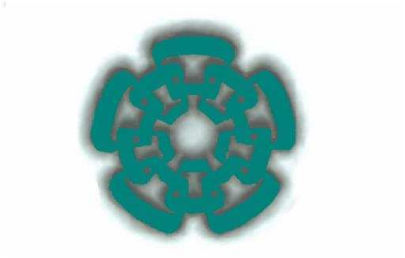


Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional

Método de campo de fuerza para la asignación de tareas en paralelo en mallas de computadoras de escritorio

Tesis que presenta:

Ed Scott Wilson García

Para obtener el grado de Doctor en Ciencias
En la especialidad de Ingeniería Eléctrica
(opción Computación)

Director de Tesis:
Dr. Guillermo Morales Luna

Trabajo realizado con el apoyo del *Programa de Becas para Estudio de Posgrado* del Instituto Mexicano del Petróleo.

Mexico, D.F. Agosto 2006

Método de campo de fuerza para la asignación de tareas en paralelo en mallas de computadoras de escritorio

Candidato: Edscott Wilson García
Supervisor: Guillermo Morales-Luna

Deseo agradecer al Dr. Tianfield Huaglory de la Glasgow Caledonian University por el apoyo y los valiosos comentarios recibidos para enriquecer la descripción del simulador de mallas dinámicas desarrollado en esta tesis. Así mismo, agradezco la valiosa ayuda del Dr. Hai Zhuge —*Chief Scientist of the China National Semantic Grid Research Project*— de la Academia China de Ciencias por el apoyo recibido en la promoción de este trabajo en los ámbitos internacionales relacionados. Por último agradezco al Dr. Díaz Pérez por los comentarios críticos que señalaron los últimos detalles que faltaban clarificar en la redacción del documento. Pero en especial deseo agradecer al Dr. Guillermo Morales Luna por su empeño y dedicación en la conducción de este trabajo de investigación.

Resumen

En este documento se propone un mejor algoritmo para el problema de asignación de tareas a procesadores en una malla de PC's no dedicados al procesamiento en paralelo bajo el modelo BSP. Dicho algoritmo se desarrolla en base al planteamiento de un campo de *seudofuerzas* —de aquí en adelante referido como *fuerzas*, con la aclaración de que no son fuerzas en el mismo sentido que las fuerzas físicas— definido en torno a una malla de computadoras de escritorio. Mediante el análisis por simulación se comparan los resultados obtenidos del método de campo de fuerza con los algoritmos canónicos para la asignación de tareas en un ambiente paralelo.

Se opta por la utilización del modelo BSP por el concepto de superpaso, un conjunto de tareas en paralelo entre barreras de sincronización. Este concepto permite analizar la manera en que podría trabajar un programa paralelo en un sistema con las características que presenta una malla de computadoras de escritorio. En consecuencia, para calificar el desempeño de un algoritmo de asignación de tareas, se mide qué tan cerca se encuentra la duración del superpaso al tiempo óptimo. Es decir, se busca la minimización del tiempo de vida del superpaso. Para ello se consideran características propias del conjunto de tareas, los procesadores y requerimientos de comunicación.

Se diseña y desarrolla un simulador para reproducir el comportamiento de una malla dinámica de computadoras. Con éste se efectúan las pruebas de rendimiento y comparación de los algoritmos de asignación de tareas.

Se estudia el detalle de un problema abstracto y otro concreto. Este último compuesto por el análisis de la resolución en paralelo de la ecuación de onda para su aplicación a la tomografía sísmica de pozos. Adicionalmente, se desarrolla un método de diferencias finitas de mayor precisión y mayor demanda de recursos de cómputo que puede ser utilizado en la resolución en paralelo de la ecuación de onda para la tomografía sísmica.

Abstract

This document proposes a better algorithm for the problem of assigning tasks to processors in a grid composed of desktop PC's which are not dedicated to parallel processing under the BSP model. Such algorithm is developed based upon the concept of *pseudo-forces* —from here on referred to as *forces* with the clarification that they are not forces in the same sense as of physical forces— defined around a grid of desktop computers. With the analysis by simulation, the results obtained from the force field method are compared with canonical task assignment methods in a parallel environment.

The BSP model is chosen in regard for the concept of *superstep*, a set of tasks performed in parallel between synchronization barriers. This concept allows for the analysis of the manner in which a parallel program may execute in a system with the characteristics presented by a grid of desktop computers. In consequence, to qualify the performance of a task assignment algorithm, the superstep duration is compared to the optimum time. For this we consider the characteristics pertaining to the set of tasks, the processors and the communication requirements.

We design and develop a simulator to reproduce the behavior of a dynamic grid of desktop computers. With this we run performance analysis and comparison with the different algorithms for task assignment.

The detail of an abstract problem and another concrete problem are studied. The latter composed by the analysis of the parallel resolution of the wave equation for the ultimate application to borehole seismic tomography. Additionally, a higher order finite differences method is developed with greater demand for computational resources which may be used for the parallel resolution of the wave equation for borehole seismic tomography.

Índice general

Resumen 3
Abstract 4

Capítulo 1. Introducción 7
1.1. Objetivo 7
1.2. Problemática 8
1.3. Términos 9
1.4. Asignación Consciente 11
1.5. Asignación Inconsciente 14
1.6. Mallas dinámicas versus estáticas 16
1.7. Modelos 21
1.8. Modelo BSP 23
1.9. Organización del documento 27

Capítulo 2. Método de campo de fuerza 30
2.1. Contribución 30
2.2. Consideraciones generales 33
2.3. Modelo del campo de fuerza 34
2.4. Diseño del algoritmo 40
2.5. Recolección de información 46
2.6. Observaciones 48

Capítulo 3. Simulación de la malla gaussiana 50
3.1. Estado del arte 51
3.2. Estrategias de asignación 55
3.3. Implementación del algoritmo 61

Capítulo 4. Evaluación del método de campo de fuerza 74

4.1. Resultados preliminares 76
4.2. Costo de tarea aleatorio 80
4.3. Aplicación a la tomografía sísmica de pozos 88
4.4. El problema de la recolección de la información 93

Conclusiones 104

Apéndice 109
La ecuación de onda en la tomografía 109
Resolución por diferencias finitas 112
Esquema de números reales 114
Esquema de números complejos 116
Análisis de estabilidad lineal 118
El método Rayuela-E4 119

Bibliografía 125

CAPíTULO 1

Introducción

En este capítulo se explica el objetivo y el problema que dan origen a la contribución principal de este trabajo de investigación. Con la exposición de la metodología de resolución del problema se detalla la manera en que la resolución del problema lleva al desarrollo de los capítulos. Se explica el razonamiento que llevó al modelo de computadora paralela seleccionado para la resolución del problema, entrando en algunos detalles acerca de las generalidades de los esquemas de asignación de tareas. Finalmente se concluye con una descripción general de los métodos para asignación de tareas a procesadores remotos, seguido por una descripción esquemática del resto del documento de tesis. El enfoque del material introductorio no es a detalle, ya que abarca varios campos y áreas distintas y un análisis exhaustivo se desviaría de los objetivos del trabajo de tesis. Para el lector interesado, hacemos citas a textos más elaborados donde se puede profundizar sobre algun tema en particular.

1.1. Objetivo

En este proyecto de investigación el objetivo fue proponer un mejor algoritmo para el problema de asignación de tareas en procesamiento paralelo —bajo el modelo *Bulk Synchronous Parallel*— a procesadores y aplicarlo al problema específico que representa una malla de computadoras de escritorio.

La contribución principal es la propuesta del método de Campo de Fuerza para la asignación de tareas paralelas en mallas de computadoras caracterizadas por distribución gaussiana de parámetros. Se detalla en el capítulo 2

1.2. Problemática

En la actualidad se tiene en las empresas del sector público una gran infraestructura de cómputo instalado en computadoras de escritorio e interconexión de red. Sin embargo en estas empresas hay severas limitaciones con respecto a los recursos para efectuar cómputo paralelo. Para que un usuario pueda ejecutar un proceso paralelo de un número limitado de hilos —utilizando sólo una fracción del total de computadoras instaladas— se requiere de un mecanismo de asignación de tareas capaz de resolver de una manera eficiente cuales nodos serán utilizados para el cómputo. Esta asignación, además, deberá tener la capacidad de variar dinámicamente conforme avanza la ejecución del programa, escogiendo diferentes subconjuntos de procesadores de la malla de acorde al cambio en la disponibilidad de recursos. La disponibilidad de recursos no es constante ni conocida, ya que se trata de una malla de computadoras dedicadas a labores de oficina.

Para resolver el problema, el primer paso es escoger un modelo de computadora paralela que permita dividir la ejecución del programa paralelo en etapas. Cada una de estas etapas estará sujeta a una asignación de tareas distintas. De esta manera se evitará comprometerse con una asignación hasta el término de la ejecución. Esto es fundamental en una malla de cómputo cuya disponibilidad en recursos varía de manera aleatoria.

Una vez que se ha escogido un modelo de computadora paralela idóneo para la resolución del problema, se procede a proponer el algoritmo de asignación de tareas. Esta propuesta refleja la contribución principal de nuestro trabajo de tesis y constituye material publicado en el *The Fourth International Conference on Grid and Cooperative Computing* [**30**]. Sin embargo el trabajo no termina con esto. Para comprobar la validez del método de asignación de tareas y efectuar un análisis comparativo con otros métodos de asignación, es necesario realizar pruebas. Dado que en la actualidad los sistemas operativos de las computadoras de escritorio no cuentan con las implementaciones

de código que harán posible este tipo de compartición de recursos en la escala que se requiere, es necesario simular estas mallas. Para esto es necesario desarrollar un simulador que sea capaz de reproducir estas condiciones y realizar pruebas comparativas con otros métodos de asignación de una manera imparcial. La importancia del simulador no puede subestimarse y por ello es que se buscó el reconocimiento internacional de esta herramienta al nivel de revista internacional arbitrada. Esto se logró en la revista *Multiagent and Grid Systems* [**32**], donde en el cuerpo de revisores se encuentra R. Buyya, autor del reconocido simulador para mallas de cómputo, GridSim [**14**].

Finalmente es necesario comprobar que la propuesta es útil. Para ello se analizan el ejemplo de la resolución en paralelo de la ecuación de onda en el problema de la tomografía sísmica petrolera. Cabe mencionar que la tomografía sísmica de prospección petrolera es una aplicación de la tecnología computacional para la localización exacta de yacimientos del hidrocarburo, tema de suma importancia. En el apéndice de esta tesis se pueden encontrar más detalles acerca de los problemas estudiados, detalles que no están directamente relacionados con el método de Campo de Fuerza, pero que son interesantes. Dado que todos los algoritmos de asignación son sujetos al problema que implica la recolección de información que caracteriza el estado del sistema, un paso adicional en la comprobación del validez del algoritmo de asignación propuesto consiste en establecer los criterios bajo los cuales este problema se ve solucionado. Este se trata en la sección 4.4.

En el apéndice se proporciona una descripción de metodologías de resolución de onda y en la (sección 4.4) se introduce un nuevo método numérico de mayor orden para la resolución de la ecuación de onda. Sin embargo, la lectura de la sección 4.4 no es vital para el desarrollo de esta tesis y puede ser omitida por el lector.

1.3. Términos

En esta sección hacemos algunas aclaraciones de términos que son utilizados a los largo del trabajo.

***Barrera de sincronización*:** : Una barrera de sincronización es una de las primitivas básicas en la programación paralela. Cuando en la ejecución de un programa paralelo se define una barrera de sincronización, ninguna tarea en ejecución puede continuar ejecutando más allá de la barrera hasta que todas las tareas hayan llegado a la barrera. Esta primitiva es utilizada para evitar las condiciones de carrera y garantizar la validez de la memoria compartida.

***Superpaso*:** : Un superpaso consiste del conjunto de tareas que se realizan entre dos barreras de sincronización en la ejecución de un programa paralelo bajo el modelo BSP.

***Asignación consciente*:** : Corresponde a una asignación de tarea que se produce a partir de una petición expresa del programa en ejecución. Es decir, la migración de tarea está codificado dentro del algoritmo de ejecución de la aplicación en ejecución.

***Asignación inconsciente*:** : Corresponde a una asignación de tarea la cual está codificado en una aplicación ajena a la tarea. Este tipo de asignación realizada por un sistema operativo o de control que desconoce las actividades que las tareas están efectuando o están por realizar.

***Algoritmo de migración*:** : Se refiere a que las tareas que emergen —o bien las que se están ejecutando— se envían para su procesamiento a procesadores remotos desde el procesador que lleva el control de inicio y terminación del algoritmo. Este es el concepto encapsulado en las instrucción `spawn` de MPI, que se puede clasificar como un ambiente primordialmente de tipo migratorio.

***Algoritmo de demanda*:** : Implica un sistema descentralizado, donde varios nodos comienzan su ejecución y van solicitando recursos según tengan disponibilidad de ejecución. Este es el modelo primordial de distribución de carga PVM, ya que

con este ambiente la manera básica al iniciar los procesos remotos no es mediante el comando `spawn` sino que al cargar el programa se especifica en cuántos nodos se desea ejecutar. Es claro que con MPI también se puede utilizar un modelo *demanda* así como en PVM se puede hacer *migración*, pero en sus inicios estos ambientes no fueron diseñados así.

***Algoritmo estático*:** : Implica un algoritmo de asignación en la que los criterios a considerar para la migración son determinados de manera anticipada y fija.

***Algoritmo dinámico*:** : Implica un algoritmo de asignación en la que los criterios a considerar para la migración son determinados de manera dinámica.

1.4. Asignación Consciente

El primer punto a resolver para resolver el problema expuesto en la sección 1.2 es determinar si la asignación de tareas será efectuada de manera consciente o inconsciente. En esta sección trataremos algunas generalidades de la asignación consciente para la definición adecuada del algoritmo de asignación de tareas en el capítulo 2.

Muchos modelos para la programación de tareas en la computación en paralelo hacen referencia a la gráfica acíclica dirigida (DAG), concepto desarrollado por Yannakakis y Papadimitriou **[63]**.

Para el problema que representa la minimización de tiempos de ejecución de modelos basados en DAG puede encontrarse en Jansen **[46]**. Además, varios algoritmos heurísticos han sido propuestos como lo son *Duplication Scheduling Heuristic* (DSH) **[50]**, *Linear Clustering with Task Duplication* (LCTD) **[70]** y *Critical-Path First Duplication* (CPFD) **[63]**. Todas esas heurísticas aplican a un paralelismo de grano fino con un conocimiento *a priori* del tiempo computacional para cada tarea. Es decir, asignación consciente. Otro algoritmo para la programación de tareas es *Bulk Communication Scheduling Heuristic* (BCSH) **[29]** que toma en consideración el tiempo de comunicación entre las tareas.

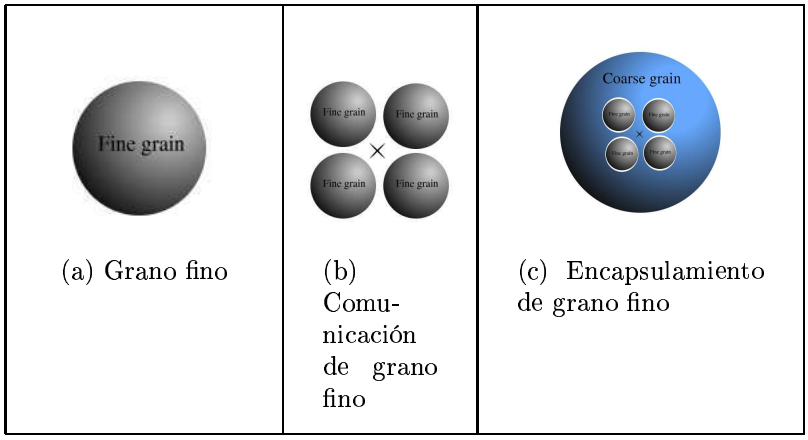
En cuanto a las estrategias de balanceo de carga de tipo *consciente*, es importante la *granularidad de paralelismo*. Los programas de grano fino intercambian un gran número de paquetes de información con poco contenido en cada paquete. Las aplicaciones de grano grueso intercambian menos paquetes de información, pero el contenido de cada paquete es mayor.

Con el paralelismo de grano fino, la cantidad de comunicación y directivas de sincronización entre procesadores participantes es muy alto. Por esta razón el paralelismo de grano fino se limita a la cantidad de procesadores que se encuentran conectados directamente entre sí, capaces de acceder a las mismas localidades físicas de memoria. Por lo general este paralelismo se limita a 2, 4 u 8 procesadores y en consecuencia se llega a un tope en la escalabilidad muy rápidamente. Los avances en el diseño de procesadores indican que en el futuro cada procesador tendrá más de una cabeza capaz de ejecutar instrucciones de manera paralela **[34]**. Es importante poder explotar esta capacidad de grano fino, aún en aplicaciones de grano grueso. Para ello, es necesario desarrollar una estructura de programación eficiente, donde se puedan combinar el grano fino con el grano grueso. En el caso de estudio que se presenta en el capítulo 4 se muestra una aplicación práctica de esta técnica.

Para realizar con éxito esta heurística es necesario encapsular los granos finos dentro de los granos gruesos. Para ello, además de encapsular los granos finos, el número de estos no debe exceder la capacidad de ejecución en paralelo del nodo que recibe el grano. Este encapsulamiento se ilustra en la figura 1.4.1.

El modelo de computadora paralela BSP se adapta a este esquema de encapsulamiento de mensajes ya que en la barrera de sincronización se tiene:

$$\sum_{i=0}^{i=n} \overleftrightarrow{m}_i = \sum_{j=0}^{j=n} \underleftrightarrow{m}_j$$



(a) Grano fino

(b) Comunicación de grano fino

(c) Encapsulamiento de grano fino

FIGURA 1.4.1.

donde $\overleftrightarrow{m}_k$ y $\underleftrightarrow{m}_k$ son los mensajes emitidos y recibidos respectivamente por el nodo k.

Haciendo una abstracción hacia un espacio continuo de información se tiene que ci_p —la cantidad de información que emite cada nodo en el intervalo $[t_1, t_2]$—, es igual a:

$$ci_p = \int_{t_1}^{t_2} b(t)dt$$

siendo $b(t)$ los bytes que se transfieren en función del tiempo. De donde se desprende que nb —la cantidad de bytes que integrarán el grano grueso de mensajes del nodo i hacia el nodo j—, será

$$nb = \int_{t_1}^{t_2} b(t, i, j)dt$$

donde $b(t, i, j)$ es la función del tiempo y los nodos de emisión y recepción que determina los bytes transferidos. En consecuencia st —la suma total de los mensajes de grano grueso—, está proporcionado por

$$st = \int_0^n \int_0^n \int_{t_1}^{t_2} b(t, i, j)\, dt\, di\, dj.$$

Para resolver el objetivo expuesto en la sección 1.1 es que el proceso de asignación de tareas sea transparente para el usuario. De otra forma

perdería generalidad y obligaría a la aplicación paralela a considerar todos los detalles de la asignación consciente.

1.5. Asignación Inconsciente

Para resolver el objetivo expuesto en la sección 1.1 es necesario tratar con un esquema de asignación inconsciente. Veamos las generalidades de este concepto antes de proceder con la selección del modelo. El lector que esté familiarizado con el tema puede saltarse esta sección.

Aunque los modelos de computación en paralelo para el análisis de la programación estática de tareas conlleva una gran importancia, así también tiene relevancia el estudio de la programación de tareas y balance de carga realizados de manera dinámica. Existe una abundancia de métodos de distribución de carga dinámica en la literatura, pero en la mayoría de los casos, cada uno es implementado para resolver un problema técnico o científico en particular, tal como el *parallel tree code* [**21**] que resuelve problemas relacionados a la simulación de galaxias en formación. Casi todos estos modelos forman parte del grupo que hemos llamado *inconsciente*.

Otros estudios teóricos como los realizados por Fonlupt [**24**] o el *Adaptive Load Distribution System* (ALDY) [**68**] han recibido amplia distribución en años recientes. El sistema ALDY constituye una propuesta para efectuar distribución de carga paralela mediante una biblioteca para aplicaciones PVM [**36**], utilizando el paradigma de *task farming* en combinación con un sistema para migración de objetos — transparente al usuario y que fuerza la migración de tareas en relación a las estadísticas de uso de sistema, sin considerar el algoritmo de la tarea en particular que se va a migrar. Con ALDY el proceso sujeto a migración se refiere como un agente [**67, 66, 79**].

En general existen dos métodos para transferir una serie de tareas de un procesador a otro. En primer lugar está la *demanda*, donde la migración es iniciada por un requerimiento remoto, y en segundo lugar la *migración*, donde la migración se inicia por condiciones resueltas en el procesador local. Un algoritmo de distribución de carga puede

utilizar cualquiera de ellos o ambos en distintas partes de ejecución de las tareas.

El método de demanda se determina por el siguiente seudocódigo, donde "AvailableResources" es un predicado que indica la disponibilidad de recursos computacionales disponibles en el nodo computacional considerado y "RequestJob" es un procedimiento que inicia la negociación para la migración de una tarea desde el nodo especificado (los subíndices to y $from$ se especifican para mayor claridad del flujo de la información):

for each $node_{to}$ **in** computers **do**
if (AvailableResources($node_{to}$))
  for each $remote_{from}$ **in** computers **do**
  if (**not** AvailableResources($remote_{from}$))
    RequestJob($remote_{from}$,$node_{to}$)
  endif
  endfor
endif
endfor

Mientras que el método de migración se describe esquemáticamente por el siguiente seudocódigo, donde "MigrateJob" es un procedimiento que inicia la negociación para la migración de una tarea hacia el nodo especificado:

  for each $node_{from}$ **in** computers **do**
  if (**not** AvailableResources($node_{from}$))
    for each $remote_{to}$ **in** computers **do**
    if (AvailableResources($remote_{to}$))
      MigrateJob($node_{from}$,$remote_{to}$)
    endif
    endfor

endif
endfor

Cuando un procesador sobrecargado migra parte de sus tareas asignadas a otro procesador, el algoritmo de verificación de carga puede encontrarse ya sea en un procesador central o bien distribuido entre los procesadores que conforman el equipo paralelo. El algoritmo de verificación de carga es el responsable de determinar las tareas a transferir y de informar a los procesadores involucrados lo qué deben hacer para completar la transferencia de manera exitosa. Algunos ejemplos de dichos algoritmos apegados a este paradigma de manera centralizada (*migración*) se pueden encontrar en **[6, 39, 47, 60]**, mientras que otros que la siguen de manera distribuida (*demanda*) se encuentran en **[28, 51, 82]**.

Existen varios métodos para implementar la determinación de disponibilidad de recursos en los sistemas, utilizando cantidades tales como los ciclos de cómputo no utilizados por unidad de tiempo y la memoria libre disponible.

La resolución del problema enunciado en la sección 1.2 nos lleva a una situación de asignación inconsciente. De tal manera, los algoritmos de asignación desconocerán la complejidad de cómputo de las tareas a asignar y tan sólo podrán contar con la información de los recursos de memoria necesarios por tarea. Este aspecto es de fundamental importancia para el diseño del simulador en el capítulo 3.

1.6. Mallas dinámicas versus estáticas

La resolución del problema que plantea resolver este trabajo de investigación de aplica a las mallas de computadoras de escritorio. Para ello debemos distinguir varios aspectos importantes relativos a esta plataforma.

Para la utilización del potencial representado por las computadoras enlazadas en red, sistemas tales como Globus **[26]** y Condor **[55]**

distribuyen las tareas en una malla de computadoras. Estos sistemas se refieren en gran detalle a los asuntos como *scheduling*, control de acceso y manejo de recursos y usuarios. El mecanismo provisto por Condor para compartir recursos —al aprovechar los ciclos no utilizados de computadoras de escritorio—, proporciona un gran poder de cómputo mediante componentes de fácil adquisición. Se han tenido los campos experimentales de Condor [**75**] y Globus en aplicaciones prácticas [**57, 41**]. Una cantidad sustancial de literatura existe sobre el tema de balanceo de carga. Dentro de ésta, un enfoque práctico puede ser encontrado en Watts [**81**] o un método alimentado por el historial en Bozyigit [**12**]. Una desventaja surge de la organización de la malla bajo la administración de un solo servidor central, configurado de manera estática. Todas las tareas esperan en una cola hasta que el recurso apropiado para su ejecución pueda ser localizado dentro de la malla por el servidor centralizado. La falla de este nodo trae como consecuencia la falla de toda la malla Condor. Más aún, no existe mecanismo para compartir la carga de trabajo que representa el manejo de la malla: la degradación del rendimiento del sistema puede surgir con un servidor central sobrecargado de tareas.

Una *malla estática* es un conjunto de computadoras en el cual los recursos computacionales se conocen y se configuran de manera estática de antemano. Más aún, los costos de comunicación entre procesadores es generalmente uniforme. Ejemplos de mallas estáticas son las supercomputadoras de memoria compartida o los cúmulos de estaciones de trabajo dedicados. En éstos, la asignación de tareas a procesadores se hace con la certeza de que los recursos serán suficientes. Las mallas estáticas comúnmente tienen configuraciones homogéneas y conexiones de red optimizadas. Una *malla dinámica*, por otro lado, es una colección de computadoras interconectadas donde el costo de comunicación entre procesadores es variable y los recursos computacionales en los nodos remotos también pueden cambiar sin previo aviso.

Es decir, si $R(i,t)$ son los recursos computacionales del nodo i en el instante de tiempo t, entonces una malla estática presenta la característica de que

$$K(t) = \sum_{i=0}^{n} \int_0^t R(i,t)dt$$

donde K es una constante. En una malla dinámica la situación es distinta ya que la caracteristica propia de estos sistemas consiste en que

$$f : t \mapsto f(t) = \sum_{i=0}^{n} \int_0^t R(i,t)dt$$

siendo f una función de tiempo, no necesariamente continua.

El sistema Condor nació como un esfuerzo para aprovechar los recursos de las estaciones de trabajo en un ambiente académico controlado. Por ello la topología se determinaba de manera inicial por condiciones de una malla estática. Conforme las computadoras personales de escritorio han sobrepasado el potencial de las estaciones de trabajo de antaño, el desarrollo de Condor comenzó a enfocarse a las computadores de uso común. Esto implicó la extensión de las condiciones de una malla estática al dominio de una malla dinámica. En Condor, por ejemplo, la calendarización y migración de tareas son exclusivas a un nodo administrador central. Las tareas esperan ser asignadas en una cola hasta que los recursos apropiados pueden ser localizados dentro de la malla. Un deterioro en el rendimiento del sistema puede ocurrir con un servidor sobrecargado. La falla de este nodo inevitablemente colapsa a la malla. Los recursos pueden ser compartidos entre varias parvadas de Condor. Esto se hace con el descubrimiento automático de estas agrupaciones de Condor más allá de los dominios administrativos **[13]**. Pero dentro de una parvada de Condor no existe mecanismo para tratar con los costos de comunicación al hacer asignación de tareas. Durante el periodo de la creación de Condor esta consideración no era importante, pero la situación ha cambiado en años recientes. Cuando los recursos computacionales de un nodo participando en una

malla cambian, la configuración estática del servidor principal de Condor debe ser actualizado. Esto no es una falla del sistema de Condor, más bien es la consecuencia de extender las consideraciones de una malla estática al territorio de las mallas dinámicas. Para elaborar nuevo software administrativo para este nuevo ambiente, las condiciones que caracterizan a las mallas dinámicas deben ser simuladas.

La combinación de las ventajas económicas del equipo de cómputo de fácil adquisición con el incremento en la complejidad de las aplicaciones científicas está marcando la pauta hacia una mayor utilización de mallas dinámicas no-dedicadas ensambladas a partir de computadoras personales. Una de las características mas prominentes de mallas de computadoras de escritorio es su tendencia de crecer no tan solo en tamaño sino en rendimiento y capacidad de memoria. Un algoritmo eficaz para la asignación de tareas debe tomar este hecho en consideración para extender la utilidad de los algoritmos por un periodo razonable de tiempo.

Las asignaciones aleatorias de tareas a procesadores usualmente tienen buenos resultados. En una malla dinámica, no obstante, los recursos disponibles en cada procesador constituyen una variable del entorno. Un proceso de asignación aleatoria que no considere esta variable incurrirá en el riesgo de asignar una tarea a un procesador donde no existan los suficientes recursos. La tarea por ende será rechazada por el procesador remoto con la consecuente pérdida de tiempo. Más aún, el siguiente intento para asignar la tarea a un procesador distinto también tiene posibilidad de errar. Conforme los requerimientos computacionales de las tareas se incrementa, la tasa de rechazo también aumenta. Las aplicaciones científicas contemporáneas son de tal complejidad que existe una demanda creciente por la capacidad de cómputo y el acceso a conjuntos más grandes de información [**27**]. En este caso, mantener información sobre los recursos que caracterizan la malla dinámica llevará a una estrategia más eficaz para la asignación de tareas.

En el caso de mallas de computadoras de escritorio, si se considera una sola latencia determinada por hardware, se incurre en un error. La topología actual se determina no sólo por el hardware, sino también por el ancho de banda disponible para que los procesos se comuniquen. Este factor dinámico no sólo depende de la localidad, la fecha y la hora, sino también de los hábitos y comportamiento de los usuarios individuales de las computadoras que conforman el ambiente paralelo.

La complejidad de las aplicaciones científicas con demanda creciente por el poder de cómputo y el acceso a conjuntos más grandes de información está marcando una pauta hacia el incremento en la utilización de mallas de computadoras personales [**27**]. En la búsqueda por un balanceo de carga, una consideración importante para maximizar la utilización de la malla dinámica es la asignación óptima de procesadores en un programa paralelo. Un esquema para la asignación de tareas de manera efectiva en una malla de computadoras de escritorio debe tomar la topología dinámica de la red en consideración, así como las variables tradicionales tales como disponibilidad y potencial de los procesadores remotos. Dado el constante crecimiento de este tipo de mallas, el método debe ser escalable, es decir, funcionar de manera mejor conforme crece el tamaño de la malla. Como ha sido mencionado por Zhuge [**85**]: "En el ambiente de la sociedad futura interconectada, los recursos fluirán de nodos de alta energía hacia los de baja energía." Las leyes y principios básicos que gobiernan este campo requieren de más investigación. El trabajo presentado en esta tesis es un paso en esa dirección.

Con respecto a la distribución de recursos entre distintas mallas de Condor, trabajo reciente por Butt y colegas [**13**] ha provisto los algoritmos para la automatización del descubrimiento de mallas Condor remotas más allá de los dominios administrativos. En su trabajo, un esquema "p2p" basado en un sustrato de ruteo de proximidad [**17**] se utiliza en conjunción con la facilidad de las mallas Condor. Pero dentro de la malla Condor, como con la mayoría de los métodos de balanceo de carga, no existe mecanismo para resolver el asunto de proximidad

y tomar este factor en consideración mientras se hace la asignación de tareas o las migraciones.

1.7. Modelos

En seguida la resolución del problema requiere visualizar la computadora paralela en términos de un modelo. Esto es porque la computadora que se quiere emular no existe de manera explícita, sino que se construirá de manera virtual a partir de una malla de computadoras caracterizadas por distribución gaussiana de parámetros. Comenzaremos con una breve revisión de los modelos de computadora paralela más populares sin entrar en los detalles que se encuentran disponibles en la literatura.

Los modelos para la computación en paralelo se requieren por la misma razón que en el cómputo secuencial: éstos actúan como una abstracción común entre los distintos lenguajes de programación. Un programa escrito de acuerdo al modelo se puede ejecutar en distintas plataformas y su comportamiento será predecible en la medida en que el modelo lo sea. En esta sección presentamos un breve esbozo de los distintos modelos de computadora paralela que existen en la literatura con miras a escoger el que será utilizado en el desarrollo de este trabajo de investigación.

Skillicorn presenta tres requisitos para un modelo de computación paralela: la independencia de la arquitectura, la congruencia y la facilidad de manejo conceptual [**71**]. Estos son detalles importantes a considerar para el desarrollo de un algoritmo para asignación de tareas.

En la computación en paralelo, existe una gran cantidad de modelos de gran valía, pero de estos hay unos que se aplican mejor a las condiciones de una malla de computadoras de escritorio, como se verá a continuación. Entre los modelos se cuentan PRAM [**25**], APRAM [**18**], BPRAM [**2**], LPRAM [**3**], WPRAM [**61**], YPRAM [**20**], HPRAM [**42**], BSP [**78**], LogP [**19**], BMF [**9**], DAG [**63**] y CLUMPS [**25**].

Algunos de estos modelos son de rendimiento y otros de arquitectura. La finalidad de este trabajo no es la determinación de cual es el mejor modelo a seguir para la programación en paralelo sobre un ambiente de mallas de computadoras de escritorio. Sin embargo para llevar a cabo el estudio detallado en esta tesis, es necesario escoger un modelo de computadora paralela. Destacamos que el método desarrollado y el estudio subsecuente puede realizarse bajo distintos modelos de computadora paralela, y esto lo señalamos en el capítulo dedicado a las conclusiones.

Entre los modelos de computadora paralela destaca el modelo BSP que busca capturar en un ambiente paralelo las ventajas que tiene el modelo von Neumann.

BSP [**78**] es un modelo puente entre hardware y programación puesto que es transparente al tipo de hardware de la computadora paralela. Esto es un punto a favor para justificar su utilización en una malla dinámica. Otro punto a su favor que justifica su utilización es la distinción entre memoria local y memoria remota (*two-level memory model*) lo cual favorece su aplicación a una malla de computadoras de escritorio. El modelo canónico BSP, sin embargo, se queda corto para los objetivos de este trabajo de investigación porque no diferencia el costo de las comunicaciones. No obstante, con la extensión del modelo [**8**] considera el efecto de la latencia y esta deficiencia queda solventada.

El modelo BSP tiene gran potencial en su aplicación a las mallas dinámicas ya que el modelo provee escalabilidad. Uno de los aspectos más relevantes de las mallas dinámicas es su habilidad para crecer, y por ende la escalabilidad es un asunto importante. En 2001 Tiskin[**77**] desarrolló un método divide-y-vencerás bajo el modelo BSP y sugirió que el balanceo de carga puede hacerse en una manera eficiente bajo el modelo BSP. Con la versión del modelo que está consciente del ancho de banda, no sólo es esto posible, sino que un algoritmo de recuperación de errores puede ser implementado con muy poco esfuerzo de cómputo adicional.

La justificación más importante por la utilización el modelo BSP es la sincronización de bloque que distingue a este modelo de los demás. Esta sincronización permite dividir la ejecución del programa paralelo en etapas. Cada una de estas etapas estará sujeta a una asignación de tareas distintas. De esta manera se evitará comprometerse con una asignación hasta el término de la ejecución. Esto es fundamental en una malla de cómputo cuya disponibilidad en recursos varía de manera aleatoria, planteado en los objetivos de la sección 1.1.

En la sección 1.8 se proporciona una explicación algunos detalles del modelo BSP que se aplican en el desarrollo del método de asignación del capítulo 2.

1.8. Modelo BSP

En esta sección profundizaremos brevemente en algunos de los conceptos más importantes del modelo BSP, dado su aplicación en la propuesta del algoritmo de asignación de tareas que se desarrolla en el capitulo 2. Esta sección puede ser revisada rápidamente, o en el caso de que el lector se encuentre familiarizado con el modelo BSP, su lectura puede ser omitida.

El modelo BSP consiste de un conjunto de parejas procesador memoria, una red de comunicaciones global, y un mecanismo para la sincronización de los procesadores mediante barreras.

Una computadora ajustada al modelo BSP opera de la siguiente forma. Una computación se compone por una secuencia de *superpasos* paralelos, donde cada uno consiste de una secuencia de pasos, seguido por una barrera de sincronización en la cual toda la comunicación se completa. Durante un superpaso, cada procesador puede realizar un número de operaciones sobre valores que residen en memoria local, puede enviar y recibir mensajes y puede manejar requerimientos remotos de lectura/escritura. La esencia del modelo BSP es la noción de superpaso, en la cual la comunicación y la sincronización se separan. Un programa BSP procede en fases, con toda la comunicación global ocurriendo entre las fases. Esta manera de realizar programación en

paralelo provee una manera eficaz para desarrollar software paralelo para sistemas escalables. Dado que las comunicaciones y la sincronización no están fusionados en un programa BSP, el programador no tiene que preocuparse de problemas como *deadlock*, que pueden ocurrir con el intercambio de mensajes de manera asíncrona. La depuración de un programa BSP también se hace más fácil para el programador. La barrera al final de un superpaso provee un punto apropiado en el cual el estado global de la computación está bien definida y puede ser interrogada. Por esta razón la depuración de un programa BSP no es más difícil que para un programa secuencial.

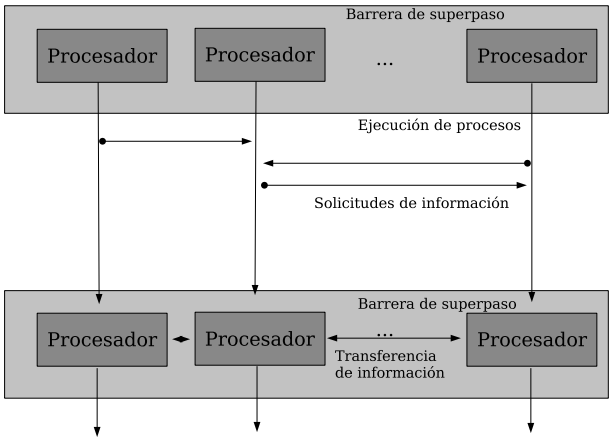


FIGURA 1.8.1. *Sincronización en el modelo BSP.*

En la figura 1.8.1 puede verse esquematizado el funcionamiento de una máquina paralela bajo el modelo BSP. Las operaciones de comunicaciones en el modelo BSP se pueden resumir con `put` y `get`. Con el primero se deposita información de la memoria local en memoria remota para otro proceso. Con el segundo se trae de la memoria remota información y se coloca en la memoria local del proceso. Ambas operaciones son primitivas de comunicación de un solo lado. Es decir, no requieren de la participación activa de los otros procesos. Además, ninguna de las dos son instrucciones bloqueantes. Todos los `put` y `get` iniciados durante un superpaso se completan antes de que comience el siguiente superpaso.

Para desarrollar el método de asignación de tareas del proyecto de investigación, se requiere de un modelo en el cual se tengan puntos en la ejecución en los que el sistema llegue a un estado definido. Esto implica que los sistemas asíncronos no serán adecuados puesto que no son determinísticos. Esta exigencia es consecuencia directa de una característica notable de una malla de computadoras, *la variabilidad*. Las mallas de computadoras de escritorio se componen de una cantidad de computadoras con distintas características que están conectadas mediante una red heterogénea. Generalmente no son construidas con la intención de proveer de una infraestructura de cómputo paralelo (en el sentido tradicional). Por otro lado, para concretar un algoritmo eficaz de asignación de tareas en estas circunstancias, es necesario conocer el estado del sistema en el momento de la asignación de tareas. Es por ello que se requiere de un modelo que tenga sincronización, como BSP. Este modelo posee algunas ventajas y desventajas. Para comenzar, hay que extenderlo con el modelo de costos [**8**] y el conocimiento de condiciones de ancho de banda. Su condición de sincronización en bloque, en cambio, es una ventaja. La manera en cómo extender el modelo BSP de la manera más eficiente es aún un problema abierto, como lo se puede comprobar con el trabajo de Kechid y Myoupo [**48**].

Éste modelo divide las aplicaciones paralelas en una serie de superpasos, cada uno de los cuales consiste de un número de hilos en paralelo con cualquier cantidad de operaciones. El aspecto más importante del modelo BSP es que la sincronización de barrera tiene lugar al final de cada superpaso. Los hilos paralelos individuales únicamente efectúan comunicación local hasta la barrera. En este punto toda la comunicación global toma lugar. Dado que la comunicación local se permite para cada hilo, es posible para un hilo efectuar una subparalelización. Esto sólo tendría sentido —claro está— si el hilo que se subparalelizará está asignado a una unidad SMP de memoria compartida donde dicha subparalelización sea deseable.

En términos generales, el modelo BSP consiste de un conjunto de nodos —cada uno con una área de memoria asociada—, una red global

de comunicaciones y un mecanismo para la sincronización de barrera de manera eficiente [**59**].

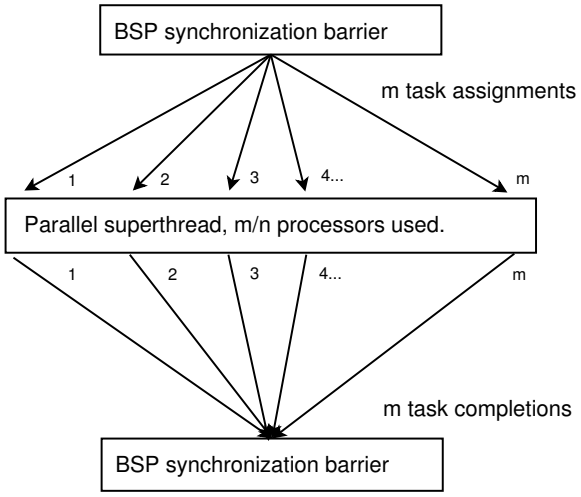


FIGURA 1.8.2. Superpaso BSP

La figura 1.8.2 representa el bloque básico para una máquina BSP con n nodos ejecutando una aplicación paralela. A la sincronización de barrera inicial, la aplicación paralela de divide en m ($m \leq n$ ya que el número de tareas a realizar en paralelo no debe superar al número de procesadores) tareas paralelas. Estas tareas entonces se asignan a nodos individuales. Para las mallas estáticas esta asignación se hace —de manera usual— aleatoriamente, pero para una malla dinámica otros métodos serían más apropiados.

Una vez que son asignadas las tareas, cada nodo actúa por cuenta propia. Estos nodos pueden enviar mensajes a otros nodos en cualquier momento pero los requerimientos de comunicación no se completan hasta la sincronización de barrera.

Para hacer una asignación de tareas en la barrera de sincronización en el ambiente de una malla dinámica, la estrategia debe considerar el estado actual de la red de comunicaciones y los recursos computacionales en los nodos remotos.

Estrategias usadas recientemente **[74]** incluyen la minimización de los costos de comunicación **[1, 37]** y el logro de un balanceo computacional **[69]** entre los nodos participantes. Mientras estas estrategias producirán buenos resultados con mallas pequeñas, métodos tales como el campo de fuerza **[30]** que consideran ambos factores que limitan a una malla dinámica producen tiempos de ejecución del programa paralelo reducido conforme crece la malla. Esto se analiza con más detalle en el capítulo 4.

Dado que todas las tareas deben sincronizarse en la barrera, un detalle en particular no deberá ser ignorado para poder combinar la variable del costo de la comunicación con la información de ciclos sin uso en nodos remotos. El estado de la red debe ser medido relativo al costo de la computación. Conforme las velocidades de los procesadores aumentan a un ritmo más acelerado que el del equipo de comunicaciones, la relación entre los dos también se incrementa. Aplicaciones paralelas con una fuerte demanda para las comunicaciones, tales como la distribución de información **[74]**, se pueden beneficiar de métodos de asignación de tarea optimizados.

Más detalles acerca del modelo de computadora BSP pueden encontrarse en la literatura en **[78, 8, 64, 48, 72, 43, 11, 10, 74]**.

1.9. Organización del documento

En el capítulo 2 se detalla el concepto de Campo de Fuerza para la asignación de tareas, aportación principal de esta tesis y material publicado en la conferencia internacional "Grid and Cooperative Computing 2005" **[30]**.

El capítulo 3 consiste del diseño e implementación de un simulador para mallas de computadoras caracterizadas por distribuciones gaussianas, cuya metodología ha sido utilizado para la validación y comparativa del método de Campo de Fuerza en relación a otras alternativas. El material de este capítulo, en conjunto con capítulo cuarto,

ha sido aceptado para su publicación en la revista internacional arbitrada, *Multiagent and Grid Systems, An International Journal* **[32]**[1]. Una vez planteado el concepto de asignación de tareas en la malla bajo estudio mediante un campo de fuerzas, nuestra investigación se enfoca a la aplicación del método para demostrar su efectividad y ventajas en relación con métodos de asignación alternativos.

En el capítulo 4 se detallan algunos casos de estudio en los cuales es posible la aplicación del método desarrollado. En este mismo capítulo, en la sección 4.4 se presenta el material publicado en la conferencia internacional "The 2006 International Conference on Grid Computing & Applications" **[31]** en el cual se muestra como se resuelve el problema que implica la recolección de la información en una malla caracterizada por distribución gaussiana de parámetros y aplicado a la ejecución de programas paralelos bajo el modelo BSP.

Una vez planteado el concepto de asignación de tareas en la malla bajo estudio mediante un campo de fuerzas, nuestra investigación se enfoca a la aplicación del método para demostrar su efectividad y ventajas en relación con métodos de asignación alternativos. Como un ejemplo, planteamos la resolución de la ecuación de onda bidimensional de la tomografía sísmica de pozos, la cual es una aplicación de la industria petrolera que se explica en la sección pertinente, simulando el comportamiento que tendrá su resolución en la malla bajo estudio.

Por último, el trabajo concluye una discusión del alcance de la metodología propuesta, sus ventajas y desventajas, sus alcances y limitaciones, y un panorama de las líneas de investigación que quedan abiertas para trabajo futuro.

Como un ejercicio matemático adicional, y como contribución a los métodos numéricos existentes, desarrollamos un algoritmo nunca reportado en la literatura para resolver ecuaciones diferenciales parciales hiperbólicas con mayor precisión. Este método fue desarrollado durante el transcurso de la investigación de este proyecto y —aunque no

[1]Revista indexada en *Zentralblatt MATH*.

está directamente realcionado con el objetivo principal de esta tesis—lo incluimos por encontrarlo de interés en la aplicación de los resultados de método de asignación de tareas desarrollado en una aplicación práctica.

CAPíTULO 2

Método de campo de fuerza

2.1. Contribución

La contribución principal de este trabajo de tesis es la propuesta del método de Campo de Fuerza para la asignación de tareas a ejecutarse en paralelo bajo el modelo BSP [1].

Lo que distingue al método de asignación por campo de fuerza es que, en términos generales, está en razón inversa al cuadrado del costo de las comunicaciones y en razón directa a los recursos computacionales en el nodo remoto multiplicado por la facilidad computacional de la tarea a ser asignada.

El modelo de Campo de Fuerza utiliza una analogía a la Ley de Coulomb:

$$F = K\frac{q_1 q_2}{r_{12}^2} \tag{2.1.1}$$

donde q_1 relativo a la tarea a ser asignada, q_2 es relativo a las características del procesador remoto y r_{12}^2 relativo al costo en comunicaciones.

Es de notar que la fuerza disminuye con el inverso cuadrado de r, lo cual se deriva de la geometría esférica que rodea a una carga puntual. En el caso de la malla dinámica de computadoras contemplada en este trabajo, ésta es la geometría que a nuestro criterio se adapta al problema. En el caso de que se aplique el método de asignación por campo de fuerza a otro tipo de malla de computo, tal como un cúmulo con topología de red conocida, podría ser favorable escoger otro tipo de relación que sea más afín a la geometría física del sistema. conformada por PCs de escritorio

[1] Referido en la sección 1.8.

La justificación que existe en representar el problema de asignación mediante un campo de fuerza surge primariamente de un argumento filosófico. Si el movimiento de partículas en un universo físico se aproxima bien mediante el uso de un campo de fuerza, entonces el movimiento de tareas en un universo virtual definido por una malla de computadoras de escritorio puede ser modelado por un campo de fuerza. El concepto de campo de fuerza ha demostrado su utilidad en la física para aproximar lo que se observa en la realidad. Las analogías para el campo de fuerza son intuitivas: las computadoras que se encuentran lejanas —alto costo en comunicaciones— deberán tener muy poco atracción por las tareas a menos de que sean muy masivas —i.e., tienen una gran disponibilidad de ciclos de cómputo—. Propuestas teóricas recientes en la física han postulado la existencia de gravitones que viajan entre cuerpos para trasmitir la fuerza de la gravedad. Estos pueden ser vistos como los paquetes de información que necesitan viajar entre computadoras con la información de ciclos disponibles y costos de comunicación. Como Zhuge lo ha apuntado [**85**], en el ambiente interconectado del futuro los recursos deberán fluir de nodos de alta energía a los de baja. La energía nos lleva a los campos de fuerza.

El término k es una constante la cual puede ser equiparada a la unidad mediante una conversión de unidades adecuada. Es de notar que la Ley de Coulomb permite fuerzas de repulsión así como de atracción. Aunque en el caso estudiado en este trabajo sería suficiente la utilización de la fuerza de atracción únicamente, se utiliza también la fuerza de repulsión, ya que esto será necesario para extender la metodología de un Campo de Fuerza Escalar (tema de este trabajo) a un Campo de Fuerza Vectorial (línea de investigación a futuro). El signo de la Fuerza sólo dependerá del numerador en la ecuación 2.1.1 ya que las distancias son siempre positivas en la analogía utilizada.

El problema en particular sobre el cual se enfoca la propuesta es la resolución del problema de asignación de tareas en una malla caracterizada por una distribución gaussiana de sus parámetros operativos. Esto no quiere decir que el método de campo de fuerza no sirve para

la asignación de tareas en una malla estática o al interior del núcleo de un chip multicabeza. Estos problemas pueden tratarse más adelante en una investigación posterior, donde sería necesario refinar o redefinir el modo de obtención de las variables q_1, q_2 y r_{12}^2. Y quizás, muy probablemente, alterar la fórmula 4.4.1 por otra donde el campo de fuerza sea más afín a la arquitectura del sistema bajo estudio.

El algoritmo propuesto va funcionando mejor conforme aumenta el tamaño de la malla de cómputo. También resuelve el problema de la variabilidad en las comunicaciones y en la cantidad de recursos disponibles en la malla. De igual manera, por la manera en que el Campo de Fuerza decae rápidamente con la distancia, resuelve el problema que representa la recolección de datos el sistema para la asignación ponderada.

Antes de comenzar con la exposición del tema, es importante señalar que todo proceso computacional se puede valorar mediante:

- La cantidad de memoria que se requiere para el almacenamiento de la información.
- El número de ciclos de computación que se requieren para efectuar todas las operaciones programadas.
- Las unidades de tiempo que se consumen por los requerimientos de comunicación con otros procesos.

El segundo punto depende de la arquitectura del hardware y puede relacionarse entre las distintas plataformas mediante la multiplicación por una constante. Por otro lado es posible tratar con los ciclos de computación en un sentido más abstracto —independiente de la arquitectura— tal como lo hacen Papadimitrou y Yannakakis **[63]** en la presentación del modelo de gráficas acíclicas dirigidas.

Con la caracterización de los procesadores y tareas se procede a definir y calificar un algoritmo de asignación. El primer punto —que consiste de los requerimientos de memoria— es un dato necesario para determinar si el nodo remoto cuenta con los recursos suficientes para realizar la tarea. El segundo punto permite obtener el tiempo total

de ejecución como una cantidad escalar. Este dato permite comparar en términos absolutos el tiempo que tarda la resolución de la tarea en distintos procesadores. Es evidente que si sólo se considera este aspecto, un algoritmo que asigne la tarea con mayor requerimiento de ciclos al procesador con más ciclos disponibles por unidad de tiempo, terminará la ejecución en el menor tiempo posible. No obstante, generalmente el conocimiento del número de ciclos que son requeridos por las tareas no es un dato conocido. Además se deben sumar los tiempos implicados por los costos de comunicación.

Con este marco de referencia, podemos determinar un *tiempo mínimo* para la ejecución de un conjunto de tareas en paralelo. De todas las posibles combinaciones tarea-procesador, aquellas que produzcan tiempos mínimos de ejecución constituirán *óptimos*. La manera en que se define este tiempo óptimo se describe en la sección 3.3.

Para el cálculo de los tiempos de ejecución es necesario conocer los ciclos que requiere cada tarea. Este dato será utilizado con fines de evaluar el desempeño de los distintos algoritmos presentados.

En cuanto a la medición de los tiempos es importante la noción de la barrera de sincronización. A partir de ésta, las tareas en paralelo proceden tan rápido como les permita la cantidad de ciclos disponibles. Al llegar a la siguiente barrera de sincronización, las tareas esperan hasta que todas hayan llegado. Por ende el tiempo de ejecución de un superpaso está determinado por la tarea que termine al último. El tiempo total de la ejecución del programa paralelo es la suma de los tiempos de cada uno de los superpasos. En este trabajo consideramos el desarrollo de un algoritmo de asignación de tareas que tenga como consecuencia tiempos más cercanos al óptimo conforme crece el tamaño de la malla. Un algoritmo que se acerque más al óptimo será *mejor* para los efectos de la investigación.

2.2. Consideraciones generales

El método de campo de fuerza consiste en determinar el campo de fuerza determinado por el costo en las comunicaciones (análogo al

concepto física de la distancia) y por las cargas de las tareas y disponibilidad de recursos en los procesadores (análogo al concepto físico de cargas electrostáticas). El método mediante el cual se calculan estos parámetros varía según el tipo de malla con la que se está trabajando. En este capítulo planteamos un método para su cálculo y aplicación para mallas dinámicas.

A continuación describimos nuestra propuesta que que utiliza consideraciones de proximidad así como disponibilidad de ciclos computacionales para optimizar la asignación de tareas en una malla dinámica. Y evaluamos el mecanismo propuesto por medio del simulador DGS presentado en el capítulo 3

2.3. Modelo del campo de fuerza

2.3.1. Descripción del modelo. Una computadora BSP [**58, 78**] consiste de un conjunto de parejas procesador-memoria, una red global de comunicaciones y un mecanismo para la sincronización eficiente de barrera para los procesadores [**59**]. En el modelo BSP la ejecución paralela se divide en *superpasos*, cada uno de los cuales consiste de un número de hilos paralelos que pueden contener cualquier número de operaciones. Estos hilos efectúan únicamente comunicación local hasta que llegan a una barrera de sincronización donde toda la comunicación global se completa.

2.3.2. Asignación de tareas. Considérese la aplicación paralela ejecutando en la máquina BSP compuesta de n nodos paralelos. La aplicación consiste de m tareas paralelas a ser ejecutadas en el superpaso i. Para asignar tareas a procesadores al comienzo de un superpaso BSP, un esquema de balanceo de carga deberá ser utilizado. Estrategias recientes [**74**] han sido guiadas por la consideración de minimizar los costos de comunicación [**1, 37**] y la obtención de un balance de carga de ciclos de procesador [**69**]. Mientras que la estrategia presentada en [**74**] es efectiva con mallas pequeñas, el método del campo de fuerza presentada en este capítulo producirá resultados que se acercan más al

tiempo mínimo de ejecución, conforme crece el tamaño de la malla de cómputo, como se muestra en el capítulo 4.

Cuando se trata con el método del campo de fuerza, hay dos consideraciones a tomar en cuenta para determinar la asignación de tareas. La primera —llamada *carga computacional*— es la combinación del costo de memoria para la tarea con la capacidad de cómputo de la computadora remota. De la configuración global de la malla dinámica, todos los nodos que no dispongan de suficientes recursos —principalmente memoria— para procesar las tareas producirán una fuerza neta positiva, y por ende serán repulsivas y se eliminarán como candidatos.

Recordando que la asignación mediante el algoritmo de campo de fuerza se resuelve mediante la maximización de la fuerza de atracción:

$$\left|F(A_f(\overrightarrow{T}), \overrightarrow{T})\right| \geq \left|F(\overrightarrow{p}, \overrightarrow{T})\right| \quad \forall \overrightarrow{p} \in P : F(\overrightarrow{p}, \overrightarrow{T}) \leq 0$$

donde $F(\overrightarrow{p}, \overrightarrow{T})$ es la fuerza producida por la combinación de un vector de procesadores $\overrightarrow{p}$ con un vector de tareas $\overrightarrow{T}$, se considera una analogía con la Ley de Coulomb para la determinación de la fuerza. Por dicha ley se tiene que un campo eléctrico es la fuerza eléctrica por unidad de carga. La dirección del campo se toma en la dirección de la fuerza que ejercería sobre una carga positiva de prueba. El campo eléctrico es radialmente externo de una carga positiva y radialmente interno hacia un punto con carga negativa. El campo eléctrico de una carga puntual puede ser obtenida de la Ley de Coulomb:

$$E = \frac{F}{q} = \frac{kQq}{qr^2} = \frac{kQ}{r^2}$$

y el campo eléctrico de cualquier número de cargas puntuales puede ser obtenido de una suma vectorial de los campos individuales.

Pudiese parecer que un problema con este método es que el inverso proporcional del inverso cuadrado hace a las fuerzas desaparecer demasiado pronto, pero en realidad esta es la fortaleza del método que

permite realizar asignaciones desde distintos puntos de la malla, conformando cada uno su propia esfera de influencia. Como consecuencia los nodos "lejanos" sólo serán viables si tienen una gran capacidad de cómputo disponibles, de lo contrario serán imperceptibles.

Regresando al modelo de campo de fuerza para distribución de tareas dentro de una malla dinámica de computación, los datos que se requieren para el cálculo del campo de fuerza incluyen la cantidad de ciclos disponibles por unidad de tiempo en la computadora remota, ξ_j. Esta variable representa el potencial de cómputo que no está siendo aprovechado en un momento dado. Otro dato importante es λ_i, el cual representa el costo en ciclos implicados por los requerimientos de memoria de la tarea i. Este dato está asociado a la velocidad con la que una computadora puede asignar punteros de memoria dinámica y llevar la información de la memoria virtual a los registros de CPU. Este es uno de los cuellos de botella en la ejecución de programas de alto rendimiento.

$$\text{costo computacional} = C_{ij} = \frac{\xi_j}{\lambda_i}$$

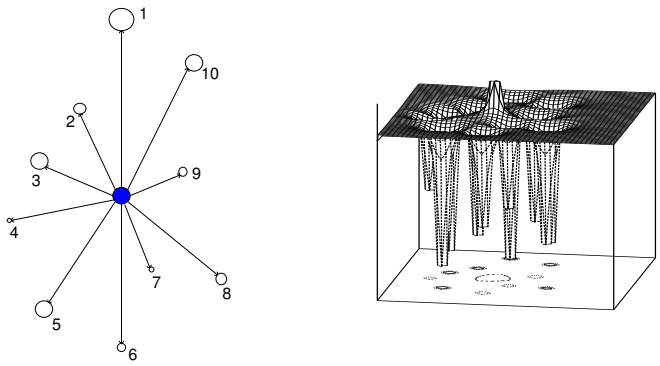
El producto de ξ_j con $1/\lambda_i$ será lo que determine el numerador en la fórmula para la determinación de la fuerza en una malla dinámica, y en términos generales representa el costo de computación para una tarea i en un procesador j.

El producto de ξ_j con $1/\lambda_i$ se representa por el tamaño del círculo en la figura 2.3.1(a) y se analiza con más detalle en la sección 2.4.3. Nótese que el algoritmo canónico voraz respecto a ciclos de cómputo asignará la primera tarea a la máquina con el mayor número de ciclos disponibles por unidad de tiempo, lo cual no es necesariamente la mejor alternativa ya que no se toma en consideración el costo en ciclos de cómputo asociado a los requerimientos de memoria de la tarea a ser asignada.

La segunda consideración de importancia es el costo en las comunicaciones. Este factor se trata con una fórmula de cuadrado inverso. Esto toma en consideración la disponibilidad de ancho de banda y la

latencia . En la figura 2.3.1(a) el costo en la comunicación se representa por la distancia del nodo central de asignación a cada una de las computadoras remotas —numeradas del 1 al 10. El algoritmo canónico voraz respecto a costo de comunicaciones asignará la primera tarea a la máquina con el menor costo en las comunicaciones, es decir, el nodo en el círculo más pequeño.

La asignación del método de campo de fuerza no es evidente de la figura 2.3.1(a) porque ambos, el tamaño y la distancia de los círculos, son tomados en consideración para determinar el efecto neto. La novedad del método del campo de fuerza estriba en la manera en que ambos, la proximidad en la red y las consideraciones de ciclos de cómputo disponibles, se conjuntan en un solo campo de fuerza para determinar cual nodo recibirá la tarea. La figura 2.3.1(b) muestra una representación tri-dimensional del campo de fuerza determinado por los nodos de la figura 2.3.1(a). La pendiente del campo de fuerza determinará en que dirección la tarea *rodará* para llegar a un nodo en particular, representado por una depresión en la superficie del campo de fuerza. La fuerza positiva indica que el nodo donde las tareas se asignan. El caso de migración de tareas no se trata en este estudio y se deja el campo abierto para investigación posterior.



(a) Nodos de la malla (b) Campo de fuerza

FIGURA 2.3.1. Consideraciones para la asignación de tareas

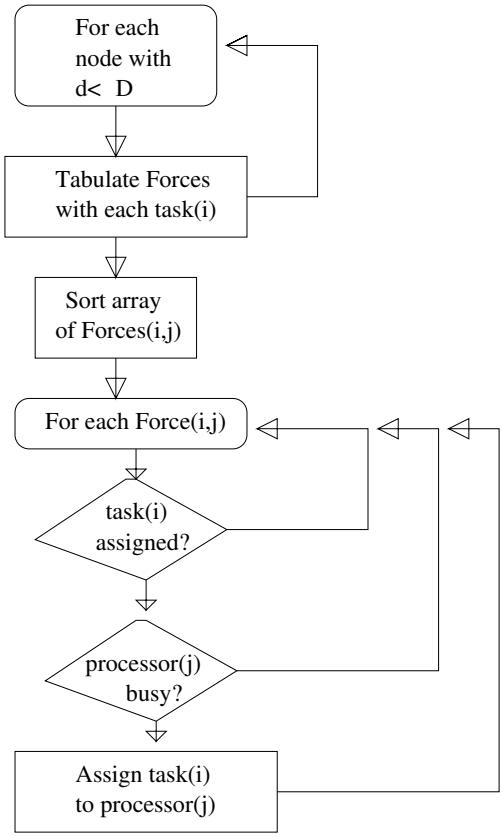


FIGURA 2.3.2. Algoritmo de asignación de tareas

El método de campo de fuerza para la sincronización opera de la siguiente manera: Al llegar a la barrera de sincronización, cada nodo envía un mensaje de terminación a todos los procesadores participando en el superpaso. De éstos, uno estará actuando como maestro de sincronización (el encargado de distribuir las tareas al inicio del superpaso). La selección del maestro dependerá de cual nodo participante en el superpaso es el primero en llegar a la barrera de sincronización. El maestro de sincronización determinará el campo de fuerza y asignará las tareas a donde el campo de fuerza es más fuerte según el algoritmo que se muestra en la figura 2.3.2.

Para calcular las fuerzas, el maestro de sincronización requiere conocer los ciclos de cómputo y memoria disponibles en cada nodo remoto

además del costo efectivo de comunicación. Es de notar que este nodo central no establece un marco de referencia absoluto ya que la fuerza decaen de manera inversamente proporcional a la distancia, lo cual limita este marco de referencia a una zona de la malla.

En la barrera se completan todas las comunicaciones y el sistema llega a un estado estable. Los valores resultantes del campo de fuerza se ordenan y los nodos con los valores más negativos (fuerza de atracción) reciben las tareas. Valores positivos del campo de fuerza indican una repulsión neta para la asignación de nuevas tareas y se generan cuando la memoria disponible del nodo es inferior a los requerimientos netos de memoria de la tarea.

2.3.3. Diseño. Al aplicar el modelo de computación BSP existe un importante aspecto que no debe ser ignorado. Todas las tareas deben sincronizarse en la barrera BSP y toda la comunicación tiene lugar en este punto. El costo de la comunicación relativa al costo de la computación es importante. Conforme la velocidad de los procesadores aumenta a una más rápida que la del equipo de comunicaciones, la razón entre las dos tenderá a aumentar [**84**]. Las aplicaciones en paralelo tales como el análisis de la información [**74**] ya tienen un mayor costo en las comunicaciones y pueden beneficiarse del método de asignación del campo de fuerza.

2.3.4. Mallas dinámicas. Mientras que la oportunidad de realizar cómputo de gran escala a un bajo costo es una clara motivación para la utilización de recursos de escritorio para el procesamiento en paralelo, la *variabilidad* de los recursos en un momento dado es el obstáculo a vencer en el diseño de una estrategia efectiva para la asignación de tareas. Los principales puntos que caracterizan una malla dinámica son la topología efectiva de red y las capacidades computacionales de los nodos individuales de cómputo. La topología efectiva de red se determina de manera instantánea de la latencia y ancho de

banda disponible en la malla dinámica. Mientras que esta topología está limitada por el hardware con el cual está ensamblada la malla, los patrones de uso de las aplicaciones fuera del contexto BSP determinarán la topología efectiva disponible a la máquina BSP y el costo de comunicaciones entre cualesquiera dos puntos dentro de la malla.

La proximidad de red es la cantidad de ciclos de cómputo que se requieren para establecer la comunicación entre cualesquiera dos nodos. Es decir, si $Cy(t)$ es la función que determina el número de ciclos en función del tiempo (esta relación generalmente es linea) y t es el tiempo que se requiere para establecer la comunicación, entonces

$$\text{proximidad} = \int_0^t Cy(t)dt.$$

Este parámetro está determinado por el equipo de comunicaciones.

La capacidad de red es la cantidad de información que puede ser transmitida entre nodos una vez que la comunicación ha sido establecida. Dado que la utilización de la red no es uniforme ni estática, la capacidad de red entre cualesquiera dos nodos no es necesariamente uniforme ni constante a través del tiempo. En este caso si $B(i, j, t)$ es la capacidad instantánea de la red entre el nodo i y el nodo j, entonces para el intervalo de tiempo $[0, t]$ la capacidad total de transmisión de datos será

$$\text{capacidad} = \int_0^t B(i, j, t)dt.$$

2.4. Diseño del algoritmo

Una consideración importante con respecto a las mallas dinámicas que se debe asumir, es que el número de tareas paralelas a ser completadas durante un superpaso es menor o igual al número de procesadores disponibles en la malla —aunque en caso contrario, tratándose de una malla dinámica, los procesos esperan hasta que haya el suficiente número de procesadores para ser asignados y comenzar su ejecución.

En el diseño de un esquema mejorado para la asignación de tareas bajo el modelo BSP con costos, se toma el concepto físico de campo

de fuerza para elaborar el algoritmo propuesto. Al hacerlo, se parte de un hecho importante: durante los últimos años la velocidad computacional ha visto aumentos a una tasa geométrica, mientras que la velocidad de las comunicaciones se ha incrementado únicamente de manera lineal **[65]**. Esto indica que las tendencias en la computación en paralelo del futuro podrían estar seriamente influenciadas por las comunicaciones.

Consideremos que los costos en ciclos computacionales para cada tarea en un superpaso BSP no necesariamente son iguales. Si C_i es el costo en ciclos para resolver una tarea en particular i, y ξ_j es la cantidad de ciclos disponibles por unidad de tiempo en el nodo j, entonces el tiempo para resolver la tarea i puede estar representado por C_i/ξ_j.

Sin considerar el costo en las comunicaciones, al tratar con una malla dinámica de tamaño m, el óptimo para n tareas paralelas es el mínimo del conjunto obtenido por la asociación de tareas a procesadores. Si el costo en comunicaciones se considera, la complejidad del problema aumenta de manera dramática. El conjunto de todos los superpasos en un problema BSP se vuelve similar a una determinación de una DAG, el cual es un problema NP-difícil.

Si un problema NP-completo puede ser resuelto, entonces el procedimiento de resolución puede ser transformado por un número finito de pasos en un procedimiento de resolución para cualquier problema de la clase NP. Mientras esto significa encontrar la solución analítica exacta, se debe reconocer que esto puede ser inalcanzable. No obstante, para una gran subclase de problemas NP-completos existe la posibilidad de encontrar métodos aproximados. No hay nada nuevo en esto: la necesidad de soluciones aproximadas ha sido identificada desde hace mucho tiempo. La diferencia con el enfoque presentado en este trabajo se detalla en la sección 2.4.1.

Mientras que el problema de calendarización de tareas es NP-difícil, muchos autores se han enfocado en el aspecto computacional **[62]**. Por otro lado, en este trabajo se toma una solución de otra rama de la Ciencia y se aplica con buenos resultados.

2.4.1. Leyes fundamentales del movimiento. Las fuerzas en la Naturaleza determinan cómo se mueven los objetos a través del espacio-tiempo y se caracterizan por campos de fuerza. No existe razón para descartar un campo de fuerza como un medio efectivo para la asignación de tareas, con las restricciones apropiadas al modelo, a menos de que uno no esté dispuesto a evaluar los resultados y consecuencias de dicha proposición. Mientras que las tareas son entidades abstractas sin ninguna masa física, estarán sujetas a un movimiento en su propio espacio abstracto. A continuación se trata de la manera en que se modifica el punto de vista de la Física Clásica a las necesidades de la Ciencia de la Computación.

Para desarrollar un algoritmo de balanceo de carga mediante una asignación adecuada de tareas emergentes en el ámbito del campo de fuerza, considérese que cualquier tarea permanecerá sin asignación a menos de que experimente una fuerza. Este enunciado también es conocido como el Principio de Galileo o la Primera Ley de Newton. La determinación de la fuerza que actúa sobre una tarea y causa que el proceso se mueva dentro de la malla dinámica esté dada por la ley del cuadrado inverso —usada en la Ley de la Gravitación Universal (2.4.1) y la Ley de Coulomb para las fuerzas electrostáticas (2.4.2).

Un esquema exitoso para el balanceo de carga que utiliza fuerzas de gravitación puede ser visto en Hui y Chanson **[45]**, aunque en este caso la fuerza de la gravedad está oculta bajo la teoría de la hidrodinámica. Este método de balanceo de carga ha encontrado aplicación con el modelo PRAM de computadora paralela. Un modelo que es menos dependiente de la arquitectura del sistema, tal como BSP, amerita una expresión más sencilla de las leyes que gobiernan la asignación de tareas. Enseguida se observa la Ley de la Gravitación Universal. La explicación de lo que significan las variables y la manera en que se hace una analogía al mundo abstracto de la asignación de tareas en cómputo paralelo se detalla *ut infra*.

$$F = G\frac{m_1 m_2}{r_{12}^2}. \tag{2.4.1}$$

Una desventaja al uso de la Ley de la Gravitación Universal por el método hidrodinámico para balanceo de carga es la ausencia de fuerzas repulsivas. El fluido hidrodinámico no puede desplazarse de un nivel más bajo a uno más alto. Sólo con fuerzas repulsivas puede un nodo rechazar tareas donde los requerimientos sobrepasen los recursos disponibles. Es de mencionar que la repulsión de tareas no se dará como un capricho del usuario, más bien se trata de un método para evitar la asignación da tareas a procesadores que carecen de los recursos necesarios para su ejecución exitosa.

Por otro lado, el modelo BSP permite la aplicación directa de la Ley de la Gravitación Universal. No obstante, las fuerzas de gravedad sólo son atractivas. Por ende, la Ley de Coulomb (2.4.2) es de mayor interés en el trabajo que aquí se desarrolla.

$$F = K\frac{q_i q_j}{r_{ij}^2}. \tag{2.4.2}$$

Al aplicar la ecuación (2.4.2), $F < 0$ implica una atracción o tendencia a mover una tarea hacia el nodo remoto, mientras que $F > 0$ implica repulsión.

¿Cómo cabe esto dentro del modelo BSP? En la barrera de sincronización el sistema llega a un estado bien definido en el cual las fuerzas atractivas y repulsivas asociadas a las tareas emergentes pueden ser determinadas. Las computadoras de la malla dinámica participando en la máquina BSP que están temporalmente fuera del contexto de la máquina (como cuando haya que efectuar tareas administrativas de mantenimiento propias del nodo) tan sólo necesitan cambiar el signo a su capacidad computacional. Con ello todas las nuevas tareas son repelidas y enviadas a otros nodos que se encuentren disponibles en la malla dinámica.

Más aún, la aplicación de la ecuación (2.4.2) bajo el modelo BSP provee la condición para la recuperación de error. Esto es porque sólo los nodos que hayan fallado y *sólo* aquellas que han fallado, exhibirán una fuerza nula, $F = 0$[2].

Cuando el programa paralelo llega a la barrera de sincronización, cualquiera de los nodos participantes puede actuar como maestro de sincronización. El sentido común indica que el primer nodo en llegar a la barrera será el maestro de sincronización para esa barrera en particular. En este punto se medirá la fuerza con los nodos restantes, uno por uno. Y si en el proceso de esta medición, cualquier nodo participante tiene $F = 0$ —costo infinito de comunicación—, el maestro de sincronización puede adicionar la tarea del nodo fallido a las propias, ceder el papel de maestro de sincronización, y permitir que el proceso paralelo se recupere de la falla del nodo remoto.

2.4.2. La distancia computacional. Para encontrar el equivalente físico para la distancia en el problema de asignación de tareas, se debe calcular el costo en las comunicaciones. Considere el caso de efectuar asignación de tareas en una máquina paralela BSP compuesta por una red de computadoras utilizando una red de uso compartido y múltiple. Con esta consideración, el ancho de banda disponible entre cualesquiera dos nodos será variable en el tiempo. Lo que es importante no es el valor instantáneo en un momento dado sino el valor promedio en el intervalo donde se localice el valor instantáneo.

2.4.3. La carga computacional. Al tratar con las cargas computacionales, el método de campo de fuerza se refiere a dos en específico. El primero yace dentro de la computadora remota y el segundo dentro de la carga a ser asignada.

Para la determinación de la fuerza de la ecuación (2.4.2), los valores de las cargas deben ser conocidos.

La carga del nodo remoto es el potencial computacional del nodo j, y es equivalente a los ciclos computacionales por unidad de tiempo

[2]siendo que el costo en comunicación es infinito, $r_{ij} = \infty$.

disponibles. Este número siempre es positivo o igual a cero. Si $q_j = 0$, entonces no existe ni atracción ni repulsión y el nodo no aceptará tareas. El hecho de que el nodo reciba o no una tarea depende de los valores del campo de fuerza para otros nodos en la malla.

2.4.4. Carga de la computadora remota. Representado por q_j, es el potencial computacional del nodo j, y es equivalente a los ciclos computacionales disponibles por unidad de tiempo. Este número siempre es positivo o igual a cero. Si el nodo j fija $q_j = 0$, entonces la atracción es nula. El hecho de que el nodo reciba o no una tarea dependerá del campo de fuerza determinado para todos los elementos de la malla.

2.4.5. Carga de la tarea. Representado por q_i, es la *facilidad* computacional asociada a los requerimientos de memoria de la tarea. La facilidad computacional —inversamente proporcional al costo (facilidad $= \frac{1}{\text{costo}}$)— es un parámetro importante que en muchas ocasiones no se le presta la atención debida: uno de los principales cuellos de botella en la ejecución de la mayoría de las tareas no yace en la velocidad del procesador sino en el movimiento de datos entre almacenaje de memoria.

En otras palabras, si la memoria disponible en el nodo j es menor que la requerida para la tarea i, entonces a q_i se le asigna un signo positivo, asegurando así una fuerza repulsiva. De otra forma el signo será negativo. Si c_i es el costo en ciclos de computación que implican los requerimientos de memoria de la tarea i, entonces $q_i = \frac{1}{c_i}$ es la carga asociada a la tarea[3]. Nótese que los requerimientos de memoria de todas las tareas tienen que ser conocidas a cualquier método de asignación que no produzca rechazos. El costo en ciclos que serán requeridos para la tarea no se utiliza en ningún momento. En la siguiente sección se hace una evaluación del desempeño del método de campo de fuerza

[3]Una distinción más puede hacerse si la memoria disponible está totalmente en RAM o parcialmente distribuida en almacenaje de disco. Este refinamiento no fue considerado en esta investigación.

con la utilización del simulador descrito en el capítulo 3. El simulador (DGS) genera valores aleatorios de las variables de la malla dinámica con concordancia con la Ley de los Grandes Números.

En resumen, siendo que C_{jk} es el costo en las comunicaciones entre el nodo receptor j y el nodo asignador k calculado en base a las consideraciones de proximidad y capacidad de la red, se tiene que la fuerza está dada por

$$F = \delta(i)\frac{q_i q_j}{C_{jk}^2}$$

donde $\delta(i)$ es el signo resultante de la resta de los requerimientos de memoria de la tarea i y los recursos de memoria del nodo j.

Al aplicar la ecuación (2.4.2), $F < 0$ implica una atracción o tendencia a mover una tarea hacia el nodo remoto, mientras que $F > 0$ implica repulsión. Con la asignación de campo de fuerza, de acuerdo a la mayor fuerza de atracción, se irán asignando las tareas.

Es de notar que la fuerza de repulsión podría ser omitida ya que una fuerza nula implica la no asignación de una tarea. Sin embargo esta consideración limitaría el método a un campo de fuerza escalar y se tiene pensado que la metodología se puede aplicar en un campo de fuerza vectorial. Para ello es necesario realizar sumas vectoriales y de tal manera dirigir la asignación hacia zonas donde la atracción es más fuerte y alejarlas de zonas saturadas. La consideración de fuerzas negativas es importante en este aspecto y el método exacto para su calculo requiere de más consideraciones que las presentadas en esta tesis.

2.5. Recolección de información

En la implementación del campo de fuerza, uno de los obstáculos a vencer consiste de la recolección de la información. Se podría argumentar que el método de asignación aleatoria, no obstante no tiene tan buen rendimiento como el método de Campo de Fuerza, éste primero presenta la ventaja de que no requiere de recolección de información. La única desventaja es que produce rechazos. Por ello esta pérdida de

tiempo puede considerarse como el costo de la recolección de información en el método de asignación aleatoria.

Sea t_i el costo en tiempo de recolección de un paquete de información del nodo i hacia el nodo central de asignación. Dado que tratamos con una red dinámica, el costo para cada nodo i será distinto.

La resolución del problema de recolección de información se hace de la siguiente manera. Dado que el método de campo de fuerza determina la asignación según una ley de cuadrado inverso al momento de estar recolectando la información existe un tiempo T de corte, el cual se determina de manera dinámica.

Es decir, van llegando los paquetes de información de la malla y según llegan se van procesando. Se puede suponer que existe una distribución normal gaussiana para los ciclos disponibles de los procesadores remotos, al igual que en los costos de comunicación.

De la información que ha llegado al tiempo $t < t_c$ se puede reconstruir en el servidor central la distribución normal para la disponibilidad de cómputo para los procesadores remotos e inferir cuáles serán los valores que faltan. La ubicación en el espacio de la malla de estos valores faltantes no es de importancia, ya que sabemos que todos están a más de una cierta distancia en cuanto al costo en las comunicaciones. De esta manera, con un número adecuado de puntos, se puede estimar el valor en la media de la distribución normal para los ciclos de cómputo disponibles en la malla dinámica.

De las tareas a asignar en el superpaso tenemos los costos por utilización de memoria que determina q_1 para cada elemento del conjunto de tareas. Los valores contenidos en los mensajes que arriban de la malla proporcionan q_2 para los nodos conocidos. De los restantes nodos podemos —de la distribución normal— acotar las q que faltan por una q_{max}: todos los nodos cuyo paquete de información no ha arribado tendrán un tiempo de comunicación mayor que t_c.

A partir de t_c y el tamaño de los paquetes de información se puede reconstruir la cota para los costos en comunicación, d_{max}. Por ende, para todos los nodos cuya información no ha llegado, el valor absoluto

del campo de fuerza es menor a

$$F \leq F_c = \frac{q_1 q_{max}}{d_{max}}.$$

donde q_1 es la carga asociada a la tarea de mayores requerimientos de memoria.

Se deben asignar tareas de 1 a n. Para todas estas se obtienen los valores F_i. De estos se obtiene un conjunto de las mayores fuerzas F_i. Cuando el ínfimo de este conjunto es mayor o igual a F_c, es obvio que es inútil seguir esperando paquetes de información y el superpaso puede proceder con la asignación de tareas.

Si bien al aumentar el número de nodos de la malla dinámica, aumenta el tiempo necesario para la recolección, también disminuye el valor de t_c y las tareas pueden ser asignadas más rápido.

La verificación de este argumento se presenta en la sección 4.4.

2.6. Observaciones

No obstante que la motivación física detrás del método de asignación de tareas mediante campo de fuerza no sea una consecuencia formal de la Ciencia de la Computacional, en la práctica las analogías hechas con las leyes de la Física convergen a una buena estrategia para la asignación de tareas en el caso de una red dinámica con tamaño creciente.

El efecto que las comunicaciones tiene en el tiempo requerido para completar un superpaso BSP es un factor que no debe ser ignorado cuando se trata con mallas de tamaño creciente. El efecto de las comunicaciones en el cómputo paralelo tendrá un impacto incrementalmente mayor conforme la velocidad de los procesadores supera el crecimiento en la velocidad de las comunicaciones.

Las estrategias de asignación de tareas que tomen ambos factores en consideración, tal como el algoritmo del campo de fuerzas, deben ser preferidas a sus contra-partidas voraces.

En el desarrollo de este algoritmo hemos considerado el campo de fuerzas como un campo escalar. No obstante, el análisis como un campo

vectorial podrá producir mejores resultados, permitiendo un balance de carga más homogéneo de la malla dinámica. Aún existe un campo abierto para la investigación en esta dirección.

CAPíTULO 3

Simulación de la malla gaussiana

El simulador de mallas dinámicas (DGS, por sus siglas en inglés) genera condiciones de carga y disponibilidad en una malla dinámica. Como una herramienta para evaluar el rendimiento de estrategias de asignación de tareas en mallas dinámicas de cómputo, el DGS puede ser utilizado para obtener evaluaciones de distintas características de malla, variando la cantidad de procesos en ejecución paralela. Este capítulo describe la construcción de este simulador. El rendimiento de varias estrategias de asignación de tareas se evalúan y comparan, incluyendo un caso de estudio para la resolución de la ecuación de onda para tomografía sísmica de pozos en el capítulo 4.

Las contribuciones principales presentadas en este capítulo y de consecuencia en el resto de esta tesis, son las siguientes:

- Describimos un algoritmo para la construcción de un simulador que considera la proximidad y ancho de banda así como la disponibilidad de ciclos computacionales.
- Evaluamos varios esquemas de asignación de tareas mediante el simulador y analizamos los resultados.
- Mostramos cuáles estrategias tienen mejor desempeño conforme el tamaño de la malla dinámica se incrementa y la cantidad de hilos paralelos se varía.

El resto del capítulo se organiza de la siguiente manera. La sección 3.1 da una introducción al los simuladores de mallas de cómputo, describiendo el estado del arte y las razones por la cuales fue necesario desarrollar un nuevo simulador —no descrito anteriormente en la literatura— para verificar el desempeño del método de asignación

de tareas por Campo de Fuerza, presentado en el capítulo 2. La sección 3.2 proporciona una descripción abstracta de los algoritmos de asignación de tareas y describe de manera concisa los esquemas utilizados para efectos de comparación de desempeño con el método de Campo de Fuerza. Por último, la sección 3.3 describe la implementación del código del simulador, incluyendo la descripción de obtención de parámetros y descripción de los pasos realizados por el código.

3.1. Estado del arte

La construcción de un simulador para reproducir resultados experimentales no es una tarea difícil. La intención presentada por el diseño del DGS es la de construir un simulador basada en un reconocido principio matemático (La Ley de los Grandes Números) para iluminar el camino para aprovechar los recursos computacionales que estarán disponibles en el futuro cercano con las mallas compuestas por un gran número de computadoras de escritorio ordinarias.

Algunos simuladores están disponibles en la actualidad para efectuar pruebas sobre la asignación de tareas en un ambiente de mallas. Estos incluyen MicroGrid [**73**], SimGrid [**16**], GridSim [**14**], y más recientemente GangSim [**23**].

El simulador MicroGrid [**73**] desarrollado en la Universidad de California es un *toolkit* específico para la arquitectura de un ambiento controlado de tipo Globus [**26**]. Por ende está enfocado hacia las aplicaciones construidad con Globus. Un importante aspecto de MicroGrid es la habilidad para verificar resultados de simulación bajo el ambiente Globus. El simulador no hace ataca el problema de examinar la ejecución de una aplicación paralela cuando la sincronización de barrera es de la mayor importancia, ni tampoco hace consideraciones matemáticas al determinar los parámetros de caracterización de mallas grandes.

El *toolkit* SimGrid, también desarrollada en la UCSD, es un sistema poderoso que permite la creación de tareas en término de sus tiempos de ejecución y recursos con respecto a la capacidad de una máquina referencia. Las tareas pueden ser asignadas de acuerdo al esquema

bajo prueba. No obstante, SimGrid se limita a una sola ente que hace las asignaciones y es difícil simular múltiples usuarios compitiendo en un ambiente de malla dinámica. No se hacen consideraciones de tipo matemáticas para la determinación de parámetros en mallas de gran tamaño para simular el rendimiento de una aplicación en paralelo.

GridSim [**14**] es un *toolkit* Java que cubre muchas de las limitaciones presentadas por SimGrid. GridSim provee un conjunto para simular distintas clases de recursos heterogéneos, usuarios, aplicaciones y asignaciones. El enfoque, en este caso así como en los simuladores previos, es hacia el cómputo distribuido y no las aplicaciones en paralelo. Con GridSim, todos los usuarios deben enviar sus tareas a un administrador central, el cual puede ser enfocado para realizar una optimización global o una satisfacción de usuarios dependiendo de la política de asignación de recursos o la optimización para usuarios de alta prioridad.

Varias limitaciones de GridSim pueden ser notadas. La asignación solo puede ser realizada desde el administrador central y no se hacen consideraciones para simular un ambiente donde las asignaciones se hacen de puntos múltiples a la malla con una topología de red en constante cambio —una de las características fundamentales de la malla dinámica. Más aún, mientras que el Java es una herramienta poderosa para la construcción de aplicaciones de alto nivel, un enfoque de más bajo nivel es más fácil de programar en un lenguaje tal como C y provee una ejecución más rápida. Este último detalle es fundamental para escalar el simulador a tamaños grandes de malla.

En una de las adiciones más recientes a la colección de simuladores de mallas, GangSim [**23**] se enfoca hacia la asignación de tareas con políticas de asignación de recursos adoptadas por sitios y organizaciones virtuales. Este simulador modelo no sólo los sitios sino que también a los usuarios de las organizaciones virtuales y es capaz de modelar políticas de utilización. El escenario donde las organizaciones virtuales aparecen y desaparecen, y donde las políticas de asignación

de recursos no son constantes no se cubre por el estado de desarrollo actual de GangSim.

El simulador de mallas de escritorio, DGS, busca cubrir el terreno no incluido por las excelente herramientas que han sido desarrollada hasta la fecha. Por esta razón una comparación de resultados producidas por el DGS y los mencionados previamente no se incluye en este trabajo de tesis: la determinación de parámetros es fundamentalmente diferente.

Adicionalmente a este último punto, los simuladores previos se han presentado como *toolkits*, mientras que el DGS se presenta como un conjunto de reglas con los cuales generar los parámetros de la malla dinámica y proceder con las políticas de asignación de tareas: el rango de aplicabilidad para el DGS es para programas paralelos bajo el modelo BSP. Estas reglas fueron programadas en lenguaje C por las conveniencias que presenta, pero pueden ser utilizadas para extender GridSim, por ejemplo. La novedad del enfoque es la utilización del la herramienta matemática implicada por la Ley de los Grandes Números para definir los parámetros de la malla. Una desventaja de esta metodología es que sólo es aplicable a mallas grandes. Una malla compuesta de unas cuantas docenas de cúmulos científicos con 100K procesadores ciertamente no estaría correctamente modelada en estas suposiciones. Por el otro lado, el número creciente de computadoras de escritorio que están siendo integradas al Internet día con día es un enfoque distinto: esta es la motivación para la construcción del simulador DGS.

Como una herramienta para evaluar el rendimiento de estrategias de asignación de tareas para mallas dinámicas, el DGS puedes ser utilizado para obtener evaluaciones comparativas de distintos tamaños de mallas y número de hilos paralelos.

Las reglas definidas por el DGS para generar los parámetros de la malla podrían ser incorporadas a cualquiera de los simuladores previos. No obstante, el propósito del simulador en este trabajo de tesis no es la extensión de simuladores existentes para complicarlos, más bien el

propósito es el desarrollar los procedimientos para tratar con mallas dinámicas de gran tamaño.

Es fundamental notar que el propósito de DGS *no* es la obtención de *benchmarks* en base a alguno de los esquemas reconocidos internacionalmente para este propósito. El fundamento del DGS es la obtención de cifras comparativa de rendimiento entre distintos esquemas de asignación de tareas. Por esta razón es indispensable la ejecución del simulador con por lo menos dos o más métodos de asignación de tareas con los mismos parámetros de la malla para cada corrida con el fin de obtener resultados relativos entre los métodos de asignación.

La dificultad técnica que existe para construir un simulador para la malla dinámica yace en la gran cantidad de computadoras individuales que componen la malla y para los cuales los parámetros de los recursos deberán ser generados a cada intervalo de tiempo para caracterizar el estado del sistema. Aunque la Ley de los Grandes Números nos indica que el método para estimar los parámetros es directo, el efectuarlo para mallas de gran tamaño es difícil dado que el simulador debe mantener el estado de la malla en memoria. Esto involucra una estructura de datos para cada nodo en la malla dinámica. Más aún, para la obtención de resultados que son matemáticamente coherentes con la presunción de Ley de los Grandes Números, las simulaciones se deberán efectuar de manera repetitiva para conformar una gran población de resultados de los cuales filtrar conclusiones estadísticas. Para ilustrar la dificultad involucrada, este trabajo incluye los resultados de una simulación de una malla dinámica de 50,000 nodos cuya ejecución en un cúmulo —bajo las condiciones descritas más adelante— dilató más de tres semanas de tiempo real en el capítulo **??**.

Una de las características más prominentes de las mallas dinámicas es su tendencia para crecer no sólo en tamaño, sino también en rendimiento y capacidad de memoria. Un esquema de asignación de tareas debe tomar este hecho en consideración para extender la utilidad de los algoritmos por un lapso razonable de tiempo.

La necesidad para el desarrollo de las reglas mediante las cuales la simulación debe establecer los parámetros sigue del desarrollo de método de asignación por Campo de Fuerza [**30**]. Un programa paralelo utilizando este método de asignación debe tener acceso a la malla en su totalidad, pero sólo utilizará una pequeña porción en cualquier momento debido a las limitaciones de la Ley de Amdahl [**40**]. La malla debe ser separada del efecto individual que los usuarios tengan sobre los parámetros: estos usuarios individuales también incluirán aquellos que estén ejecutando un programa paralelo bajo el modelo BSP con un número pequeño de hilos por superpaso (el pequeño es relativo al tamaño total de la malla). Esto se modelo de la mejor manera por la Ley de los Grandes Números.

3.2. Estrategias de asignación

El caso en que uno o más nodos remotos que están ejecutando alguna tarea en particular falle, no está considerado en el simulador. Esto es por que el propósito de la herramienta es la de producir una comparativa entre distintos métodos de asignación y la falla en realización de tareas afectarán a los distintos métodos de asignación de manera equitativa y por ello no tendrán efecto en los resultados relativos. Ninguno de los métodos utilizados para la comparativa utiliza la duplicación de tareas. La utilización de duplicación de tareas puede ser utilizado como una herramienta complementaria por cualquier método de asignación y podría ser utilizado para extender el simulador en el futuro.

En muchos casos, los requerimientos adicionales en comunicación y computación por la recolección de información para hacer que funcione un esquema de tarea-procesador, puede exceder los beneficios que otorga la estrategia de asignación. A continuación se hace un análisis de este problema y se determinan las condiciones cuando una estrategia de asignación es favorable o no.

Sea $\mathbb{P} = \{P_1, P_2, , \ldots\}$ un conjunto de procesadores y $\mathbb{T} = \{T_1, T_2, , \ldots\}$ un conjunto de tareas. Ambos conjuntos son finitos. Sea

$$T^{(m)} = \{(T_1, T_2, \ldots, T_m) \quad | \quad \forall i \leq m : T_i \in \mathbb{T} \quad \& \\ \forall i, j \leq m : i \neq j \Rightarrow T_i \neq T_j\}$$

el conjunto de los vectores de dimensión m de tareas distintas a pares. Sea

$$P^{(m)} = \{(P_1, P_2, \ldots, P_m) \quad | \quad \forall i \leq m : P_i \in \mathbb{P} \quad \& \\ \forall i, j \leq m : i \neq j \Rightarrow P_i \neq P_j\}$$

el conjunto de los vectores de dimensión m de procesadores distintos a pares. Sea $A : T^{(m)} \rightarrow P^{(m)}, \overrightarrow{T} \mapsto A(\overrightarrow{T}) = \overrightarrow{P}$ una función de asignación de tareas.

Sea

$$\tau_0 : \mathbb{T} \times \mathbb{P} \rightarrow \mathbb{R}; (T, P) \mapsto t$$

donde t es el tiempo de ejecución de T en P. Sea

$$\tau_m : T^{(m)} \times P^{(m)} \rightarrow \mathbb{R}; (\overrightarrow{T}, \overrightarrow{P}) \mapsto \tau_m(\overrightarrow{T}, \overrightarrow{P}) = \sum_{i=1}^{m} \tau_0(T_i, P_i).$$

Para efectos de notación, escribiremos:

$$\overrightarrow{T} \bullet \overrightarrow{P} := \langle \overrightarrow{T} | \overrightarrow{P} \rangle := \tau_m(\overrightarrow{T}, \overrightarrow{P}).$$

Sea $\mathbb{A}_m$ —en un sentido abstracto— el conjunto de los algoritmos de asignación de tareas. Para cada método de asignación $A \in \mathbb{A}_m; A : T^{(m)} \rightarrow P^{(m)}$ definimos

$$\tau_m^A : T^{(m)} \rightarrow \mathbb{R}; \overrightarrow{T} \mapsto \tau_m^A(\overrightarrow{T}) = \overrightarrow{T} \bullet A(\overrightarrow{T})$$

Además se tiene el costo para la ejecución del algoritmo de asignación: $\sigma : \mathbb{A}_m \rightarrow \mathbb{R}; A \mapsto \sigma(A)$.

Finalmente se tiene el costo total para la ejecución del superpaso bajo el esquema de asignación de tareas A:

$$\upsilon_A : T^{(m)} \rightarrow \mathbb{R}; \overrightarrow{T} \mapsto \overrightarrow{T} \bullet A(\overrightarrow{T}) + \sigma(A).$$

Definición: Se dice que un algoritmo $A : T^{(m)} \rightarrow P^{(m)}$ es *inviable* si

$$\forall \overrightarrow{T} \in T^{(m)} \forall \overrightarrow{P} \in P^{(m)} : \upsilon_A(\overrightarrow{T}) > \overrightarrow{T} \bullet \overrightarrow{P}.$$

Definición: Se dice que un algoritmo $A : T^{(m)} \rightarrow P^{(m)}$ es *viable* si no es inviable. Es decir:

$$\exists \overrightarrow{T} \in T^{(m)}, \exists P \in P^{(m)} : \upsilon_A(\overrightarrow{T}) \leq \overrightarrow{T} \bullet \overrightarrow{P}.$$

El hecho de que un algoritmo A sea viable no quiere decir que su aplicación será favorable, tan solo indica que puede serlo.

Sea I_m la imagen de τ_m:

$$I_m = \left\{ r \in \mathbb{R} | \exists \overrightarrow{T} \in T^{(m)}, \overrightarrow{P} \in P^{(m)} : r = \overrightarrow{T} \bullet \overrightarrow{P} \right\}.$$

$I_m \in \mathbb{R}$ es finito y su función característica o su problema de decisión conlleva una cierta complejidad. Un número real $\mu \in \mathbb{R}$ es umbral de A si $\exists r \in I_m : \mu \leq r$.

*Definición:*Se dice que $A : T^{(m)} \rightarrow P^{(m)}$ es $\mu-buena$ [1] si

$$\forall \overrightarrow{T} \in T^{(m)} : \upsilon_A(\overrightarrow{T}) < \mu.$$

Muchas estrategias de asignación, aunque sean viables, tienen la característica de tener un umbral μ demasiado alto cuando se aplican a las condiciones de las mallas estáticas —particularmente para super computadoras de memoria compartida o cúmulos dedicados de estaciones de trabajo— esta premisa no necesariamente se aplica a las mallas dinámicas. Esto es porque los tiempos de ejecución dependen de los recursos computacionales de la malla, y estos recursos están definidos por la función de tiempo $f(t)$ (no necesariamente continua) donde $R(i,t)$ son los recursos en función del tiempo y los nodos:

$$f(t) = \sum_{i=0}^{i=n} \int_0^t R(i,t)dt$$

En dicho caso, conviene determinar si una estrategia de asignación es *favorable* o no.

[1]Observación: si A es viable entonces existe un umbral μ tal que A es $\mu-$buena.

Sea $A : T^{(m)} \to P^{(m)}$ una estrategia de asignación cualquiera. La función $\overrightarrow{T} \mapsto \upsilon_A(\overrightarrow{T})$ es una variable aleatoria real sobre el conjunto $T^{(m)}$. Sea μ_A su primer momento, es decir, su esperanza o valor promedio $\mu_A = \sum_{r \in I_m} r\text{Prob}[\upsilon_A = r]$. La estrategia $A \in \mathbb{A}_m$ se dice ser *favorable* si para cualquier estrategia de asignación aleatoria $\overline{A} : T^{(m)} \to P^{(m)}$ (a cada tarea "actual", $\overline{A}$ le asigna uno de cualquiera de los procesadores disponibles), se tiene $\mu_A \leq \mu_{\overline{A}}$.

Esta noción de ser favorable depende del primer momento de la variable aleatoria υ_A y nociones similares resultan cuando se consideran otros parámetros estadísticos. Así, en vez del promedio μ_A se puede considerar la mediana $M_A = \min\left\{r_0 | \sum_{r \in I_m, r \geq r_0} \text{Prob}\left[\upsilon_A = r\right] \geq 2^{-1}\right\}$.

En la construcción del DGS, la computadora paralela conformada por la malla dinámica se contempla bajo el modelo BSP con consideraciones de costo para la latencia y el ancho de banda. En este modelo, la ejecución del programa se divide en una serie secuencial de superpasos. Los hilos paralelos están contenidos dentro de estos superpasos. Para incrementar el rendimiento de una computadora que se comporta bajo esta especificación de modelo, el tiempo para la terminación del superpaso debe ser reducida.

3.2.1. El método de asignación aleatoria. Éste es el esquema más común para asignar una tarea a un procesador remoto. Consiste en asignar de manera aleatoria la tarea a cualquier procesador disponible. Una ventaja de este método de asignación es que no se requiere información del estado de los procesadores remotos ni del costo en las comunicaciones. El método se adapta mejor a las mallas estáticas donde tanto la configuración de la red como de los recursos del procesador remoto se encuentran listados en una localización central.

Este método puede resultar contraproducente con mallas dinámicas por dos razones. La primera es que el procesador remoto que recibe una tarea puede no tener suficientes recursos de memoria y por ende deberá rechazar la tarea. En consecuencia la asignación falla y debe ser intentada de nueva cuenta. La segunda razón es que ambas variables

—el costo en las comunicaciones y los recursos en nodos remotos— son variables en una malla dinámica y por ende es improbable que la asignación optimice los recursos disponibles en la red.

En el método de asignación aleatoria se tiene que los tiempos de ejecución de las tareas en paralelo está bien representado por su primer momento μ_A por lo que una estrategia de asignación que produce menores tiempos de ejecución en las tareas, aunado al tiempo de asignación, puede considerarse como una estrategia favorable.

3.2.2. La asignación voraz respecto a ciclos computacionales. Éste se caracteriza por tener como principal factor a la cantidad de ciclos de computación sin utilizar por unidad de tiempo disponibles en el nodo remoto **[5]**. Para la distribución de n tareas paralelas, las computadoras remotas con la mayor cantidad de ciclos disponibles por unidad de tiempo serán asignadas las tareas en orden, siempre y cuando los recursos de memoria sean mayores o iguales a aquellos que requieren las tareas. Es decir, para el algoritmo de asignación voraz $(A_r(\overrightarrow{T}))$ se maximizan los ciclos de cómputo disponibles en los procesadores remotos $(R(\overrightarrow{P}))$ con respecto al conjunto de procesadores en la malla:

$$R(A_r(\overrightarrow{T})) \geq R(\overrightarrow{P})\ \forall \overrightarrow{P} \in P^{(m)}.$$

Cuando se trata con tareas donde los requerimientos de comunicaciones son pequeños o insignificantes, éste debe ser el método preferido. No obstante, las aplicaciones donde hay poca necesidad para las comunicaciones son menos cada día.

3.2.3. La asignación voraz respecto a costo de comunicación. Éste se caracteriza por tener como principal factor al costo de las comunicaciones entre el nodo que asigna y el nodo remoto **[5]**. A las computadoras remotas con el menor costo de comunicación —y con suficientes recursos de memoria— les serán asignadas las tareas que componen el superpaso. Es decir, para el algoritmo de asignación voraz $(A_c(\overrightarrow{T}))$ se minimizan los costos de comunicación hacia los procesadores remotos $(C(\overrightarrow{P}))$ con respecto al conjunto de procesadores en

la malla:

$$C(A_c(\overrightarrow{T})) \leq C(\overrightarrow{P}) \; \forall \overrightarrow{P} \in P^{(m)}.$$

Para aplicaciones con una utilización de mucha información donde los ciclos computacionales no están en alta demanda, éste sería el método a elegir.

3.2.4. La asignación por campo de fuerza. El método de asignación por campo de fuerza es el que se propone en este trabajo de tesis y no ha sido propuesto anteriormente en la literatura científica.

Éste considera la comunicación y los ciclos disponibles y se combina en un campo de fuerza escalar. La fuerza neta de atracción determina cuáles computadoras remotas les serán asignadas las tarea que componen el superpaso. Computadoras remotas que no tienen suficientes recursos producirán una fuerza repulsiva.

Otra política de asignación de tareas que utilice ambas variables representadas por la información del costo de computación y comunicaciones seguramente estará disponible. De hecho, las políticas que utilicen ambas variables para resolver el problema de asignación no sólo son infinitas en número, sino que además forman un espacio de funciones cuyo número de elementos no es ni siquiera numerable. El simulador presentado es una metodología para comparar estos métodos entre si al ser confrontados con las condiciones de la malla dinámica.

Un problema de los métodos que utilizan información de tareas y de procesadores para realizar la asignación, consiste en que son de orden de complejidad $O(NM)$. Esta dificultad, sin embargo, se minimiza mediante la aplicación con el método de Campo de Fuerza, como se demuestra en el capítulo **??**, ya que la fuerza que ejercen los nodos para la atracción de tareas decae rápidamente con el costo computacional. Esto permite aplicar el método de una manera local desde distinto puntos de acceso a la malla para distintos usuarios. De tal manera que el orden se transforma a $O(nm)$, donde $n \ll N$, y $m \ll M$.

3.3. Implementación del algoritmo

Para la implementación del algoritmo del simulador, es necesario comenzar con la determinación de los parámetros que caracterizarán a la malla dinámica.

Las tendencias generales que caracterizan una malla de computadoras de escritorio se pueden observar desde un punto de vista estadístico de la información recabada por Kondo, *et al.* [**49**] y puede ser utilizada como soporte para la aplicación de la Ley de Grandes Números de la teoría de la probabilidad.

La Ley de Grandes Números indica que sin importar la distribución de las variables que caracterizan una malla de escritorio, el promedio de los valores observados convergerá a una distribución normal. En otras palabras, dado un gran número de puntos, la función de probabilidad puede ser simulada por una distribución de Gauss:

$$Y(x) = Y(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2}(x-\mu)^2/\sigma^2},$$

con media μ y desviación estándar σ. El DGS genera valores para las variables que caracterizan la malla de escritorio y las tareas a ser asignadas en cualquier instante de tiempo. Un hecho notable es que esta presunción es válida para una colección grande de puntos. Por lo tanto, esta herramienta de evaluación es más aplicable a mallas de computadoras de escritorio de tamaño creciente, en los cuales el número de nodos es de mas de mil computadoras.

La primera variable para ser asignada es el valor instantáneo de los ciclos computacionales sin utilizar que están disponibles en cada computadora dentro de la malla dinámica. Considerando que la mayor parte de los usuarios estarán utilizando un procesador de palabras, navegando el Internet o procesando su correo electrónico —si acaso no están distraidos de la pantalla— la cantidad de ciclos disponibles será alta. Los usuarios que estén ejecutando programas que requieran de mucho CPU y poca interacción de entrada/salida mediante dispositivo de interconexión con la interfase humana (el teclado, por ejemplo), y

que dejarán pocos ciclos de computación disponibles, estarán distribuidos a lo largo de las orillas de la distribución de Gauss.

La última variable a ser asignada se refiere a la cantidad de recursos de memoria disponibles en cada computadora. Esta variable —la cual es independiente de los ciclos computacionales disponibles— es básicamente la cantidad de memoria virtual que esta computadora remota puede asignar a una tarea remota sin incurrir en problemas de subir y bajar datos del *swap* de disco. Debido a que cada tarea a ser asignada tiene una cantidad predeterminada de recursos de memoria necesarios para su ejecución, esta variable será utilizada para determinar si una tarea en particular puede ser asignada a un procesador remoto en específico o no. La mayoría de las aplicaciones actuales para el usuario de escritorio —tales como correo electrónico, procesador de texto, navegadores de Internet— suelen ser acaparadores de la memoria, especialmente con cada nueva versión. Esto determina que el número de máquinas con alta disponibilidad de memoria para procesos remotos tenderá a estar a lo largo de las orillas de la distribución de Gauss.

Por ende el perfil general de la máquina típica de la malla dinámica de computadoras de escritorio será de un alto índice de ciclos disponibles y de baja cantidad de memoria libre. Para generar estos valores, una distribución de Gauss se divide en un número de puntos que concuerda con la cantidad de computadoras en la malla. Entonces cada punto de las distribuciones de Gauss se asigna de manera aleatoria a una computadora remota en específico. La asignación de cada una de las distribuciones es independiente de las otras, como se menciona arriba.

La tercera variable que caracteriza a la malla de escritorio es el costo de la comunicación entre el procesador remoto y el procesador donde las tareas se generan. Como la mayoría de los usuarios no estarán subiendo o bajando archivos la mayor parte del tiempo, las computadoras con alto costo estarán a lo largo de las orillas de la distribución de Gauss. La asignación de costo se hace de la misma forma en que para las primeras dos variables.

Finalmente, el diseño trata con las variables que caracterizan las tareas remotas a ser asignadas. Cada una de estas tareas tendrá ciertos requerimientos de memoria y costo computacional. Ambas de estas variables se obtienen de una distribución de Gauss en la manera descrita arriba. Los requerimientos de memoria de cada tarea deben ser conocidos de antemano por *cualquier* mecanismo de asignación de tareas que evite el problema de los rechazos. El costo computacional, por el otro lado, es generalmente una variable desconocida. Por esto último, el simulador no hace esta variable conocida a cualquiera de los esquemas donde se evalúa el rendimiento. No obstante, esto se utiliza para tabular todas las combinaciones tarea/procesador para obtener un mínimo absoluto. Esta evaluación es la parte que requiere más poder de cómputo con respecto al simulador, como se describe abajo.

Considerando que el simulador está diseñado para evaluar una computadora paralela actuando bajo el modelo BSP, el mínimo es la combinación de las tareas a procesadores en el cual el tiempo máximo para todas las resoluciones de tarea es la más pequeña. En otras palabras:

Sea $P = \{p_1, p_2, \ldots, p_n\}$, el conjunto de procesadores que hay en la malla, y $T = \{t_1, t_2, \ldots, t_m\}$. el conjunto de tareas a ser resueltas en paralelo dentro del superpaso. Dado que tratamos con mallas dinámicas de gran tamaño que serán utilizadas para ejecutar tareas paralelas de un superpaso BSP limitadas por la Ley de Amdahl, podemos suponer que $m \leq n$. Entonces, para toda $t \in T$ y toda $p \in P$, existe un tiempo de ejecución $\tau(t, p)$. τ es por ende un mapeo $\tau : T \times P \rightarrow \mathbb{N}$. Defínase una asignación de tareas ϕ como un mapeo inyectivo $\phi : T \rightarrow P$. Entonces existen $C_m^n = n!/n!(m-n)!$ combinaciones de donde escoger a ϕ. Sea Φ el conjunto de todas las asignaciones y para cada $\phi \in \Phi$ definase: $\omega(\phi) = \text{Max}|_{t \in T}(\tau(t, \phi(t)))$. Entonces, el problema de optimización consiste en encontrar una asignación $\phi_0 \in \Phi$ tal que $\omega(\phi_0) = \text{Min}|_{\phi \in \Phi}\omega(\phi)$. Nótese que pueden existir múltiples asignaciones óptimas ϕ_0, ϕ_1, $\ldots$, ϕ_i.

Para la tabulación de características de nodos individuales que componen la malla dinámica, los valores instantáneos en cualquier momento no son muy útiles. El valor característico ciertamente estaría mejor representado por un valor promedio durante un intervalo particular de tiempo donde se localiza el valor instantáneo (es decir, un promedio móvil). Por ello, si el valor instantáneo es $\beta(t)$ (que pudiese ser el costo de las comunicaciones o la disponibilidad de ciclos en la computadora remota), el promedio móvil $\breve{\beta}(t)$ estaría dado por:

$$\breve{\beta}(t) = \frac{1}{\Delta t} \int_{t-\Delta t}^{t} \beta(\xi) d\xi.$$

Desde un punto de vista práctico, para hacer aplicable este método, se requiere aplicar un algoritmo numérico de integración. La regla de los trapezoides producirá buenos resultados con un número grande de puntos $\beta(t_0)$, donde $t_0 \in [t - \Delta t, t]$.

En resumen, los parámetros que el DGS determinará mediante la distribución de Gauss son los siguientes:

- Ciclos de cómputo disponibles por unidad de tiempo en cada procesador.
- Memoria disponible en cada procesador.
- El costo de las comunicaciones hacia cada procesador remoto.
- Los requerimientos de memoria para cada tarea.
- El costo computacional en ciclos para cada tarea.

Esto provee condiciones distintas de la malla para cada corrida del simulador y requiere que sea ejecutado una gran cantidad de veces para obtener resultados estadísticos representativos. La figura **??** muestra los resultados para 1000 corridas del simulador donde 20 tareas simultáneas componen el superpaso para ser asignado en una malla dinámica con 1000 computadoras remotas. En esta gráfica se observa que el tiempo de ejecución del superpaso obtenido al usar el método voraz respecto a ciclos de cómputo.

3.3.1. Parámetros utilizados en la experimentación. Para la obtención de los parámetros utilizados en los experimentos, de acuerdo a la Ley de los Grandes Números, se generó una distribución normal para el costo de comunicaciones, para los ciclos disponibles de cómputo y para los recursos de memoria disponibles. Esta distribución normal para el costo de comunicaciones de muestra en la figura 3.3.1 con los nodos individuales de la malla dinámica ordenados según la distribución normal. Para cada corrida individual del simulador (miles de corridas son necesarias bajo distintas condiciones) el identificador de nodo dentro de la malla se asigna de manera aleatoria al número de la malla en la distribución normal. En una sola corrida del simulador —por ejemplo— los parámetros utilizados para los costos de comunicación se muestra en la figura 3.3.2. De la misma manera, los parámetros para los ciclos de cómputo disponibles y recursos de memoria se generan y se muestran en las figuras 3.3.3–3.3.4.

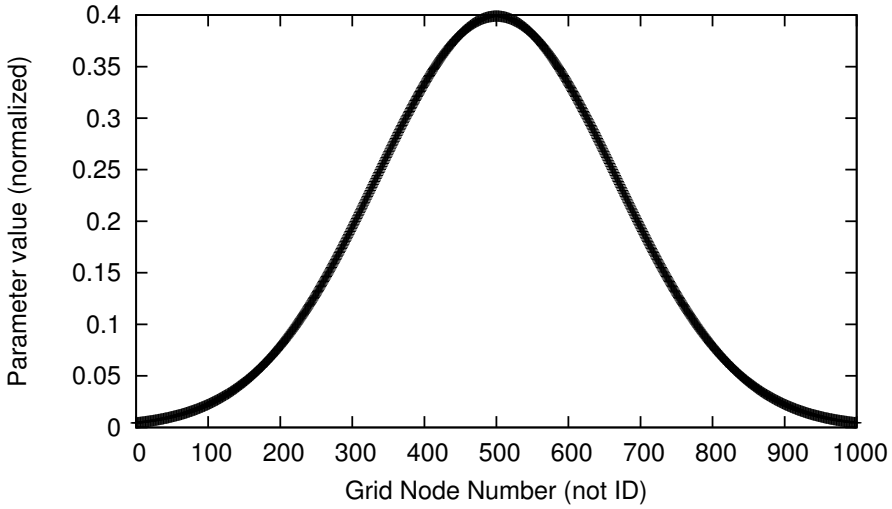


FIGURA 3.3.1. Costo de comunicaciones para nodos de la malla (ordenados)

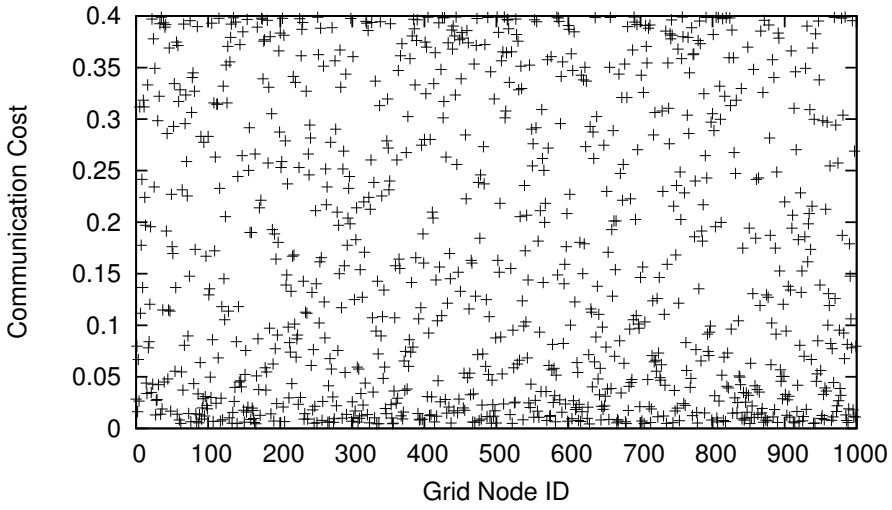


FIGURA 3.3.2. Costo de comunicaciones para nodos de la malla (asignados)

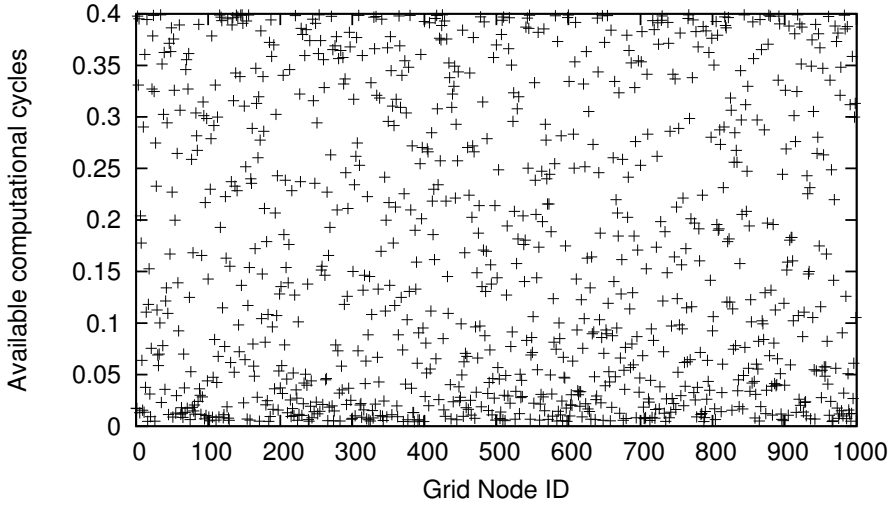


FIGURA 3.3.3. Ciclos de cómputo disponibles para nodos de la malla (asignados)

Como se observa, el valor promedio de estas distribuciones normales son iguales. Esto permite variar los parámetros del sistema mediante una variación lineal de la distribución gaussiana para tomar en cuenta

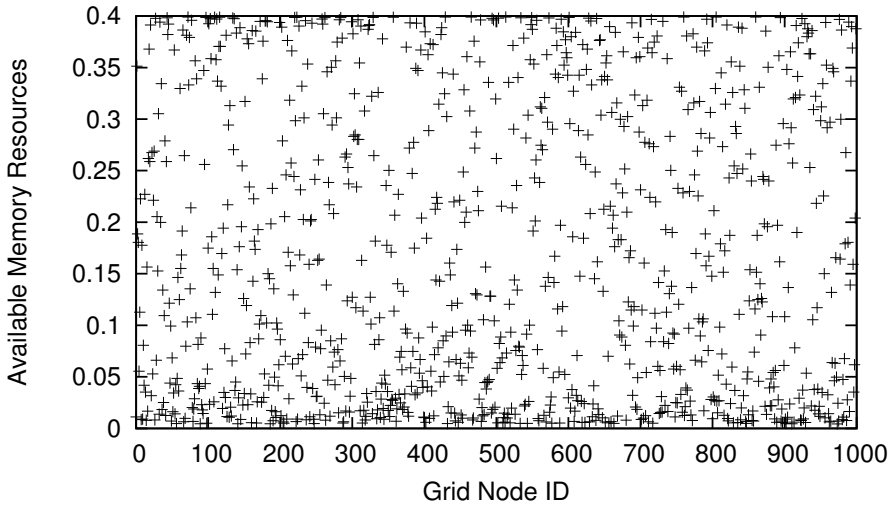


FIGURA 3.3.4. Recursos disponibles de memoria para nodos de la malla (asignados)

factores como congestión de red o una decreciente disponibilidad de ciclos disponibles en la malla.

En cuanto a la generación de los parámetros que caracterizan las tareas a ser asignadas, se utilizaron dos enfoques. El primero fue suponiendo una distribución normal tal como la que se muestra en la figura 3.3.5. Con dicha distribución, los ciclos computacionales y recursos de memoria requeridos por cada tarea fueron asignados de manera aleatoria a cada identificador de tarea, como se muestra en las figuras 3.3.6–3.3.7. El segundo enfoque, utilizado en el caso de estudio de la sección 4.3 consiste en analizar el problema paralelo en particular y asignar los parámetros que caracterizan propiamente a las tareas.

3.3.2. Procedimiento de la simulación. La entrada de parámetros para cada corrida incluye el tamaño de la malla (N), la relación entre comunicación y cómputo para la aplicación (R), y el número de hilos paralelos a ser ejecutados en el superpaso (M).

1. Asignar un número de identificación a cada nodo dentro de la malla y a cada tarea a ser ejecutada en el superpaso.

2. Dividir el intervalo $[-3, 3]$ en N segmentos.

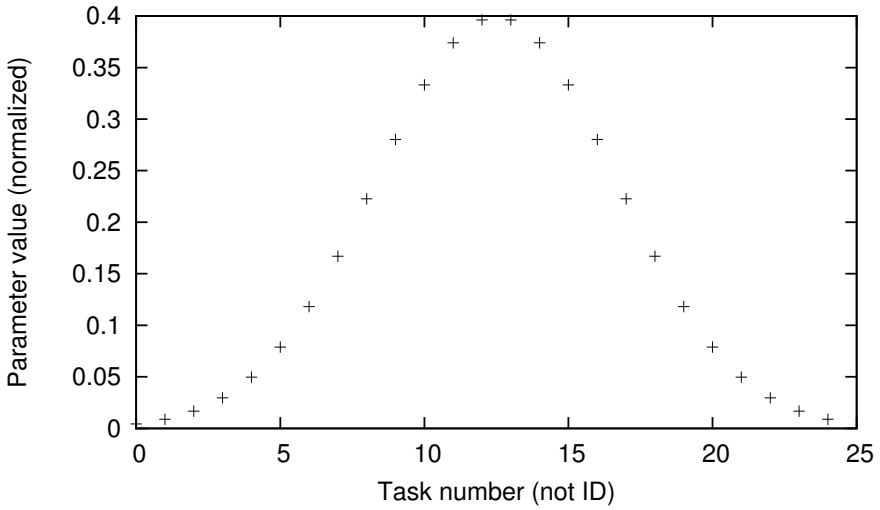


FIGURA 3.3.5. Ciclos de cómputo requeridos para las tareas paralelas (ordenados)

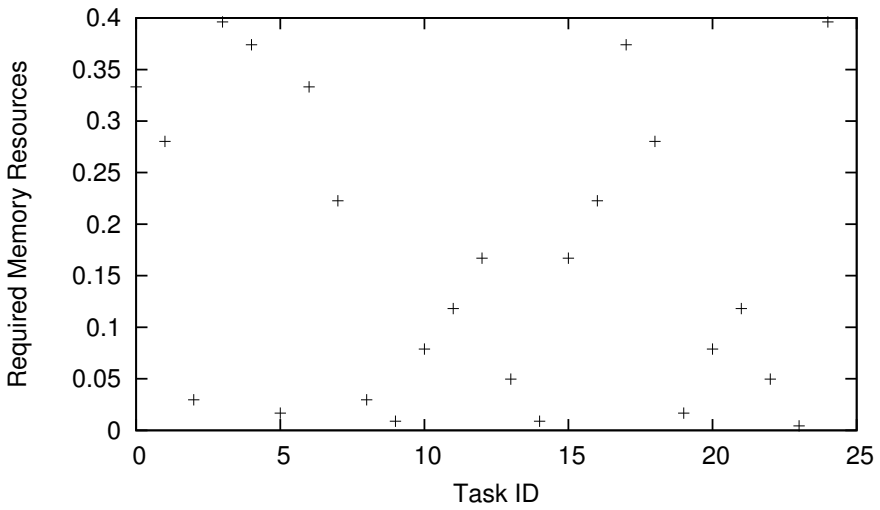


FIGURA 3.3.6. Recursos de memoria requeridos para las tareas paralelas (asignados)

3. Para cada punto medio de segmento obtener el valor de la curva de distribución gaussaina (arreglo $G_1[N]$).

4. Efectuar asignaciones aleatorias de cada identificador de nodo a los elementos del arreglo $G_1[N]$. Esto será los ciclos de cómputo

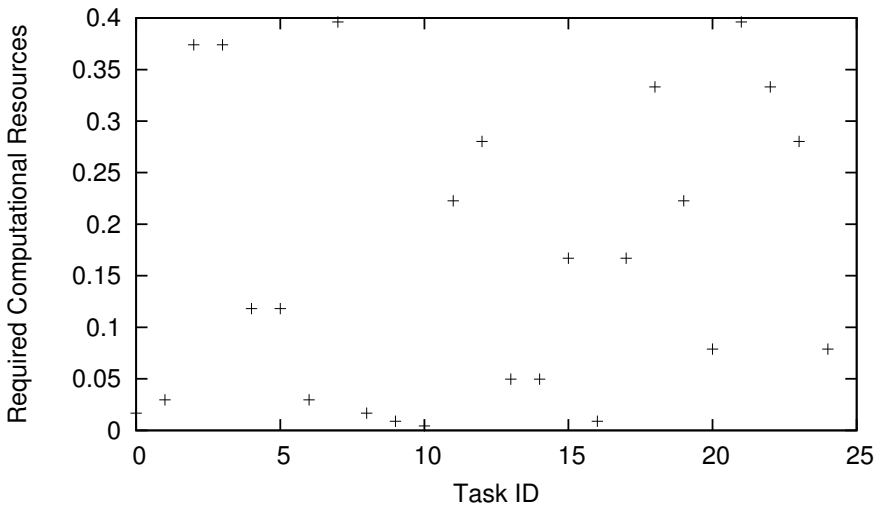


FIGURA 3.3.7. Ciclos de cómputo requeridos para las tareas paralelas (asignados)

disponibles para los nodos remotos ($A[N]$). Las unidades de medida se transforman para que coincidan con la expresión más sencilla de la distribución gaussiana.

5. Efectuar una asignación aleatoria distinta para cada identificador de nodo a los elementos del arreglo $G_1[N]$ multiplicado por la razón de costo de comunicación a cómputo R. Este será el costo de comunicación para los nodos remotos, ($B[N]$).

6. Efectuar una tercera asignación aleatoria distinta de cada identificador de nodo a los elementos del arreglo $G_1[N]$. Esto será la disponibilidad de recursos de memoria en los nodos remotos ($C[N]$). Las unidades una vez más se transforman para que coincidan con la expresión más sencilla de la distribución gaussiana.

7. Si se conocen los requerimientos de memoria de las tareas (como en el problema de tomografía presentado en la sección 4.3), asignar estos valores al arreglo $T_m[M]$ y proceda al paso 11, de lo contrario continúe en el paso 8.

8. Dividir el intervalo $[-3, 3]$ en M segmentos.

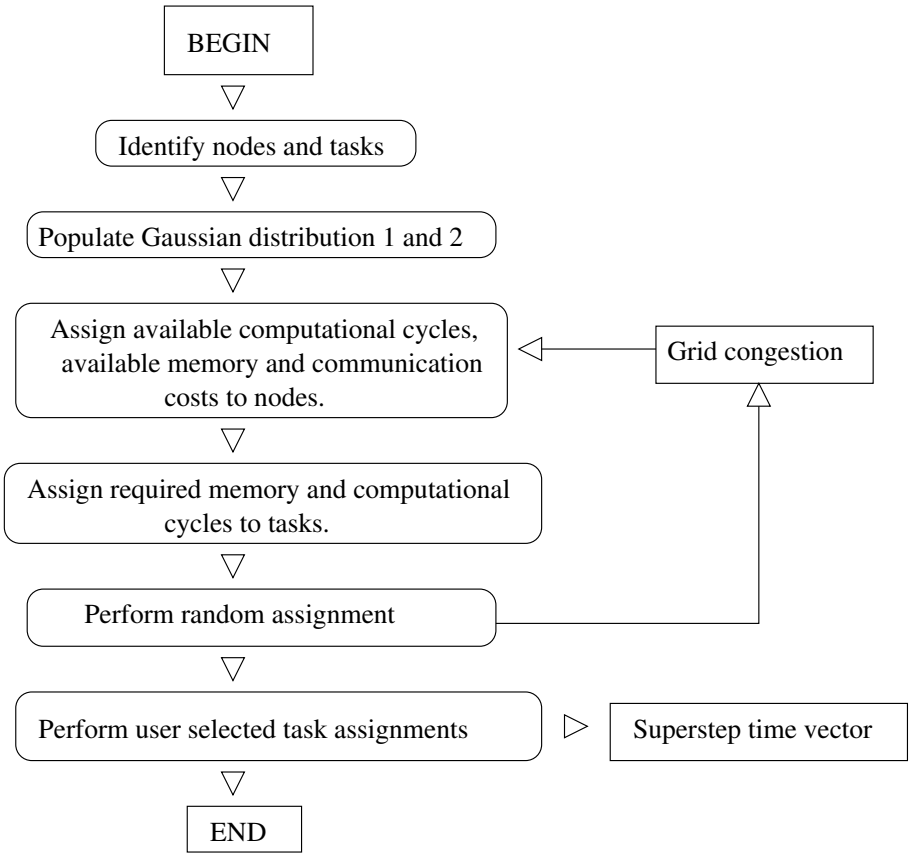


FIGURA 3.3.8. Flujo interno del simulador

9. Para cada punto medio de segmento obtener el valor de la curva de distribución gaussaina (arreglo $G_2[N]$).

10. Efectuar asignaciones aleatorias de cada identificador de nodo a los elementos del arreglo $G_1[N]$. Esto será los ciclos de cómputo disponibles para los nodos remotos ($A[N]$). Las unidades de medida se transforman para que coincidan con la expresión más sencilla de la distribución gaussiana. Esto coincide con las unidaded del punto 6.

11. Si se conocen los requerimientos de memoria de las tareas (como en el problema de tomografía presentado en la sección 4.3), asignar estos valores al arreglo $T_m[M]$ y proceda al paso 15, de lo contrario continúe en el paso 12.

12. Dividir el intervalo $[-3, 3]$ en M segmentos.

13. Para cada punto medio de segmento obtener el valor de la curva de distribución gaussaina (arreglo $G_2[N]$).

14. Efectuar asignaciones aleatorias de cada identificador de tarea a los elementos del arreglo $G_2[N]$. Esto será los ciclos de cómputo requeridos por las tareas ($T_c[N]$). Las unidades de medida se transforman para que coincidan con la expresión más sencilla de la distribución gaussiana. Esto coincide con las unidades del punto 4.

15. Con todos los parámetros que caracterizan la malla en los arreglos A, B y C, efectuar una asignación aleatoria de tareas a los nodos remotos.

16. Si alguna tarea asignada tiene más requerimientos de memoria que aquellos disponibles en el nodo asignado, tabular la falla y sumar el tiempo asociado al costo de comunicaciones para el nodo remoto al tiempo del superpaso para el método de asignación aleatoria.

17. Repetir el punto 16 hasta que todas las tareas se hayan asignado de manera exitosa o todos los nodos se han agotado. Si todos los nodos se han agotado, tabular una mensaje de "congestión de malla" y repita el procedimiento a partir del punto 4. El tiempo final del superpaso se almacena como el primer elemento del arreglo $D[M]$. El tiempo de superpaso será el tiempo máximo de ejecución más el tiempo máximo de comunicación de todas las tareas, aunado al tiempo perdido debido a rechazos.

18. En este punto el simulador puede ser configurado para utilizar toda la información en los arreglos A, B y C (continuar en el punto 20) o bien simular una condición real donde los nodos que se encuentran demasiado lejanos se ignoran.

19. Los arreglos A, B y C, se codifican como elementos de un arreglo estructurado $S[A, B, C]$, y así se habilita la utilización de un "quick–sort" sobre S el cual acomodará a los tres con el mismo criterio. S se ordena de acuerdo a B y se reduce al eliminar los elementos de la cola. El número de elementos a considerar en los esquemas de asignación de tareas es configurable por el usuario y se fija como un múltiplo del número de tareas a ser asignadas.

20. El tiempo de superpaso para cada método de asignación se obtiene de S. El método escogido para programar lo descrito arriba es intensivo de memoria y evita mover variables entre localidades de memoria: S se ordena mediante acomodo de punteros para cada uno de los métodos de asignación de tareas.

21. Un vector conteniendo los tiempos de superpaso para todos los métodos considerados se agrega a un archivo de texto el cual pueda ser alimentado a software estadístico o perfilado para elaboración de gráficas. Este segundo paso incluye la obtención de la media, el promedio, el máximo y el mínimo para todos los métodos de asignación analizados.

22. Si el número de simulaciones efectuadas es igual al número de simulaciones solicitadas, detener la ejecución. De lo contrario, repetir el procedimiento a partir del punto 4.

Este procedimiento se ilustra en la figura 3.3.8. Como se puede observar, las reglas para la simulación son simples y directas y de fácil reproducción. No obstante, el procedimiento ilustrado en la figura 3.3.8 debe ser repetido un número grande de veces. Los resultados para cada método de asignación de tareas se conserva en un arreglo separado.

En el procedimiento descrito arriba, la implementación se realizó en lenguaje C con el compilador GNU. Todos los arreglos fueron optimizados por la utilización de arreglos de apuntadores. De tal manera sólo se movieron los apuntadores los cuales eran referidos más de una vez por los distintos arreglos. Esto —además de generar un código más rápido— evitó cualquier duplicación de valores en memoria.

La versión distribuida en el cúmulo de 200 nodos localizado en el Instituto Mexicano del Petróleo se obtuvo mediante la generación de "shell scripts" generados automáticamente mediante programas en lenguaje Perl. La ejecución y control de los procesos de simulación que se generaban en el cúmulo también se logró mediante "shell scripts" automáticamente generados.

Una pregunta que puede surgir es si los métodos de asignación tendrán un efecto en los parámetros a ser medidos para la asignación

de tareas en sí. Mientras que esto ciertamente es cierto, el efecto se distribuye de forma uniforme por la malla dinámica, produciendo sólo un error sistemático.

CAPíTULO 4

Evaluación del método de campo de fuerza

En este capítulo se comienza con una descripción de los resultados preliminares del simulador DGS en la sección 4.1. En la sección 4.2 se hace un estudio comparativo del desempeño de los métodos de asignación de tareas descritos en el capítulo 3 con el método de Campo de Fuerza.cuando el costo de las tareas a realizar es aleatorio. En la siguiente sección, 4.3, se hace una simulación de la resolución del problema de la tomografía de pozos, donde los costos de las distintas tareas del superpaso son conocidos. Por último, terminamos con la sección 4.4 en donde se analiza el problema de recolección de la información, con lo cual se observa que una fragmentación de la malla no altera el buen desempeño del método propuesto en este trabajo de tesis.

En las simulaciones descritas a continuación, las corridas cortas (tales como aquellas que produjeron las figuras **??**–**??**), fueron ejecutadas en una Pentium-IV@2.2GHz. Las corridas más elaboradas (figuras **??**–**??**) fueron resueltas en un cúmulo dedicado de 200 processor Pentium-III@1GHz conectado con enlace Gigabit Ethernet conmutado, en el Instituto Mexicano del Petróleo. Estas corridas tuvieron una duración típica de 36 a 96 horas de tiempo real antes de su terminación. Una sola simulación no es demasiado intensivo sobre el procesador, pero cada punto en las gráficas más elaboradas representa el resultado estadístico de 1000 corridas del simulador. Más aún, al aumentar el número de tareas paralelas en el proceso, los requerimientos de memoria del programa aumentan de manera significativa. Un arreglo con todas las combinaciones tarea–procesador se mantiene en memoria para obtener una aproximación al valor mínimo teórico.

Las simulaciones fueron distribuidas en el cúmulo al crear un proceso para cada conjunto de tareas (*i.e.*, 5, 6, 7, ..., 50) que conformaron los hilos a ser resueltos durante el superpaso BSP. Cada uno de estos procesos generó recursos y condiciones de malla distintos para 1000 escenarios (que no debe ser confundido con las simulaciones donde se mantuvo constante el tamaño de malla en 1000 elementos) para procesar los resultados y obtener la mediana entre otros estadísticos. Los otros valores obtenidos incluyeron la media, el máximo, el mínimo y el valor en tiempo *secuencial*, es decir, el tiempo que el superpaso tomaría para completar si todos los hilos fuesen ejecutados de manera secuencial en el procesador de referencia. Por ello, para cada conjunto de parámetros (costo de comunicación bajo, normal y alto) se generaron 45 procesos, lo cual equivalía a una utilización del 67.5 por ciento de los procesadores del cúmulo. Más adelante, cuanto se realiza la simulación de resolución del problema de tomografía de pozos, el número de hilos en el superpaso permanece constante y se generan procesos distintos para cada tamaño de malla con un paso de 100 nodos. En este caso la utilización del cúmulo fue al 100 por ciento mientras muchas tareas permanecían en cola.

Las unidades de tiempo en todas las gráficas subsecuentes están expresada relativas a los ciclos computacionales de un procesador de referencia, de igual manera en que lo hace el simulador GridSim [**14**]. Estas unidades abstractas de tiempo se utilizan en lugar de unidades de tiempo real porque el enfoque del simulador es producir resultados comparativos para distintos esquemas de asignación de tareas bajo las mismas condiciones de comunicaciones y disponibilidad de recursos. Es de notar que esto no es lo mismo que predecir el desempeño en tiempo real de una aplicación paralela BSP en particular sobre una malla dinámica.

4.1. Resultados preliminares

El propósito de la evaluación es determinar cual estrategia es *mejor* para la asignación de tareas en una malla dinámica. Cada proceso puede ser calificado con las siguientes consideraciones:

- La cantidad de memoria requerida para el almacenaje de los datos.
- La cantidad de ciclos de computación que se requieren para completar todas las operaciones programadas.
- Las unidades de tiempo consumidas por los requerimientos de comunicación.

Estrictamente hablando, el segundo punto dependerá del hardware y puede ser asociado entre distintas plataformas por una proporcionalidad lineal. Por otro lado, los ciclos de cómputo pueden ser referidos en un sentido abstracto —independiente de la arquitectura— tal como lo hacen Papadimitriou and Yannakakis [**63**] en la presentación del modelo de las gráficas acíclicas dirigidas. En este trabajo se utiliza el mismo enfoque.

Con estas cantidades, los méritos de cualquier esquema particular para la asignación de tareas puede ser evaluado. El primer punto —requerimientos de memoria— es necesario para determinar si la computadora remota tiene los recursos necesarios para completar la tarea. El segundo punto —ciclos requeridos por la tarea y ciclos disponibles por unidad de tiempo en el nodo remoto— permiten obtener el *tiempo pared* de ejecución del programa. No obstante, el conocimiento de la cantidad de ciclos que cada tarea requiere no es generalmente conocido. Además, en la tabulación de una tarea paralela, el tiempo asociado al costo en las comunicaciones también debe ser considerado.

Para el cálculo del óptimo es necesario conocer el número de ciclos que cada tarea requerirá. Esto se utilizará para evaluar el rendimiento de los distintos algoritmos que se comparan (estos algoritmos, claro, no podrán utilizar esta información). Un algoritmo será *mejor* que otro si los tiempos obtenidos más cercanos a los valores óptimos.

Las figuras 4.1.1–4.1.3 reflejan información puntual para las simulaciones de un programa de aplicación con 20 hilos paralelos en los cuales el principal cuello de botella es la disponibilidad de ciclos de cómputo, pero donde los requerimientos en comunicaciones no son insignificantes. Cada punto de la gráfica representa el promedio del tiempo de superpaso obtenido en 1000 simulaciones con la metodología de asignación correspondiente.

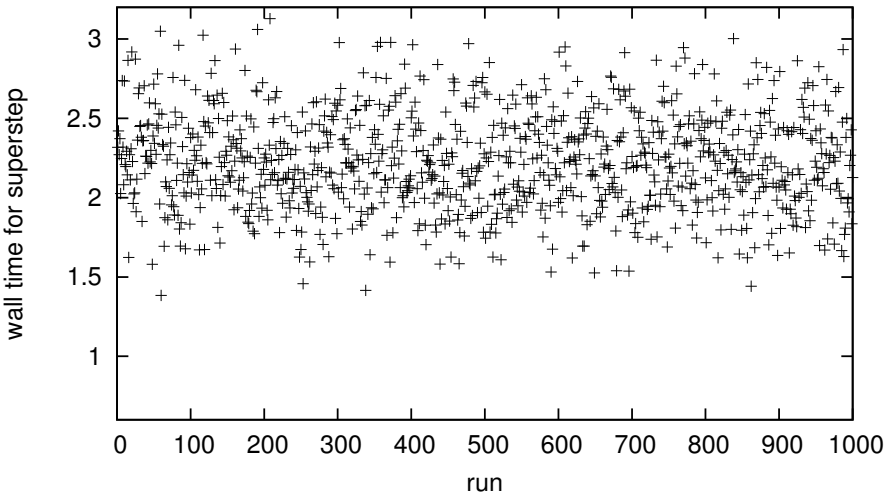


FIGURA 4.1.1. Tiempo de superpaso para 20 tareas en una malla de 1000 nodos con costo de tarea aleatoria (método avaricioso en ciclos de cómputo)

La figura 4.1.1 muestra los resultados para el tiempo promedio del superpaso de 1000 simulaciones cuando la asignación se hace mediante el método avaricioso en ciclos de cómputo. El eje horizontal representa las corridas individuales del simulador. Para cada corrida los recursos son asignados y representan condiciones de malla distintas. Todos los puntos de simulación en la gráfica corresponden a una malla de 1000 nodos. El eje vertical se expresa en unidades de tiempo relativos — *i.e.*, tiempo equivalente para una cantidad fija de ciclos de cómputo en un procesador base de referencia. En esta figura los parámetros están pesados según la transformación lineal descrita en el capítulo 3

para reflejar condiciones de comunicación normales. En esta gráfica se observa que los tiempos caen en el intervalo [1.5, 3.1].

En seguida mostramos la figura 4.1.2. En esta figura el método

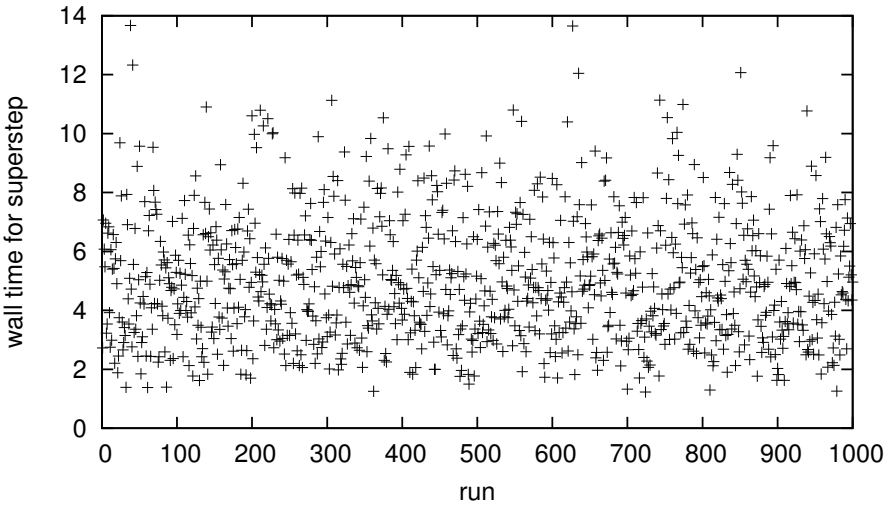


FIGURA 4.1.2. Tiempo de superpaso para 20 tareas en una malla de 1000 nodos con costo de tarea aleatoria (método avaricioso en comunicaciones)

de asignación de tareas utilizado es el avaricioso por comunicaciones. Al igual que en la figura anterior, en esta gráfica cada punto del eje horizontal representa el promedio de duración del superpaso de 1000 corridas individuales del simulador. Para cada corrida los recursos son asignados y representan condiciones de malla distintas. La gráfica corresponden —de igual manera— a una malla de 1000 nodos. El eje vertical se expresa en unidades de tiempo relativos al procesador de referencia. Se observa que los tiempos de superpaso caen en intervalo [1.0, 14.0]. Esto es debido a que el cuello de botella de las tareas no son las comunicaciones y por lo tanto se obtienen resultados menos favorables que los que presenta el método avaricioso en ciclos de cómputo de la figura 4.1.1. Se nota que la escala en la figura 4.1.2 es diferente,

y esto refleja un desempeño pobre del método avaricioso en comunicaciones cuando los requerimientos en ciclos de cómputo son el factor dominante.

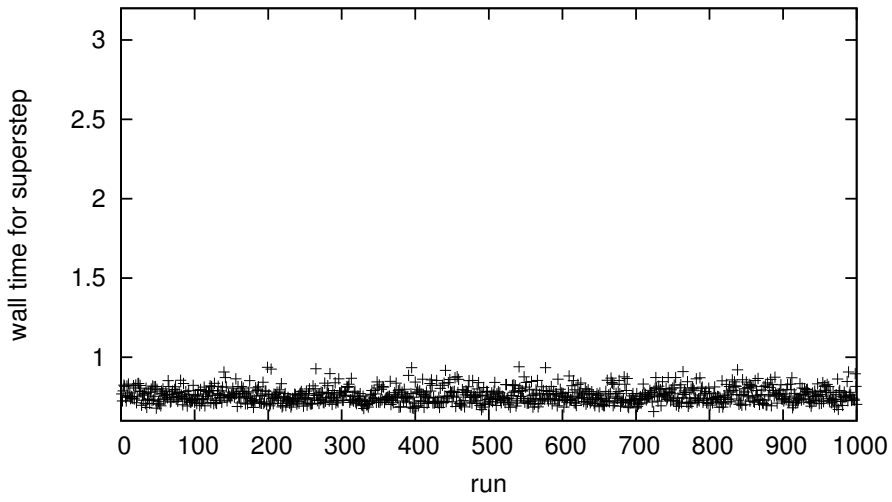


FIGURA 4.1.3. Tiempo de superpaso para 20 tareas en una malla de 1000 nodos con costo de tarea aleatoria (método Campo de Fuerza)

En la figura 4.1.3 se muestran las simulaciones con los mismos parámetros, pero tabulando el promedio del tiempo de superpaso obtenido en 1000 simulaciones conforme al método de asignación por Campo de Fuerza. Una vez más, el eje horizontal representa los promedios de duración de superpaso. Para cada corrida los recursos son asignados y representan condiciones de malla distintas. La malla tiene un tamaño de 1000 nodos. El eje vertical se expresa en unidades de tiempo relativos. Se observa que el desempeño del superpaso cae en el intervalo [0.1, 1.0], el cual es mejor que el desempeño del método avaricioso en ciclos de cómputo de la figura 4.1.1. Esto indica que aunque las comunicaciones no son tan importantes para que el método avaricioso en comunicaciones tenga un buen desempeño, no son despreciables y son suficientes para que el método de Campo de Fuerza concentre los

tiempos de superpaso en un intervalo más estrecho y de menor valor absoluto.

De las figuras 4.1.1–4.1.3 el análisis inicial indica que el método de Campo de Fuerza es superior a los otros dos métodos estudiados, sin embargo se requiere de mayor análisis para determinar bajo qué condiciones esto es verdad. En otras palabras, la información de estas figuras es indicativa de tendencias que deben ser estudiadas con mayor detalle, que se realiza en las siguientes secciones. Para este efecto, al obtener los resultados estadísticos de una serie de gráficas como las mostradas en las figuras 4.1.1–4.1.3, para un número distinto de tareas paralelas simultáneas y para tamaños de malla distintos, los patrones emergentes se pueden observar. Esto se realiza en las secciones 4.2–4.3.

4.2. Costo de tarea aleatorio

En las siguientes pruebas conducidas, el primer experimento consistió en analizar un conjunto de tareas en paralelo. Para ello se obtuvieron 1000 resultados del tipo mostrados en las figuras 4.1.1– 4.1.3, es decir, para cada distinto método de asignación: aleatorio, avaricioso en ciclos de cómputo, avaricioso en comunicaciones, Campo de Fuerza y aproximación al mínimo se tabularon los promedios de 1000 simulaciones en 1000 ocasiones. Es decir, se realizaron 10^6 simulaciones para cada conjunto de tareas en paralelo, tabulando en cada simulación los los resultados de cinco métodos de asignación. Esto se repitió para condiciones de costo bajo, normal y alto de comunicaciones, y también se repitió para conjuntos de 5 hasta 50 tareas en paralelo. Con ello el total de simulaciones que se requirió para las figuras 4.2.1–4.2.3 fue de $3 * 45 * 10^6 = 1{,}35 * 10^8$. Un número similar de simulaciones debe ser efectuado para reproducir los resultados que aquí se presentan. En base a otros experimentos realizados y que no se reproducen en este trabajo por no ser relevantes al objetivo fundamental de este trabajo de tesis, se puede apuntar que un número mayor número de simulaciones no produce resultados muy distintos de los que aquí se presentan.

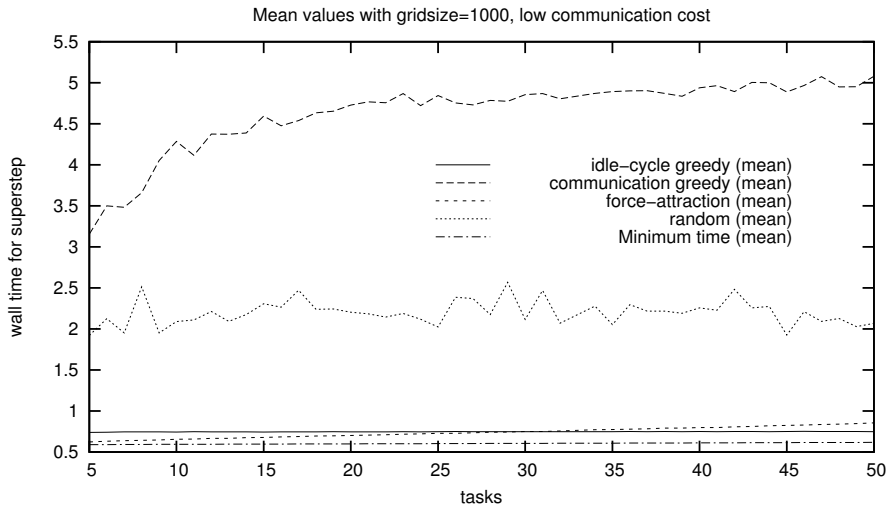


FIGURA 4.2.1. Resultados comparativos para tareas con costo bajo de comunicacion

En la figura 4.2.1 se muestra el resumen estadístico de las medias en las simulaciones efectuadas en donde las tareas tienen un costo bajo de comunicaciones. Como es de esperarse, el método avaricioso en comunicaciones es el que peor desempeño presenta. El método de asignación aleatoria, aún considerando los retrasos causados por rechazos, lo supera y se encuentra el rango de tiempo [2.0, 2.5]. Con resultados muy similares podemos encontrar el método avaricioso en ciclos de cómputo, el método de Campo de Fuerza y la aproximación al mínimo. En este caso se observa en la figura 4.2.1 que el método de Campo de Fuerza es mejor que la asignación voraz en ciclos de cómputo y que sólo cuando el número de elementos en el conjunto de tareas en paralelo supera 30 es que se invierte la relación.

En la figura 4.2.2 que se muestra adelante, se observan el resumen estadístico de las medias en las simulaciones efectuadas en donde las tareas tienen un costo normal de comunicaciones. En este caso el método voraz en comunicaciones supera a la asignación aleatoria únicamente cuando hay menos de 10 tareas paralelas en el conjunto a ser asignado. Por otro lado, el método de Campo de Fuerza —que toma en

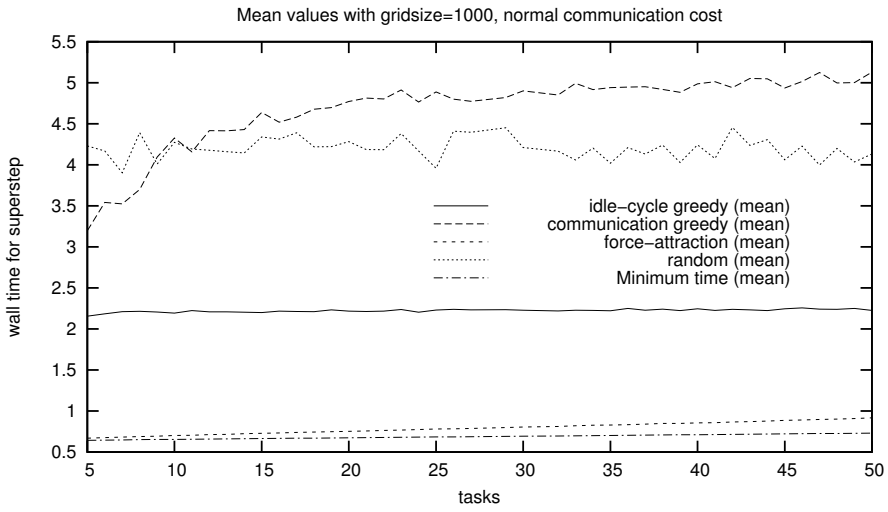


FIGURA 4.2.2. Resultados comparativos para tareas con costo normal de comunicación

consideración a las comunicaciones— conduce a un mejor desempeño que cualquier otro método y se desvía muy poco de la aproximación al mínimo.

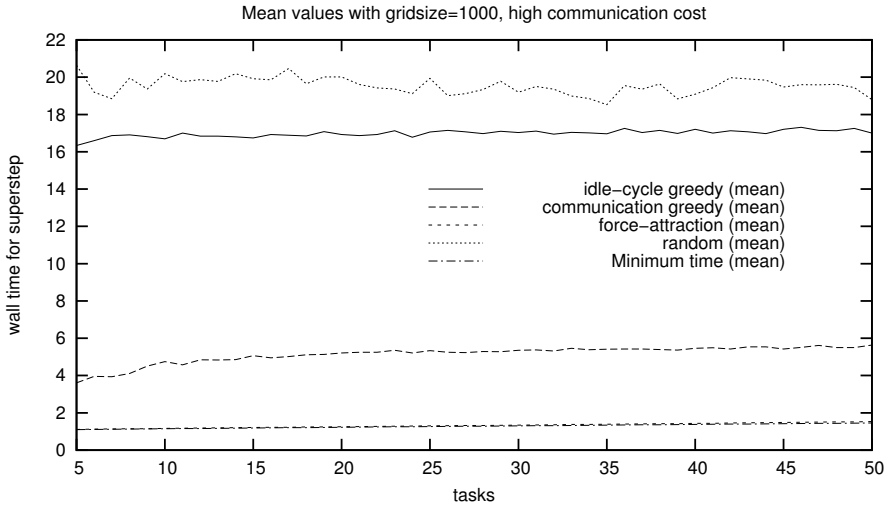
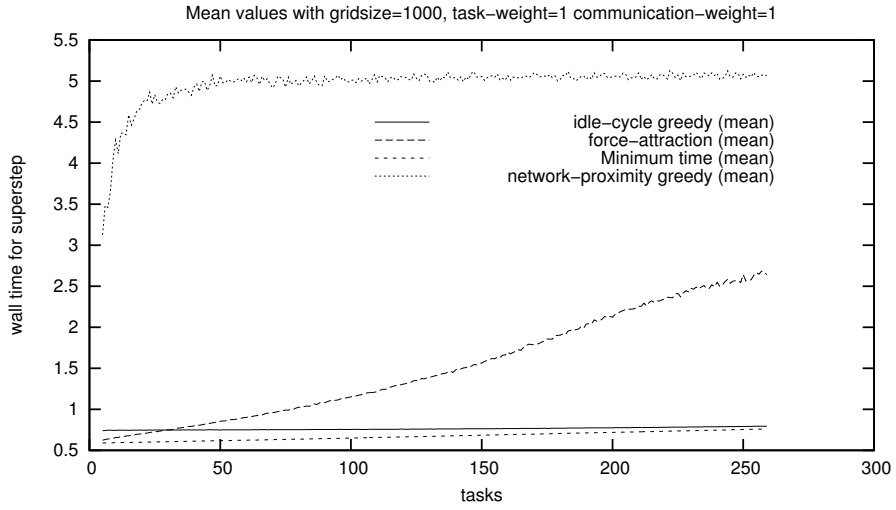


FIGURA 4.2.3. Resultados comparativos para tareas con costo alto de comunicación

En la figura 4.2.3 observamos el comportamiento cuando el costo en comunicaciones es alto. Como es de esperarse, el método voraz en comunicaciones supera al voraz en ciclos de cómputo, y ambos son mejores que el método aleatorio. El método de Campo de Fuerza está tan cercano a la aproximación al mínimo que en la gráfica no se pueden distinguir uno del otro. Esto muestra que el método de Campo de Fuerza es casi insuperable por los otros métodos bajo estas condiciones. También es de notar que las aplicaciones con un alto costo de comunicaciones cobran cada día mayor importancia, como lo menciona Sujithan [**74**].

En seguida procedemos a realizar más simulaciones, aumentando el número de elementos en el conjunto de tareas paralelas. Para ello realizamos el mismo procedimiento de experimentos con conjuntos de cardinalidad $\{51, 52, \ldots, 255\}$ y resumimos los resultados estadísticos de las medias en las figuras 4.2.4–4.2.6.

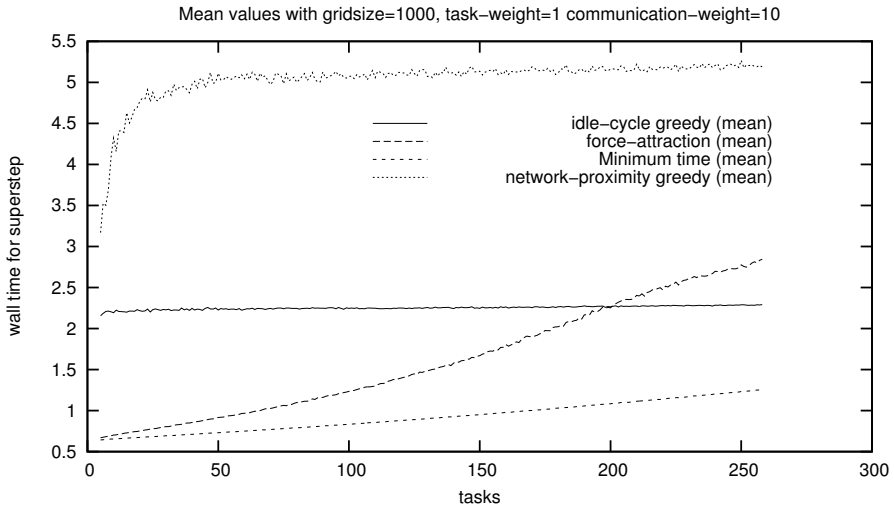


Bajo costo de comunicación

FIGURA 4.2.4. Valores medios para distintos costos de comunicación

En la figura 4.2.4 se observa que después de que el conjunto de tareas en paralelo rebasa la cardinalidad de 50, el método voraz en comunicaciones se estabiliza en un valor de 5. Este valor refleja un

desempeño pobre, como es de esperarse para tareas donde las comunicaciones no son significativas. Otro comportamiento que se observa es que el desempeño del método de Campo de Fuerza también va decrementando, aunque la pendiente de la curva es cada vez menor conforme se avanza por el eje coordenado. El método voraz en ciclos de cómputo permanece con un buen desempeño a lo largo del estudio. En detalle importante que debe notar de esta gráfica es que la cardinalidad del conjunto de tareas al final del experimento se acerca al 25 por ciento de la capacidad de la malla, por lo cual es necesario analizar qué ocurre cuando la malla tiene mayores dimensiones.

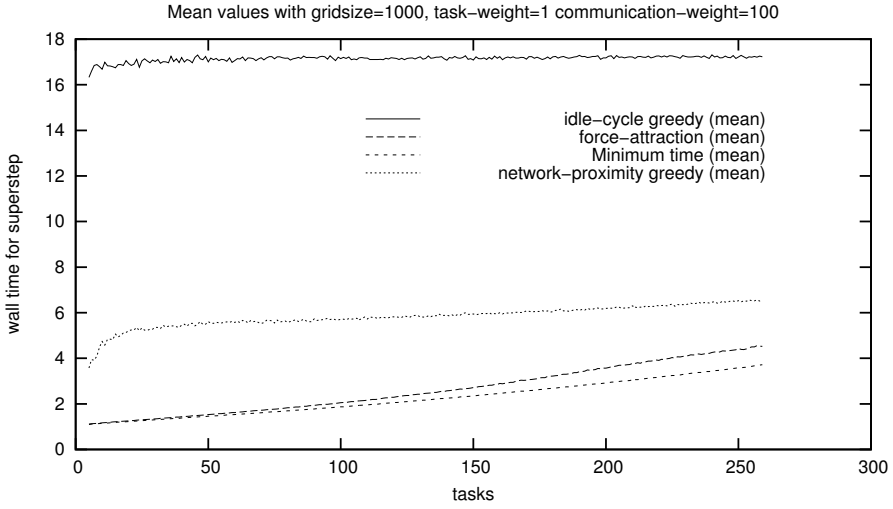


Costo normal de comunicación

FIGURA 4.2.5. Valores medios para distintos costos de comunicación

En la figura 4.2.5 se observan los resultados para un programa paralelo donde el costo del cómputo aún excede el costo en las comunicaciones. Muchas aplicaciones en paralelo caen en esta categoría. En esta figura se aprecia que después de que el conjunto de tareas en paralelo rebasa la cardinalidad de 200 (una quinta parte del tamaño de la malla dinámica), el método de Campo de Fuerza tiene peor desempeño que el voraz en ciclos de cómputo. Se observa que este último método tiene un desempeño casi constante a lo largo de los experimentos. Por otro

lado, aún con un costo normal de comunicaciones, el método voraz en comunicaciones no justifica su uso pues su desempeño mejora poco en relación a la que se observa en la figura 4.2.4.



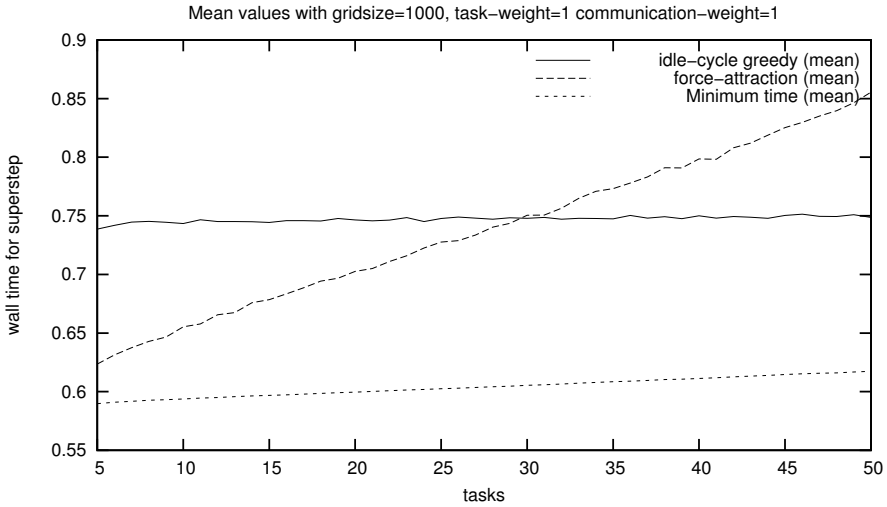
Alto costo de comunicación

FIGURA 4.2.6. Valores medios para distintos costos de comunicación

En la figura 4.2.6 se muestran los resultados de esta serie de experimentos cuando se toma un costo de comunicaciones importante. Es decir, se analizan tareas donde los costos de comunicación son significativos en relación a los costos de cómputo. En este caso el método voraz en comunicaciones supera al voraz en ciclos de cómputo. No obstante, en ninguna circunstancia supera al método de Campo de Fuerza. Este último método sólo se despega de la aproximación al óptimo cuando la cardinalidad del conjunto de tareas a ser asignadas supera 100.

En el futuro cercano el tamaño de las mallas dinámicas tenderá a incrementarse, y por esta razón es importante analizar el rendimiento de los algoritmos de asignación de tareas en relación al crecimiento de la malla. En las figuras 4.2.7–4.2.9 se resumen los resultados estadísticos de las simulaciones que muestran estas tendencias. Para su realización fue necesario repetir las simulaciones anteriores pero con dos tamaños

distintos de malla dinámica. Estas simulaciones se realizaron para mallas dinámicas con poblaciones de 5000 y 10000. Todas las simulaciones se realizaron bajo el supuesto de bajo costo de comunicaciones, dado que en esta circunstancia el método de Campo de Fuerza tuvo su peor desempeño.



Bajo costo de comunicación (detalle)

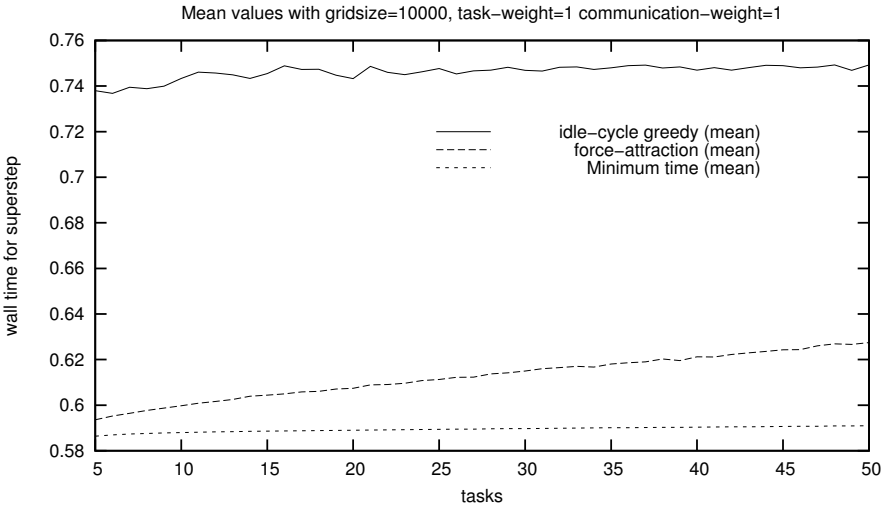
FIGURA 4.2.7. Valores medios para bajo costo de comunicación tamaño de malla creciente

En la figura 4.2.7 se muestra el detalle de la figura 4.2.4, donde se aprecia claramente el punto de intersección entre los desempeños del método voraz en ciclos de cómputo y el método de Campo de Fuerza.

En la figura 4.2.8 se observa el comportamiento con una malla dinámica cinco veces más grande que la que se presenta en la figura 4.2.7. En este caso, con los mismos conjuntos de tareas paralela a asignar, el método de Campo de Fuerza es mejor que cualquiera de los otros métodos analizados. También se nota que la pendiente que define el deterioro en el desempeño de método de Campo de Fuerza ha disminuido notablemente con relación a la que muestra en la figura 4.2.7. Esto indica que el método de Campo de Fuerza es escalable y trabaja mejor a mayor población de la malla dinámica.

Bajo costo de comunicación (detalle malla 5X)

FIGURA 4.2.8. Valores medios para bajo costo de comunicación tamaño de malla creciente



Bajo costo de comunicación (detalle malla 10X)

FIGURA 4.2.9. Valores medios para bajo costo de comunicación tamaño de malla creciente

En la figura 4.2.9 se observa el comportamiento con una malla dinámica diez veces más grande que la que se presenta en la figura 4.2.7.

En este caso, con los mismos conjuntos de tareas paralela a asignar, el método de Campo de Fuerza es notablemente mejor que cualquiera de los otros métodos analizados. También se nota que la pendiente que define el deterioro en el desempeño de método de Campo de Fuerza sigue disminuyendo con relación a la que muestra en la figura 4.2.8. Esto confirma que el método de Campo de Fuerza incrementa su desempeño en relación a los otros métodos analizados conforme se crece la población de la malla dinámica. Este es un detalle muy importante para un método de asignación a aplicarse en una malla dinámica.

En los resultados que arriba se muestran, se observa que conforme crece el número de computadoras participantes en la malla dinámica, el método por campo de fuerza funciona mejor que los voraces. El método aleatorio no presenta mucha variación al crecimiento de la malla. Aún en aplicaciones donde el costo en las comunicaciones es bajo, el método de Campo de Fuerza es una opción importante a considerar cuando se utilizan mallas dinámicas grandes para efectuar cómputo en paralelo bajo el modelo BSP.

Estos resultados indican la dirección que los algoritmos de asignación de tareas deben tomar con las aplicaciones en paralelo en mallas dinámicas de tamaño relativamente pequeño. El efecto conforme crece el tamaño de malla confirma la tendencia del algoritmo de campo de fuerza a sobrepasar a los otros métodos en cuanto a rendimiento.

4.3. Aplicación a la tomografía sísmica de pozos

En la industria petrolera, los yacimientos se encuentra a cada vez mayor profundidad. Debido a que estos yacimientos están atrapados dentro de delgadas capas de la corteza terrestre, la economía determina que los pozos de explotación se deben perforar de tal manera de que entren al yacimiento de manera horizontal y atraviesen el yacimiento a lo largo de la trayectoria más larga. Esto asegurará que la superficie de contacto será lo mayor posible y la producción petrolera estará maximizada durante la vida del pozo.

CUADRO 1. Valor relativo del costo de cómputo para las tareas en una tomografía sísmica con 12 pozos

Grupo	Elementos	Costo
1	6	1.0000
2	12	0.9659
3	12	0.8660
4	12	0.7071
5	12	0.5000
6	12	0.2588

Para determinar la configuración geométrica exacta del yacimiento petrolífero y determinar el punto de entrada horizontal, se realizan varios pozos de exploración alrededor de la zona potencial del yacimiento. Ondas sísmicas se generan y se reciben en estos pozos. Al medir el retardo en la velocidad de propagación, las diferencias en la densidad de la roca se pueden determinar y una reconstrucción de la geometría del yacimiento se puede obtener por métodos inversos. Un enfoque distinto es la de utilizar el método directo de resolver la ecuación de onda. El método directo no se usa de manera general debido a la complejidad del problema. Pero conforme los recursos de cómputo continúan creciendo y se agrupan en mallas dinámicas, la posibilidad yace en el futuro. El siguiente material muestra lo que se puede esperar de la utilización de distintos métodos de asignación de tareas para este problema en particular. Por otro lado, los métodos numéricos para la resolución de ecuaciones diferenciales parciales —como la ecuación de onda— se irán tornando más complejos y más demandantes de poder de cómputo para la obtención de mejores resultados. Un ejemplo de como desarrollar un método de diferencias finitas para la obtención de mejores resultados —al precio de una mayor demanda de recursos de cómputo— lo podemos encontrar en el apéndice.

Para una tomografía sísmica de 12 pozos con configuración simétrica, existen 66 trayectorias de ondas sísmicas para ser resueltas mediante la ecuación de onda. Cada trayectoria de onda —ilustrado en la

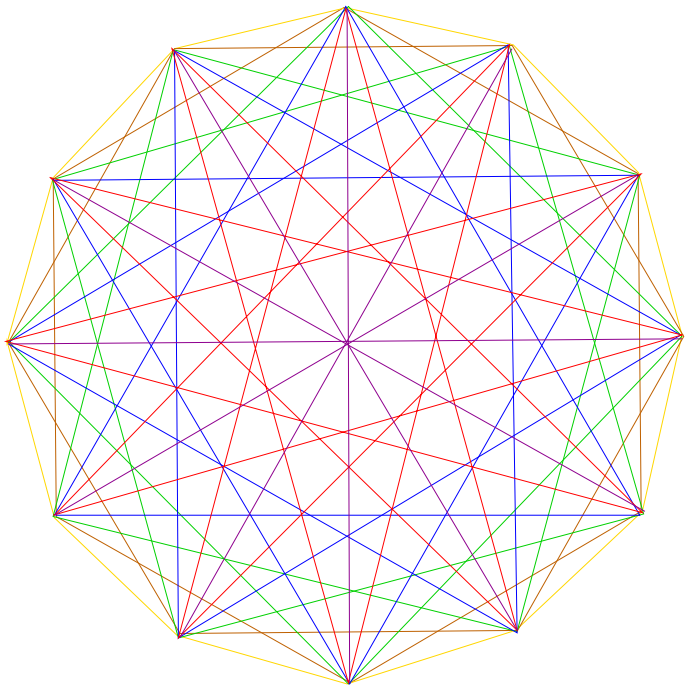


FIGURA 4.3.1. Trayectorias sísmicas para las ondas sísmicas en una tomografía de 12 pozos en colocación simétrica

figura 4.3.1— representa una tarea a ser resuelta durante el superpaso. En esta figura los pozos exploratorios están localizados alrededor de un yacimiento potencial y las trayectorias de onda a ser resueltas para cada iteración se ilustran por las líneas que conectan a los pozos. La ecuación diferencial parcial a resolver es hiperbólica y puede ser resuelta por métodos finitos bien conocidos como los de Lax–Wendroff o de MacCormack o bien por métodos nuevos como el de la Rayuela-E4 presentado en el apéndice. Métodos más complicados tales como variaciones de Godunov no se requieren debido a que no se ven involucrados las discontinuidades de fase ni fuertes ondas de choque. Por ende los requerimientos de cómputo para resolver cada problema de onda es directamente proporcional a la distancia entre las ondas. Los requerimientos de cómputo para todas las trayectorias de onda no son

iguales, pero existe una relación de proporcionalidad. Existen seis grupos. El primero tiene diámetro completo y seis elementos. Todos los demás grupos tienen 12 elementos. Tomando el diámetro completo como el costo de cómputo normalizado a la unidad, la relación entre los grupos se muestra en la tabla 4.3.

Si una sola instancia del problema de onda requiere del 10 % de una computadora de escritorio típica (este porcentaje dependerá de la complejidad relativa del problema y puede ser variada como se desee para producir resultados del simulador), el DGS puede generar las tareas paralelas con la complejidad apropiada a partir de la información de la tabla 4.3.

Las figuras 4.3.2–4.3.4 muestran los resultados de las simulaciones conforme crece el tamaño de la malla.

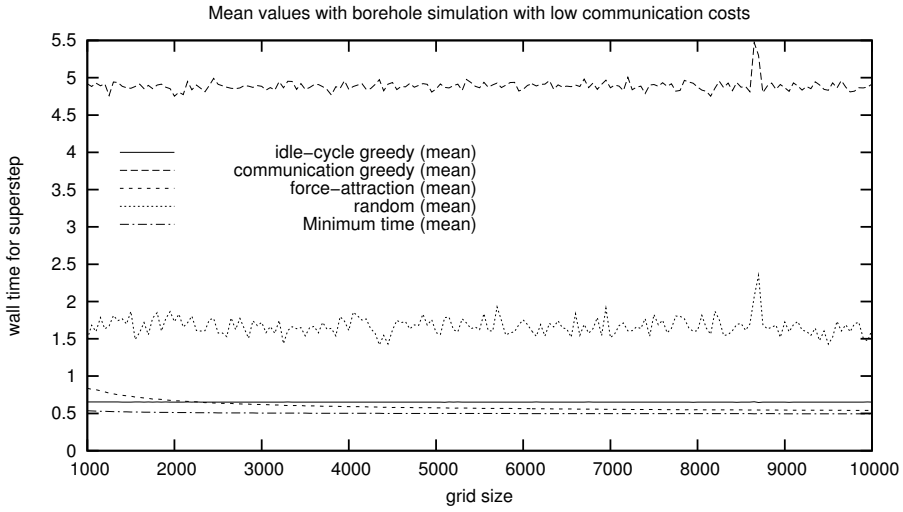


FIGURA 4.3.2. Resultados comparativos para bajo costo de comunicaciones en la simulación de pozos de tomografía

En la figura 4.3.2 se muestran los resultados estadísticos de las medias cuando la malla dinámica está interconectada con equipo sofisticado de comunicaciones que se encuentra subutilizado. En esta circunstancia el efecto del costo de las comunicaciones es bajo. Esta figura

muestra que mientras que el método de Campo de Fuerza comienza con tiempos mayores que el voraz respecto a ciclos de cómputo, los tiempos mejoran después de que el tamaño de la malla dinámica llega a 2000 procesadores y continua bajando conforme crece la malla con mayor número de nodos.

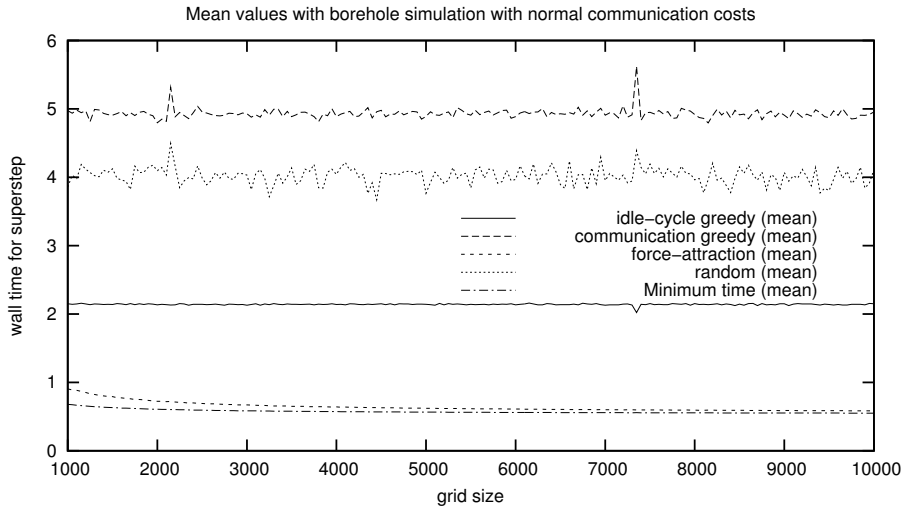


FIGURA 4.3.3. Resultados comparativos para costo normal de comunicaciones en la simulación de pozos de tomografía

En la figura 4.3.3 la red de comunicaciones de la malla dinámica restringe de manera normal la resolución del problema. En este caso la situación muestra que el método de Campo de Fuerza supera todos los otros métodos con respecto al rendimiento del superpaso. De la figura se nota que el método propuesto en este trabajo rápidamente se acerca a la aproximación al mínimo tiempo de ejecución posible para el superpaso. El método voraz en ciclos de cómputo permanece casi constante en 2 unidades de tiempo sin importar el tamaño de la malla. Los métodos aleatorio y voraz en comunicaciones también permanecen constantes —salvo fluctuaciones hacia arriba y abajo— sin importar el tamaño de la malla dinámica.

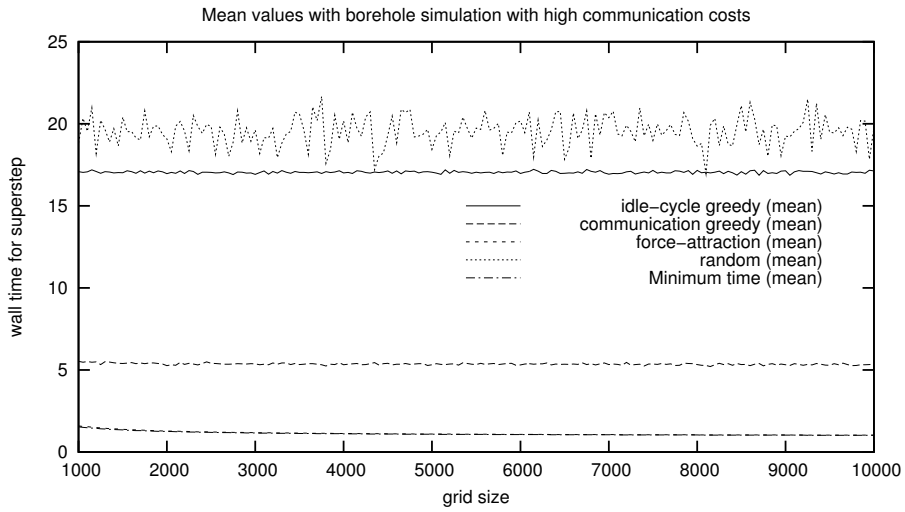


FIGURA 4.3.4. Resultados comparativos para costo alto de comunicaciones en la simulación de pozos de tomografía

En la figura 4.3.4 se tiene que los costos en la comunicación son críticos, y los resultados son inequívocos con respecto a los méritos del método de asignación del Campo de Fuerza. En este caso el desempeño del método de Campo de Fuerza es indistinguible de la aproximación al mínimo. Los desempeños de los otros métodos permanecen constantes sin importar el crecimiento de la malla dinámica. Esto confirma la escalabilidad del método de Campo de Fuerza en el problema de la tomografía de pozos.

4.4. El problema de la recolección de la información

En la implementación del campo de fuerza, el mayor obstáculo a vencer consiste de la recolección de la información. Aunque se ha visto que el método de asignación aleatoria no tiene tan buen rendimiento como el método de campo de fuerza, éste primero presenta la ventaja de que no requiere de recolección de información. Por otro lado presenta la desventaja de que produce rechazos.

Utilizando el simulador DGS es posible tabular el número de rechazos que se producen al utilizar el método de asignación aleatoria.

Con ello para serie de mil corridas del simulador se obtiene la media del número de rechazos.

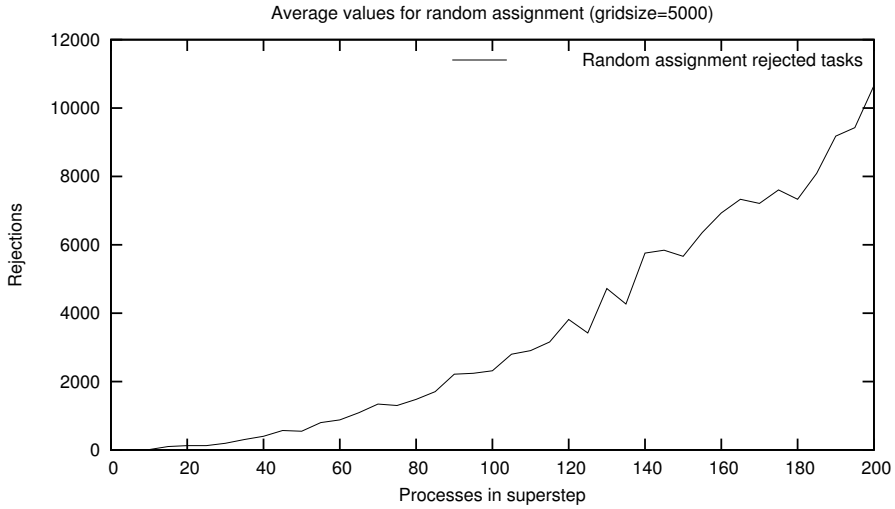


FIGURA 4.4.1. Rechazos de tareas asignadas por método aleatorio

En la figura 4.4.1 se observan los rechazos que se detectan al utilizar el método de asignación aleatoria con el simulador DGS. En esta figura se muestra el promedio de tareas que son rechazadas por nodos remotos debido a la insuficiencia de recursos. Cada punto del eje coordenado representa el promedio de 1000 simulaciones independientes con parámetros distintos. Se nota que el numero de rechazos aumenta de manera significativa conforme aumenta la cardinalidad del conjunto de tareas paralelas a se asignadas. De esta forma, cuando el conjunto de tareas a ser asignado es muy pequeño, casi no se producen rechazos. Pero conforme va creciendo se encuentra que la cantidad de rechazos se incrementa rápidamente. Cada tarea rechazada consume el doble del tiempo que hubiese tardado en llegar la información que permitiera determinar de antemano si el nodo es capaz de resolver la tarea en cuestión. La tarea debe ser enviada y el mensaje de rechazo tiene que regresar. Por ello la pérdida de tiempo puede considerarse como

un costo de recolección de información para el método de asignación aleatoria.

A continuación se presenta los resultados que se obtienen al aplicar la técnica de cortar la recepción de paquetes de información después de que un cierto número de éstos haya arribado, desarrollado en la sección 2.5. Por las consideraciones anteriores, según el tamaño de la malla y el número de tareas a resolver en el superpaso, es de esperar que habrá números de corte donde los métodos de asignación ponderados —como el del campo de fuerza— superarán a cualquier asignación aleatoria.

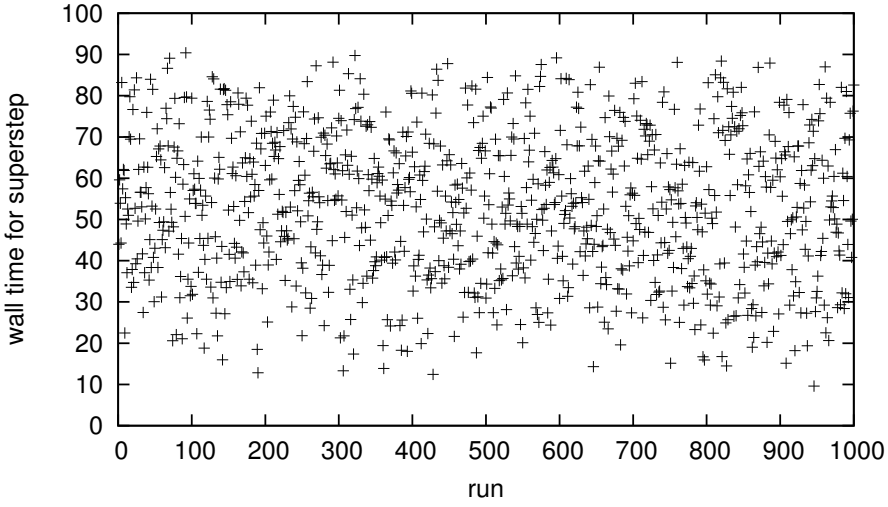


FIGURA 4.4.2. Promedio de tiempo de para superpaso con asignación aleatoria

En la figura 4.4.2 podemos ver los resultados que se obtienen para la ejecución del simulador DGS mediante la asignación aleatoria. El conjunto de tareas a ser asignado en la experimentación tiene una cardinalidad de 20. Una vez más, en esta gráfica, cada corrida en el eje coordenado representa el promedio de 1000 simulaciones bajo condiciones de malla dinámica distintas. En general se puede observar una distribución aleatoria de tiempos de superpaso en el intervalo [10,90].

Esta tendencia confirma que una asignación aleatoria de tareas a procesadores producirá a su vez un tiempo de superpaso aleatorio.

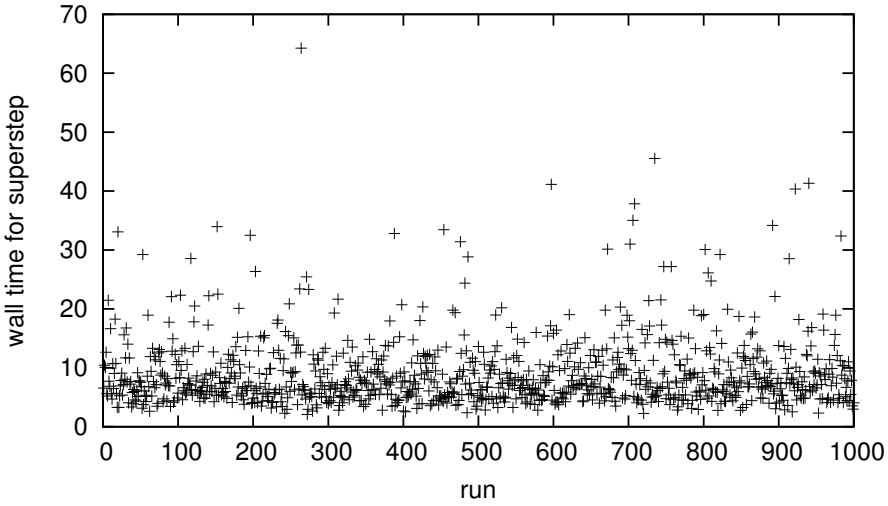
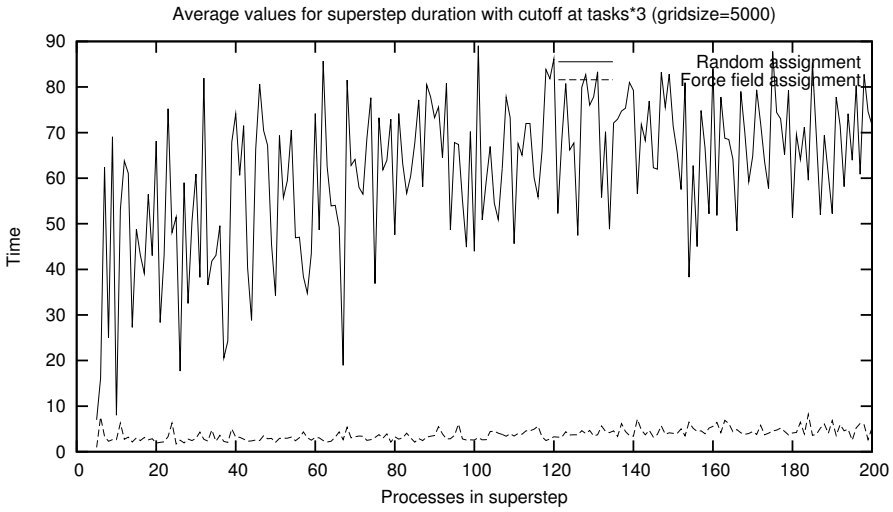


FIGURA 4.4.3. Promedio de tiempo de para superpaso con por Campo de Fuerza

En la figura 4.4.3 podemos ver los resultados que se obtienen para la ejecución del simulador DGS mediante el método de Campo de Fuerza. En esta figura se consideran 20 procesos en una malla dinámica poblada con 5000 nodos, pero se hace un corte de información: solamente se utilizan los primeros 50 mensajes que llegan para la aplicación del método. Es decir, los 4500 nodos con costo de comunicación más alto no se consideran al aplicar el algoritmo del Campo de Fuerza. En los resultados se nota que aún con esta fragmentación de la malla se encuentra que los tiempos del superpaso están en el intervalo [3,65], con la mayoría de los resultados concentrándose en el intervalo [3,10]. Es decir, el método de Campo de Fuerza aún provee buenos resultados. Esto es de esperarse, ya que el planteamiento teórico del método indica que la fuerza decae muy rápidamente con el costo en comunicaciones. Por esta razón los nodos con alto costo de comunicaciones —que son ignorados en el experimento— de todas manera no tendrían mucho efecto, a menos de que tuvieran una cantidad muy grande de

ciclos disponibles. Esto último no ocurre en la malla dinámica, ya que estamos trabajando con computadoras personales de escritorio con características similares entre si.

En los siguientes experimentos tomamos el número de mensajes a esperar como un múltiplo de las tareas que serán asignadas. De esta forma podemos analizar el efecto que se tiene al incrementar la cardinalidad del conjunto de tareas a ser asignado. De las figuras 4.4.4–4.4.6 se puede observar para distintos valores de corte en la recepción de mensajes de información, que el método de asignación por campo de fuerza supera ampliamente en cuanto al rendimiento en la terminación del superpaso a una asignación aleatoria.

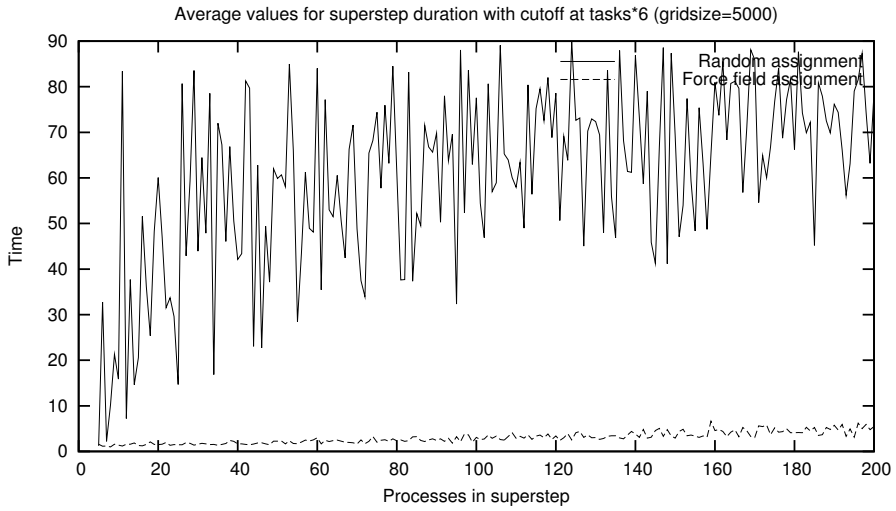


Corte cuando mensajes arribados=tareas*3

FIGURA 4.4.4. Duración del superpaso

En la figura 4.4.4 se observan los resultados en tiempo de terminación del superpaso cuando se dejan de recibir mensajes después de que hayan arribado el triple de tareas que se ejecutarán en el superpaso. Para el método de asignación aleatoria se muestran tiempos de superpaso bastante aleatorios, pero en promedio por encima de las 50 unidades de tiempo relativos a los ciclos de cómputo del procesador de referencia. Así mismo, se puede observar para el método de Campo de

Fuerza poca variación conforme aumenta el número de tareas: mantiene un tiempo de superpaso por abajo de las 10 unidades de tiempo. En este experimento el tamaño de la malla permanece constante en 5000 nodos.

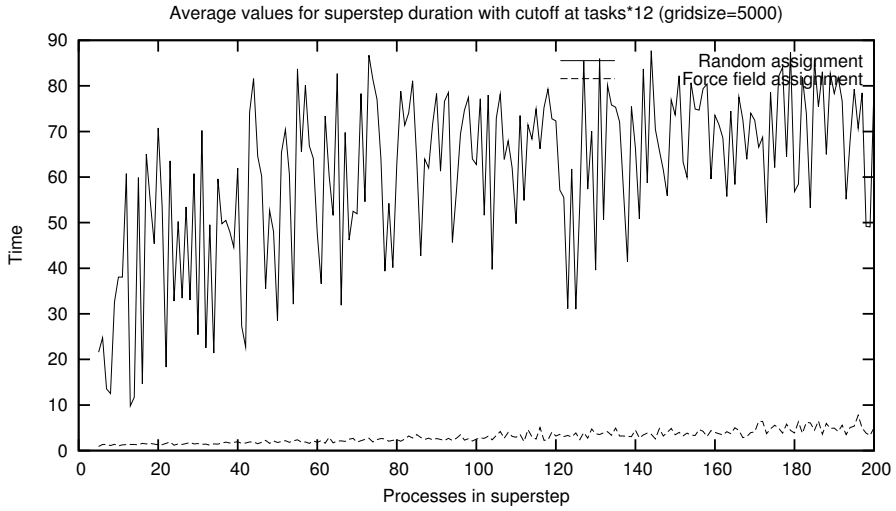


Corte cuando mensajes arribados=tareas*6

FIGURA 4.4.5. Duración del superpaso

En la figura 4.4.5 se observa el resultado de la misma experimentación sobre la malla de 5000 nodos. En este caso la curva de los tiempos de superpaso por asignación aleatoria sigue manteniendo un comportamiento errático. Por otro lado los tiempos que reporta el método de Campo de Fuerza cuando la cantidad de mensajes es 6 veces el número de tareas a ser asignadas, la curva de tiempo de superpaso comienza a estabilizarse. Esto indica que las condiciones para la obtención de una q_{max} que se plantea en la sección 2.5 son alcanzados. Según se plantea en la sección 2.5 la q_{max} será una cota para la cargas remotas. Esta cota será el valor máximo de carga para todos los mensajes que tengan un tiempo de transmisión mayor a los del umbral t_c. Al alcanzar el umbral t_c ya no es necesario seguir esperando paquetes de información. Esto es consecuencia directa del método de Campo de Fuerza, ya que la fuerza decae muy rápidamente con la distancia y los nodos que se encuentran

lejanos no contribuyen significativamente en la asignación de tareas, a menos de que su disponibilidad en ciclos de cómputo sea muy grande. Esta última condición no se cumple en el caso de la malla dinámica, por la manera en que está conformada por computadoras de escritorio de características similares.

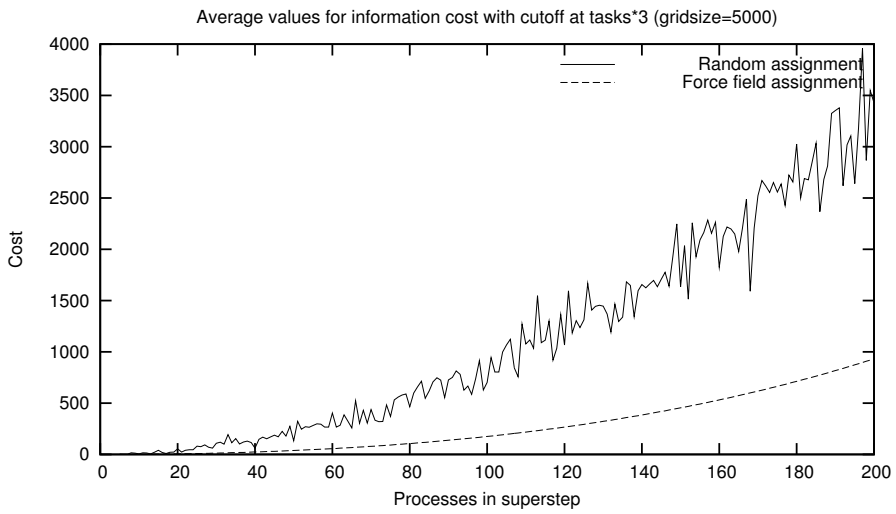


Corte cuando mensajes arribados=tareas*12

FIGURA 4.4.6. Duración del superpaso

En la figura 4.4.6 analizamos el comportamiento cuando el número de mensajes a recibir es no más de doce veces las tareas a ser asignadas. En los resultados de esta figura no se nota mucha diferencia con respecto a las anteriores y se concluye que es innecesario esperar más mensajes antes de aplicar el método de Campo de Fuerza para asignar las tareas del superpaso.

En seguida es necesario analizar el tiempo que el método aleatorio y el método de Campo de Fuerza invierten en la recolección de información. Para esto recordemos que el tiempo perdido por rechazos en el método aleatorio cuenta como tiempo invertido en recolección de información.

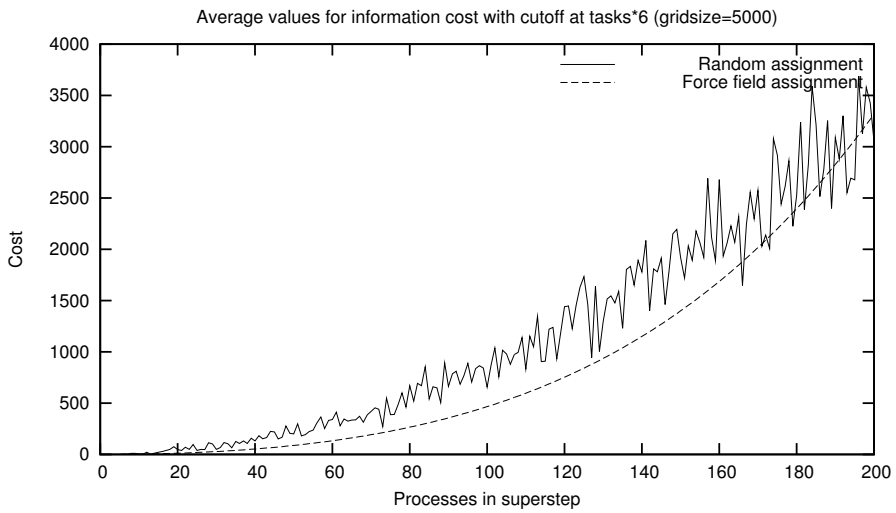


Corte cuando mensajes arribados=tareas*3

FIGURA 4.4.7. Costo de la recolección de información

En la figura 4.4.7 se observa el comportamiento en cuanto al costo de la comunicaciones. En esta figura el corte de mensajes ocurre después de recibir tres veces la cantidad de tareas a asignar. El tamaño de la malla dinámica es de 5000 nodos. El costo por recolección de información en el método de asignación aleatorio es la curva superior de patrón errático que crece de manera casi lineal conforma aumenta el número de tareas a ser asignadas. Por otro lado, en la parte inferior, se observa como una curva continua el costo de información del método de Campo de Fuerza. La manera en que se obtiene este valor es tabulando el tiempo que toma la recepción del último mensaje recibido. Dado que los parámetros de la red son generados por el simulador DGS, este es un valor que podemos tabular. De la figura 4.4.7 es claro que el método de Campo de Fuerza supera a la asignación aleatoria ya que tiene menores tiempos en costo de comunicación y en ejecución del superpaso para el rango de tareas considerado.

En seguida procedemos a analizar el costo de información para los casos en que se esperan seis y doce veces la cantidad de las tareas a ser asignadas.



Corte cuando mensajes arribados=tareas*6

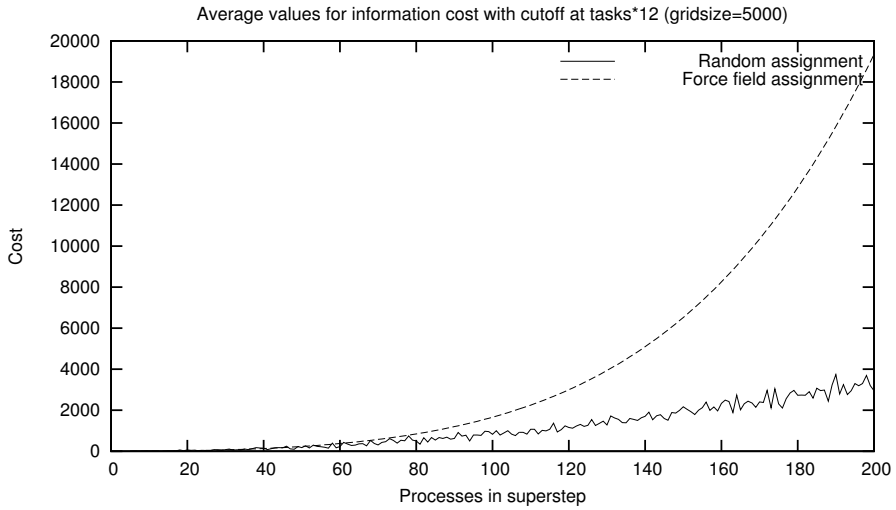
FIGURA 4.4.8. Costo de la recolección de información

En la figura 4.4.8 se muestra el caso en que se espera seis veces el número de tareas a ser asignadas. En este caso la diferencia en costo de información es menor, pero aún es superior el método de asignación por Campo de Fuerza dado que los tiempos invertidos en la recolección de información son menores. Sin embargo la diferencia observada en cuanto al tiempo de superpaso entre las figuras 4.4.5 y 4.4.6 nos indica que el esfuerzo de esperar más mensajes no vale la pena. La q_{max} ha sido rebasada desde las condiciones del experimento mostrado en la figura 4.4.4.

En el caso de que se esperen mensajes para un total de hasta seis veces el número de tareas a ser asignadas en el superpaso, tiempos son equivalentes a lo que se pierden por rechazo de tareas mediante el método aleatorio de asignación. Pero los tiempos de superpaso para este caso, mostrados en la figura 4.4.5, son sustancialmente mejores para la asignación por Campo de Fuerza.

Para efectos de completar el estudio a falta analizar el tiempo de recolección de información para el caso en el que espera la recepción

de doce veces la cantidad de mensajes. Eso lo hacemos en la siguiente figura.



Corte cuando mensajes arribados=tareas*12

FIGURA 4.4.9. Costo de la recolección de información

En la figura 4.4.9 existe un momento en que el costo de recolección por el método de Campo de Fuerza es mayor que el producido por los rechazos en la asignación aleatoria. Se observa que la escala de la gráfica es distinta, ya que los tiempos por recolección de información mediante el método de Campo de Fuerza se elevan rápidamente a partir de que se tienen más de 40 tareas paralelas a ser asignadas. En este caso la cantidad de mensajes que esperados equivale a casi un 10 % de los nodos que componen la malla. Es decir, asignar las cuarenta tareas en los 480 nodos más cercanos de una malla con población de 5000 computadoras.

En este último caso habría que cotejar la relación entre el costo de comunicaciones y el costo de cómputo de la aplicación para apreciar el efecto global en el rendimiento del programa en ejecución. No obstante, como el tiempo de ejecución del superpaso se mantiene relativamente constante cuando se recibe un número de hasta tres veces la cantidad de las tareas a asignar, es lógico que la aplicación de estos esquemas

en los que se esperan más mensajes no ha de aplicarse en la práctica, especialmente si se codifica dentro del algoritmo el ajuste a la curva de Gauss para las capacidades de cómputo de la malla y se obtiene la cifra óptima para el corte de mensajes. Para todos los mensajes que tardan más en llegar que este límite, se considera que la distancia en el campo de fuerza es infinita y por ende la fuerza de atracción es nula.

Conclusiones

En este capítulo hacemos una reseña crítica del método de asignación de tareas propuesto, incluyendo sus limitaciones y sus fortalezas. Finalizamos con una discusión de las líneas de investigación relacionadas que se han abierto y que meritan estudios posteriores.

- Se desarrolló la metodología para caracterizar a las mallas dinámicas de cómputo de gran tamaño mediante la Ley de Grandes Números y se aplicó para la codificación de un simulador para medir los rendimientos de los distintos métodos de asignación. Con ello se va más allá de los simuladores propuestos en la literatura **[23, 7, 14]** y los resultados han sido aceptado para su publicación en una revista internacional arbitrada **[32]**.
- Se desarrollo y verificó el escalamiento del método de asignación de tareas mediante Campo de Fuerza para las mallas de computadoras de escritorio. La comparación del método con otros esquemas canónicos de asignación resultó favorable. Los resultados del desarrollo del método han sido publicados en el congreso internacional *Grid and Cooperative Computing* efectuado en la ciudad de Beijing en el 2005 y publicados por Springer-Verlag **[30]**.
- Se hizo el planteamiento teórico de la manera en que el problema de la recolección de la información puede ser resuelto para el método de asignación por Campo de Fuerza y se verificaron estos resultados mediante las simulaciones de mallas de computadoras de escritorio con un tamaño de 5000 nodos. Estos resultados fueron publicados por CSREA Press y presentados

en el congreso internacional *Grid and Cooperative Computing* celebrada en la ciudad de Las Vegas en 2006 **[31]**.

- Aunque sólo se relaciona de manera indirecta con el objetivo principal de esta tesis, una contribución adicional desde el punto de vista de desarrollo tecnológico es el análisis del problema de la simulación de la tomografía de pozos mediante la ecuación de onda cuando se aplica una distribución de tareas mediante el método de campo de fuerzas en una malla dinámica. Aunado a esto, se elaboró un método numérico de mayor precisión para la resolución de ecuaciones diferenciales parciales hiperbólicas por diferencias finitas mediante una extensión al plano complejo (método Rayuela-E4). En este campo existe una búsqueda por métodos de mayor orden **[83]** y el método Rayuela-E4 es una nueva alternativa. Este método está en proceso de revisión para su publicación y se encuentra descrito en el apéndice al final de este documento.

En este estudio la aplicabilidad del método de Campo de Fuerza se limita a la asignación de tareas de las aplicación paralela programada bajo el modelo BSP, en una malla de computadoras de escritorio. La característica principal de las mallas de computadoras de escritorio es que los parámetros que las caracterizan son dinámicos y pueden ser modelados mediante distribuciones normales. Por otro lado una de las fortalezas incluye en que es un método que permite la asignación de tareas desde distintos puntos de la malla de computadoras de escritorio: los conjuntos de tareas a ser asignados para una aplicación paralela encuentran limitaciones por la Ley de Amdahl y en consecuencia cada asignación ocupa una parte reducida del universo global de procesadores disponibles en la malla.

Un importante aspecto de este trabajo de investigación es que no es completa y deja muchas líneas de investigación abiertas. Entre las líneas abiertas de investigación podemos señalar las siguientes.

¿Cual es el múltiplo óptimo de la cardinalidad del conjunto de tareas a ser asignadas que corresponda a la cantidad de mensajes a esperar antes de iniciar la asignación? Este es un problema que requiere analizar la naturaleza de los programas paralelos en ejecución.

¿Cuál es la aplicación del método de Campo de Fuerza bajo otros modelos de computadora paralela? En este caso habría que valorar medidas de rendimiento, escalabilidad y reproducibilidad de la manera en la que se efectuó para el modelo BSP.

¿Cuál es la aplicabilidad del método de Campo de Fuerza a otros sistemas que no presenten las características de la malla de computadoras de escritorio? En este caso habría que hacer un estudio de la topología de la máquina paralela en cuestión para determinar la forma óptima de la ecuación que determina el campo. Al variar la forma de la ecuación se pueden producir campos con geometrías distintas, como la cilíndrica por ejemplo.

¿Cual es la forma ideal de calcular las fuerzas repulsivas para determinar rechazos? En el trabajo desarrollado en esta tesis se trabajo con un campo de fuerzas escalar. Pero el concepto puede ser extendido a un campo de fuerzas vectorial y de esta forma las fuerzas repulsivas toman importancia considerable. De tal manera se permitiría optimizar los recursos en área específicas de la malla de computadoras de escritorio. Es decir, la aplicación de fuerzas repulsivas y direcciones para establecer áreas que rechazarán tareas aunque haya nodos en su interior con alguna capacidad de procesamiento disponible. Más trabajo se tiene que desarrollar en esta dirección antes de hacer afirmaciones a favor o en contra.

Otra línea abierta en la investigación se refiere a la utilización de duplicación de tareas para el método de Campo de Fuerza. ¿Cómo se afecta su rendimiento y cual es la comparativa con utilizar la duplicación de tareas en los otros métodos de asignación analizados? Esta duplicación de tareas sería un paso para evitar deterioro en el desempeño que surgen con la falla de nodos en la realización de una tarea.

Para su estudio habría que extender el DGS en consecuencia. Habría que el modelo de Campo de Fuerza para hacer una asignación idónea.

La resolución de problemas en una malla dinámica de computadoras de escritorio es factible gracias a la gran cantidad de computadoras de escritorio que existen en la industria y que no están siendo utilizadas a su máxima capacidad. Por esta razón el método de Campo de Fuerzas para la asignación de tareas resulta un buen balanceador de carga para aplicaciones en paralelo bajo el modelo BSP cuando este problema se resuelve en una malla dinámica. En este caso, el tiempo de ejecución del superpaso es la variable objetivo a optimizar.

Existen varios métodos para implementar la determinación de disponibilidad de recursos en los sistemas, utilizando cantidades tales como los ciclos de cómputo no utilizados por unidad de tiempo y la memoria libre disponible. La implementación del código para ejecutar programas mediante la asignación de tareas por el algoritmo de Campo de Fuerza no se contempla en este trabajo y es tema de trabajo futuro.

De igual manera, un tema para desarrollo es la implementación de algoritmos para la distribución de la información que caracteriza el estado de la malla de cómputo y resolver los problemas prácticos que esto conlleva. Es decir, en esta tesis no se tiene planteado el método por el cual se enviaran los paquetes de información que caracterizan a la malla. Este es un problema de implementación que podría ser resuelto de distintas maneras pero que va más allá del objetivo de investigación de esta tesis.

Se debe notar que el método de campo de fuerza es de orden de complejidad $O(nm)$, ya que depende de las tareas y los procesadores. Esta dificultad, sin embargo, se minimiza mediante la aplicación con el método de campo de fuerza, ya que la fuerza que ejercen los nodos para la atracción de tareas decae rápidamente con el costo computacional. Esto permite aplicar el método de una manera local desde distinto puntos de acceso a la malla para distintos usuarios, haciendo un corte después de recibir un número limitado de paquetes de información.

Se ha visto en este trabajo que el método de campo de fuerza supera a los otros métodos contemplados cuando el tamaño de la malla dinámica de cómputo crece en tamaño. Además, dado que es posible calcular de manera dinámica las distribuciones normales que caracterizan el estado de la disponibilidad de recursos del sistema, se puede obtener la cota para el costo en comunicaciones para aplicar el método de asignación por campo de fuerza sin afectar de manera apreciable el desempeño del método.

Apéndice

En este apéndice se incluye material relacionado de manera indirecta con los objetivos principales. Entre esto se incluye una descripción de la ecuación de onda en la tomografía sísmica de yacimientos petrolíferos. En seguida se detalla la manera en que se resuelve el problema tradicionalmente por diferencias finitas. Después hacemos un planteamiento de cuarto orden mediante números reales. Continuamos con la extensión del método al plano de los números complejos. Para el El método desarrollado se hace el analisis de estabilidad lineal y por último se hace una comparativa de desempeño con el método de MacCormack, demostrando las condiciones en las cuales el nuevo método supera al establecido.

La ecuación de onda en la tomografía

Una de las ecuaciones que mayor impacto han tenido en el siglo XX con continua trayectoria en lo que va del presente es la ecuación de onda. En la Física se ha determinado con certeza que todas las partículas del universo existen como una dualidad particula/onda. Con este antecedente, no es de sorprenderse que la ecuación de onda juegue un importante papel en la prospección petrolera.

A partir de la década de los setenta, la producción petrolera en México tuvo un importante auge que subsiste hasta la fecha. Una característica importante de este auge ha sido el hecho de que una gran parte de las reservas localizadas y explotadas se encuentra en el lecho marino de la plataforma continental. Esto no quiere decir que los yacimientos en tierra firme se hayan agotado. Pero entonces, ¿por qué

perforar pozos en zona marina cuando es más barato hacer lo mismo en tierra firme? La respuesta yace en términos económicos de la prospección.

A diferencia de los océanos que cubren una gran extensión, los yacimientos petroleros se encuentran distribuidos en cúmulos de material orgánico que quedó atrapado entre capas de roca que se formaron por acontecimientos geológicos. El reto de la ingeniería es localizar estos yacimientos y, ciertamente, la localización es más eficaz en la plataforma continental, por lo que veremos a continuación.

Cuando el material orgánico que formó el petróleo quedó atrapado bajo capas de roca, durante millones de años sufrió complejas transformaciones químicas y entre los productos de estas reacciones están los hidrocarburos ligeros tales como metano y propano. Una característica de los hidrocarburos ligeros es su elevada presión de vapor, lo cual incrementa la presión que ejerce el yacimiento a la roca que lo rodea. La consecuencia natural es la deformación de la capa de roca en una especie de promontorio o pústula enterrada. Por esta razón, si se localizan estos promontorios, existe una gran probabilidad de que haya un yacimiento petrolífero asociado. Pero si los yacimientos se localizan a gran profundidad, encontrar estas deformaciones en las capas de roca es la primera parte del problema que hay que resolver.

Para lo anterior se aplica el método de la tomografía de reflexión. Las ondas sísmicas, que se caracterizan por una longitud de onda de uno hasta varios cientos de metros, son capaces de atravesar la corteza terrestre, pero a su vez, como cualesquiera ondas, presentan una cierta reflexión cuando la densidad del medio por la que viajan cambia. De esta forma, si generamos ondas sísmicas en superficie y detectamos los tiempos e intensidades de las reflexiones de la onda, podemos determinar las distancias que existen entre el punto de generación de la onda sísmica y las capas de roca de distinta densidad. Una sola detonación no sirve de gran cosa, pero efectuando una gran cantidad de detonaciones en distintas puntos permite generar una imagen tridimensional. El problema de hacer esto en tierra firme es que el procedimiento es

muy oneroso y se dificulta por la topografía del terreno o la negativa de los habitantes a permitir daños en sus parcelas.

Por otro lado, el mismo estudio es mucho más económico y preciso si se realiza en el mar sobre la plataforma continental. En este último caso, lo único que se requiere es un barco equipado con geoposicionador y un cable de varios kilómetros de longitud equipado con geófonos para detectar la reflexión de las ondas sísmicas generadas por una serie continua de detonaciones. Con ello se obtiene una malla de puntos experimentales con los que es posible conocer a gran detalle las irregularidades en las capas de roca a profundidad. Sin embargo, esta primera utilización de las ondas sísmicas no es suficiente.

Con la tomografía de reflexión es posible identificar zonas geológicas donde existe alta probabilidad de encontrar yacimientos petrolíferos. Sin embargo, no se tiene la certeza de que exista petróleo o la forma exacta en la que está distribuida. Dado que la perforación marina es mucho más cara que la terrestre, y los yacimientos están atrapados entre capas horizontales de roca, es deseable perforar el yacimiento de manera horizontal de tal forma que el pozo lo atraviese por su parte más ancha. Para conocer la forma del yacimiento es necesario recurrir a la tomografía sísmica de pozos exploratorios, que es el principal interés de este trabajo.

Antes de describir la tomografía de pozos, veamos por qué la ecuación de onda nos permite efectuar dos tipos de estudio radicalmente distintos con la misma onda.

La ecuación de onda homogénea y unidimensional puede expresarse así:

$$P_{tt} + a^2 P_{xx} = 0,$$

donde P es la perturbación acústica de presión (Pa) y a es la velocidad de la onda en el medio. La ecuación de onda también se puede escribir como un sistema de dos ecuaciones diferenciales de convección

acopladas:

$$\rho u_t + P_x = 0$$
$$P_t + \rho a^2 u_x = 0$$

donde u es la velocidad acústica.

Dado que la velocidad acústica depende de la densidad del medio por el que viaja la onda, habrá un aletargamiento en la velocidad cuando pase por un medio de menor densidad. Este detalle es la clave que permite detectar los yacimientos mediante la tomografía sísmica de pozos.

Resolución por diferencias finitas

Muchos problemas en la física-matemática pueden ser reducidos a ecuaciones diferenciales parciales (EDP). De especial importancia son las EDP de segundo orden[**76**]:

$$\Phi(x, y, u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}) = 0.$$

De manera análoga, la ecuación puede ser escrita para un número mayor de variables independientes. La ecuación es lineal con respecto de la derivada de mayor orden si tiene la forma:

$$a_{11}u_{xx} + 2a_{12}u_{xy} + a_{22}u_{yy} + \Phi_1(x, y, u, u_x, u_y) = 0.$$

Las ecuaciones diferenciales parciales lineales se pueden distinguir en las elípticas, parabólicas e hiperbólicas. De particular interés en campos tales como la dinámica computacional de fluidos son las ecuaciones diferenciales hiperbólicas, las cuales tienen la forma canónica:

$$u_{x_1x_1} = \sum_{i=2}^{N} x_i x_i + \Phi_2.$$

Debido al tiempo finito de propagación de señales en las EDP hiperbólicas, las ecuaciones en diferencias finitas (EDF) se pueden utilizar para resolver este tipo de ecuaciones de manera aproximada. Generalmente una solución analítica a la EDP no es posible, y por ello una

solución numérica se debe utilizar para aproximar la solución. Al hacer esto, la solución exacta a la EDP se aproxima por una función f, la cual sólo se conoce por los valores que toma dentro del dominio del problema conforme la variable de tiempo se avanza. En este trabajo tratamos con el problema de Cauchy para la ecuación de onda homogénea, una EDP lineal hiperbólica:

$$F_{tt} + F_{xx} = 0.$$

La teoría matemática nos indica que existe exactamente una función F que satisface la EDP y que pasa por un punto dado. Los métodos de marchar EDP hiperbólicas aproximarán a F con una solución numérica, a la cual nos referimos en este trabajo como f.

Al elaborar algoritmos para la generación de f, las técnicas modernas siguen uno de dos enfoques. La primera es clásica en su naturaleza: la f se asume de clase C-infinito (todas las derivadas son continuas). Con este supuesto, la f puede ser expresada como una serie de expansión de Taylor, lo que permite el desarrollo de formulas de aproximación para las derivadas que aparezcan en la EDP. Por ello, la naturaleza continua de la solución aproximada se asemeja a la solución exacta de la teoría matemática. Por otro lado, Godunov **[35]** propuso un enfoque distinto para obtener f al asumir que la solución es continua únicamente por secciones, convirtiéndose en discontinua entre los elementos de la malla de iteración. Para cada elemento de la malla se resuelve la ecuación de conservación (forma integral de la EDP). Este método se utiliza como una alternativa o en combinación con los métodos clásicos para obtener soluciones aproximadas a problemas de EDP con discontinuidades fuertes. Este es el caso en el simulado de explosiones nucleares o de trayectorias de flujo para gases de combustión en flujos supersónicos, mas no así para el problema de la prospección petrolera.

El incremento en el poder de cómputo ahora permite que la EDF pueda ser resuelto mucho más rápido que antes. Por ello, incrementos en la calidad de los resultados deben ser buscados como una meta en la utilización de este poder creciente de cómputo. En el enfoque clásico

para establecer una EDF asociado a una EDP, las derivadas parciales se aproximan por medio de una expansión de Taylor truncada. Al truncar después del primer término de la derivada, por ejemplo, una ecuación de primero orden (en relación a la precisión de la derivada) se obtiene:

$$f_x(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{4.4.1}$$

Y al truncar después del segundo termino de la derivada se obtiene una aproximación de segundo orden:

$$f_x(x) \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \tag{4.4.2}$$

Aplicando la aproximación de primer orden de tiempo y de segundo orden al espacio, se obtiene el método FTCS. El método BTCS utiliza las mismas aproximaciones pero las aplica en el sentido inverso del tiempo, con lo cual se obtiene un sistema de ecuaciones que debe ser resuelto por métodos iterativos para cada paso de tiempo. Otros métodos clásicos que utilizan aproximaciones de primero y segundo orden son los de contra-viento (*upwind*), Lax[**52**], Lax-Wendroff[**53**], MacCormack[**56**] y el Rayuela-FTCS/BTCS (*hopscotch*)[**38**]. Una descripción ilustrativa de estos métodos y su aplicación a la ecuación de onda puede encontrarse en el libro de Hoffmann [**44**].

En este trabajo se sigue el enfoque clásico, pero se extiende el desarrollo hacia el plano complejo para poder utilizar el método de la Rayuela con una aproximación de cuarto orden para la derivada espacial. El método desarrollado constituye una mejora en cuestión de precisión a los algoritmos clásicos utilizados en la actualidad.

Esquema de números reales

Como se muestra en el trabajo de Alford [**4**], Levander [**54**] y Yao [**83**], un método de mayor orden para resolver la ecuación de onda por simulación numérica para el propósito de la prospección petrolera

ha sido recibido con interés por parte de la industria petrolera internacional. La diferencia con los citados métodos y el aquí presentado es el nivel de complejidad. Nuestro enfoque es más fácil de implementar y también puede ser aplicado a otras ecuaciones diferenciales parciales hiperbólicas que requieren de simulación numérica.

Comencemos al examinar la aproximación de cuarto orden para la derivada centrada simétricamente. Esta ecuación se obtiene mediante manipulación algebraica de la expansión de Taylor y el truncamiento de los términos a partir de la quinta potencia:

(4.4.3)
$$f'(x) \approx \frac{2}{3\Delta x}\left[f(x+\Delta x) - f(x-\Delta x) - \frac{1}{8}\left(f(x+2\Delta x) + f(x-2\Delta x)\right)\right]$$

Considerando la técnica convencional de resolución que transforma la ecuación de onda en un sistema de dos ecuaciones de convección acopladas, las EDF basadas en la aproximación anterior llevan a la formula de iteración que se muestra en las ecuaciones siguientes. Esto se hace al combinar las aproximaciones de cuarto orden para la variable espacial y la de primer orden para la variable de tiempo:

(4.4.4)
$$f(t+1, x_i) = f(t, x_i) - \frac{2a}{3}\left(g(t, x_{i+1}) - g(t, x_{i-1}) - \frac{1}{8}(g(t, x_{i+2}) - g(t, x_{i-2}))\right)$$

(4.4.5)
$$g(t+1, x_i) = g(t, x_i) - \frac{2a}{3}\left(f(t, x_{i+1}) - f(t, x_{i-1}) - \frac{1}{8}(f(t, x_{i+2}) - f(t, x_{i-2}))\right)$$

Mientras que la EDF que resulta de la aproximación de cuarto orden espacial es sencilla de implementar para la simulación numérica, un análisis de tipo Warming-Hyett **[80]** no es concluyente en cuanto a la consistencia de la EDF con la EDP original a ser simulada. La experimentación numérica sugiere la inconsistencia, lo cual no es correcto como demostramos abajo.

Esquema de números complejos

El primer detalle que se puede notar de la formula de aproximación es que mientras los métodos de primer y segundo orden requieren ir una distancia Δx para obtener una diferencia finita, para un método de cuarto orden sólo se encuentran dos puntos a lo largo del eje real a una distancia Δx. Los otros puntos se localizan al doble de esa distancia. Si aquellos puntos pueden ser reemplazados por puntos a la misma distancia Δx, entonces el eje complejo debe ser utilizado. Usando esta idea, la solución a la EDP puede ser visualizado no como una curva en la dimensión real, sino como una superficie en el plano complejo. El Teorema de Cauchy-Kovalevskaya nos garantiza que la solución se extiende hacia el plano complejo y que este enfoque es válido.

Considere las expansiones de Taylor siguientes:

(4.4.6)

$$f(x + \mathrm{i}\Delta x) = f(x) + \mathrm{i}f'(x)\Delta x - \frac{1}{2!}f''(x)\Delta x^2 - \frac{\mathrm{i}}{3!}f'''(x)\Delta x^3 + O(\Delta x^4)$$

(4.4.7)

$$f(x - \mathrm{i}\Delta x) = f(x) - \mathrm{i}f'(x)\Delta x - \frac{1}{2!}f''(x)\Delta x^2 + \frac{\mathrm{i}}{3!}f'''(x)\Delta x^3 + O(\Delta x^4)$$

Lo que sigue de estas ecuaciones y de la manipulación algebraica direcat es la fórmula de aproximación de cuarto orden en formato complejo:

(4.4.8)

$$f'(x) \approx \frac{1}{4h}\left[f(x + \Delta x) - f(x - \Delta x) - \mathrm{i}\left(f(x + \mathrm{i}\Delta x) + f(x - \mathrm{i}\Delta x)\right)\right]$$

Para la ecuación de onda, las EDF basadas en la aproximación anterior llevan a las formulas de iteración que se muestran a continuación. Esto se hace al combinar aproximaciones de cuarto orden para la variable espacial y de primer orden para la variable de tiempo:

$$f(t+1,x_i) = f(t,x_i) - \tfrac{a}{4}\,(\, g(t,x_{i+1}) - g(t,x_{i-1}) - \mathrm{i}(g(t,x_i+\mathrm{i}\Delta x) - g(t,x_i-\mathrm{i}\Delta x))\,) \tag{4.4.9}$$

$$g(t+1,x_i) = g(t,x_i) - \tfrac{a}{4}\,(\, f(t,x_{i+1}) - f(t,x_{i-1}) - \mathrm{i}(f(t,x_i+\mathrm{i}\Delta x) - f(t,x_i-\mathrm{i}\Delta x))\,) \tag{4.4.10}$$

Los valores para la función a lo largo del eje imaginario se pueden aproximar de la serie de Taylor en el nivel de tiempo correspondiente al reemplazar los valores de las derivadas por aquellos obtenidos por manipulación algebraica:

$$f'(x) = \frac{2}{3h}\left[a - b - \frac{1}{8}c + \frac{1}{8}d\right], \tag{4.4.11}$$

$$f''(x) = \frac{4}{3\Delta x^2}\left[a + b - \frac{1}{16}(c+d) - \frac{7}{8}f(x)\right], \tag{4.4.12}$$

$$f'''(x) = -\frac{1}{\Delta x^3}\left[a - b - \frac{1}{2}(c-d)\right], \tag{4.4.13}$$

$$f''''(x) = \frac{4!}{6\Delta x^4}\left[-a - b + \frac{1}{4}(c+d) + \frac{3}{2}f(x)\right], \tag{4.4.14}$$

$a = f(x+\Delta x)$, $b = f(x-\Delta x)$, $c = f(x+2\Delta x)$, $d = f(x-2\Delta x)$.

Para su referencia más adelante, los algoritmos que utilizan las ecuaciones anteriores para calcular los valores de la función solución a lo largo del eje complejo se refieren como **imagplus** and **imagminus**. Los valores que no están sobre el eje real no tienen ningun significado físico, pero son útiles desde el punto de vista matemático.

La EDF que se obtiene para la ecuación de onda a partir de las aproximaciones complejas es consistenten con la ecuación diferencial parcial, como lo demuestra un análisis por el método de Warming and Hyett **[80]**.

Análisis de estabilidad lineal

En esta sección se explica por que el método hacia adelante en tiempo con aproximación de cuarto orden para el espacio es incondicionalmente inestable para la ecuación de onda. La técnica utilizada es la indicada en [**44**]. Esto es lo que se espera dada su semejanza con el método FTCS el cual también es incondicionalmente inestable. Veamos el análisis de estabilidad por el método de von Neumann. Para la ecuación de convección, la fórmula de iteración hacia adelante en tiempo es:

$$\begin{aligned} f(t+1,x_i) = \quad & f(t,x_i) - \tfrac{a}{4}\,(\, f(t,x_{i+1}) - f(t,x_{i-1}) \\ & -\mathrm{i}(f(t,x_i+\mathrm{i}\Delta x) - f(t,x_i-\mathrm{i}\Delta x))\,) \end{aligned}$$

De aquí se obtiene el término generalizado de la serie de Fourier en la dirección real:

$$f(x_j) = c\mathrm{e}^{\mathrm{i}jk\Delta x},$$

donde la constante

$$k = \frac{\pi}{L}m$$

se conoce como el número de onda. Con ello se obtiene el término generalizado en la dirección imaginaria:

$$f(x_j \pm \mathrm{i}\Delta x) = c\mathrm{e}^{\mathrm{i}k(j\Delta x \pm \mathrm{i}\Delta x)} = f(x_j)\mathrm{e}^{\mp k\Delta x},$$

de donde se desprende:

$$\begin{aligned} f(t+1,x_j) &= f(t,x_j) - \frac{af(t,x_j)}{4}\left[\mathrm{e}^{\mathrm{i}k\Delta x} - \mathrm{e}^{\mathrm{i}k\Delta x}\right] - \mathrm{i}f(t,x_j)\left[\mathrm{e}^{-k\Delta x} - \mathrm{e}^{k\Delta x}\right] \\ &= f(t,x_j)\left[1 - \frac{a}{4}(2\mathrm{i}\,\mathrm{sen}\theta - 2\mathrm{i}\cos\theta)\right]. \end{aligned}$$

Por ello el factor de amplificación:

$$G = 1 - \frac{a}{4}(2\mathrm{i}\,\mathrm{sen}\theta - 2\mathrm{i}\cos\theta).$$

Lo cual indica que el método hacia adelante en el tiempo es incondicionalmente inestable. Por otro lado, el método hacia atrás en tiempo es incondicionalmente estable para la ecuación de onda como veremos a continuación.

La fórmula de iteración para la EDF es:

$$\begin{aligned} f(t,x_i) = \quad & f(t+1,x_i) - \tfrac{a}{4}[f(t+1,x_{i+1}) - f(t+1,x_{i-1}) \\ & -\mathrm{i}(f(t+1,x_i+\mathrm{i}\Delta x) - f(t+1,x_i-\mathrm{i}\Delta x))] \end{aligned}$$

De donde se obtiene:

$$|G| = \left| \frac{1}{1 - \frac{ai}{2}\,\mathrm{sen}\theta - 2\mathrm{i}\cos\theta} \right|$$

Por lo que el método hacia atrás en tiempo es incondicionalmente estable. La extensión de los resultados de la ecuación de convección a la ecuación de onda es inmediata.

El método Rayuela-E4

El enfoque utilizado por Gourlay[**38**] se utiliza en esta investigación para hacer que el método sea explícito y alcance la estabilidad. La técnica de Gourlay consiste en utilizar hacia adelante en tiempo para los puntos pares de la malla y hacia atrás en tiempo para los puntos impares.

Debido a lo similar del método Rayuela-Gourlay con el Rayuela-E4, no nos detendremos en este punto y continuaremos directamente con el método Rayuela-E4.

La primera parte del método requiere el cálculo de todos los puntos pares para el siguiente nivel de tiempo por medio del método de cuarto orden hacia adelante en tiempo. Para ello se obtienen las siguientes fórmulas de iteración para la ecuación de convección:

(4.4.15)

$$f(t+1,x_i) = f(t,x_i) - \frac{2a}{3}\left(g(t,x_{i+1}) - g(t,x_{i-1}) - \frac{1}{8}(g(t,x_{i+2}) - g(t,x_{i-2})) \right)$$

La fórmula de iteración para la segunda ecuación de convección es enteramente análoga.

(4.4.16)
$$g(t+1,x_i) = g(t,x_i) - \frac{2a}{3}\left(f(t,x_{i+1}) - f(t,x_{i-1}) - \frac{1}{8}(f(t,x_{i+2}) - f(t,x_{i-2}))\right)$$

Para implementar el método de cuarto orden hacia atrás en tiempo para los puntos nones, la ecuaciones anteriores no son suficientes ya que requieren de puntos nones así como de puntos pares. Para resolver este problema, se utiliza la fórmula de iteración compleja:

(4.4.17)
$$\begin{aligned} f(t+1,x_i) = f(t,x_i) - \frac{a}{4}\,(\, & g(t,x_{i+1}) - g(t,x_{i-1}) \\ & -\mathrm{i}(g(t,x_i+\Delta x\mathrm{i}) - g(t,x_i-\Delta x\mathrm{i}))) \end{aligned}$$

(4.4.18)
$$\begin{aligned} g(t+1,x_i) = g(t,x_i) - \frac{a}{4}\,(\, & f(t,x_{i+1}) - f(t,x_{i-1}) \\ & -\mathrm{i}(f(t,x_i+\Delta x\mathrm{i}) - f(t,x_i-\Delta x\mathrm{i}))) \end{aligned}$$

Las ecuaciones anteriores no involucran ningún punto non, así que son adecuadas para el método de la rayuela. La única dificultad es que los valores de las funciones a lo largo del eje imaginario también se requieren. En el nivel actual de tiempo no hay problema con esto, ya que pueden ser calculados de las expansiones de Taylor con las formulas imagplus e imagminus. Para los valores de la solución aproximada a lo largo del eje imaginario en el siguiente nivel de tiempo, el método Rayuela-E4 utiliza un enfoque de predicción/corrección, como aquellos de los métodos de MacCormack o de Lax-Wendroff de dos pasos. Con estos valores predichos, los valores corregidos para los puntos nones en el siguiente nivel de tiempo se obtienen con las ecuaciones anteriores para cerrar el ciclo iterativo.

El algoritmo del método se describe de la siguiente manera:

```
for each timelevel do
  for each EvenPoint do
    FTS4
  endfor
  for each Point do
    imagplus and imagminus
  endfor
  for each ImaginaryPoint do
    maccormack
  endfor
  for each OddPoint do
    BTS4
  endfor
endfor
```

Los resultados numéricos para el problema ejemplo de la ecuación de onda verifican que el método Rayuela-E4 constituye una mejora sobre otros métodos clásicos, tales como Rayuela-FTCS/BTCS o el de MacCormack.

Considere el problema:

$$P_{tt} = a^2 P_{xx},$$

donde P es la perturbación de presión acústica (Pa) y a es la velocidad del sonido. Esta ecuación se obtiene al combinar las siguientes dos ecuaciones:

$$\rho u_t + P_x = 0$$

$$P_t + \rho a^2 u_x = 0$$

Para ejemplificar la resolución de la ecuación de onda **[44]** se puede considerar el siguiente problema de Cauchy: un ducto infinito se llena con un fluido incompresible para el cual la densidad = 1.0 y la velocidad

de onda acústica $a = 1{,}0$ m/s. El fluido inicialmente está en reposo y tiene una distribución acústica de presión dada por:

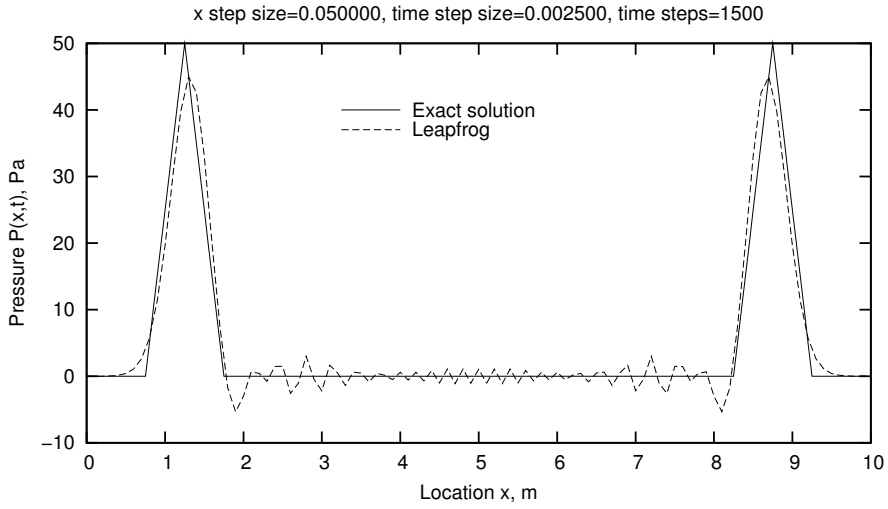
$$P(x,0) = 200(x-1), \qquad 1{,}0 \leq x \leq 1{,}5$$

$$P(x,0) = 200(2-x), \qquad 1{,}5 \leq x \leq 2{,}0$$

La solución exacta es una onda que se mueve en la dirección positiva y negativa.

Las figuras 4.4.10–4.4.12, muestran la solución aproximada junto a la solución exacta para tres métodos distintos despues de 150 iteraciones. De estas figuras se puede notar que la solución aproximada de todos los métodos a excepción del método Rayuela-E4 se van quedando atrás de la solución exacta. El método Rayuela-E4 predice la velocidad de onda con mayor certeza.

FIGURA 4.4.10. Nivel de tiempo 150 para el método Leapfrog



Para el caso de la tomografía de pozos aplicado a la prospección petrolera con la intención de redecir la configuración de yacimientos, el punto más importante a medir en la ecuación de onda es precisamente la alteración en la velocidad de la onda. Por ello es importante

FIGURA 4.4.11. Nivel de tiempo 150 para el método MacCormack

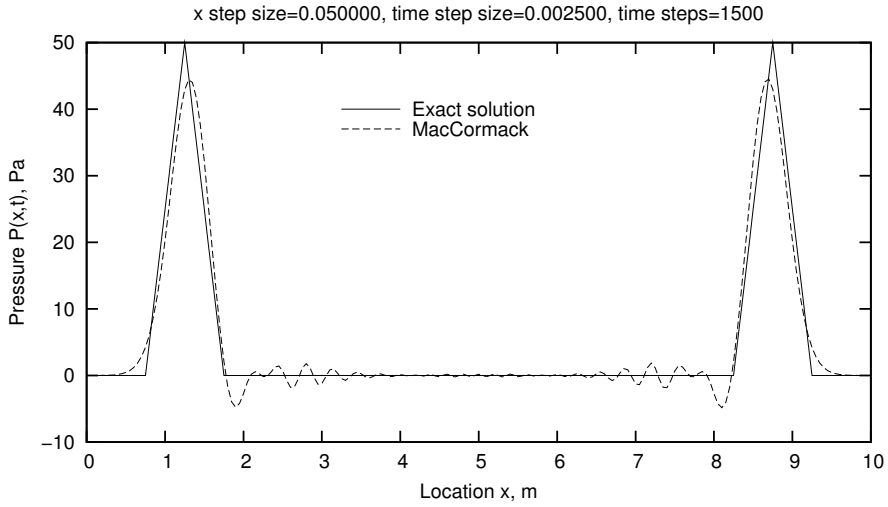
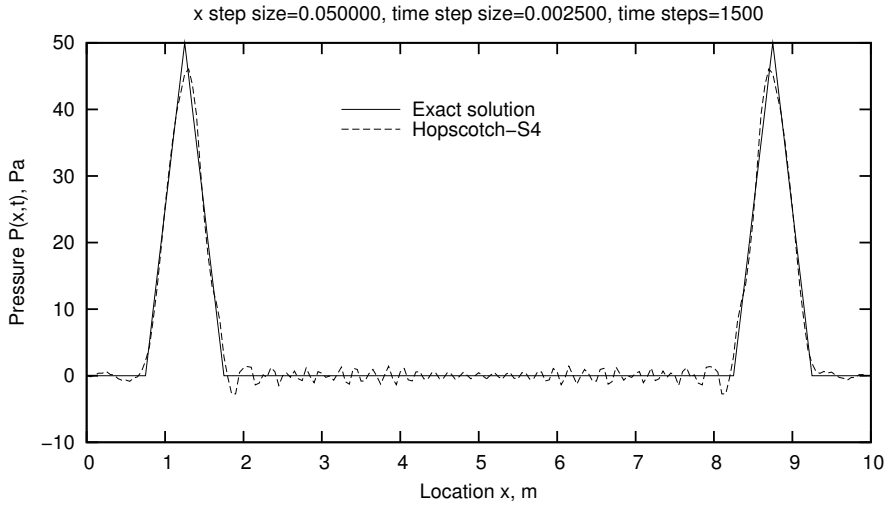


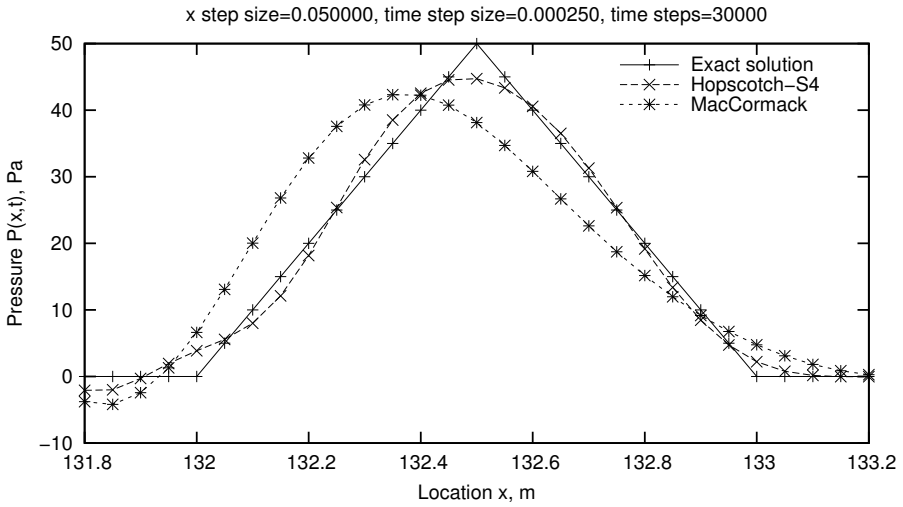
FIGURA 4.4.12. Nivel de tiempo 150 para el método Rayuela-E4



contar con un método numérico que sea más exacto en la predicción de la velocidad de onda. El método Rayuela-E4 desarrollado como parte del programa de estudio de posgrado para empleados del IMP, en colaboración con el CINVESTAV, es un método que se ajusta a esta

exigencias de la ingeniería. Esta mejora en precisión de predicción de la velocidad de onda puede apreciarse mejor en la figura 4.4.13 donde se compara el método Rayuela-E4 con el método de MacCormack con la mismas condiciones de iteración en la malla, después de 30000 iteraciones.

FIGURA 4.4.13. Contraste de los métodos Rayuela-E4 y MacCormack



Bibliografía

[1] M. Adler, J. W. Byers, and R. M. Karp. Scheduling parallel communication: The h-relation problem. Technical report, International Computer Science Institutue, Berkeley, 1995. Technical Report TR-95-032.

[2] A. Aggarwal, A. K. Chandra, and M. Snir. On communication latency in pram computations. In *Proceedings of SPAA 89*, pages 11–21, 1989.

[3] A. Aggarwal, A. K. Chandra, and M. Snir. Communication complexity of prams. *Theoretical Computer Science*, 71(1):3–28, 1990.

[4] R. M. Alford, K. R. Kelly, and D. M. Boore. Accuracy of finite-difference modeling of the acoustic wave equation. *Geophysics*, 6:834–842, 1974.

[5] S. Ali, J. K. Kim, Y. Yu, S. B. Gundala, S. Gertphol, H. J. Siegel, A. A. Maciejewski, and V. Prasanna. Greedy hueristics for resourse allocation in dynamic distributed realtime heterogeneous computing systems. In *The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2002)*, pages 519–530. CSREA Press, 2002.

[6] J. Arabe and C. Murta. Auto-balanceamento de carga em programas paralelos. In *Proceedings of the VIII Brazilian Symposium on Computer Architecture and High Performance Computing*, pages 161–171, 1996.

[7] R. M. Badia, J. Labarta, J. Gimenez, and F. Escalé. Dimemas: Predicting mpi applications behaviour in grid environments. In *Workshop on Grid Applications and Programming Tools*, 2003.

[8] A. Baumker, W. Dittrich, and F. M. Heide. Truly efficient parallel algorithms: 1-optimal multisearch for an extension of the bsp model. *Theoretical Computer Science*, (203), 1998.

[9] R. S. Bird. Lectures on constructive functional programming. *Constructive Methods in Computer Science*, NATO ASI Series F55:151–216, 1989.

[10] R. H. Bisseling. Basic techniques for numerical linear algebra on bulk synchronous parallel computers. In *Proceedings First Workshop on Numerical Analysis and Applications*, pages 46–57. Springer-Verlag, 1996.

[11] O. Bonorden, B. Juurlink, I. von Otte, and I. Rieping. The paderborn university bsp (pub) library- design, implementation and performance. Technical report, University of Paderborn, 1998. Technical report tr-rsfb-98-063.

[12] M. Bozyigit. History-driven dynamic load balancing for recurring applications on networks of workstations. *The Journal of Systems and Software*, 51:61–72, 2000.

[13] A. R. Butt, R. Zhang, and Y. C. Hu. A self-organizing flock of condors. In *Proceedings Super Computing 2003*, pages 15–21. ACM, 2003.

[14] R. Buyya and M. Murshed. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14:1175–1220, 2002.

[15] D. K. G. Campbell. *CLUMPS: a candidate model of efficient general purpose parallel computation.* PhD thesis, University of Exeter, 1994.

[16] H. Casanova. Simgrid: A toolkit for the simulation of application scheduling. In *Proceedings 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, page 430. IEEE Computer Society, 2001.

[17] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks. Technical report, Microsoft Research, 2002. Technical Report MSR.TR-2002-82.

[18] R. Cole and O. Zajicek. The apram: Incorporating asynchrony into the pram model. In *Proceedings of the 1989 ACM Symposium on Parallel Algorithms and Architectures*, 1989.

[19] D. Culler, R. Karp, D. Patterson, A. Sahay, J. Schauser, E. Santos, R. Subramonian, and T. von Eicken. Logp: Towards a realistic model of parallel computation. In *Fourth ACM SIGPLAN Symposium on Principles and PRactice of PArallel Programming*, pages 1–12, 1993.

[20] P. de la Torre and C. P. Kruskal. Towards a single model of efficient computation in real parallel machines. In *Parallel Architectures and Languages Europe, Lecture Notes in Computer Science.* Springer-Verlag, 1991.

[21] J. Dubinski. A parallel tree code. *New Astronomy*, (1):133–147, 1996. `http://www.cita.utoronto.ca/ dubinski/treecode/treecode.html`.

[22] C. Dumitrescu, I. Raicu, and I. Foster. Di-gruber: A distributed approach to grid resource brokering. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 38, Washington, DC, USA, 2005. IEEE Computer Society.

[23] C. L. Dumitrescu. Gangsim: A simulator for grid scheduling studies. In *ACM International Symposium on Cluster Computing and the Grid (CCGRID'05).* ACM, 2005.

[24] C. Fonlupt, P. Marquet, and J.-L. Dekeyser. Data-parallel load balancing strategies. *Parallel Computing*, 24(11):1665–1684, 1998. `http://citeseer.nj.nec.com/58809.html`.

[25] S. Fortune and J. Wyllie. Parallelism in random access machines. In *Proceedings of the 10th Annual Symposium on Theory of Computing*, pages 114–118. ACM, 1978.

[26] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputing Applications and High Performance Computing*, 2(11):115–128, 1997.

[27] I. Foster(Ed.) and C. K. (Ed.). *The GRID: Blueprint for a New Computing Infrastructure*. Morgan Kauffmann Publishers, 1999.

[28] M. Franklin and V. Govindan. A general matrix iterative model for dynamic load balancing. *Parallel Computing*, (22):969–989, 1996.

[29] N. Fujimoto, T. Baba, T. Hashimoto, and K. Hagihara. On message packaging in task scheduling for distributed memory parallel machines. *International Journal of Foundations of Computer Science*, 12(3):285–306, 2001.

[30] E. W. García and G. Morales-Luna. Design of the force field task assignment method and associated performance evaluation for desktop grids. In *Proceedings of the 2005 Fourth International Conference on Grid and Cooperative Computing*, pages 1009–1020. Springer-Verlag, 2005.

[31] E. W. Garcia and G. Morales-Luna. A solution to the information retrieval problem in desktop grid task assignment using the force field model. In *The 2006 International Conference on Grid Computing and Applications (GCA 06)*, pages 204–210. CSREA Press, 2006.

[32] E. W. Garcia and G. Morales-Luna. Simulation for bulk synchronous parallel superstep task assignment in desktop grids characterised by gaussian parameter distributions. *Multiagent and Grid Systems*, (Accepted for publication), scheduled for 2007.

[33] P. B. Gibbons. A more practical pram model. In *Proceedings of the 1989 ACM Symposium on Parallel Algorithms and Architectures*, pages 158–168. ACM, 1989.

[34] W. W. Gibbs. A split at the core. *Scientific American*, 291(9):74–77, 2004. `http://www.sciam.com`.

[35] S. K. Godunov. A difference scheme for the numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sbornik.*, 36:271–306, 1959.

[36] C. Gold and T. Schnekenburger. Using the ALDY load distribution system for PVM applications. In *PVM*, pages 278–287, 1996. `http://citeseer.nj.nec.com/gold96using.html`.

[37] M. T. Goodrich. Communication-efficient parallel sorting. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 247–256. ACM, 1996.

[38] A. Gourlay. Hopscotch: A fast second-order partial differential equation solver. *Journal Inst. Maths. Applics.*, 6:375–390, 1970.

[39] N. Guil and E. Zapata. Fast hough transform on multiprocessors: a branch and bound approach. *J. of Parallel and Distr. Comp.*, (45):82–89, 1997.

[40] J. L. Gustafson. Reevaluating amdahl's law. *Communications of the ACM archive*, 31(5):532–533, 1988.

[41] I. F. Haddad and E. Paquin. Mosix: A cluster load-balancing solution for linux. *Linux J.*, 2001(85es):6, 2001.

[42] T. Heywood and S. Ranka. A practical hierarchical model of paralel computation. *Journal of Parallel and Distributed Computing*, 16(3):212–232, 1992.

[43] J. M. D. Hill, S. R. Donaldson, and A. McEwan. Installation and user guide for the oxford bsp toolset implementation of bsplib. Technical report, Oxford University Computing Laboratory, 1998.

[44] J. D. Hoffmann. *Numerical Methods for Engineers and Scientists*. McGraw-Hill, Inc., 1993.

[45] C. C. Hui and S. T. Chanson. Hydrodynamic load balancing. *IEEE Trans. Parallel and Distributed Systems*, 10(9), 1999.

[46] K. Jansen, R. Solis.Oba, and M. Sviridenko. Makespan minimization in job shops: a linear time approximations chemes. 2003.

[47] M. Kaddoura and S.Ranka. Runtime support for parallelization orf data-parallel applications on adaptive and nonuniform computational environment. *Jorunal of parallel and Distributed Computing*, (43):163–168, 1997.

[48] M. Kechid and J. Myoupo. Towards a more realistic bsp cost model. In *Proceedings: Eighth International Conference on High-Performance Computing in the Asia-Pacific Region*, pages 3–12. IEEE, 2005.

[49] D. Kondo, M. Taufer, C. L. Brooks, H. Casanova, and A. A. Chien. Characterizing and evaluating desktop grids: An empirical study. Technical report, San Diego Supercomputer Center and University of California, San Diego, 2004. Work supported by the National Science Foundation under Grant ACI-0305390.

[50] B. Kruatrachue. *Static task scheduling and packing in parallel processing systems*. Department of Electrical and Computer Engineering, Oregon State University, Cornvallis, 1987.

[51] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms*. Benjamin/Cummings, 1994.

[52] P. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Comm. Pure and Appl. Math.*, 2:159–193, 1954.

[53] P. Lax and B. Wendroff. Systems of conservation laws. *Comm. Pure and Appl. Math.*, 13:217–237, 1960.

[54] A. R. Levander. Fourth-order finite-difference p-sv seismograms. *Geophysics*, 53:1425–1436, 1988.

[55] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *Proceedings 8th International Conference on Distributed Computing Systems (ICDCS 1988)*, pages 104–111, 1988.

[56] R. MacCormack. The effect of viscocity in hypervelocity impact cratering. *Institute of Aeronautics and Astronautics*, pages 69–354, 1969.

[57] K. Malik, O. Khan, T. Mobashir, and M. Sarwar. Migratable sockets in cluster computing. *J. Syst. Softw.*, 75(1-2):171–177, 2005.

[58] W. F. McColl. *General Purpose Parallel Computing*, pages 337–391. Cambridge University Press, 1993.

[59] W. F. McColl. *Scalable Computing*, pages 46–61. Springer-Verlag, 1995.

[60] W. L. M.J. Zaki and S. Parthasarathy. Customized dynamic load balancing for a network of workstations. *J. of Parallel and Distr. Comp.*, (43):156–162, 1997.

[61] J. M. Nash, P. M. Dew, and M. E. Dyer. *Scalable parallel application design*, pages 47–79. University of Westminster.

[62] R. Niewpoort, T. Kielmann, and H. Bal. Efficient load balancing for wide-area divide-and-conquer applications. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel PRogramming*, pages 34–43. ACM, 2001.

[63] C. Papadimitriou and M. Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comput.*, 19(2):322–328, 1990.

[64] D. Pountain. Parallel processing in bulk. Byte Core Technologies. `http://www.byte.com/art/9611/sec5/art5.htm`, 1996.

[65] R. R. Schaller. Moore's law: past, present and future. *Spectrum IEEE*, 34(6):52–59, 1997.

[66] T. Schnekenburger. Cooperating agents: Language support and load distribution. `http://citeseer.nj.nec.com/192702.html`.

[67] T. Schnekenburger. Using Agents for Parallel Programming in Workstation Networks. pages 331–334. http://citeseer.nj.nec.com/107994.html.

[68] T. Schnekenburger. The ALDY Load Distribution System. Technical report, 1995. http://citeseer.nj.nec.com/1714.html. TUM-19519 und SFB 342/11/95 A.

[69] H. Shi and J. Schaeffer. Parallel sorting by regular sampling. *Journal of Parallel and Distributed Computing*, 4(14):361–372, 1992.

[70] B. Shirazi, H. Chen, and J. Marquis. Comparative study of task duplication static scheduling versus clustering and non-clustering techniques. *Concurrency: Practice and Experience*, 7(5):371–390, 1997.

[71] D. B. Skillicorn. Architecture independent parallel computation. Technical report, Queen's University, Ontario, 1990. Technical Report ISSN-0836-0227-90-268.

[72] D. B. Skillicorn, W. F. McColl, and J. M. D. Hill. Questions and answers about bsp. Technical report, Oxford University Computing Laboratory, 1996. http://www.bsp-worldwide.org/. Tecnical report PRG-TR-15-96.

[73] H. Song, X. Liu, D. Jacobson, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. The microgrid: A scientific tool for modeling computational grids. *Scientific Programming*, 3(8):127–141, 2000.

[74] K. R. Sujithan. Towards a scalable parallel object database - the bulk synchronous parallel approach. Technical report, Wadham College Oxford, 1996. Technical Report PRG-TR-17-96.

[75] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency and Computation: Practice and Experience*, 17(2–4):323–356, 2005.

[76] A. N. Tijonov and A. A. Samarsky. *Ecuaciones de la física matemática.* Ed. Mir, 1983.

[77] A. Tiskin. A new way to divide and conquer. *Parallel Processing Letters*, 4, 2001.

[78] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[79] P. L. Vegas. In international conference on parallel and distributed processing techniques and applications. http://citeseer.nj.nec.com/522255.html.

[80] R. F. Warming and B. J. Hyett. The modified equation approach to the stability and accuracy analysis of finite-difference methods. *J. Comp. Phys.*, 14:159–179, 1974.

[81] J. Watts and S. Taylor. A practical approach to dynamic load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 9(3):235–248, 1998.

[82] M. Willebeek-LeMair and A. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on parallel and Distributed Systems*, (4):979–993, 1993.

[83] Z. Yao and G. F. Margrave. Fourth-order finite-difference scheme for p and sv waves propagating in 2d transversely isotropic media. *CREWES Research Report*, 11, 1999.

[84] A. Yu. The future of microprocessors. *IEEE Micro*, 16(6):46–53, 1996.

[85] H. Zhuge. The future interconnetion environment. *IEEE Computer*, 4(38):27–33, 2005.