red); cuando se trabaja una aplicación en un entorno de trabajo local, estos factores pueden ser un problema, y más aún cuando se trabaja con aplicaciones donde se desea buscar canales de comunicación utilizando las vías de tráfico de Internet, es decir utilizar un entorno *Multicluster* de ejecución Aumage (2002) y Argollo (2006).

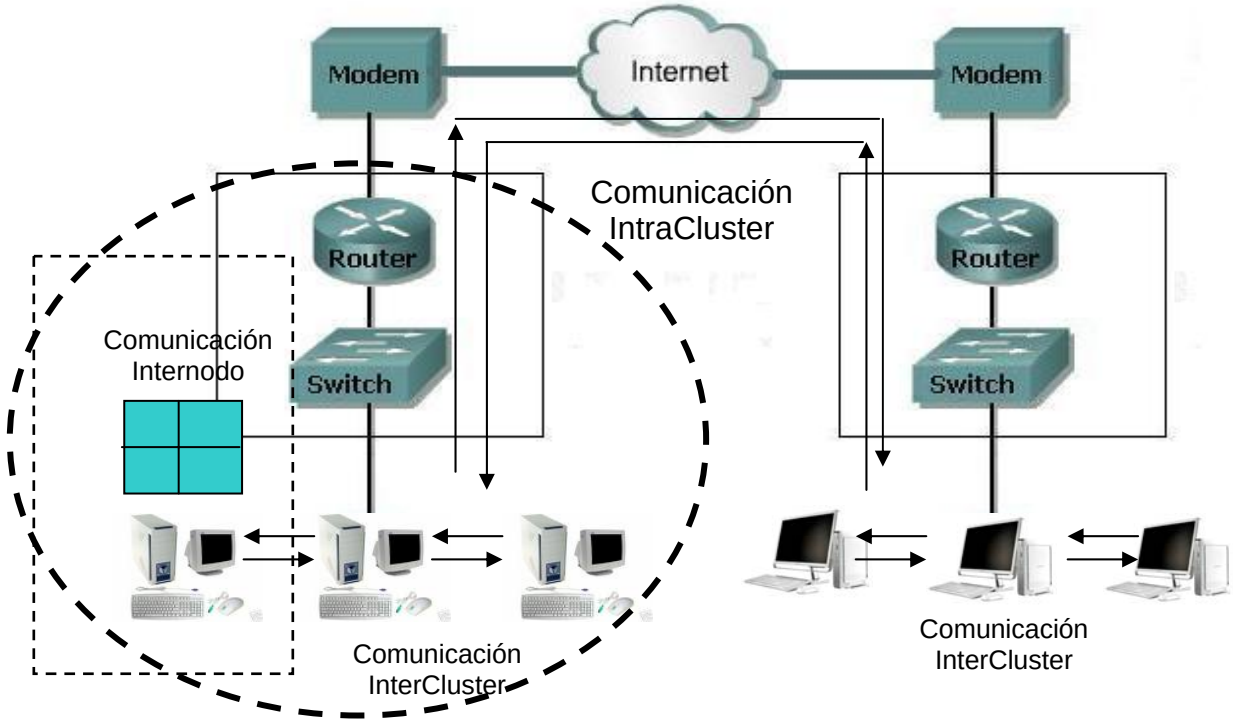


Figura 3.2 Comunicación de una aplicación SPMD en un *Multicluster*

Cada una de las comunicaciones efectuadas en un ambiente *Multicluster*, pueden tener velocidades distintas por lo que el problema de mantener la aplicación sincronizada es otra dificultad a la hora de ejecución si se trabaja en un entorno heterogéneo, en el caso particular de ésta investigación, se desea estudiar el comportamiento de las aplicaciones SPMD en un cluster de cómputo, para controlar dos de las tres comunicaciones que incluye un ambiente *multicluster*.

Las comunicaciones Internodo e Intercluster, son dos tipos de comunicaciones; la primera es ejecutada por cada celda que es asignada a un nodo y que se comunica con las otras celdas internas, y la segunda es la comunicación de los nodos de cómputo del cluster que ejecuta la aplicación. En ambos casos la velocidad de transmisión de los mensajes es variable, en el caso de las comunicaciones Internodo son comunicaciones a nivel de memoria por lo que son comunicaciones rápidas. La tendencia de las nuevas arquitecturas incluyen sistemas *multicores,* que permite que se asignen procesos a cada core en el nodo, por lo que la comunicación Intranodo en éste tipo de arquitectura se realiza a través de memoria compartida. Al contrario en la comunicación Intercluster, se utiliza el sistema de paso de mensajes por lo que son usados como canal de comunicaciones los medios de transmisión de la red local.

En las comunicaciones Intercluster, se pueden presentar retardos de transmisión por el tráfico que es generado por otros nodos, por lo que las comunicaciones se pueden ver afectadas y por ende la aplicación se retrase a nivel de los tiempos de ejecución. Las aplicaciones SPMD deben ser bien controladas para que el costo del volumen de comunicaciones no genere un overhead sobre la ejecución de la aplicación.

3.2 Definición del Problema de Investigación.

Para lograr una ejecución eficiente de las aplicaciones SPMD, se debe controlar el aspecto de las comunicaciones, en el caso particular el número de comunicaciones en un cluster que influyen en la ejecución de la aplicación son

dos tipos, las Internodo e InterCluster, por lo que se busca desarrollar cambios mediante estrategias de mapping y scheduling de celdas, dependiendo de la granuralidad definida en la aplicación. Al desarrollar éstas estrategias se puede establecer una relación entre las celdas o tareas fronteras que tienen comunicación con lo vecinos, y las tareas del nodo que contiene comunicaciones internas, con el objetivo de aumentar el *Speedup* de la aplicación y mantener los niveles de eficiencia preestablecidos. En la figura 3.3 se puede detallar una aplicación SPMD, con un conjunto de tareas de las cuales una parte son definidas como celdas fronteras y la otra parte como tareas internas de la aplicación.

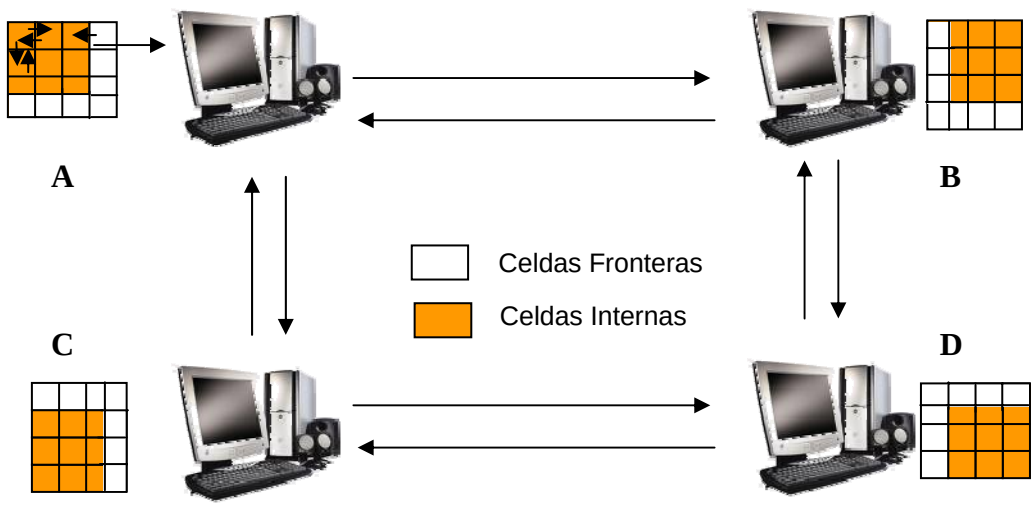


Figura 3.3 Esquema de una Aplicación SPMD asignada a 4 Nodos

En la figura 3.3, se puede detallar el conjunto de celdas asignadas a cada uno de los nodos, un ejemplo es el reflejado por el nodo A, el cual presenta que las celdas asignadas internas tienen una velocidad de comunicación rápida con relación a las celdas definidas como frontera, el motivo es que las celdas internas usan la memoria para trasmitir, mientras las celdas fronteras comunican al nodo vecinos a través de la red de comunicaciones del cluster.

Para establecer un nivel de eficiencia de la aplicación SPMD se parte de la idea de solapar el cómputo con la comunicación, no obstante cada comunicación como se menciono anteriormente, tiene comunicaciones entre las celdas que van a velocidades de transferencias distintas, por lo que se debe estudiar el

comportamiento de los tiempos de comunicación, así como las variaciones que pueden existir a nivel de transmisión del enlace de comunicación, por lo que se que se deba asignar un conjunto de celdas al nodo, de forma de mantener una ratio de cómputo comunicación, que mantenga un umbral de eficiencia en la ejecución de la aplicación y determinar el número de elementos fronteras que serán ejecutadas por la aplicación.

3.2.1 Mapping SPMD

Al establecer una relación de cómputo-comunicación, y generando una política para solapar el cómputo de las celdas internas con las comunicaciones, se puede obtener que la ejecución de las celdas interna se puedan ejecutar al mismo tiempo que se puede comunicar la información con los vecinos, por lo que se busca obtener una relación para asignar un conjunto de celdas internas que comunican a velocidades más rápidas que las celdas identificadas como frontera, con el número de celdas de frontera que debe contener el nodo. Las celdas fronteras comunican a velocidades inferiores que las internas, por lo que el objetivo es mantener una ejecución de tal forma que las tiempo de emitir las comunicaciones de las fronteras, sean igual o similar al tiempo de cómputo del conjunto de celdas internas, es decir mantener un control de comunicaciones Internodo y Intercluster que emitan las celdas asignadas a los nodos.

Entonces, para realizar esto se busca diseñar un modelo de mapping de celdas, que permita la ejecución de la aplicación manteniendo el patrón de solapamiento de cómputo y comunicación, no obstante cuando se asigna el conjunto de celdas a un nodo, se establece lo siguiente: la cantidad de celdas asignadas por el eje “x” y la cual se denotará con nx y por otra parte la cantidad de celdas del eje “y” o ny. En la figura 3.4, se visualiza la asignación de los ejes del conjunto de celdas.

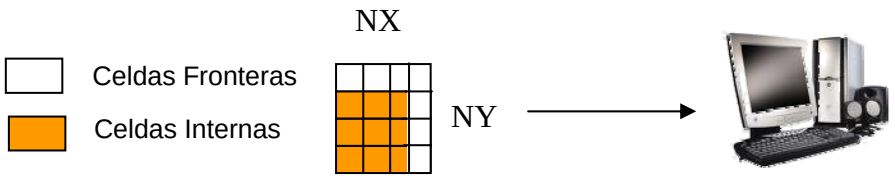


Figura 3.4 Asignación de Celdas a un Nodo

Al establecer las coordenadas, el número total de celdas asignadas esta representada por:

$$\text{ConjuntoCelda} = \text{NX} * \text{NY} \quad (1)$$

Sin embargo, el conjunto de celdas a ser ejecutadas por el nodos y que su tamaño ha sido determinado por la ecuación (1), se debe desglozar el conjunto de celdas que son internas y las celdas de fronteras, por lo que la ecuación queda subdividida como la siguiente ecuación.

$$\text{ConjuntoCelda} = \text{CF} + \text{CI} = \text{NX} * \text{NY} \quad (2)$$

Tabla 3.1 estable un resumen de los parámetros para determinar el conjunto de celdas que integran al nodo.

Tabla 3.1 Parámetros para determinar las celdas globales en un nodo

Símbolo	Descripción
CF	Cantidad de Celda Fronteras
CI	Cantidad de Celda Internas
NX	Valor del conjunto en la coordenada X
NY	Valor del conjunto en la coordenada Y
ConjuntoCelda	Cantidad resultante del número de Celdas

En el caso particular, se esta diseñando un esquema de ejecución homogéneo, por lo que la cantidad de asignación a cada nodo tiene el mismo tamaño y por lo que la subdivisión es de acuerdo al número de Nodo “K”, que incluya el sistema de cómputo. La subdivisión se determina de acuerdo al conjunto global de celdas que integran la aplicación y las cuales las denotamos como “F” en el caso de las filas y “C” en el caso de las columnas, la figura 3.5, muestra un conjunto de celdas que deben ser subdivididas de acuerdo a la distribución lógica de los nodos.

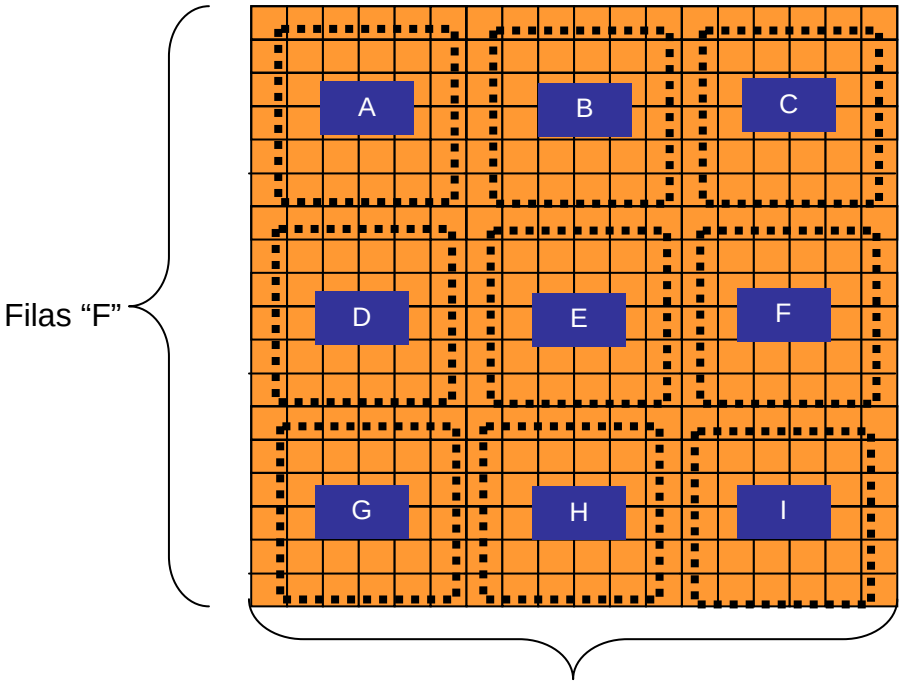


Figura 3.5 Conjunto de Celdas SPMD

En el caso de la fígura 3.5, se tiene un conjunto de celdas las cuales deben ser asignadas a nueve nodos con una distribución 3x3, e decir 3 nodos en el eje de las filas y 3 en el eje de las columnas. Cada segmento de asignación tendrá entonces las coordenadas en “X” y “Y”, para cada uno de los nodos, por lo que el cálculo de las variables nx y ny explicadas en la ecuación (1), se determinan de la siguiente forma.

$$NX = C / KC \quad (3)$$

$$NY = F / KF \quad (4)$$

Tabla 3.2 Calculo de Coordenadas X / Y, para distribución en Nodos

Símbolo	Descripción
NX	Valor del conjunto en la coordenada X para el Nodo
NY	Valor del conjunto en la coordenada Y para el Nodo
C	Cantidad de Celdas Totales en Columnas
F	Cantidad de Celdas Totales en Filas
KC	Nodos en la topología lógica en Columnas
KF	Nodos en la topología lógica en Filas

Una vez obtenido los valores de subdivisión se puede tener una asignación de celdas, la cual depende de la capacidad de cómputo de la aplicación y del volumen de comunicaciones, por lo que la idea es generar trozos que sean asignados a cada nodo, las letras de la figura 3.5 indican el nodo al cual se le asignará ese trozo para ejecución.

Los trozos inicialmente se pueden generar de forma homogénea (Figura 3.5), para realizar pruebas iniciales, sin embargo, pueden seleccionarse con heterogeneidad, dependiendo del factor de cómputo de cada uno de los nodos, y del tipo de comunicación, si es de dos, tres o cuatro comunicaciones con los nodos vecinos (Figura 3.9).

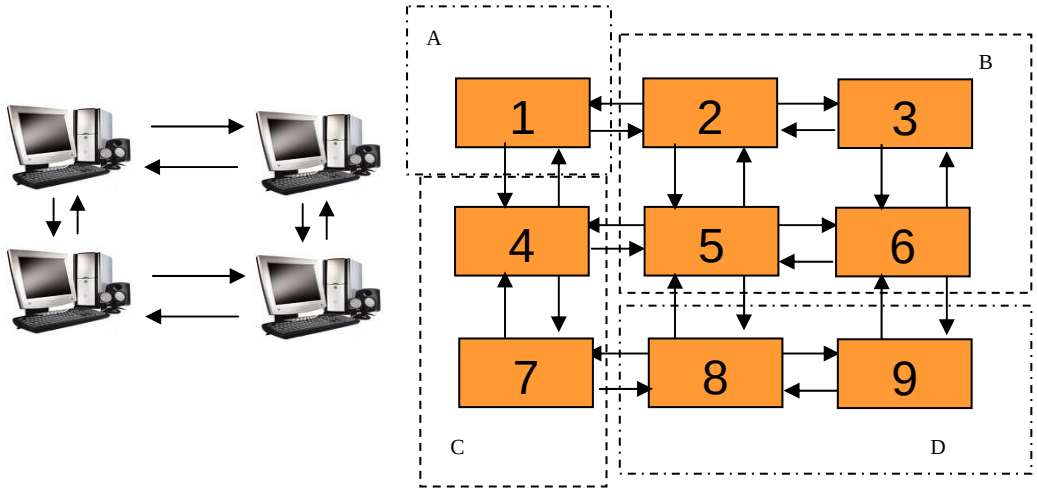


Figura 3.6 Mapping Heterogéneo de Celdas

En la figura anterior, se tiene un caso con 4 nodos y 9 celdas, la asignación en este tipo de caso debe ser totalmente heterogénea y el volumen de comunicaciones también, debido a que existe disparidad de celdas fronteras, el objetivo en este caso es asignar las celdas de acuerdo a la capacidad de cómputo de cada nodo que integre el sistema de cómputo, uno de los factores que se debe tener presente es el tiempo de comunicación de las celdas fronteras y la cantidad de celdas internas que se ejecutan.

Al obtener entonces las coordenadas, de las ecuaciones (3) y (4), se hace la subdivisión homogénea del conjunto de celdas, sin embargo se presenta los casos donde cada nodo tendrá un subconjunto de celdas, y las mismas serán

asignadas de acuerdo a la topología lógica asignada. La figura 3.7, muestra un ejemplo de asignación de coordenadas de los nodos tanto a nivel de “KF” y “KC”, que son la cantidad de nodo en filas y cantidad de nodos asignados a las columnas respectivamente.

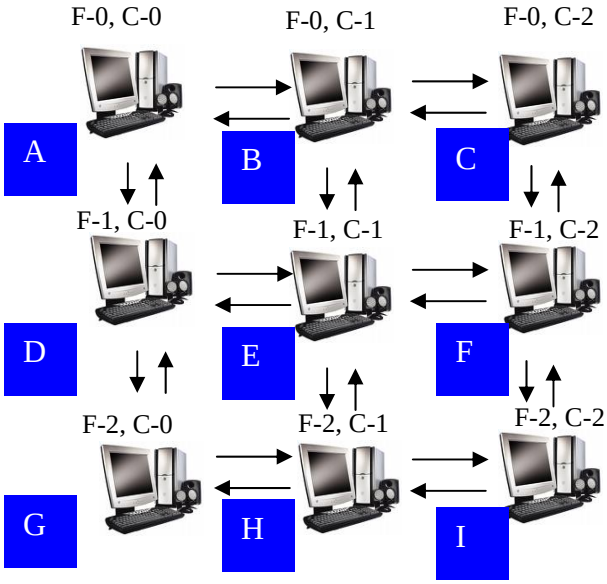


Figura 3.7 Distribución Lógica de los Nodos

La topología lógica de la figura 3.7, muestra que existen nodos con comunicaciones variables, en el caso del nodo identificado con F=0, C=0 (A) , se tiene que la comunicación es realizada con 2 vecinos, con B y con D, no obstante, el nodo identificado con F=0, C=1 (B) , tiene 3 comunicaciones con A, C y con E, por otra parte en el caso del nodo identificado con F=1, C=1 o (E), tiene 4 comunicaciones con B, D, H y F, es decir de acuerdo a la ubicación del nodo dentro de la topología lógica, tiene un conjunto de comunicaciones que es variable y que influye en la cantidad de celdas fronteras que se deban ejecutar.

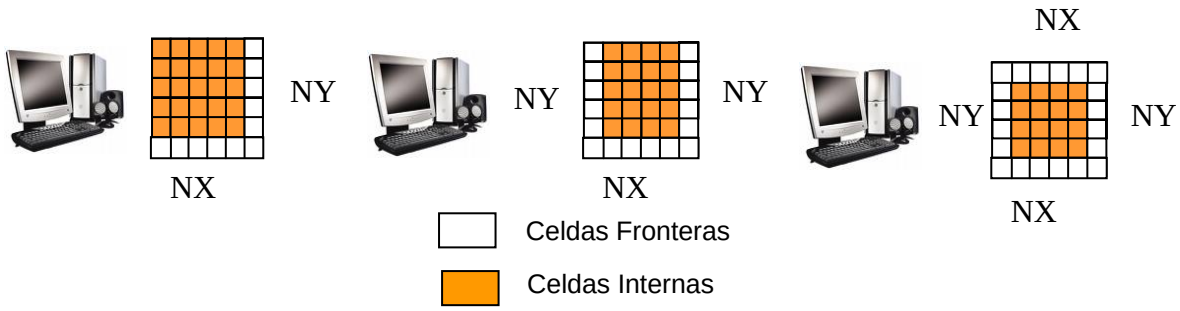


Figura 3.9 Casos de Celdas Fronteras.

La figura 3.9, describe la cantidad de celdas fronteras que pueden presentarse cuando se asignan un conjunto de celdas a los nodos de cómputo, por lo tanto la cantidad de celdas fronteras que tiene cada nodo, va de acuerdo al número de fronteras que contenga en cada coordinas por el número de celdas de la coordenadas, es decir

$$CF = (p * NX + p1 * NY) - (p * p1) \quad (5)$$

En la tabla 3.3 se describen los parámetros de la ecuación

Tabla 3.3 Cálculo de Celdas Fronteras.

Símbolo	Descripción
NX	Valor del conjunto en la coordenada X para el Nodo
NY	Valor del conjunto en la coordenada Y para el Nodo
p	Cantidad de fronteras en el eje X
p1	Cantidad de fronteras en el eje Y
CF	Cantidad de Celdas Fronteras

Al determinar, las celdas fronteras ecuación (5) y teniendo el conjunto global de celdas que son asignadas a cada nodo ecuación (1), entonces se puede establecer la cantidad de celdas internas de acuerdo a:

$$CI = ConjuntoCelda - CF \quad (6)$$

En la tabla 3.4 se describen los parámetros de la ecuación

Tabla 3.4 Cálculo de Celdas Internas

Símbolo	Descripción
CF	Cantidad de Celda Fronteras
CI	Cantidad de Celda Internas
ConjuntoCelda	Cantidad resultante del número de Celdas

Un ejemplo que permita mostrar las ecuaciones anteriores, se puede establecer como: supóngase que se tiene un conjunto de 900 celdas x 900 celdas en 9 procesadores con una distribución lógica 3x3, entonces si determinamos que los valores de F=900, C=900 KF=3 y KC=3, entonces se aplica la ecuación (3) y (4) para calcular los valores de NX y NY, dando como resultado, que el valor de NX=300 y NY=300.

Al obtener la cantidad de NX y NY, se puede calcular el conjunto de celdas globales que tendrá cada uno de los nodos, en éste caso se aplica la ecuación (1) y se obtiene.

ConjuntoCelda= 300 * 300 = ConjuntoCelda = 90000 Celdas.

Al obtener el total general de las celdas, se debe determinar el número de elementos fronteras que tendrán cada uno de los nodos, eso se describirá en la siguiente tabla 3.5, donde se detalla la cantidad de celdas de fronteras, aplicando la ecuación (5) cuando los nodos tienen 2,3 y 4 comunicaciones respectivamente,

Tabla 3.5 Ejemplo de Cálculo Celdas Fronteras e Internas.

Ubicación Fila	Ubicación Columna	Conjunto Celda	Cant. Comunic.	Cant. Fronteras NX (p)	Cant Fronteras NY (p1)	NY	NX	Celdas Fronteras Totales
0	0	90000	2	1	1	300	300	599
0	1	90000	3	1	2	300	300	898
1	1	90000	4	2	2	300	300	1196

Una vez culminado el calculo se determina entonces cuantas son las celdas Internas a ejecutar, aplicando la ecuación (6) y se detalla en la tabla 3.6.

Tabla 3.6 Cálculo de Celdas Internas

Ubicación Fila	Ubicación Columna	Cant. Comunic.	Conjunto Celda	Celdas Fronteras Totales	Celdas Internas
0	0	2	90000	599	89401
0	1	3	90000	898	89102
1	1	4	90000	1196	88804

La tabla 3.6, muestra la cantidad de celdas a computar de acuerdo a las que se describen como frontera y como celdas internas, entonces, se puede determinar, que el número de celdas internas es mayor que el de las celdas fronteras por lo que se puede expresar lo siguiente

$$CF < CI \qquad (7)$$

Obteniendo, esta relación de igualdad, se determina que el tiempo de cómputo del conjunto de celdas fronteras, es menor al de las que integran las celdas internas, por lo que la figura 3.10, muestra el comportamiento de cómputo y considerando que las comunicaciones son variables motivado al uso de canales de comunicación distinto, que en el caso de las celdas internas es la memoria y en el caso de las celdas fronteras son lo canales de la red, entonces tenemos un diagrama como el siguiente.

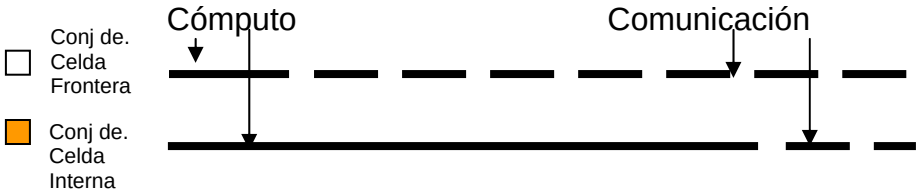


Figura 3.10 Ejecución y Relación Cómputo Comunicación.

En la figura 3.10, se muestra, que el tiempo en computar de las celdas fronteras es menor que el de las celdas internas, esto motivado a la cantidad de celdas que tiene cada uno de los conjunto tanto el de celdas fronteras (CF), como el de las celdas internas (CI), donde CF >CI (7), esta relación entonces depende de las velocidades de transmisión, a la hora de comunicar a las celdas vecinas la información, por lo tanto lo que se desea es buscar un relación de cómputo y comunicación, donde cada conjunto pueda terminar al mismo tiempo, tanto el cómputo interno como las comunicaciones de las fronteras, lograr éste esquema permite que las comunicaciones estén solapadas y por ende no generen un overhead sobre la aplicación, lo que mejoraría la eficiencia del entorno.

3.2.2 Modelo de Evaluación

El modelo de ejecución de aplicaciones SPMD, es necesario en la presente investigación, para determinar el tiempo de ejecución y los niveles de eficiencia que se desea estudiar en la aplicación. Los parámetros de ejecución dependen de la propia aplicación y no son variables así se cambie el entorno de trabajo; es decir, el número de tareas que se tienen en un problema siempre se mantienen, aunque se realicen variaciones del entorno, igualmente el nivel de comunicaciones. Las variaciones pueden existir a nivel de tiempos de respuesta, tanto en el cómputo como en la comunicación, en relación a la comunicación está influenciado por el canal de transmisión, y por el tráfico de la red.

La ejecución de las celdas influye en el tiempo de ejecución de la aplicación, por lo que la función descrita en el capitulo 2 relacionada con el tiempo de ejecución es expresada como la ecuación (8).

$$\text{Tejecución} = \text{Tcómputo} + \text{Tcomunicación} + \text{Tsolapamiento} + \text{Tocio} \qquad (8)$$

En la tabla 3.7 se describen los parámetros de la ecuación

Tabla 3.7 Parámetros de Cálculo Tiempo de Ejecución

Símbolo	Descripción
Tcómputo	Tiempo Total en computar las celdas
Tcomunicación	Tiempo en Comunicar
Tsolapamiento	Tiempo Solapado entre Cómputo y Comunicación
Tocio	Tiempo que el procesador no esta en uso.

Entonces, el tiempo de ejecución de una aplicación SPMD está compuesto por elementos de los que se pueden mencionar, el tiempo de cómputo, el tiempo de comunicación, el tiempo de solapamiento y el tiempo de ocio. Por lo que, se puede establecer que el tiempo de ejecución de la aplicación como una relación entre los factores anteriormente descritos y quedando representado como la ecuación (8), estos valores determinan el comportamiento del tiempo de ejecución de la aplicación.

- **Tiempo de Cómputo.**

El modelo computacional de una aplicación, está compuesto por el número total de tareas a ejecutar como trabajo, en el caso de las aplicaciones SPMD, que mantienen el mismo código de programa, se puede establecer que el conjunto de cálculos de la aplicación están conformados por la misma cantidad de operaciones o la complejidad de trabajo que tiene la aplicación. El tiempo de cómputo de cada una de las celdas se compone por la ecuación (9)

$$\text{Tcómputo(iter)} = \text{Tn} * (\text{CF} + \text{CI}) \quad (9)$$

En la tabla 3.8 se describen los parámetros de la ecuación

Tabla 3.8 Parámetros de Cálculo Tiempo de Cómputo en una Iteración

Símbolo	Descripción
Tcómputo(iter)	Tiempo Total en computar las celdas en una iteración
CF	Cantidad de Celda Fronteras
CI	Cantidad de Celda Internas
Tn	Tiempo en ejecutar una celda

- **Tiempo de Comunicación.**

Los parámetros relacionados con la comunicación, son evaluados de acuerdo al volumen de comunicación que las celdas desean transmitir con otras celdas en la aplicación SPMD. Las comunicaciones en un entorno de cluster están compuestas de dos tipos de comunicaciones las Internodo y las Intercluster, en ambas comunicaciones el flujo de información a comunicar a los vecinos es el mismo; la diferencia fundamental es el método de comunicación a utilizar y el tiempo de transmisión

Las comunicaciones internodo, se realizan directamente en la memoria del nodo, por lo que la velocidad de comunicación es muy elevada por ser dentro del mismo nodo, estos valores de comunicación son despreciables y no se pueden solapar

con el cómputo, debido a que necesita la información de memoria para poder ejecutar la tarea en el nodo.

$$TcommInternodo = Lmem + CDTM / TasTraf \quad (10)$$

En la tabla 3.9 se describen los parámetros de la ecuación

Tabla 3.9 Parámetros de Tiempo de una Comunicación Internodo

Símbolo	Descripción
TasTraf	Tiempo de Transferencia de memoria.
Lmen	Latencia de Acceso a Memoria
CDTM	Cantidad de datos a copiar de memoria
TcommInternodo	Tiempo de una Comunicación Internodo

La comunicación Intercluster es realizada por paso de Mensaje, por lo que utiliza la red de Interconexión para la transmisión de datos, esto genera una latencia de comunicación mayor influenciada por los medios de transmisión y los dispositivos de interconexión que en el primer esquema. La comunicación es expresada como

$$TcommIntercluster = LMPI + TamMens / BW + LR \quad (10)$$

En la tabla 3.10 se describen los parámetros de la ecuación

Tabla 3.10 Parámetros de Tiempo de una Comunicación Intercluster

Símbolo	Descripción
TamMens	Tamaño del Mensaje a enviar
BW	Ancho de Banda del enlace de comunicación
LMPI	Latencia de Preparar un Mensaje de MPI
LR	Latencia de la Red
TcommIntercluster	Tiempo de una comunicación Intercluster

Entonces, al describir el entorno de las comunicaciones queda, expresado por la cantidad de comunicaciones Internodo y las comunicaciones Intercluster.

$$Tcomunicación = P1 \times \left(\alpha + \frac{CDTM}{TasTraf} \right) + P2 \times \left(\beta + \frac{TamMess}{BW} + \Lambda \right) \qquad (11)$$

Donde, los valores de P1 y P2, son las comunicaciones de cada tipo y Tcomunicación el tiempo global de comunicaciones de una iteración.

3.2.3 Scheduling SPMD

Para cumplir un solapamiento con miras de mejorar la eficiencia, se debe establecer el orden de ejecución de cada una de las celdas de acuerdo al orden de prioridad (Scheduling de las celdas). El orden de ejecución de las celdas depende de la prioridad de la misma en el caso de ser un celda con dos comunicaciones a nodos vecinos, entonces se le asigna la prioridad de ejecución más alta (1), en el caso de que solamente mantenga una comunicación con un vecino a nivel de red, entonces se le asigna la prioridad (2) y por último si la comunicación es solamente interna (3), esto establecerá un orden de ejecución inicial donde las fronteras se ejecutaran primero y luego se ejecutaran las celdas internas, cuando se este comunicando la frontera con los otros nodos (Ver Figura 3.10).

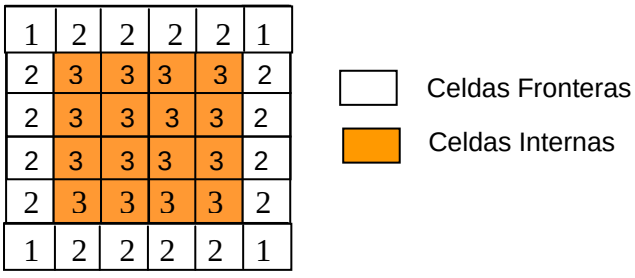


Figura 3.11 Orden de Ejecución de Acuerdo a la Prioridad de la Celda

Sin embargo, existe un relación entre el mapping de las celdas o la cantidad de celdas asignadas y la ejecución, motivado al tiempo que puede tardar el enlace de comunicación en enviar el mensaje, por ejemplo si se busca una relación de cómputo y comunicación, que permita que las celdas se puedan ejecutar simultáneamente, se debe considerar que pueden existir 3 casos. El primero es cuando existe un tiempo de comunicaciones más alto que el tiempo de cómputo interno, lo que generaría un entorno con C*ommunication Bounded* (Figura 3.11),

CF CI

esto genera que la aplicación en cada iteración de cómputo tenga retrasos, motivados a los niveles de comunicaciones.

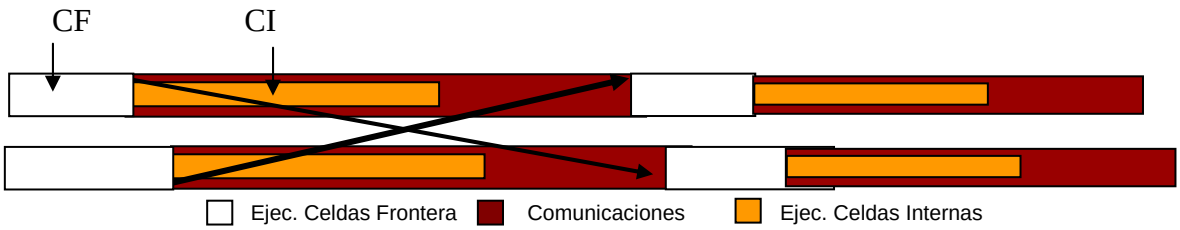


Figura 3.11 Caso de Comunication Bounded.

Por otra parte, se puede presentar que se busque el número de celdas internas que se deben ejecutar de forma tal que exista la relación o se mantenga una aplicación *Computation Bound*, o con alto nivel de cómputo, esto es descrito en la figura 3.12, donde la cantidad de celdas asignadas para el cálculo interno, supera a las comunicaciones.

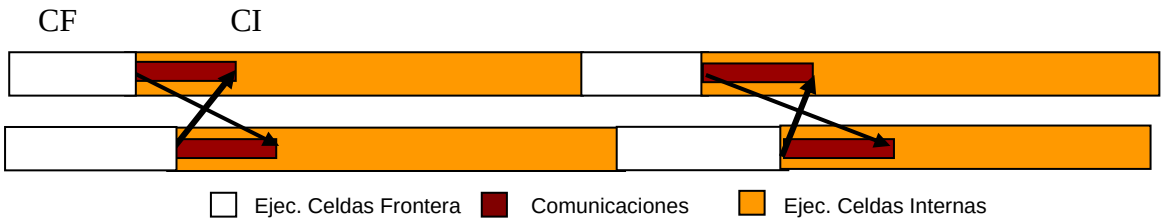


Figura 3.12 Caso de Computation Bounded

Sin embargo, se puede buscar un punto ideal donde se solape el cómputo y la comunicación, para asignar la cantidad de cómputo que excede el nodo y asignárselo a otros nodos, se debe mantener una relación entre el cómputo y la comunicación de forma de cumplir la siguiente relación, derivada de las ecuaciones (6) que calcula el número de celdas internas y la comunicación de las celdas fronteras realizadas por paso de mensajes (10), por lo tanto queda una relación.

$$Tn * CI \geq LMPI + (\text{TamMens})/\text{BW} + \text{Lr} \qquad (12)$$

La relación de la ecuación anterior permite, que se mantenga un nivel de solapamiento entre el cómputo y la comunicación, donde a más cerca de la igualdad esten los valores, se entiende que existe un mejor solapamiento, por ende se mantendría el nivel de eficiencia, debido al solapamiento total de las comunicaciones (Figura 3.13).

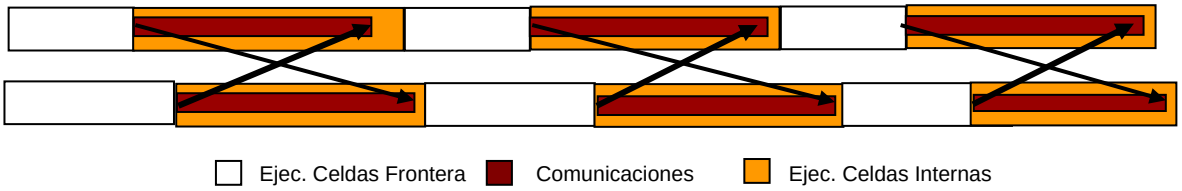


Figura 3.13 Caso Aceptable

Para tomar en consideración los tres casos anteriores, se debe establecer el número de celdas que deben ser asignadas a un nodo, las cuales pueden ser calculadas de acuerdo a las ecuaciones (1)(5)(6), con el objetivo de mantener el solapamiento, donde el cómputo debe ser mayor o igual al volumen de comunicaciones intranodos. Un esquema de mapping de celdas permite dar una relación entre los elementos que comunican y los que van a computar, el ideal es mantener un control del volumen de comunicaciones si se desea obtener una ganancia considerable en la aplicación.

Cada caso es particular debido al tipo de nodo en el cual se está ejecutando, en caso de un entorno homogéneo el valor de n es igual para cada uno de los nodos, esto permite que el análisis este enfocado a nivel de los nodos con mayor número de comunicaciones.

El mantener esta relación permite que la aplicación SPMD sea *computation bound* , o se mantenga un esquema de solapamiento ideal que cumpla la igualdad de la ecuación (12), esto origina que las comunicaciones no retrasan los tiempos de ejecución de la aplicación, no obstante se busca el elemento ideal el cual debe ser regido por la igualdad de la ecuación.

En el siguiente capítulo, se puede detallar la evaluación experimental que apoya la idea del solapamiento utilizando un mapping de celdas y esquemas de

scheduling que permiten la ejecución eficiente de las aplicaciones SPMD en ambientes distribuidos.

Capitulo 4. Evaluación Experimental

4.1 Introducción

El siguiente capítulo explica los experimentos realizados que permiten comprobar la pregunta planteada inicialmente en el capitulo 1. Asimismo, expone el estudio de acuerdo a los objetivos planteados para el desarrollo del trabajo.

Las aplicaciones SPMD tienen un comportamiento particular con respecto al resto de los paradigmas de programación paralela, debido al volumen de comunicaciones con las celdas vecinas, que cada una de las celdas integrantes de la aplicación debe realizar. Lo anterior, conlleva a visualizar el comportamiento de una aplicación SPMD y observar cómo es el patrón de comunicaciones y el cómputo con cada una de las celdas.

El estudio está basado en una aplicación SPMD de dos comunicaciones con los vecinos, que permite calcular la transferencia de calor de un cuerpo, el cálculo de la aplicación se realiza con diferencias finitas, y contiene un conjunto de iteraciones para calcular cada celda que integra la aplicación. Sin embargo para realizar la presente investigación, se han realizado modificaciones al patrón de comunicaciones de forma que la comunicación no sea solamente a dos vecinos, sino que sea ejecutada de acuerdo al patrón que se desea estudiar que integra cuatro vecinos.

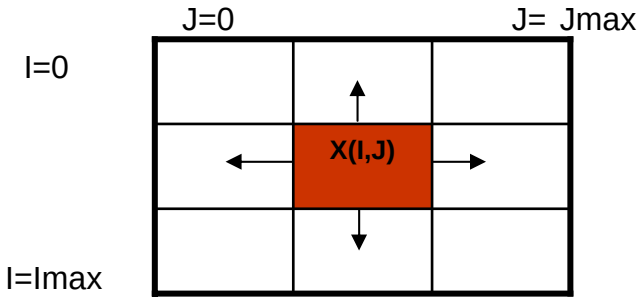
Para el desarrollo de los experimentos, se esquematizó la distribución de los datos de acuerdo al esquema de mapping. Al asignar el conjunto de elementos se comienza la ejecución, de acuerdo al orden de prioridad de las celdas, en este caso se observan las trazas de ejecución, para visualizar el comportamiento de la aplicación y la asignación a cada nodo. Por otra parte, se evaluó la asignación de diferentes tamaños de problema a cada nodo, mostrando los resultados con trazas de ejecución.

Al visualizar los resultados, se puede detallar el solapamiento entre el cómputo y la comunicación, obteniendo un punto donde se determina el solapamiento que podría mencionarse como ideal.

Finalmente, se evalúan las métricas de rendimiento de la aplicación, aplicando los conceptos de mapping y scheduling. Éste análisis se realiza comparando el patrón de 2 comunicaciones entre los nodos, es decir la aplicación que originalmente se obtuvo y luego se analizó el patrón de 4 comunicaciones solapadas, observándose los niveles de eficiencia y escalabilidad.

4.2 Entorno Experimental a nivel de Software

La aplicación de transferencia de calor, se encarga de calcular el comportamiento de los puntos de una región, comunica su información a los elementos vecinos, y luego vuelve a computar a través de un cálculo de diferencias finitas. La figura 4.1, muestra la comunicación de la aplicación y el cálculo que cada celda debe realizar en cada iteración de la aplicación.



$$X(I,J)_{n+1} = X(I,J)_n + ax\,[X(I+1,J)_n + X(I-1,J)_n - 2*X(I,J)_n\,] + ay[X(I,J-1)_n + X(I,J-1)_n - 2 * X(I,J)_n\,]$$

Figura 4.1 Comportamiento de Cálculo de la aplicación Heat2d.

En cada una de las iteraciones, se obtiene un gráfico o un fichero que demuestra el comportamiento de los puntos, y la generación de calor de los mismos, a mayor número de iteraciones o ciclo de cálculo de todos los puntos que conforman la aplicación, se obtienen resultados más precisos acerca de la aplicación.

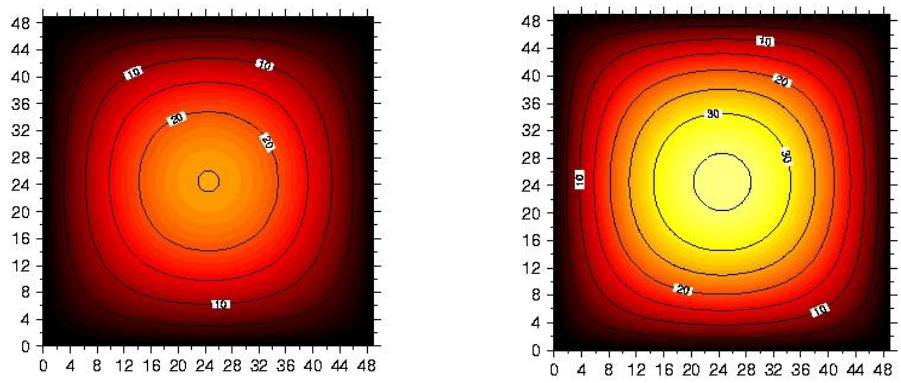


Figura 4.2 Iteraciones de la Aplicación SPMD heat2d

Por otra parte, la aplicación SPMD de transferencia de calor o heat2d, está originalmente diseñada para que el modelo de comunicación de los nodos sea de 2 comunicaciones con 2 vecinos; es decir, se comporta como la imagen de la figura 4.3, por lo que el nivel de comunicaciones con los vecinos es solamente bidireccional, por lo que comunica, computa y luego comunica los resultados a los vecinos, este paso es repetido de acuerdo al número de iteraciones que se configure en la ejecución de la aplicación.

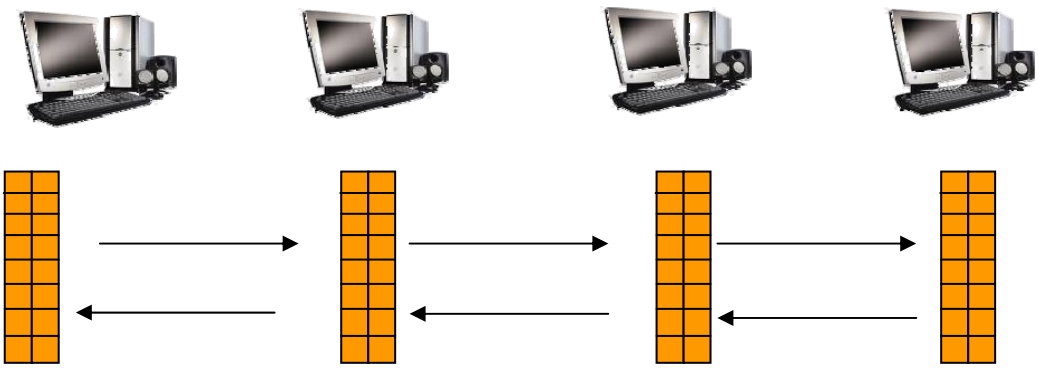


Figura 4.3 Comportamiento de aplicación heat2d

Una vez obtenido los resultados, los mismos son devueltos a un nodo central que se comporta como master, en realidad ese nodo no ejecuta operación. Para realizar el presente estudio, se tuvo que realizar una modificación en el patrón de comunicaciones, el objetivo es ésta modificación es analizar el comportamiento de una aplicación con un patrón de cuatro comunicaciones solapadas por el cómputo, podría obtener mejores niveles de prestaciones en la aplicación. Mantener un patrón de comunicaciones como el mostrado en la figura 4.4

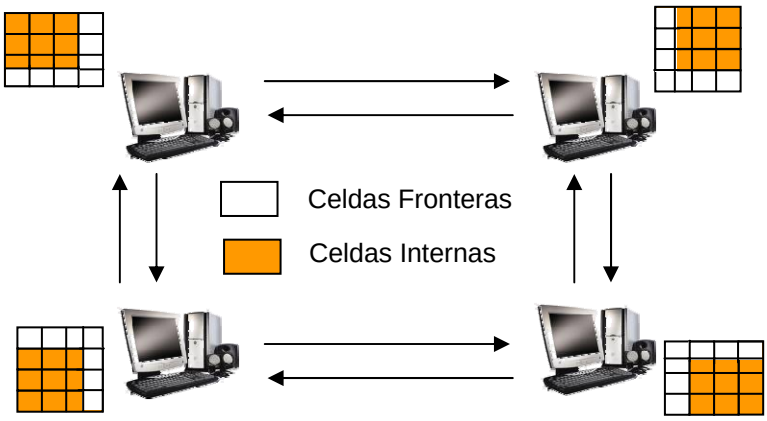


Figura 4.4 Cambio en el patrón de Comunicaciones.

En el caso presentado no se generan modificaciones al cómputo, sino al nivel de comunicaciones por lo que los resultados en ambos casos son iguales.

4.3 Entorno Experimental a nivel de sistema

La experimentación fue esquematizada, para la ejecución de la aplicación en un cluster de ocho nodos. Todos los nodos con características a nivel de arquitectura igual, por lo que se manejaba un entorno de cómputo homogéneo, la distribución lógica de los nodos, se diseñó con los comandos MPI_Cart_create(), MPI_Cart_coords(), y MPI_Cart_rank(). Estos comandos retornaban las coordenadas de cada uno de los nodos, lo que permitía una asignación del orden de comunicación de cada uno de los nodos.

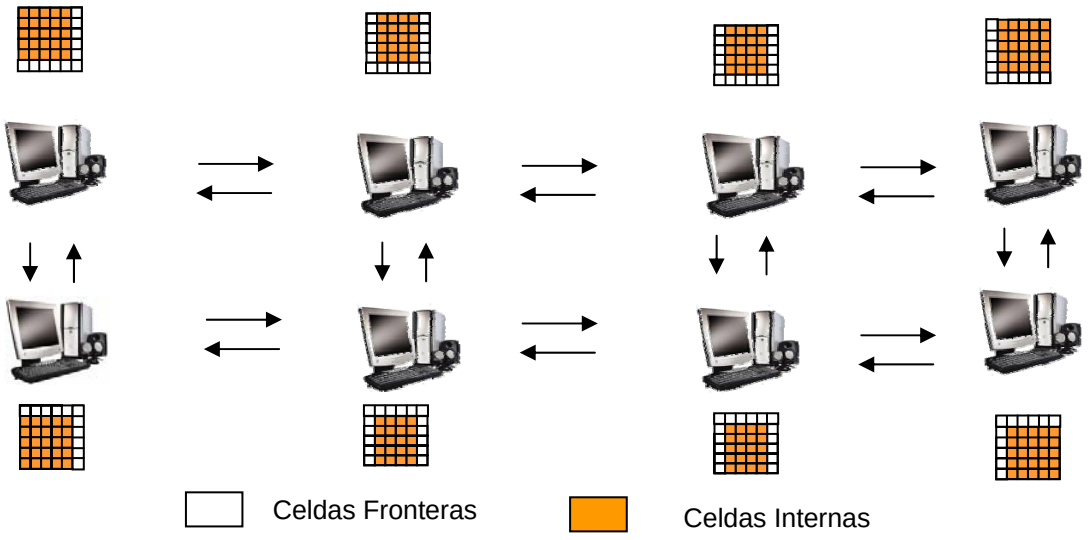


Figura 4.5 Distribución Lógica del Entorno de Trabajo

En la figura 4.5, se detalla el orden de la distribución de los nodos que se asignó a la ejecución de la aplicación, en el caso de este conjunto de nodos, solamente se tienen nodos con dos y tres comunicaciones con vecinos, la aplicación está diseñada para 4 comunicaciones para cada nodo, en el caso de no tener sino 2 o 3, se filtran las comunicaciones no enviando mensaje por el lado que no tenga comunicaciones, un ejemplo es el detallado en la figura 4.6

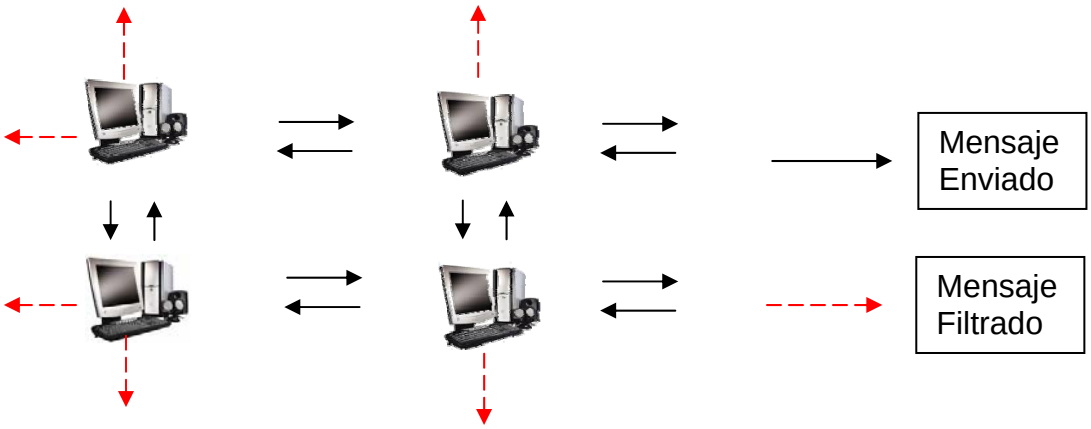


Figura 4.6 Filtrado de Envió de Mensaje

Los mensajes son filtrados por condicionales de envío. Esto permite que la aplicación pueda ser ejecutada en un número de nodos que tenga 2, 3 y 4 comunicaciones respectivamente.

La arquitectura de los nodos de ejecución esta conformada por 8 nodos, con procesadores Pentium IV de 3.0 Ghz, memoria 1 Gb y con una arquitectura de red conmutada de 1 Mbps, full duplex.

La ejecución de la aplicación se realizó 25 veces por cada experimento, tomando los valores promedios de las ejecuciones, y calculando las desviaciones que varían de acuerdo al tamaño de las celdas, y al conjunto de datos que se ejecuten de la aplicación. La herramienta de paso de mensajes MPI, utilizando mensajes asíncronos MPI_Isend() y MPI_Irecv(), y uso de pthreads para el solapamiento de cómputo y comunicación, todos los datos han sido adquiridos a un fichero y luego los valores cargados al programa Excel para el diseño de las gráficas para su posterior análisis de resultados.

4.4 Análisis de los Resultados.

Los resultados obtenidos están orientados a cubrir los objetivos inicialmente planteados en la investigación, se comienza la descripción y análisis de acuerdo a la necesidad para la ejecución, en éste orden de ideas, se realizó la descripción del modelo de mapping, seguidamente por el proceso de scheduling y se evalúan las prestaciones obtenidas

- **Modelo de Mapping**

El establecimiento de un método de mapping de asignación a cada nodo tomando en consideración el número de celdas y nodos descritos en el capitulo 3, se partió inicialmente con la configuración de una distribución lógica de los nodos, asignándole un identificador de coordenadas, lo anterior permite que la ejecución y las comunicaciones se realicen hacia el nodo correcto de la distribución.

La partición de las celdas, se realizó de forma homogénea lo que permite obtener trozos de celdas del mismo tamaño para cada nodo, para esto se obtuvo la mejor división de acuerdo al número de nodos, por lo que el primer paso fue obtener la mejor distribución lógica de los nodos, lo cual representa generar una estructura que contenga coordenadas de filas y columnas de identificación.

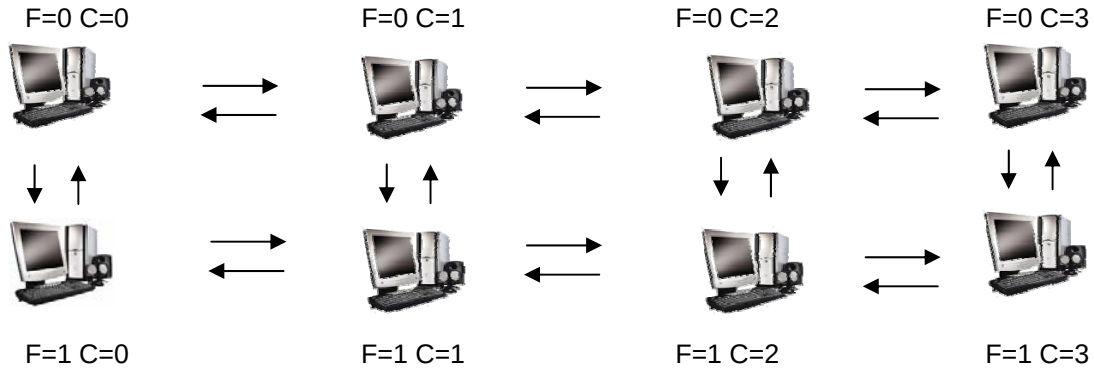


Figura 4.7 Distribución Lógica de los Nodos Experimentales

Al conocer el identificador de cada nodo, se puede comenzar a realizar la distribución; la distribución de las celdas fue homogénea por lo que se buscó repartir la misma cantidad de elementos a cada nodo como se mencionó en el capitulo 3. Una vez conseguida la distribución lógica se procedió a conseguir un

factor de diferencias entre los nodos, para poder hacer la separación de los resultados, la cantidad de celdas por columnas y por filas, y hacer una generación de trozos como la mostrada en la tabla 4.1, los valores obtenidos fueron generados en distribuciones desde 200x200 celdas, hasta 6000x6000

Tabla 4.1 Celdas Asignadas a cada Nodo luego del Mapping

X	Y	Nodo	NX	NY
200	200	8	50	100
300	300	8	114	225
400	400	8	100	200
500	500	8	125	250
1000	1000	8	250	500
1500	1500	8	375	750
2000	2000	8	500	1000
2500	2500	8	625	1250
3000	3000	8	750	1500
3500	3500	8	875	1750
4000	4000	8	1000	2000
5000	5000	8	1250	2500
6000	6000	8	1500	3000

Esta distribución permite que la ejecución tenga la misma cantidad de cómputo, la diferencia en cada nodo es la cantidad de comunicaciones a los vecinos y el cómputo de borde para cada uno, dependiendo de la distribución lógica que se asigne a cada nodo.

```
dim_size[0] = *(b);    dim_size[1] = *(b+1);
periods[0] = 1;    periods[1] = 1;
MPI_Cart_create(MPI_COMM_WORLD, 2, dim_size, periods, 0,
&communicator);
MPI_Cart_coords(communicator, rank, 2, coords);
/*Determinando los vecinos de acuerdo a las coordenadas*/
nc[0] = coords[0]-1;
if(nc[0]<0) nc[0]=coords[0];
nc[1] =coords[1];
MPI_Cart_rank(communicator, nc, &nb[0]);
nc[0] = coords[0]+1;
if(nc[0]==dim_size[0]) nc[0]=coords[0];
nc[1] = coords[1];
MPI_Cart_rank(communicator, nc, &nb[1]);
nc[0] = coords[0];
nc[1] = coords[1]+1;
if(nc[1]==dim_size[1]) nc[1]=coords[1];
MPI_Cart_rank(communicator, nc, &nb[2]);
```

Figura 4.8 Código de Distribución Lógica de los Nodos

Al obtener una distribución como la descrita en la tabla 4.1 para cada nodo, se debe establecer que los elementos de borde estarán sujetos a la cantidad asignada y a la posición del nodo, lo último por la distribución lógica, para este caso se cálculo con las ecuaciones (1) (5) (6) descritas en capitulo 3, obteniendo valores diferentes dependiendo de si el nodo tiene dos o tres comunicaciones.

- **Modelo de Scheduling**

Se estableció el orden de ejecución de las celdas de la aplicación SPMD en el cluster de cómputo (Scheduling), partiendo que el número de celdas a ser asignadas para cada nodo es mayor que uno, se consideró la forma de ejecución analizando el patrón de comunicaciones de las aplicaciones SPMD.

El método para ejecutar cada una de las celdas, se ha esquematizado de acuerdo al orden de prioridad que se le asignan en el conjunto de celdas; por ejemplo, las celdas con mayor prioridad de ejecución son aquellas que mantienen comunicación con dos vecinos, seguidas de las celdas con comunicaciones con un vecino, y por último se realiza la ejecución de las celdas internas, el motivo de este orden de prioridad son las latencias de los enlaces de comunicación, como se observó en el capítulo 3, la latencia de comunicación de la red es mayor que las generadas dentro del nodo, por ésta razón se determina este orden de prioridad de ejecución.

Tomando la ejecución de las celdas en un nodo y considerando las prioridades descritas en el capitulo 3, se deben ejecutar las celdas en el orden representado en la figura 4.9.

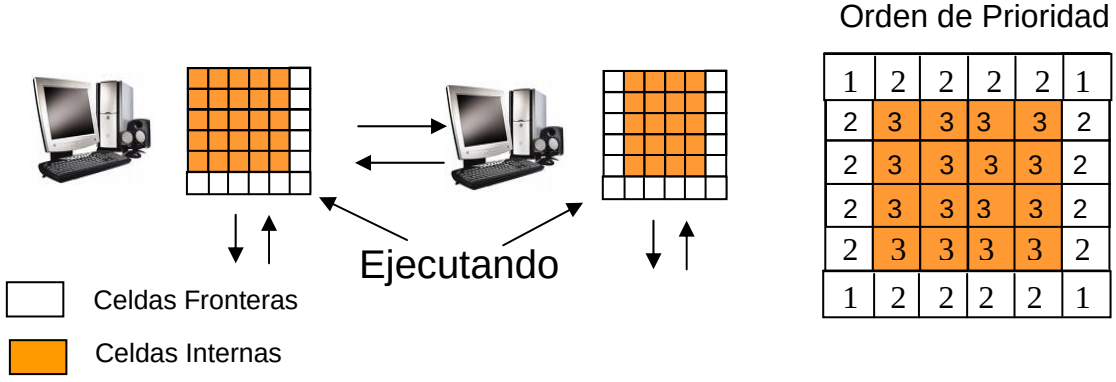


Figura 4.9 Planificación de las celdas del Problema

A partir de las prioridades se determina el orden de ejecución de cada celda de forma de permitir el solapamiento de cómputo de las celdas internas y la comunicación de las fronteras, es importante acotar, que el éxito del solapamiento depende de la ratio entre el cómputo interno y la comunicación del borde.

En la figura 4.9, se describe el comienzo de ejecución de las celdas de borde y luego comienza la comunicación de la información, se observa que existen nodos con dos y tres comunicaciones a los vecinos, lo que origina un aumento tanto en el cálculo del borde como en la comunicación; luego que se ejecuta el borde se comienza a realizar la ejecución de las celdas internas y la comunicación, esto se mantendrá solapado.

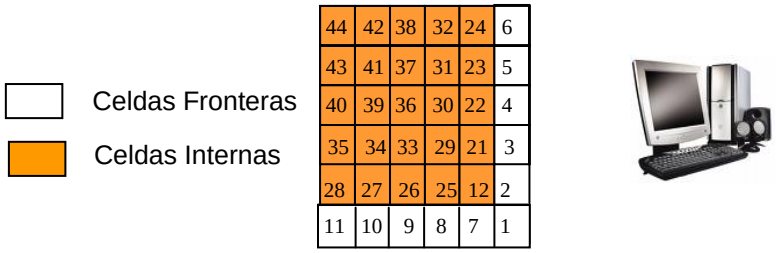


Figura 4.10 Ejecución de Celdas de acuerda a la prioridad de comunicación

En la figura 4.10, la numeración indica el orden de ejecución, es importante acotar, que no es la prioridad de ejecución, esta numeración es una vez establecidas las prioridades de la figura 4.9

El orden de ejecución de las celdas fronteras, permite que se mantenga un solapamiento del cómputo interno y de la comunicación, el planificar la ejecución de las celdas origina que no exista pérdida de tiempo, o que el procesador este ocioso por mala planificación de ejecución, lo anterior originado, por la información que tienen que ser transferida de los nodos o celdas vecinas. El retraso que puede generar las comunicaciones, ocasiona que la aplicación no tenga niveles de eficiencia aceptables y la velocidad de cómputo se ve afectada por la ejecución de la aplicación.

Al unir el mapping con el orden de ejecución se puede obtener resultado evaluando el comportamiento en un nodo y el conjunto total de la aplicación que serán descritos de acuerdo a los siguientes escenarios.

- Escenario 1:

La estructura del primer escenario, permite observar el comportamiento de las comunicaciones en dos nodos, el primero que tiene 2 comunicaciones y que esta identificado con las coordenadas F=0, C=0, del esquema experimental y el segundo de 3 comunicaciones identificado con F=0, C=1, aquí se analiza el comportamiento de las comunicaciones, tomando los tiempos de cada uno de los elementos a estudiar. El tamaño del problema es variable de 100 celdas por 100 celdas en cada eje, obteniendo los valores promedios de acuerdo al volumen de comunicación que tiene la aplicación en cada uno de los casos.

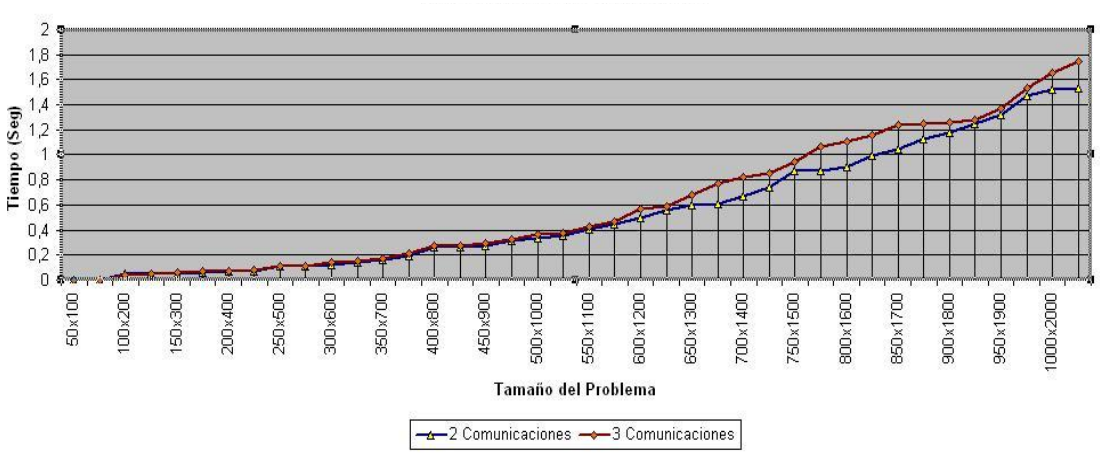


Figura 4.11 Comportamiento con Nodos de 2 y 3 comunicaciones

La figura 4.11, muestra la influencia en los tiempos de comunicación de los nodos con mayor cantidad de comunicación, para cada caso se utilizó una ejecución en ocho nodos, al observar los resultados obtenidos, se visualiza que el nodo con tres comunicaciones con sus vecinos, mantiene un nivel más alto de comunicaciones, a diferencia de los nodos con dos vecinos. Por lo tanto el nodo con 3 comunicaciones es el que dirige la política de solapamiento dentro del entorno. Cada valor obtenido se refleja de acuerdo al número de celdas que contiene el nodo, en el caso de eje X, se describe el tamaño del problema y en el eje y el tiempo que tarda en enviar el mensaje.

Estudiando este mismo escenario, se incluyo el cómputo del nodo con 3 comunicaciones obteniendo los valores de la gráfica 4.12.

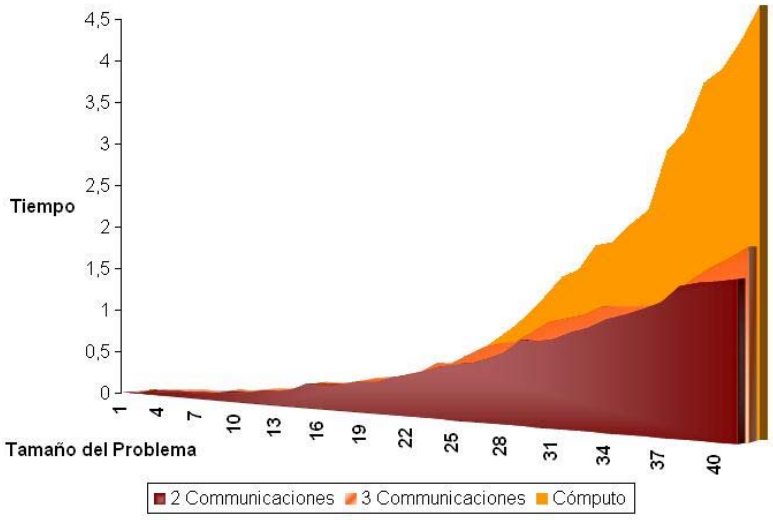


Figura 4.12 Relación Cómputo y Comunicación

Los resultados obtenidos, muestran el punto de solapamiento entre las comunicaciones y el cómputo de las celdas internas, encontrando determinar el punto que puede visualizar el punto ideal de solapamiento, el cual es establecido por la visualización del cómputo con respeto a los valores de 3 comunicaciones. Este podría considerar con un valor de solapamiento.

Asimismo, la Figura 4.12 detalla, que los nodos con tres comunicaciones mantienen valores en tiempo mayores a los obtenidos por los nodos de dos comunicaciones, por lo que se debe tomar en consideración para el solapamiento, y validar la ecuación que mantiene el cómputo por encima de los patrones de comunicaciones ecuación (12) del capitulo 3.

Al visualizar este primer escenario, se puede describir la influencia que tiene el número variable de comunicaciones en el entorno.

- Escenario 2:

En el segundo escenario se mide los problemas presentado de la mala selección del conjunto de datos a computar con el patrón, en el siguiente caso se toman los valores de acuerdo al tamaño del conjunto de celdas y son mostrados en la tabla 4.2, que contiene los valores promedios de ejecución.

Tabla 4.2 Resultados Promedios de Cómputo y Comunicación

Dist_Nodo	NX	NY	Tamaño a Comunicar MB	Tiempo 2 Comm.	Tiempo 3 Comm	Cómputo
8	50	100	0,038808	0,00079	0,00082235	0,00070833
8	114	225	0,088208	0,002475	0,002131	0,00087563
8	100	200	0,157608	0,045937	0,040635	0,00093532
8	125	250	0,247008	0,046928	0,047024	0,00141633
8	150	300	0,356408	0,053474	0,057474	0,00206933
8	175	350	0,485808	0,061995	0,065995	0,002657
8	200	400	0,635208	0,068011	0,072731	0,00360033
8	225	450	0,804608	0,075603	0,079022	0,00427067
8	250	500	0,994008	0,109681	0,111809	0,00571533
8	275	550	1,203408	0,108563	0,111365	0,00642733
8	300	600	1,432808	0,125492	0,138851	0,00841967
8	325	650	1,682208	0,142122	0,153457	0,00936167
8	350	700	1,951608	0,161116	0,173769	0,011133
8	375	750	2,241008	0,193083	0,211413	0,012539
8	400	800	2,550408	0,26326	0,269884	0,01306933
8	425	850	2,879808	0,268351	0,2718351	0,07115
8	450	900	3,229208	0,276648	0,292939	0,076376
8	475	950	3,598608	0,316061	0,329292	0,02157933
8	500	1000	3,988008	0,332978	0,363173	0,060361
8	525	1050	4,397408	0,356495	0,373425	0,14520233
8	550	1100	4,826808	0,407748	0,425585	0,24014867
8	575	1150	5,276208	0,446786	0,467172	0,34297133
8	600	1200	5,745608	0,495293	0,564549	0,45543567
8	625	1250	6,235008	0,554546	0,584546	0,641698
8	650	1300	6,744408	0,5953623	0,677534	0,71381233
8	675	1350	7,273808	0,612478	0,769766	0,84009333
8	700	1400	7,823208	0,667969	0,81726	0,979379
8	725	1450	8,392608	0,739767	0,851605	1,12558833
8	750	1500	8,982008	0,875086	0,948135	1,30961667
8	775	1550	9,591408	0,873318	1,067381	1,51718467
8	800	1600	10,220808	0,907218	1,107497	1,604405
8	825	1650	10,870208	0,992337	1,152644	1,843934
8	850	1700	11,539608	1,044135	1,2415862	1,883352
8	875	1750	12,229008	1,131624	1,244569	2,081617
8	900	1800	12,938408	1,181449	1,2552321	2,21733973
8	925	1850	13,667808	1,244282	1,269748	2,806176
8	950	1900	14,417208	1,317483	1,374916	2,99730233
8	975	1950	15,186608	1,476885	1,530859	3,45991767
8	1000	2000	15,976008	1,521094	1,656253	3,59160933
8	1025	2050	16,785408	1,530607	1,746332	3,87459657
8	1050	2100	17,614808	1,5675632	1,865623	4,186323

Tomando los elementos de la tabla anterior y graficando la columna de 3 comunicaciones con la de cómputo, se obtiene lo representado en la figura 4.13,

que muestra el comportamiento cómputo y comunicación con el nodo de 3 comunicaciones.

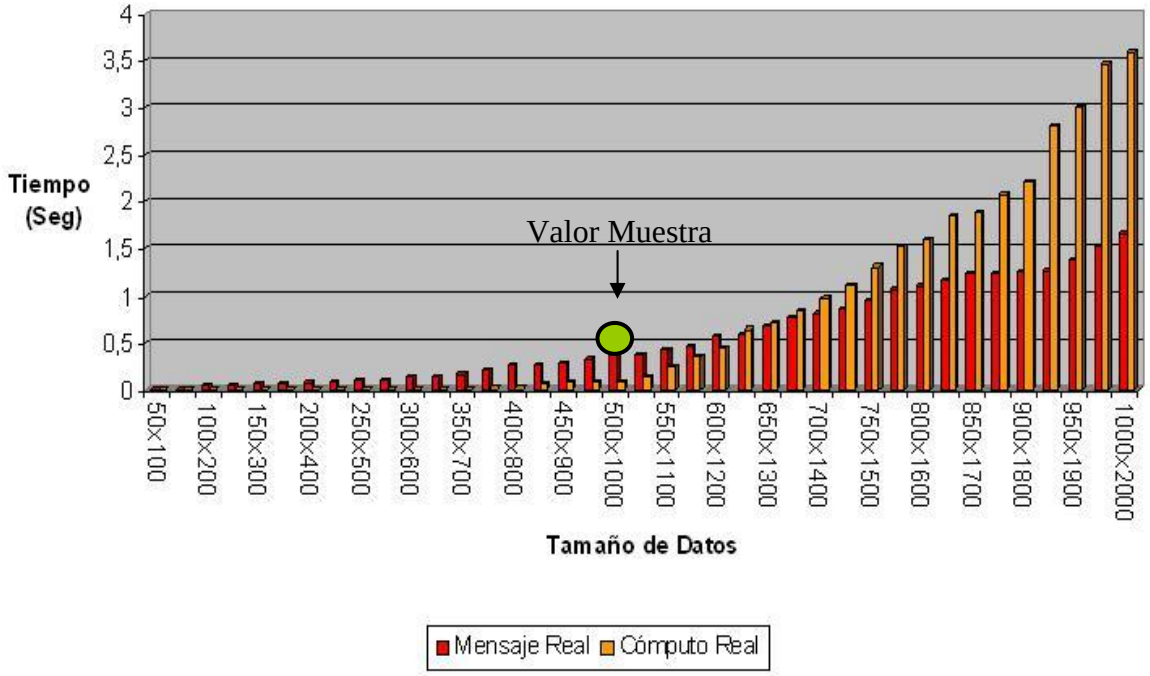


Figura 4.13 Nodo de 3 comunicaciones: Relación Cómputo y Comunicación

En la figura se detallan los diferentes tamaños asignados al nodo y los tiempos en comunicar y ejecutar de acuerdo al volumen de comunicación.

Si se selecciona un punto de referencia, como el descrito en la figura anterior, se puede presentar el caso, que el tiempo de comunicaciones sea mayor al tiempo de cómputo del conjunto de celdas, esto motivado a los tiempos que puede tardar la arquitectura de paso de mensajes y la red en transmitir, en éste caso particular las comunicaciones se desbordan, lo que origina que en cada iteración de la aplicación se retenga, por la espera de la información enviada por los nodos vecinos, este caso se comentó en el capítulo anterior denominado Communications bound. Es importante acotar que todos los cálculos han sido determinados con el valor medio de 25 ejecuciones tanto a nivel de cómputo como con las comunicaciones. El valor adquirido por la muestra refleja con una traza de ejecución, cómo las comunicaciones desbordan a la ejecución de la aplicación, la Figura 4.14 muestra la aplicación en 8 nodos.

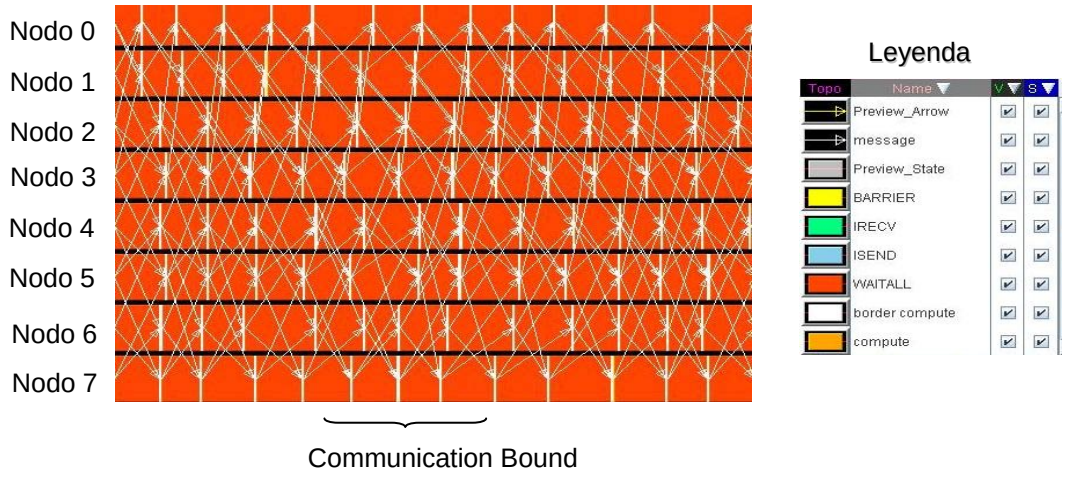


Figura 4.14 Trazas de Ejecución Communicaction Bound.

Por otra parte, al tomar una selección adecuada de la relación cómputo y comunicación, se pueden obtener mejores resultados de solapamiento, si se determina el punto de intersección de las barras de cómputo y comunicación (Figura 4.15), se puede luego detallar las trazas de ejecución, observando un mejor solapamiento entre el cómputo y la comunicación.

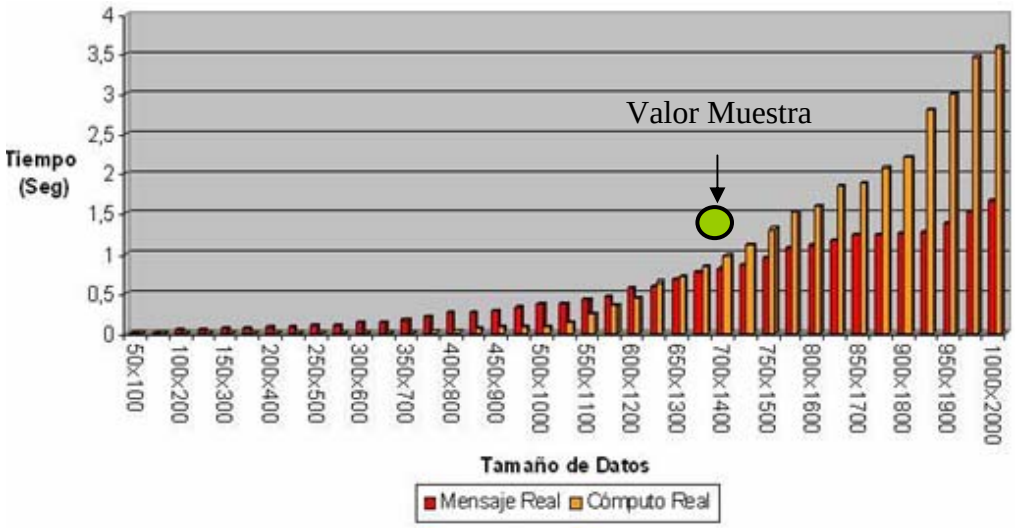


Figura 4.15 Nodo de 3 comunicaciones: Relación Cómputo y Comunicación

Al obtener este punto de muestra, se puede conseguir una relación de cómputo y comunicación, con mayor grado de solapamiento, las trazas de ejecución de la figura 4.16 muestran el solapamiento obtenido en ese punto seleccionado de la figura 4.15.

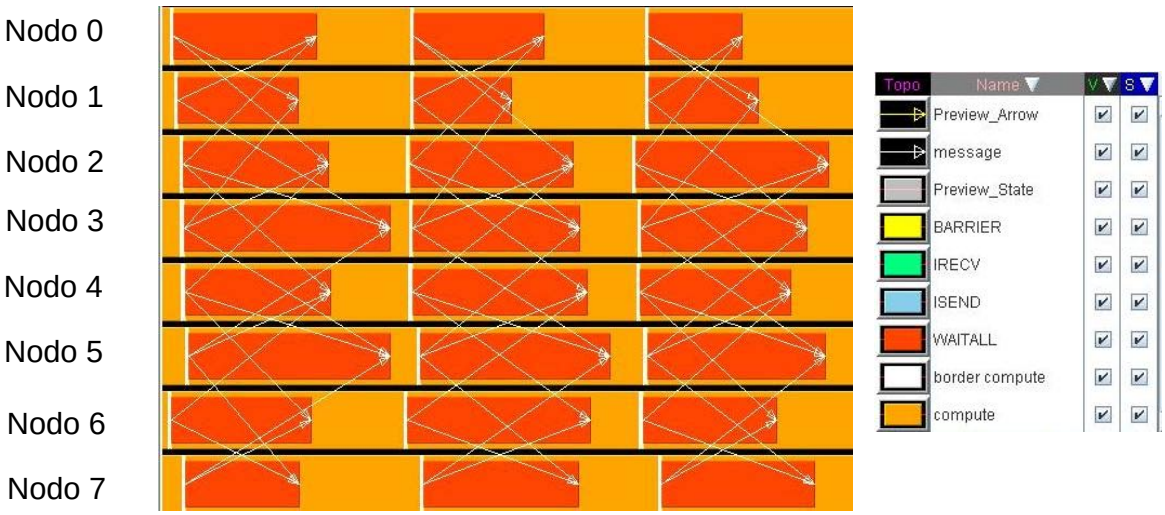


Figura 4.16 Trazas de Ejecución con Solapamiento

Los valores obtenidos, detallan una diferencia en cada nodo, motivado a que la partición de las celdas es de forma homogénea, por lo que los nodos con mayor cantidad de comunicaciones (más vecinos), tardan más tiempo en enviar los mensajes.

Si se tiene un entorno homogéneo de cómputo, se puede seleccionar el punto de solapamiento entre el cómputo y la comunicación, estudiando las variables que integran la ecuación de solapamiento descrita en el capítulo 3 de la presente investigación, donde la influencia está orienta al cómputo de cada una de las celdas y el tiempo en enviar los mensajes por paso de mensaje; éste último punto, va determinado con la latencia de la red o enlace de comunicaciones, por el número de comunicaciones y el tamaño de elementos que se desea enviar a los nodos.

En los problemas SPMD, para que se mantengan altos niveles de eficiencia, se deben controlar el sistema de comunicaciones con las celdas vecinas, que en algunos casos utilizan paso de mensaje, memoria común, o enlaces de comunicaciones de red, por lo que es importante hacer una subdivisión del conjunto de celdas que cada nodo procesa; el objetivo es mantener un dominio de descomposición de la aplicación (Foster, 1994), por lo que se asignará un conjunto de celdas a cada nodo para que se ejecuten. Si se mantiene un patrón de comunicación con 4 vecinos, se debe considerar que la cantidad de

comunicaciones por algunos nodos serán mayores que las generadas por otros, por lo que inicialmente se realizó una subdivisión homogénea para probar la idea inicial, como lo es controlar las comunicaciones solapando cómputo y comunicación, mejorando así los niveles de eficiencia y de Speedup.

- Escenario 3.

En este escenario se calculan los tiempos de ejecución tomando en consideración la aplicación de transferencia de calor, sin ninguna modificación y la cual estará identificada como (A. original) en el desarrollo de las figuras, por otra parte se nombrara (A. modelada) a la aplicación que mantiene el patrón de solapamiento de cómputo y comunicación.

La figura 4.17 muestra la ganancia de aplicar, al aplicar un modelo de mapping y scheduling para la ejecución de la aplicación, descritos anteriormente.

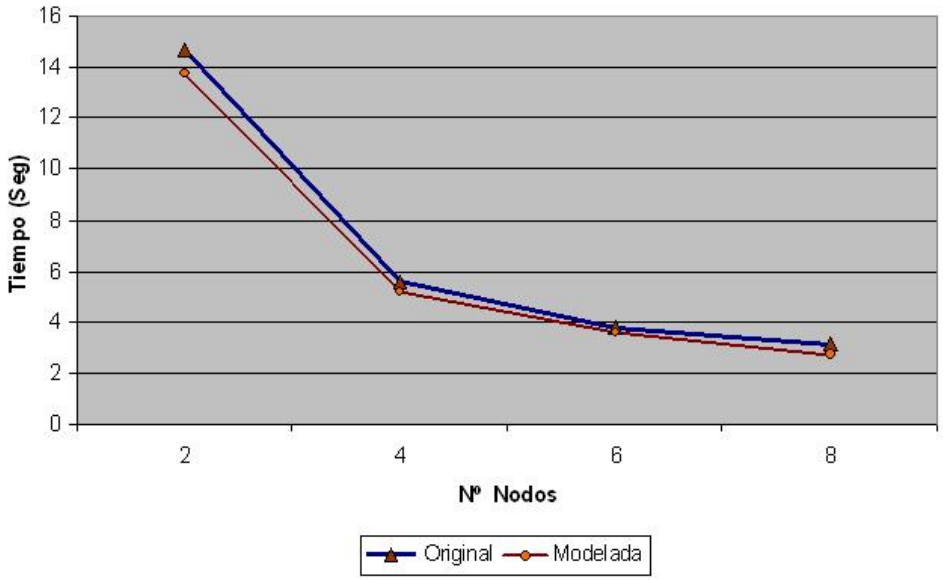


Figura 4.17 Tiempo de Ejecución, Tamaño 2000x2000 celdas

En el caso de la figura 4.17, se puede visualizar la ganancia de la aplicación utilizando el modelo presentado, contra la ejecución el modelo original de la aplicación de transferencia de calor, aunque ambas líneas tienen poca separación, el volumen de ganancia a nivel de los tiempos de ejecución se

comienza a observar a medida que el tamaño de celdas aumenta, el motivo es por el cómputo del conjunto de celdas.

En la figura 4.18, se aumenta el tamaño del problema y se puede destacar la disminución de los tiempos de ejecución de la aplicación.

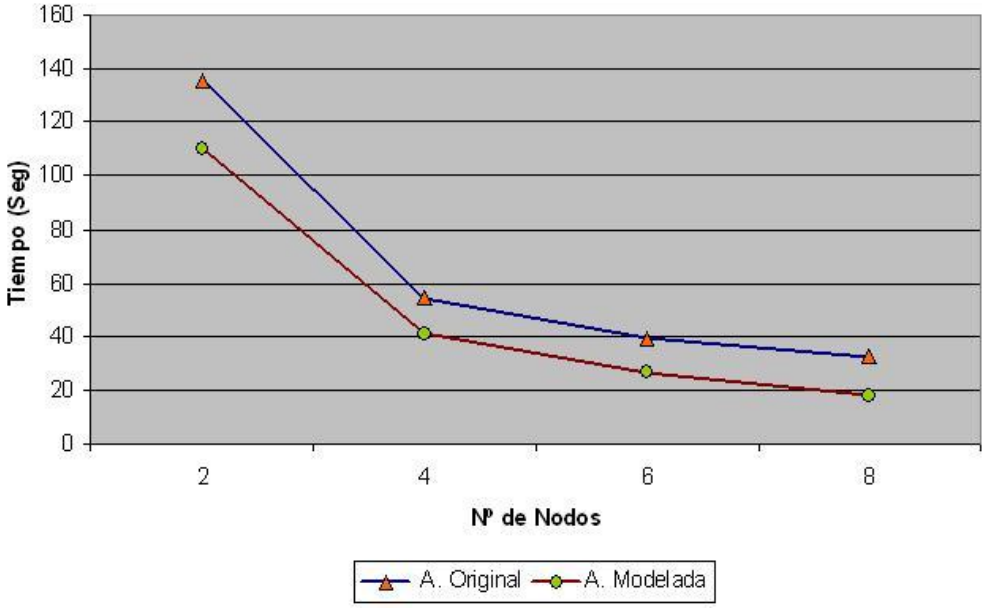


Figura 4.18 Tiempo de Ejecución, Tamaño 6000x6000 celdas

Al analizar la figura 4.18, se observa el aumento de la ganancia del tiempo de ejecución al solapar el cómputo con la comunicación, caso contrario a lo presentado en la figura 4.17 donde el nivel de ganancia es inferior. Los problemas que incluyen mayor cantidad de celdas permiten tener alto volumen de cómputo interno, por lo que las comunicaciones están totalmente solapada.

Con ésta gráfica (figura 4.18) se describe que el comportamiento a medida que aumenta el tamaño y es comparado con la aplicación original de transferencia de calor, se obtienen niveles de ganancia con los tiempos de ejecución. La prueba se realizó en un cluster con ocho nodos, por lo que sería importante visualizar en un cluster más grande para detallar el punto de escalabilidad del modelo.

A la hora de medir los valores de los tiempos de ejecución y el Speedup, se hacen con el objetivo de visualizar el comportamiento de la eficiencia en el entorno, al mantener altos niveles de eficiencia en un cluster, se puede comenzar

a extrapolar el modelo a un entorno con mayor comunicaciones como el *Multicluster*.

Al obtener esos tiempos de ejecución se procede a estudiar la aceleración de forma de presentar una mejora del Speedup de la aplicación manteniendo un nivel de eficiencia; una vez ejecutada la aplicación en un entorno conformado con ocho nodos de cómputo, y ejecutando la aplicación con un número de 25 veces con 50 iteraciones por ejecución, se obtienen valores medios de los tiempos de ejecución.

Realizando una ejecución serie de la aplicación, y comparando con la aplicación original de la transferencia de calor y el patrón obtenido del solapamiento, se pueden obtener ganancias a nivel de la velocidad computacional de la aplicación, empleando el patrón para aplicaciones SPMD solapadas. Los valores obtenidos del Speedup, son mostrados en dos figuras (4.13 y 4.14) cada una con un tamaño de problema distinto (número de celdas totales).

En el caso de la figura 4.19 se muestran los tiempos obtenidos para una distribución de 2000 x 2000 celdas en varias distribuciones de nodos, encontrando los siguientes resultados.

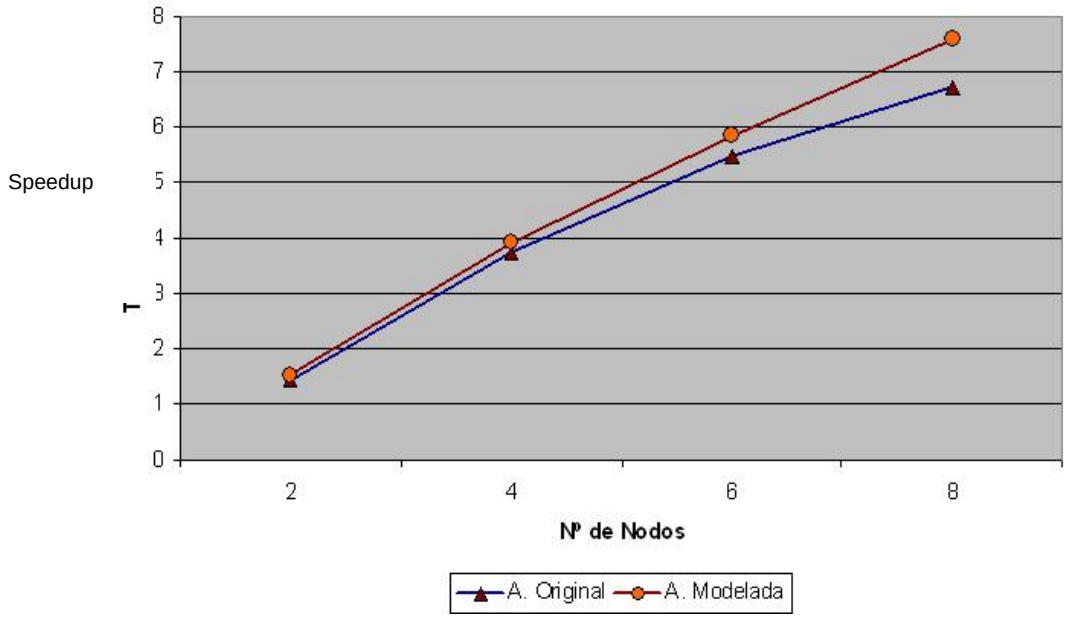


Figura 4.19 Speedup 2000x2000 Celdas

Al conseguir los resultados mostrados, se puede visualizar la ganancia a nivel de velocidad computacional que se obtiene al aplicar la ejecución primero en las celdas de borde, y luego ejecutar las celdas internas mientras se comunica, en este caso la ganancia observada es menor a la que se puede detallar cuando se aumenta el problema como en el caso siguiente (Figura 4.20).

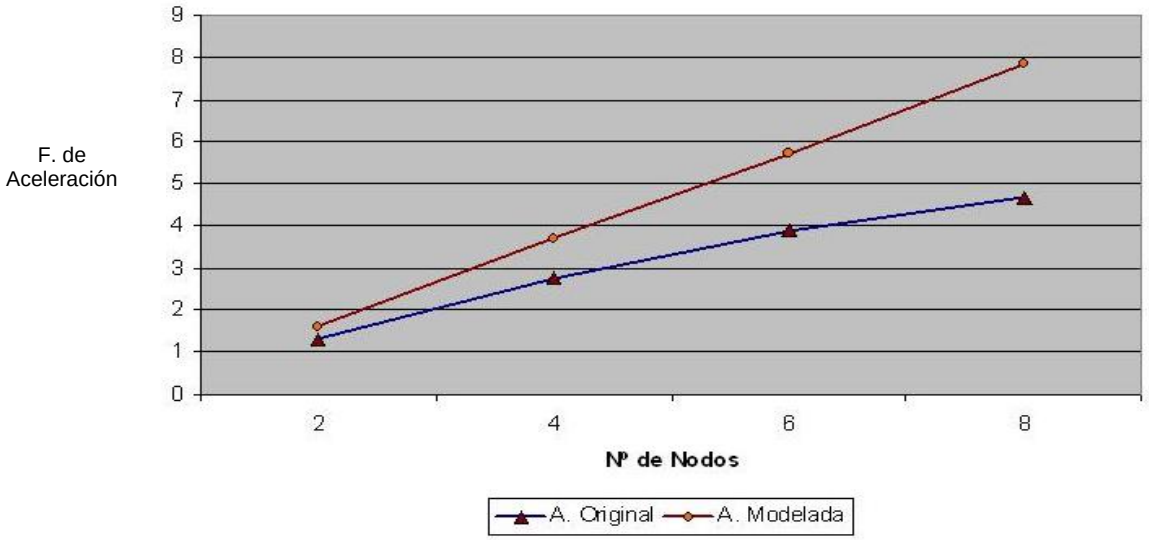


Figura 4.20 Speedup 6000x6000 Celdas

Al aumentar el número de celdas se puede observar una mejor ganancia de la velocidad computacional, los factores iniciales, son el solapamiento de las comunicaciones, y el orden de ejecución de la aplicación.

Finalmente, se detallan las ganancias en la eficiencia del uso de los recursos, es presentada por la figura 4.21, en el caso de usar 6000X6000 celdas, el comportamiento de utilización de los diferentes nodos es incremental, motivado al solapamiento en la aplicación modelada se obtienen valores de eficiencias por encima del 80%, lo que determina que mientras está computando se están solapando los mensajes, esto origina que los niveles de eficiencia aumenten, y permite que los tiempos de ejecución de la aplicación disminuyan, obteniendo mejores prestaciones en la ejecución de la aplicación.

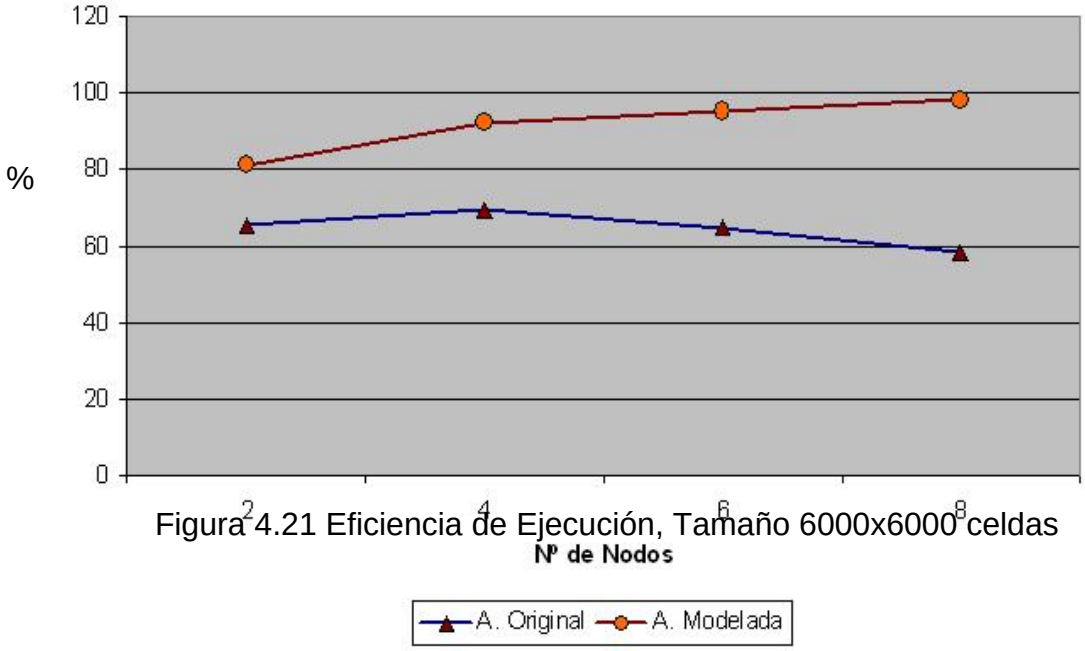


Figura 4.21 Eficiencia de Ejecución, Tamaño 6000x6000 celdas

Por otra parte, se visualiza que la curva de eficiencia disminuye en la aplicación original de transferencia de calor, lo que permite observar que a medida que la aplicación aumenta el tamaño de las celdas disminuye la eficiencia en el patrón original, motivado a las comunicaciones, caso contrario cuando se modela con el patrón propuesto de solapamiento donde los niveles de eficiencia se mantienen en un valor que en el peor de los casos puede establecer un rango por encima del 80%, para un cluster con 8 nodos.

Al culminar la ejecución y los experimentos, se puede establecer que en un entorno homogéneo, se pueden controlar las comunicaciones, sin embargo, a la hora de utilizar un entorno heterogéneo o un *Multicluster*, se deben parametrizar los volúmenes de comunicaciones tanto InterCluster como internodos, por lo que no solamente se deben controlar no solo las comunicaciones que se han solapado en la presente memoria. No obstante sirve como base fundamental para comenzar a determinar que la ejecución de las aplicaciones SPMD, se pueden realizar de manera eficiente, y que se pueden pasar a otro nivel debido a que se pueden controlar las comunicaciones.

Capitulo 5. Conclusiones y Líneas Abiertas.

A continuación, se describen las conclusiones alcanzadas con el desarrollo de ésta investigación, asimismo se plantean las líneas abiertas que sirven de base para los trabajos futuros de investigación, relacionados con las aplicaciones SPMD en ambientes de *Multiclusters*.

5.1 Conclusiones.

Las aplicaciones SPMD son usadas en entornos Grid o ambientes *Multicluster*, debido a que es un tipo de aplicación muy fácil de paralelizar, debido a que cada nodo es asignado un único programa y con diferente distribución de celdas de datos. El tema de aplicaciones de predicciones meteorológicas cada día es de más importancia y este tipo de aplicaciones son diseñadas bajo el enfoque de intercambio de información de vecinos, similar a los esquemas de programación SPMD. Los factores de precisión de éste tipo de aplicaciones han originado que la cantidad de cómputo de las aplicaciones aumente y por ende se tengan que buscar alternativas para que los cálculos sean dados en menores tiempos.

Es por esto que la ejecución de las aplicaciones SPMD, sigue una tendencia de ser ejecutadas en ambientes de cómputo con mayor performance, dentro de los cuales tenemos los ambientes *Multicluster*. Sin embargo, este tipo de ambientes tienen altas diferencias en las latencias de los enlaces, lo que origina que las comunicaciones de las aplicaciones SPMD tengan que controlar éste factor.

El presente trabajo se ha enfocado en generar una política de scheduling y mapping que permita controlar el volumen de comunicaciones tanto a nivel del cluster, como a nivel del nodo de cómputo, lo anterior buscando una relación de cómputo y comunicación que se pueden solapar. El solapamiento permite que los tiempos de comunicaciones no interfieran en la ejecución de la aplicación.

Lo importante de una ejecución eficiente de un SPMD, es conocer el cómputo de cada celda y el volumen de comunicaciones que tiene a través de los enlaces,

una vez obtenidos esos valores se puede establecer una relación de la ratio de cómputo y comunicación que permite tener solapadas las comunicaciones con el cómputo interno.

Al mantener las reglas de gestión de comunicaciones, se origina que la aplicación pueda escalar mejor, generando que la misma tenga una mejor velocidad computacional y se pueden definir unos valores o umbrales de eficiencia. Esto indica que se pueden gestionar las comunicaciones por lo que se podrían solapar los resultados a un ambiente *Multicluster*, siempre y cuando el volumen de comunicaciones sea controlado y que la aplicación tenga gran cantidad de cómputo.

Al mapear y hacer el orden de planificación de celdas por prioridad de ejecución, permite que los tiempos de ejecución se pueden disminuir, esto motivado al solapamiento que se puede esquematizar con la planificación planteada.

Los objetivos planteados en la fase de la investigación, ha sido cubiertos a nivel de un cluster de cómputo homogéneo, no obstante la experimentación muestra resultados preliminares que apuntan que los tiempos de comunicaciones pueden ser gestionadas con un esquema de solapamiento, sin embargo hay que profundizar el estudio para controlar las comunicaciones de los enlaces de Internet que son muy variables y que tienen alta latencias, por lo que se generan nuevos elementos a estudiar si se desea ejecutar aplicaciones SPMD en ambientes *Multicluster*.

De los estudios experimentales realizados, se han podido extraer las siguientes conclusiones y que aportan descripciones por cada uno de los objetivos planteados para la fase planteada de la investigación.

- Al estudiar el comportamiento de las aplicaciones SPMD en un cluster, se ha mostrado que una variable para el ajuste entre el cómputo y la comunicación depende de la posición del nodo dentro de la distribución lógica que se le asigne, esto por el volumen de comunicaciones que posee y la cantidad de cómputo frontera que ejecuta.

- Al subdividir las celdas en forma homogénea, se realizó una prueba que demuestra que se puede aplicar un entorno de solapamiento entre el cómputo y la comunicación. Sin embargo, observando las gráficas de ejecución de la aplicación, se puede describir que existe influencia en la cantidad de cómputo de cada nodo, por lo que se deben generar asignaciones de celdas de acuerdo a la capacidad de cómputo de cada uno de los nodos, originando heterogeneidad de cómputo.

- Con relación a la manera de hacer el scheduling de celdas, se puede concluir que dando prioridades de ejecución de cada celda de acuerdo a las prioridades descritas en el texto, se pueden obtener niveles de ganancia a niveles de performance de ejecución de la aplicación, lo que conlleva a una mejor ejecución de la aplicación en el entorno distribuido.

- Al obtener un buena ejecución del modelo de aplicación, se obtiene que las prestaciones de las mismas se reflejen con resultados, donde la eficiencia es por encima del 85% y el Speedup mantiene una curva que permite demostrar que se eleva la escalabilidad de la aplicación, en el caso del SPMD, depende de la estructura de la misma, donde exista alto volumen de cómputo para que el mismo pueda ser asignado a otros nodos. Los tiempos de ejecución al tomar un punto de solapamiento, se pueden adquirir resultado por debajo de los de la aplicación de transferencia de calor, sin ninguna modificación.

Los objetivos que fueron planteados han sido cubiertos por la presente investigación, sin embargo deja nuevos desafíos que a continuación se mencionan. Las posibles líneas abiertas de investigación que pueden continuarse desarrollando sobre este tema se describen a continuación.

5.2 Líneas Abierta

Apuntar a la ejecución de aplicaciones SPMD en ambientes *Multicluster* de manera eficiente genera un conjunto de desafíos que pueden desarrollarse en la

aplicación de nuevas áreas que permitan demostrar la ejecución de la aplicación en un entorno totalmente heterogéneo.

Dentro de los primeros pasos, es empezar a estudiar el comportamiento de un ambientes heterogéneo, por lo que se debe hacer una asignación de celdas de forma heterogénea, ésta distribución afinara el mapping de las celdas, por lo que se debe asignar de acuerdo a la capacidad de cómputo que cada nodo puede tener. Esto originara que la distribución de la carga de trabajo sea dependiente de las comunicaciones y del cómputo de cada nodo.

Asimismo, se debe comprobar un entorno de ejecución con un *Multicluster*, donde se puedan controlar las comunicaciones, en el caso se tendrán tres tipos, Intracluster, Internodo, Intercluster, por lo que las variaciones se deben controlar bajo los tres aspectos, de acuerdo a las comunicaciones para poder solapar el cómputo con la comunicación.

Por otra parte, se debe montar una estructura que modele el comportamiento de las aplicaciones SPMD, para establecer el conjunto de variables con el objetivos de diseñar un modelo que permita describir el entorno global de comunicaciones. Lo anterior, permite que se desarrolle un modelo de ejecución que permita mantener niveles de eficiencia sobre un umbral del 80%, elevando la velocidad computacional de la ejecución de la aplicación.

Referencias

Argollo E (2006), “Performance Prediction and Tunning in Multicluster Enviroment” Tesis Doctoral, Universidad Autónoma de Barcelona.

Aumage O (2002), Heterogeneous Multi-Cluster Networking with the Madeleine III Communication Library, ipdps, p. 0085b, International Parallel and Distributed Processing Symposium: IPDPS 2002 Workshops

Bal H, Plaat A, y Bakker M. (1998), Optimizing Parallel Applications for wide-area Clusters. Proceeding of the 1st Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing. (IPPS/SPDP) Pág. 784-790.

Barreto, M., Ávila, R. B., and Navaux, P. O. (2000). The MultiCluster Model to the Integrated Use of Multiple Workstation Clusters. In Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing (May 01 - 05, 2000). J. D. Rolim, Ed. Lecture Notes In Computer Science, vol. 1800. Springer-Verlag, London, 71-80.

Buyya R. (1999) “High Performance Cluster Computing”, Pretince Hall NJ, USA, 1999. Pág 12.

Castro M. (2007), Programación con Listas de Datos para Computo Paralelo en Clusters, Tesis Doctoral, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional.

Culler, D. E., Gupta, A., and Singh, J. P., (1999) Parallel Computer Architecture: a Hardware/Software Approach. 1st. Morgan Kaufmann Publishers Inc.

Cruz C , (2005) Estrategias Coordinadas Paralelas Basadas en Soft Computing para la solución de Problemas de Optimización, Universidad de Granada, Tesis Doctoral, Pág. 18.

Cremonesi, P. and Gennaro, C, (2002). Integrated Performance Models for SPMD Applications and MIMD Architectures. IEEE Trans. Parallel Distrib. Syst. 13, 12 (Dec. 2002), 1320-1332

Flynn, M. J. (2000). Very high-speed computing systems. In Readings in Computer Architecture, M. D. Hill, N. P. Jouppi, and G. S. Sohi, Eds. Morgan Kaufmann Publishers, San Francisco, CA, 519-527.

Foster I, (1994), Designing and Building Parallel Programs, Addison Wesley Publishing Company.

Lee A, Eric Coe, B. Scott Michel, James Stepanek, Ignacio Solis, J. Matt Clark, Brooks Davis (2003), "Using Topology-Aware Communication Services in Grid Environments," *ccgrid*, p. 534, Third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03).

Leopold, C. (2001) Parallel and Distributed Computing: a Survey of Models, Paradigms and Approaches. John Wiley & Sons, Inc.

Moreno L (2005), Computación Paralela y Entornos Heterogéneos, Tesis Doctoral Universidad de la Laguna, Pág. 40.

Oliveira A, Argolo G, Iglesias P Martins S y Plastino A, (2005). Evaluating a Scientific SPMD Application on a Computational Grid with Different Load Balancing Techniques, F.F. Ramos et al. (Eds.): ISSADS, LNCS 3563, pp. 301–311

Plaat, A., Bal, H. E., and Hofman, R. F. (2001) Sensitivity of parallel applications to large differences in bandwidth and latency in two-layer interconnects. Future Gener. Comput. Syst. 17, 6 (Apr. 2001), 769-782

Silva L. and Buyya R (1999), “Parallel Programming Models and Paradigms, High Performance Cluster Computing: Programming and Applications”, Rajkumar Buyya (editor) pag 10.

Weissman J. Grimshaw A. (1995) A framework for partitioning parallel computations in heterogeneous environments, Concurrency: Practice and Experience Volume 7, Issue 5 , Pages455 – 478

Weissman J, (2001), Prophet: Automated Scheduling of SPMD Programs in Workstation Network, Concurrency: Practice and Experience, Volume 11, Issue 6 , Pages301 – 321

Wilkinson B. y Allien M. , (1999) Parallel Programing,. Pretince Hall