INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECANICA Y ELECTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO
E INVESTIGACIÓN

IMPLEMENTACIÓN DEL MDFDT EN 3D MEDIANTE CÓMPUTO PARALELO, PARA EL ESTUDIO DE PROPAGACIÓN ELECTROMAGNÉTICA EN MEDIOS COMPLEJOS

T E S I S

QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES

PRESENTA:

ING. ABIMAEL RODRÍGUEZ SÁNCHEZ

DIRECTORES DE TESIS:

DR. MAURO ALBERTO ENCISO AGUILAR
M. EN. C. JESÚS ANTONIO ÁLVAREZ CEDILLO

México D.F. Diciembre 2011

INSTITUTO POLITÉCNICO NACIONAL

SIP-14

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de **México, D. F.** siendo las **18:00** horas del día **22** del mes de **NOVIEMBRE** del **2011** se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de E.S.I.M.E.-Zacatenco. para examinar la tesis titulada:

"IMPLEMENTACIÓN DEL MDFDT EN 3D MEDIANTE CÓMPUTO PARALELO, PARA EL ESTUDIO DE PROPAGACIÓN ELECTROMANÉTICA EN RECINTOS CERRADOS"

Presentada por el alumno:

RODRÍGUEZ	**SÁNCHEZ**	**ABIMAEL**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: | **B** | **0** | **9** | **1** | **8** | **6** | **8** |

aspirante de:

MAESTRO EN CIENCIAS EN INGENIERÍA DE TELECOMUNICACIONES

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Los Directores de tesis

DR. MAURO ALBERTO ENCISO AGUILAR

M. en C. JESÚS ANTONIO ÁLVAREZ CEDILLO

Presidente

Segundo Vocal

DR. JORGE ROBERTO SOSA PEDROZA

M. en C. JESÚS ANTONIO ÁLVAREZ CEDILLO

Tercer Vocal

Secretario

M. en C. MIGUEL SÁNCHEZ MERAZ

DR. LUIS MANUEL RODRÍGUEZ MÉNDEZ

EL PRESIDENTE DEL COLEGIO

DR. JAIME ROBLES GARCÍA

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

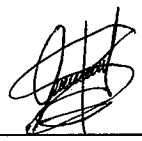
INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México D.F. el día 9 del mes de Diciembre del año 2011, el (la) que suscribe Abimael Rodríguez Sánchez alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Telecomunicaciones con número de registro B091868, adscrito a la Sección de Estudios de Posgrado e Investigación de la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Zacatenco, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección de Dr. Mauro Alberto Enciso Aguilar y el M. en C. Jesús Antonio Álvarez Cedillo y cede los derechos del trabajo intitulado Implementación del MDFDT en 3D mediante cómputo paralelo, para el estudio de propagación electromagnética en recintos cerrados, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección abima7@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Abimael Rodríguez Sánchez

Dedicatoria

Para mi familia, base del empeño y perseverancia en mis estudios y a la mujer más hermosa Beatriz Gayosso apoyo incondicional durante este proyecto.

Agradecimiento

Agradezco en primer lugar a Jesucristo por darme la oportunidad de poder terminar una etapa tan hermosa en mi vida y que sin él nada de esto sería posible.

Agradezco sinceramente el apoyo brindado por mis directores de Tesis, Mauro Alberto Enciso Aguilar y Jesús Antonio Álvarez Cedillo, de quienes de no haber tenido el impulso, ánimo y supervisión del desarrollo de mi proyecto de Tesis, no hubiera sido posible concretar esta etapa de mi vida. Muy en especial quiero reconocer el papel del Dr. Mauro, quien siempre estuvo al tanto de mis avances y de las posibilidades de proyección para mí, ya sea estudiando un Doctorado o comenzando mi labor profesional muchas gracias Dr. Mauro.

De la misma manera agradecer al M. en C. Jesús Antonio Álvarez Cedillo por su tiempo, conocimiento y apoyo incansable a lo largo de este trabajo quien sin conocerme me brindó su apoyo e incluso su amistad... gracias

Asimismo a al Dr. Luis Rodríguez Méndez, M. en C. Miguel Sánchez Meraz, Dra. Martha Galaz y el Dr. Jorge Sosa quienes me ayudaron con sus correcciones oportunas durante todo este trabajo pero muy en particular al Dr. Sosa por cambiar mi manera de ver el Electromagnetismo y por su paciencia hacia mi persona.

Agradecer el apoyo de mis padres y hermana, quienes fueron mi principal apoyo moral y fuente de alegrías durante mis estudios incluyendo a mi sobrino que es como un hijo Arielito.

Al amor de mi vida, Beatriz parte de mi vida así como de la Maestría, por su completo apoyo en las situaciones fáciles como en los problemas, así como su tiempo brindado hacia mi persona...Te amo

Quiero agradecer a los profesores de ESIME con quienes cursé las materias del posgrado, sus enseñanzas me han quedado bien grabadas y de la misma manera a todos los compañeros de la Maestría con quien compartí momentos inolvidables en esta fascinante etapa de nuestra formación académica.

Por último y no menos, un agradecimiento bastante especial al M. en C. Salvador Coss, M. en C. Manuel Benavides y al Doctor Jorge Sosa por el trabajo desarrollado en la tesis de Maestría [1], ya que fué mi punto de partida y quienes jugaron un papel importante asesorándome en todo tiempo durante mi trabajo de investigación.

Gracias a todos por confiar en mí y ayudarme a salir adelante. Que Dios los bendiga!!!

Índice general

Índice de figuras

Índice de tablas

Resumen

Este trabajo optimiza e implementa una herramienta para el desarrollo de algoritmos que demandan de una gran cantidad de recursos de cómputo como es el caso del Método de Diferencias Finitas en el Dominio del Tiempo (FDDT) tridimensional, el cual es utilizado para modelar la propagación electromagnética en recintos cerrados contemplando mesas, paredes, escritorios, computadoras, etc. El Método FDTD modela estructuras en el dominio del tiempo utilizando las ecuaciones diferenciales de Maxwell. Dentro de las ventajas de este método radica el hecho de no utilizar ningun tipo de aproximación empírica sobre otros métodos de simulación. Emplear el MDFDT permite analizar los efectos de reflexión, refracción y difracción de una onda electromagnética que se desplaza en el espacio libre y que incide sobre diferentes obstáculos. Desafortunadamente requiere grandes cantidades de memoria y tiempos de simulación bastantes largos.

Esta tesis aplica procesamiento paralelo para la solución del MDFDT tanto en dos como tres dimensiones. Para ello los programas de simulación requieren ser segmentados a cada procesador para reportar sus condiciones. Se describe el lenguaje de programación utilizado, las estrategias analizando el incremento de velocidad de procesamiento, la eficiencia computacional del algoritmo implementado así como la descripción de la implementación de la herramienta computacional. Se presentan los resultados de las simulaciones de propagación electromagnética, que se realizan en las oficinas para la verificación del correcto funcionamiento del algoritmo.

Abstract

This work develops a parallel Finite-Difference Time Domain (FDTF) technique which is used to simulate scattering objects like indoor electromagnetic propagation considering tables, walls,desks,computers, etc. In additon to this, it has been implemented a tool for development of algorithms that need large computational resources. The FDTD method simulates structures in the time-domain using a direct form of Maxwelś curl equations. This method has the advantage that any does not use empirical approximation. We use the MDFDT to analyze the effects of reflection, refraction and diffraction of a electromagnetic wave that propagates an indoor environment.

This requires large amounts of memory and long simulation times.

This thesis applies parallel processing to solve the FDTD method for two and three dimensions. To this end, the simulation needs to be segmented to each processor to report their conditions. We describe the programming language used, the strategies for parallelization, analyzing the processing increase in speed as well as the computational efficiency of the algorithm along the description of the implementation of the computational tool.

Finally we present results of simulations of electromagnetic propagation in order to verify the correct operation of the algorithm.

Justificación

En la Maestría en Ciencias en Ingeniería de telecomunicaciones de la SEPI ESIME IPN Unidad Zacatenco, se desarrollan diversos proyectos sobre electromagnetismo computacional, empleando diversas técnicas numéricas las cuales pueden ser muy demandantes en recursos de cómputo, por lo tanto, ha surgido la necesidad de explorar el desarrollo de nuevas aplicaciones. Dichos proyectos involucran el análisis de propagación electromagnética en medios no isotrópicos y no homogéneos así como dependencia en el dominio de la frecuencia. Los algoritmos presentan un principal problema ya que este tipo de aplicaciones implican el manejo de programas directos con datos de gran escala, realización de simulaciones, programas iterativos así como matrices de dimensiones grandes. Esto demanda grandes recursos de cómputo lo cual aún en estos tiempos es muy caro, por lo que es necesario explorar nuevas alternativas. Dentro de dichas alternativas se encuentran el caso de las técnicas de paralelización, las cuales permiten optimizar los algoritmos de proyectos de investigación basada en el modelado numérico, debido a que permiten mejorar principalmente en tiempo muchos de los algoritmos y por lo tanto obtener mejores resultados. La principal ventaja de este tipo de técnicas es la de poder trabajar con simulaciones en regiones con dimensiones eléctricas mucho más grandes y complejas. Dichas simulaciones son muy costosas en cuanto al tiempo de procesamiento, por lo que la nueva alternativa de técnicas de paralelización permitirá optimizar el algoritmo, resolver dichos problemas en un menor tiempo con mejores resultados y permitir un avance importante en el área de las comunicaciones. Se propone un cluster de computadoras como alternativa para cubrir la necesidad del cómputo de alto desempeño, de tal manera que se cubra la necesidad con el menor costo posible.

Glosario de térmimos

Absorción: es la transformación de parte de la energía electromagnética que se propaga en calor. Es la forma en la que la materia toma la energía de un fotón.

Alfanumérico: es el campo de datos o conjunto de caracteres que incluye letras,números y otros caracteres especiales como símbolos de puntuación, espacios, símbolos matemáticos. Información que integra datos alfabéticos y numéricos.

Algoritmo: es el conjunto de instrucciones que configuran el procedimiento paso a paso para resolver un determinado problema en una cantidad finita de tiempo. A partir de los algoritmos se forman los programas.

Algoritmo de Yee: Algoritmo que transforma las Ecuaciones de Maxwell en ecuaciones de diferencia, discretizándolas en tiempo y espacio.

Arquitectura de red: conjunto de reglas que gobiernan la interconexión y la interacción de los componentes de una red, incluye los formatos para los datos, los protocolos, las estructuras lógicas, etc.

Asíncrona: es la comunicación en la que emisor y receptor no se hallan sincronizados al principio de la transmisión de información. Los datos pueden llegarle al receptor en cualquier momento.

Atenuación: Disminución de la energía de una onda (y por lo tanto, también de su amplitud) durante la propagación por un medio.

Balanceado de cargas: es la distribución de procesos y actividades de comunicación, incluso a lo largo de una red de servidores, de modo que ningún dispositivo quede colapsado.

BIOS: es un programa grabado en la memoria ROM por el proveedor. Este sistema ad-

ministra las entradas y salidas de la computadora, y reconoce las características del equipo periférico.

Bit: unidad mínima de almacenamiento de la computadora.

Buffer (memoria intermedia): es la zona de la memoria que almacena temporalmente datos durante la transferencia de la información, se usa normalmente para equilibrar las diferentes velocidades operativas de los componentes de la unidad central de procesamiento.

Bus: canal a través del cual las señales pasan de unas partes de un circuito a otras.

Buscador: es la herramienta que permite ubicar contenidos en la Red, buscando en forma booleana a través de palabras clave.

Byte: es la unidad de información en memoria o almacenada en disco, que se utiliza normalmente para representar un carácter, la cual equivale a 8 bits.

Cabecera: porción frontal de un paquete que contiene información de protocolo, como direcciones, para dirigir el paquete a través de la red.

Cable: es el medio de transmisión de alambre o fibra óptica recubierto de una capa protectora.

Capa PML: Capa Perfectamente Acoplada (Perfectly Matched Layer) la cual crea una capa alrededor de la región de cálculo que absorbe la onda en un perfil de conductividad polinomial.

Caché: buffer de alta velocidad que agiliza el intercambio de información entre un dispositivo y el procesador.

CD-ROM: es un dispositivo de almacenamiento que sólo permite la lectura de los datos que contiene.

Centinelas: instrucciones para programación paralela basadas en lenguaje Fortran.

Circuito integrado (chip): circuito electrónico microscópico que agrupa varios millones de transistores y se fabrica con unas costosísimas técnicas.

Cliente: es el software que trabaja en una unidad central de procesamiento local para poder hacer uso de algún servicio de una unidad remota.

Cliente/servidor: sistema de organización de interconexión de computadoras según el cual funciona Internet, así como otros tantos sistemas de redes.

Cluster: es la unión de dos o más computadoras las cuales son unidas a través de una interconexión de red para ejecución en paralelo.

Cluster Beowulf: conjuntos de computadoras unidas por una red Ethernet las cuales tienen la características de el manejo de memoria Distribuida.

Cluster Mosix: es la unión de computadoras las cuales son interconectadas por medio de una red el cual crea múltiples procesos y en forma transparente pueden migrar estos procesos a los diferentes nodos conectados a la red.

Código fuente: es la forma en la que el programa de computación es escrito por el programador. El código fuente está escrito en lenguaje de programación, el cual es compilado en código objeto, en código máquina o ejecutable por un programa interpretador.

Código objeto: es el código de cómputo generado por un procesador de lenguaje de código fuente como un ensamblador o un compilador.

Coeficiente de Transmisión: factor que proporciona el porcentaje de energía de una señal incidente que se refracta y, por lo tanto, se transmite hacia el interior del segundo medio.

Comando: es la instrucción del sistema, concebida para una tarea específica.

Comando Externo: son los comandos utilizados con menos frecuencia que otros, por lo que no se cargan automáticamente en memoria, ahorrando espacio y almacenándose en el disco.

Compilador: herramienta software que transforma un programa fuente en código objeto, es decir, en un conjunto de instrucciones con sus datos en formato reconocible por la unidad central de procesamiento, aunque incompleto.

Computadora: dispositivo electrónico programable que es capaz de recibir datos, en un formato especificado, someterlos a un proceso y emitir, una información o señales de control, como resultado de ese proceso.

Conductividad: medida de la respuesta de las cargas libres de un medio en presencia de un campo eléctrico externo.

Conductores: materiales que presentan una conductividad superior a 10^5 S/m.

Conexiones: es la transferencia de información vía electrónica entre los sistemas de dos o más organizaciones.

Constante de propagación: factor de decaimiento de la intensidad del campo electromagnético. Número de onda complejo.

Constante dieléctrica: Para un mismo medio (misma mezcla de materiales), en unas mismas condiciones físicas es un valor constante característico.

Controlador: también llamado driver, es el programa que controla la forma en que se comunica la computadora con un determinado dispositivo hardware.

CUDA: son las siglas de Compute Unified Device Architecture (Arquitectura de Dispositivos de Cómputo Unificado) que hace referencia tanto a un compilador como a un conjunto de herramientas de desarrollo creadas por nVidia que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU de Nvidia.

Desempeño computacional: muestra mejoras tanto en mejoras en tiempo de ejecución así como el de la eficiencia computacional.

Difracción: efecto que se produce cuando la energía incide en un único elemento de tamaño relativamente grande de manera que las condiciones del medio varían bruscamente de un punto a otro.

Directivas: instrucciones propias de lenguaje C y C++ las cuales son usadas para la ejecución de tareas en paralelo.

Directorio: es el lugar del disco donde se almacenan los archivos de forma tal que el sistema operativo pueda encontrarlos cuando sea necesario.

Dispersión: distribución aleatoria de la energía cuando incide sobre objetos de dimensiones del orden de longitud de onda o inferior.

DNS, (servidor de nombres de dominios): sistema de computadoras que se encarga de convertir (resolver) las direcciones electrónicas de Internet en la dirección IP correspondiente y viceversa.

Dominio: es la identificación alfanumérica de un nodo o servidor en Internet. Es el último término de una dirección Internet.

Eficiencia computacional:se mide porcentaje y establece el tiempo de uso de los recursos computacionales conforme se incrementa el número de equipos.

Ethernet: estándar de configuración de redes locales a través de cable UTP alcanzabndo altas velocidades como Gigabit Ethernet.

Firewall: son los conjunto de programas de protección y dispositivos especiales que ponen barreras al acceso exterior a una determinada red privada.

Frecuencia: número de ciclos que por segundo efectúa una onda del espectro radioeléctrico.

FTP:es el protocolo para la transferencia de archivos. Norma específica que regula el intercambio de archivos entre diferentes máquinas y sistemas.

Ganglia: monitor de procesos en línea.

Gateway: es el dispositivo de comunicación entre dos o más redes locales (LANs) y remotas, usualmente capaz de convertir distintos protocolos, actuando de traductor para permitir la comunicación.

GPU: Unidad de Procesamiento Gráfico el cual es un procesador dedicado al procesamiento de gráficos.

GPUMAT: estandard en Matlab que permite correr códigos en un GPU

Hardware: es el conjunto de dispositivos físicos de los que se compone una unidad central de procesamiento.

Hilo:unidad de procesamiento mas pequeña que puede ser planificada por la computadora.

Homogeniedad: se da cuando las constantes del campo eléctrico, magnético y de conducción son constantes.

HTML: es el lenguaje que define textos, destinado a simplificar la escritura de documentos estándar. Es la base estructural en la que están diseñadas las páginas de la World Wide Web.

HTTP: es el protocolo que sirve de base para el intercambio de información en el sistema WWW.

Hub: es el dispositivo que concentra los cables de una red local.

Interfase: es el dispositivo hardware o protocolo de programación encargado de realizar la adaptación que haga posible la conexión entre dos sistemas o elementos de unidad central de procesamiento, entre unidades o con el usuario.

Internet: es la red mundial formada por la conexión de redes locales, regionales y nacionales en la que se intercambian datos y se distribuyen tareas de procesamiento.

Intranet: es la utilización de la tecnología de Internet dentro de la red local (LAN) y/o red de rea amplia (WAN) de una organización.

IP: protocolo de Internet el cual provee un método para fragmentar (deshacer en pequeños paquetes) y rutear (llevar desde el origen al destino) la información.

Isotropía: se da cuando sus propiedades físicas son idénticas en todas las direcciones.

Iteración: representa la repetición de las instrucciones del programa de forma autónoma por contraposición a la recursividad en un lenguaje de programación.

Lenguajes de programación: son el conjunto de instrucciones que permiten al programador resolver problemas a través de la creación de un programa ejecutable por la computa-

dora.

Linux: núcleo de sistema operativo libre tipo Unix.

Ley de Amhdal: determina la mejora máxima de un sistema cuando se desea paralelizar.

Memoria compartida: permite crear zonas de memoria ejecutada por varios procesos.

Memoria de las computadoras: es el término que se emplea para denotar las partes de la computadora donde se almacenan datos.

Memoria distribuida: se da cuando cada procesador tiene su propia memoria.

Microcomputadora: es el tipo de máquina más común, se suele encontrar en todo tipo de organización.

Microondas: son las ondas electromagnéticas de frecuencia elevada.

Microprocesador: circuito integrado que contiene los circuitos
necesarios para que una unidad central de procesamiento de pequeño tamaño lleve a cabo sus cálculos y comunicaciones con los otros componentes del sistema.

Monitor: dispositivo hardware que convierte en señal visible la información suministrada por la computadora, a través de una pantalla.

Monitoreo: es el seguimiento preciso a todas las fases del programa y a los aspectos críticos.

Monoprocesador: computadoras con un sólo procesador o un núcleo.

MPI: Interfaz de Paso de Mensajes bajo el esquema de memoria distribuida para aplicaciones en paralelo.

Multiacceso: es la forma de funcionamiento de una unidad central de procesamiento en el que un conjunto de usuarios puede interactuar, mediante terminales, con los programas que se estén ejecutando.

Multihilo: manejo de muchos hilos de programación.

Multiplataforma: es el programa o dispositivo que puede utilizarse sin inconvenientes en distintas plataformas de hardware y sistemas operativos.

Multiproceso: permite que un programa pueda tener varias partes que se ejecuten simultáneamente.

Multiprogramación: es el modo de operación de una unidad central de procesamiento en el que en un momento dado se pueden estar ejecutando varios programas, cada uno en una etapa diferente de su funcionamiento normal.

Multitarea: es la capacidad de algunas computadoras y sistemas operativos para ejecutar

varias aplicaciones al mismo tiempo, dividiendo las operaciones del procesador en tareas concurrentes.

Multiusuario: unidad central de procesamiento utilizable por varias personas que pueden estar usándola al mismo tiempo.

Nodo: es el punto de una red, en general. Conexión o punto de conmutación en una red.

OPENMP: es una interfaz de programación de aplicaciones (API) para la programación multiproceso de memoria compartida en múltiples plataformas.

Onda electromagnética: tipo de onda que posee dos campos: un campo eléctrico y otro magnético, los cuales oscilan entre si. Posee amplitud y dirección.

Página web: es la unidad que muestra información en la Web.

Paralelismo: es la capacidad que permite a varios programas ejecutarse a la vez sobre una misma máquina o, en el otro extremo que un mismo programa se ejecute de forma distribuida y simultáneamente en varios procesadores.

Parámetro: información añadida a la instrucción que inicia una determinada aplicación.

Parametros electromagnéticos: parámetros que definen el medio electromagnéticamente: conductividad, permitividad dieléctrica y permeabilidad magnética.

Periférico: es el dispositivo externo conectado a la unidad central de procesamiento, y utilizado generalmente para la entrada y salida de datos.

Permeabilidad magnética: capacidad de inmanación de un medio en presencia de un campo magnético externo. Constante de proporcionalidad entre la intensidad del campo magnético externo y la inducción magnética.

Permitividad dieléctrica: capacidad de polarización de un medio en presencia de un campo eléctrico externo. Constante de proporcionalidad entre el campo eléctrico externo aplicado y el desplazamiento eléctrico.

Ping: es la herramienta que permite averiguar si existe un camino (comunicación) de TCP/IP entre dos computadoras de cualquier parte de Internet.

Plataforma: es el fundamento tecnológico de un sistema de cómputo; en general, es una combinación específica de equipo de cómputo (hardware) y sistema operativo.

Procesamiento multinúcleo: son las computadoras que tienen más de una unidad de procesamiento o CPU, y cada CPU en esencia es una sola PC.

Procesamiento de datos: es la función que ofrece la tecnología de la información para al-

macenar, modificar, realizar cálculos, ordenar, transmitir, y presentar información con medios electrónicos.

Proceso paralelo: es el uso simultáneo de dos o más computadoras para solucionar un problema.

Programa: es el conjunto de instrucciones codificadas que ordenan a la computadora llevar a cabo determinada tarea; son instrucciones y datos escritos en un lenguaje de programación y almacenados en formato electrónico, que ordenan a la computadora la ejecución de cierta tarea.

Propagación multitrayectoria: cuando las ondas pueden alcanzar un destino por diversos caminos gracias a la reflexión, reflección o difracción.

Pruebas de desempeño: involucran una o más de las siguientes: pruebas de esfuerzo y sincronización, en donde se pone a prueba al máximo el sistema; pruebas de configuración, compatibilidad y recuperación, cuando se combina el sistema con otro de capacidad, memoria y velocidad menor; y pruebas de regresión.

Red de área local: es la red de computadoras interconectadas, distribuida en la superficie de una sola oficina o edificio. También llamadas redes privadas de datos.

Reflexión: cuando una onda incide sobre una superficie rebota hacia el mismo medio.

Refracción: cambio de dirección cuando la onda pasa de un medio a otro.

Sistema de nombre de dominio: es el método que convierte las direcciones Internet endirecciones IP. (Domain Name System)

Sistema operativo: es el programa que controla la forma en que la computadora utiliza sus recursos, entre ellos la memoria, el espacio de almacenamiento en disco, la interfase con los equipos periféricos y la interfase con el usuario. Esta constituido por programas (software) que llevan a cabo las tareas básicas para el funcionamiento de las computadoras.

Software: es el conjunto de instrucciones mediante las cuales una computadora puede realizar las tareas ordenadas por el usuario. Está integrado por los programas, sistemas operativos y utilidades.

Supercomputadora: es el tipo más grande, rápido y caro de computadora.

Superusuario: también llamado administrador del sistema, necesario cuando se desea realizar diversas tareas que requieren ciertos privilegios.

TCP: protocolo de control de transmisión; norma orientada a conexión, el cual es un con-

junto de protocolos de comunicación que se encargan de la seguridad y la integridad de los paquetes de datos que viajan por Internet.

TCP/IP: norma de comunicación en Internet, compuesta por dos partes: el TCP/IP. El IP desarma los envíos en paquetes y los rutea, mientras que el TCP se encarga de la seguridad de la conexión, comprueba que los datos lleguen todos, completos, y que compongan finalmente el envío original. (Transmission Control Protocol/Internet Protocol)

Telecomunicaciones: toda emisión, transmisión o recepción de signos, señales, escritos, imágenes, voz, sonidos o información de cualquier naturaleza que se efectúa a través de hilos, radioelectricidad, medios ópticos, físicos, u otros sistemas electromagnéticos.

Telnet: programa que permite el acceso remoto a un host. Utilizado para conectarse y controlar computadoras ubicadas en cualquier parte del planeta.

Tiempo compartido: técnica de proceso que consiste en la utilización de un mismo equipo por varios usuarios a la vez.

Tiempo de ejecución: tiempo que se encarga en procesar un programa la computadora.

Tiempo de sincronización: tiempo necesario para poder sincronizar los hilos para una aplicación en paralelo.

Unix: sistema operativo multitarea capaz de administrar varios puestos de trabajo desde un servidor central, capaz de aprovechar al máximo la capacidad de la unidad central.

Usuario: es todo aquel empleado que por las funciones que desempeña, usa o tiene acceso a bienes informáticos.

www: es un sistema global de información que se basa en el hipertexto, a través de un lenguaje llamado HTML, y es una manera de acceder a Internet y una forma de presentar la información con un interfaz gráfico que facilita la búsqueda de documentos.

Capítulo 1

Introducción

1.1. Introducción

El avance de la tecnología y las necesidades del hombre han hecho que la investigación científica, en cualquiera de sus ramas, tenga una gran importancia, exigiendo el desarrollo de nuevas alternativas que permitan reducir las fronteras del conocimiento humano. Principalmente el campo de acción de las ondas electromagnéticas, y su forma de estudio, se vuelve primordial en el estudio para el desarrollo de las comunicaciones.

En el año de 1873 fueron caracterizados los fenómenos electromagnéticos a través de un conjunto de ecuaciones por conducto de James Clerk Maxwell, en las que los campos eléctrico y magnético varían continuamente en espacio y tiempo, y su interación mutua produce una onda. Las ecuaciones de Maxwell, aplicadas a problemas reales pueden resolverse analíticamente o numéricamente. Sin embargo, por mucho tiempo este tipo de problemas reales se resolvieron analíticamente y tomaba bastante tiempo la resolución de estos mismos. Se pretendía dar solución mediante el uso de múltiples mediciones, lo cual llevó a cierto tipos de modelos, como son el caso específico del tratamiento de antenas. En los modelos particulares la gran cantidad de datos nos lleva a suponer la solución de una gran cantidad

de cálculos. Pero no fué hasta que se comenzó a hacer uso de las computadoras que se obtuvo la precisión, ya que es posible poder simular diferentes tipos de estructuras. Con el uso de las computadoras, se introduce una nueva alternativa que da lugar a explorar un área que no era tan conocida como es el caso del electromagnetismo computacional y con ello un gran avance al ya no tener que realizar innumerables cálculos o mediciones manualmente, que permitieran poder establecer algún modelo. El electromagnetismo computacional nos permite entre otras cosas, el poder simular y analizar los fenómenos electromagnéticos como reflexión, refracción y difracción a través de algunos métodos numéricos. Es por ello que la labor de la investigación sobre la propagación electromagnética ha venido a revolucionar por completo ya que se han tenido resultados en cantidades de tiempo menores y con mayor exactitud. Actualmente existen métodos computacionales que nos permiten simular estos comportamientos, entre ellos están: Método de diferencias finitas en el dominio del tiempo, Método de momentos, etc. Estos son de vital importancia ya que nos pueden describir el comportamiento electromagnético variando ciertos parámetros. Sin embargo aunque esta nueva técnica tiene grandes ventajas, está limitado por la capacidad que tiene el equipo de cómputo al procesar grandes volúmenes de procesos ya que estos cálculos son muy costosos en cuanto equipo de cómputo se refiere. El Método FDTD tiene 2 principales ventajas sobre análisis empíricos. Esto provee una solución directa a las ecuaciones de Maxwell sin mucha complejidad y toma en cuenta tanto el campo eléctrico y magnético en modelos 2D y 3D.

Este capítulo establece los objetivos de la investigación, y resalta la estructura en la cual esta organizado este trabajo. Asimismo da un panorama actual de los métodos de simulación.

1.2. Objetivos

- Desarrollar e implementar una herramienta para el desarrollo de algoritmos que demandan de una gran cantidad de recursos de cómputo como es el caso del algoritmo de Yee tridimensional para el modelado de propagación electromagnética en sistemas complejos a través de procesamiento multinúcleo paralelo.

- Desarrollar un cluster como herramienta de monitoreo y generación de códigos de ejecución en paralelo.

- Estudiar el comportamiento de la propagación electromagnética en recintos cerrados, considerando obstáculos presentes en ambientes típicos de oficinas.

1.3. Motivación: Panorama actual

Mientras el desempeño de sistemas computacionales sigue creciendo en magnitud y complejidad (cientos de millones de transistores en un chip, procesadores multinúcleo, redes en internet, etc.) la demanda de memoria, ciclos de cómputo y consecuentemente el tiempo requerido para procesar información, se ha convertido en el cuello de botella en virtud de la demanda por nuevos y mejores sistemas.

Sin embargo, la creciente cantidad de transistores integrables en un chip, ha dado lugar a la categoría de Circuitos ULSI y VLSI (Ultra / Very Large Scale Integration), en los cuales se añade la dificultad de diseñar arquitecturas correctas que a pesar de la progresiva densidad de transistores con la que están hechos (billones de ellos en un solo cm^2), sean completamente funcionales y se desempeñen acorde a lo planeado. En la fig 1.1 se muestra la ley de Moore

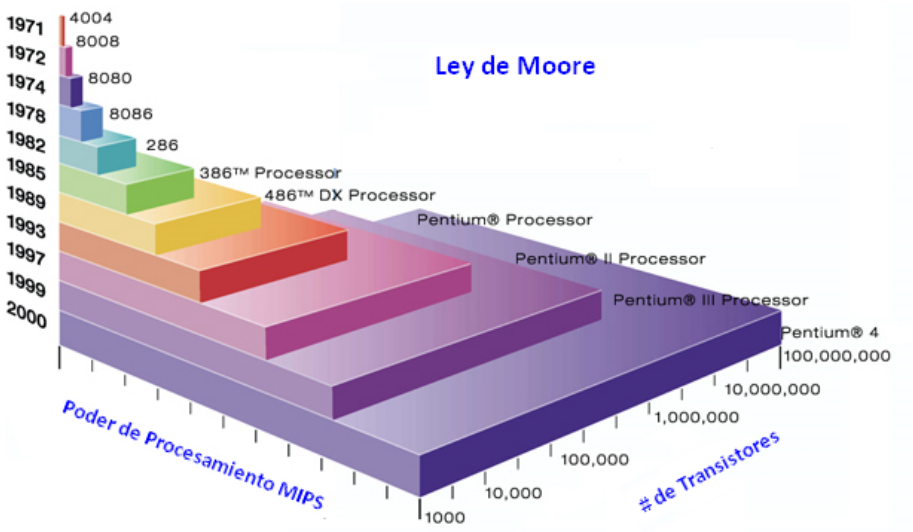


Figura 1.1: Ley de Moore

[2], apoyada por las nuevas tecnologías de diseño y los sintetizadores cada vez más confiables, son claros indicadores de que la integración con mayor densidad de transistores en una oblea de silicio seguirá en ascenso. Dichos indicadores muestran una proyección hacia arquitecturas

de computadoras que permitan poder dar solución cada vez más y mejor a diferentes métodos computacionales motivando a invertir tiempo y recursos en áreas que antes era imposible incursionar.

La finalidad de esta investigación al utilizar técnicas de programación paralela es aprovechar al máximo las plataformas de cómputo paralelo y distribuido (por ejemplo Clusters, Grids) con el fin de acelerar las simulaciones complejas, además de optimizar algoritmos que ya han sido analizados en diferentes estructuras y poder comprobar (validar) resultados. En este proyecto aprovecharemos las ventajas del diseño jerárquico de los lenguajes de programación paralelo para implementar el FDTD tanto en dos y tres dimensiones, en particular el lenguaje Fortran, pero de igual forma en lenguaje C y scripts en Matlab. En primera instancia, planeamos utilizar la granularidad por módulo en el contexto de Opemp, que es compatible con compiladores C, C++, Fortran; para generar la técnica de paralelización adecuada al problema presentado. Se implementa una herramienta para la ejecución de códigos en paralelo con su respectivo monitoreo y análisis para caracterizar el comportamiento de los algoritmos y así tener una métrica sobre del uso de los bloques de datos que componen al método.

Este trabajo esta organizado en 5 capítulos y 3 anexos.

En el capítulo 2 se describe el estado del arte del Método de Diferencias Finitas en el Dominio del Tiempo. Se discuten conceptos fundamentales del método FDTD como los distintos esquemas en diferencia finita, la obtención del algoritmo de Yee en dos y tres dimensiones, condiciones de frontera y tipos de fuente de onda utilizados.

En el capítulo 3 se describe el estado del arte de las Arquitecturas paralelas y Estrategias de paralelización del algoritmo de Yee. Asimismo se tratan sobre conceptos como son el cómputo paralelo, técnicas de paralelización bajo diferentes esquemas de memoria así como los diferentes cluster más utilizados hoy en día.

En el capítulo 4 se detalla la implementación del Algoritmo de Yee tanto modo serial como modo paralelo del MDFDT para dos y tres dimensiones.

En el capítulo 5 se presentan los resultados obtenidos durante este trabajo. Se presentan gráficas de desempeño de velocidad y eficiencia computacional. Asimismo se compara la

ejecución de dichos algoritmos con diferentes arquitecturas incluyendo el cluster Mosix implementado en este trabajo con la finalidad de validar los resultados obtenidos.

En el capítulo 6 se presentan las conclusiones obtenidas durante el desarrollo de este trabajo.

En el Anexo A se muestra a detalle la implementación de un Cluster tipo Mosix, desde la instalación del sistema operativo, configuración de red, instalación de kernel, instalación y configuración de mosix, instalación de compiladores de intel C, C++ y fortran así como la instalación y configuración de monitor del sistema Ganglia.

En el Anexo B se muestra las publicaciones generadas a partir de este trabajo.

En el Anexo C se muestra el código fuente paralelizado y optimizado empleado en la simulación del Método de Diferencias Finitas en el Dominio del Tiempo en tres dimensiones para la propagación electromagnética en interiores.

Al final de este trabajo el lector tendrá un panorama claro acerca de las técnicas de paralelización, implementación del MDFDT en dos y tres dimensiones, así como las herramientas para la construcción de un cluster Mosix.

1.4. Comparación entre Modelos de 2 y 3 Dimensiones

Modelos en tres dimensiones toman en cuenta cambios en la estructura física. En cambio modelos en dos dimensiones hacen una aproximación y no tienen variación en alguno de los tres ejes. Figura 1.2 muesta un ejemplo de un modelo 2d. En este caso una sección es tomada en el plano y-z y además no toma en cuenta ningun cambio en la dirección x. Por esta razón el modelado en 3D es normalmente utilizado para modelar medios físicamente o eléctricamente no uniformes. Desafortunadamente modelos en 3D consumen un gran tiempo de ejecución.

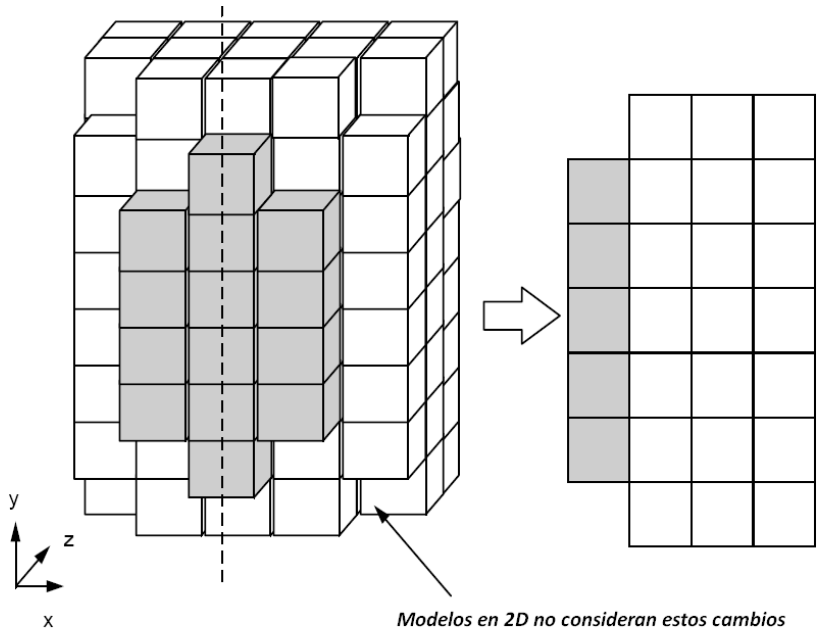


Figura 1.2: Modelos 3D y 2D

1.5. Métodos de Simulación

El modelado de propagación electromagnética involucra representar un sistema caracterizado por un modelo matemático. El tipo de modelo usado normalmente depende de sus parámetros como son el tiempo total de la simulación, el tamaño de la región a simular, el tipo de resultado que se requiere, la frecuencia de operación, etc. Esta sección discute los principales métodos de simulación electromagnética y han sido incluidos con la finalidad de entender las ventajas y desventajas de los métodos de simulación electromagnética utilizados durante este trabajo de tesis. Para diferentes áreas existen:

- Métodos de Volumen
- Métodos de Superficie
- Método de Rayos
- Métodos Híbridos

La figura 1.3 muestra una clasificación de los principales Métodos de modelados utilizados:

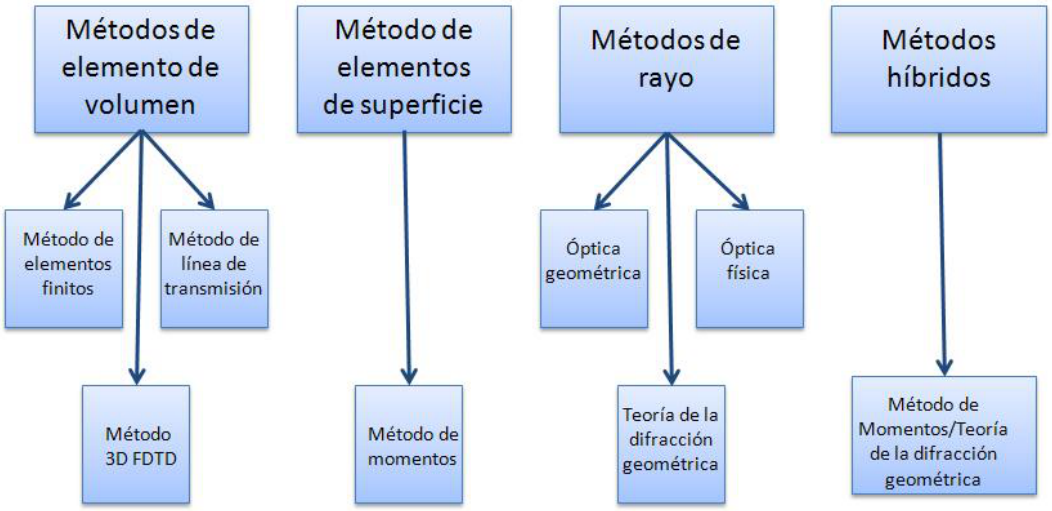


Figura 1.3: Métodos de modelado

1.5.1. Métodos de Volumen

El Método de elementos de volumen involucra hasta estructuras 3D, o elementos del sistema junto con la descripción de los materiales. Figura 1.4 muestra un ejemplo de una estructura convertida en un número de elementos. El modelado de cada elemento podría variar de elemento a elemento.

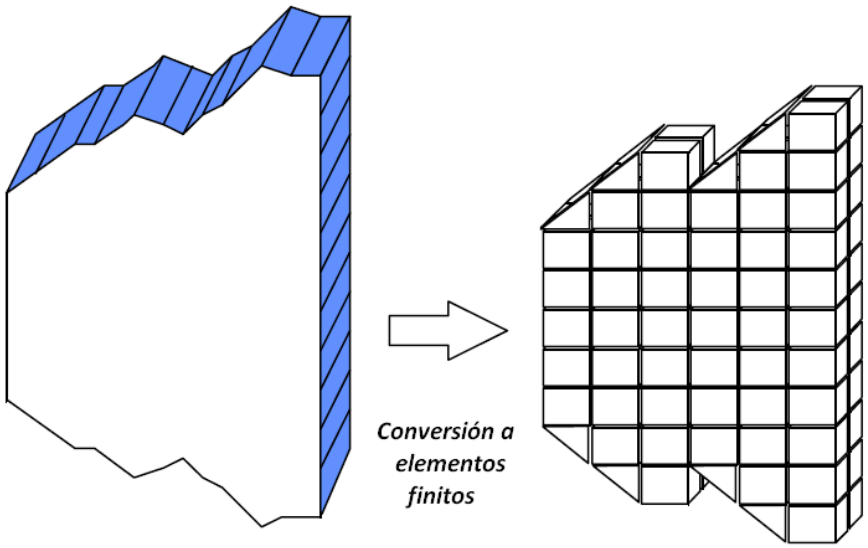


Figura 1.4: Conversión a elementos finitos

Método Finito

El método de elemento finitos divide la estructura física en pequeños elementos los cuales están compuestos de formas simples, como cubos(para 3D) y cuadrados o triángulos (para 2d). Esos elementos son modelados a través de la propagación electromagnética o en algunos casos, modelados por materiales eléctricos. El modelo completo es construido después de conectar las entradas y las salidas de los elementos con los vecinos.

Método de Línea de Transmisión (TLM)

El método TLM es un método en el dominio del tiempo en el que una onda electromagnética es propagada a través de una línea de transmisión[3]. El método TLM toma en cuenta las propiedades y condiciones de frontera al establecer las propiedades de la línea de transmisión. Sistemas con modo transversal eléctrico o magnético tienen dos lineas de transmisión equivalentes para cada modo[4]. Como referencia existe Christopoluos [5] y Hoefer [6] para más información sobre el método TLM

Método FDTD-3D

Algoritmo de Yee basado en las ecuaciones de Maxwell definen el método de Diferencias Finitas en el Dominio del Tiempo (FDTD). El método FDTD es una simulación en el dominio del tiempo que involucra el cálculo de cada uno de los campos eléctrico y magnético para dar solución a diferentes problemas relacionados con reflección, refracción o difracción electromagnética. La principal ventaja es que proporciona una solución directa a las ecuaciones diferenciales de Maxwell sin muchas complejidades en comparación con métodos empíricos[7].

El capítulo 2 discute el método FDTD en tres dimensiones con más detalle.

1.5.2. Métodos de elementos de superficie

En los métodos de elementos de superficie, el campo eléctrico y magnético no penetra dentro de los elementos [8]. Métodos de superficie generalmente requiere menos elementos

que los métodos de volumen, pero las propiedades del material son más complicadas de definir.

Método de Momentos

El método de de momentos (MoM) es una método de elementos de superficie y es uno de los más usados en el electromagnetismo computacional. Cuando dielétricos son usados con el MoM, cambia el problema en un elemento de volumen. Desafortunadamente, estas técnicas incrementa el tiempo de simulación.

1.5.3. Métodos de Rayo

El método de rayo envuelve el trazado de la ruta de los rayos cuando son reflejados o difractados de un objeto. La cantidad y dirección de la difracción y reflección depende de el tipo de superficie (tanto geométricamente como el material). Existen soluciones para un amplio margen de materiales conductores, pero para problemas con dieléctricos las soluciones son limitadas. El trazado del rayo domina el tiempo de simulación y seguido es complicado estimarlo. Afortunadamente almacenamiento de información no es un problema. Los principales métodos utilizados son:

- Teoría de difracción geométrica(GTD)
- Óptica física(PO)
- Óptica geométrica(GO)

Esos métodos son generales y sólo aplicados en electromagnetismo para la simulación de antenas reflectivas. El método de óptica geométrica es útil para diámetros de apertura los cuales son largos en términos de longitud de onda. Como la apertura del reflector disminuye, el patrón de radiación llega a incrementar dominando la difracción

1.5.4. Métodos híbridos

Los métodos híbridos involucra una combinación de dos o más métodos de volumen, superficie o trazado de rayos.

1.6. Conclusiones

Modelado de propagación electromagnética es generalmente útil cuando se requiere analizar estructuras complejas o simulaciones. Los Métodos de elemento Finito, MDFDT, and TLM analizan las estructuras en grupos de elementos interconectados. El Método de Elemento Finito basado en frecuencia modela cada uno de los elementos con su equivalente caracterstica de frecuencia. El método FDTD y el método TLM son métodos basados en tiempo e involucran modelos iterativos discretos en intervalos de tiempo. Ellos difieren en la técnica en que ellos modelan cada uno de los elementos. El método TLM modela los usos equivalentes en la línea de transmisin para cada elemento, a diferencia del Método FDTD que modela la propagación a través de elementos usando una forma discreta de las ecuaciones de Maxwell.

Se ha mostrado, que el MDFDT en 3D proveé una directa solución a las ecuaciones diferenciales de Maxwell tomando en cuenta el campo eléctrico como el magnético en un modelo tridimensional. TLM a diferencia de FDTD utiliza aproximaciones empíricas. Simulaciones en el Dominio del Tiempo tienen ventajas sobre soluciones en el dominio de la frecuencia ya que normalmente requieren relativamente mucho tiempo en completar las simulaciones y las estructuras generalmente no son facilmente de simular. Computadoras modernas involucran mejoras en equipo computacional. En general, soluciones en el dominio del tiempo tiene ventajas sobre soluciones en el dominio de la frecuencia ya que ellos pueden ser utilizados en el procesamiento paralelo reduciendo el tiempo de ejecución de cada una de las simulaciones (serán discutidos en capítulo 4 y 5). Por esas razones el MDFDT en 3D ha sido escogido como estudio principal en esta investigación.

Capítulo 2

Método de Diferencias Finitas en el Dominio del Tiempo

2.1. Introducción

Este capítulo discute la teoría de el Método FDTD y su aplicación para propagación electromágnetica. Capítulo IV aplican el Método FDTF aplicado a simulación de propagación electromagética en recintos cerrados.

2.2. Método de Diferencias Finitas en el Dominio del Tiempo

El Método de Diferencias Finitas en el Dominio del Tiempo (FDTD) utiliza las ecuaciones de Maxwell las cuáles definen la propagación electromagnética de una onda y la relación entre el campo eléctrico y magnético, las cuales quedan expresadas de la siguiente manera:

$$\frac{\partial \overrightarrow{B}}{\partial t} = -\nabla \times \overrightarrow{E} - \overrightarrow{M} \tag{2.1}$$

$$\frac{\partial \overrightarrow{D}}{\partial t} = \nabla \times \overrightarrow{H} - \overrightarrow{J} \tag{2.2}$$

$$\nabla \cdot \overrightarrow{D} = \rho_v \tag{2.3}$$

$$\nabla \cdot \overrightarrow{B} = 0 \tag{2.4}$$

Donde:

$\overrightarrow{E}$ es la Intensidad de campo eléctrico $\left(\frac{volts}{metro}\right)\frac{V}{m}$

$\overrightarrow{D}$ es la Densidad de flujo eléctrico $\left(\frac{coulomb}{metro^2}\right)\frac{C}{m2}$

$\overrightarrow{H}$ es la Intensidad de campo magnético $\left(\frac{ampere}{metro}\right)\frac{A}{m}$

$\overrightarrow{B}$ es la Densidad de flujo magnético $\left(\frac{weber}{metro^2}\right)\frac{Wb}{m2}$

$\overrightarrow{J}$ es la Densidad de corriente eléctrica $\left(\frac{ampere}{metro^2}\right)\frac{A}{m2}$

$\overrightarrow{M}$ es la Densidad de corriente magnética $\left(\frac{volt}{metro^2}\right)\frac{V}{m2}$

ε es la Permitividad eléctrica $\left(\frac{farad}{metro}\right)\frac{F}{m}$

ε_r es la Permitividad relativa $\left(Escalar\right)$

ε_0 es la Permitividad del vacio $= 8.854 \times 10^{-12}\frac{F}{m}$

μ es la Permeabilidad magnética $\left(\frac{Henry}{metro}\right)\frac{H}{m}$

μ_r es la Permeabilidad relativa $\left(Escalar\right)$

μ_0 es la Permeabilidad del vacio $= 4\pi \times 10^{-7}\frac{H}{m}$

La relación entre la densidad de flujo eléctrico y el campo eléctrico queda expresada para materiales lineales, isotrópicos y homogéneos de la siguiente manera:

$$\overrightarrow{D} = \varepsilon\overrightarrow{E} = \varepsilon_r\varepsilon_0\overrightarrow{E} \tag{2.5}$$

La relación entre la inducción magnética y el campo magnético esta dado por:

$$\overrightarrow{B} = \mu\overrightarrow{H} = \mu_r\mu_0\overrightarrow{H} \tag{2.6}$$

Considerando un medio sin pérdidas eléctricas ni magnéticas,queda expresada como $\overrightarrow{J}$ y $\overrightarrow{M}$ pueden actuar como fuentes independientes de energía eléctrica y magnética

$\overrightarrow{J}_{fuente\ y}\ \overrightarrow{M}_{fuente}$. Un medio con pérdidas eléctricas y magnéticas no dispersivas,queda expresada como:

$$\overrightarrow{J} = \overrightarrow{J}_{fuente} + \sigma\overrightarrow{E} \tag{2.7}$$

$$\overrightarrow{M} = \overrightarrow{M}_{fuente} + \sigma^*\overrightarrow{H} \tag{2.8}$$

Donde:

σ: Conductividad Elétrica$\left(\frac{Siemens}{metro}\right)\frac{S}{m}$

σ^*: Conductividad Magnética$\left(\frac{Siemens}{metro}\right)\frac{S}{m}$

La solución de todo el esquema de acuaciones enfocadas a la aplicación de interés involucra el uso del del método de diferencias finitas.

2.3. Aproximación con Diferencias Finitas

El método basado en diferencias finitas es una aproximación para encontrar la solución numérica de las ecuaciones que gobiernan el modelo matemático de un sistema continuo. Básicamente, en una solución por diferencias finitas, las derivadas son reemplazadas por aproximaciones en diferencias finitas, convirtiendo entonces un problema de ecuaciones diferenciales en un problema algebraico fácilmente de resolver. Para ello se hace uso de

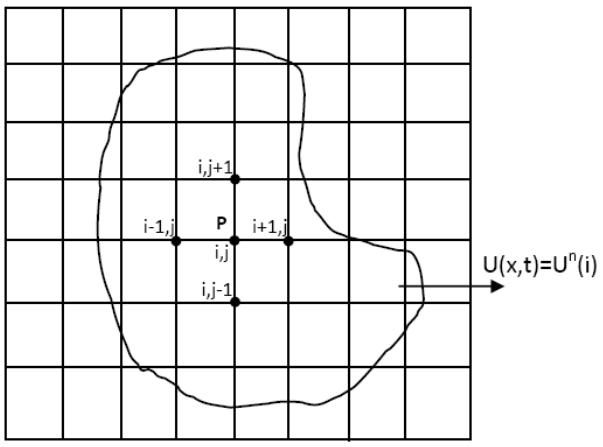


Figura 2.1: Discretización de la región espacio-temporal mediante una malla

un conjunto de puntos finitos los cuales son identificados con los nodos de una malla, en las

que está dividido la región de cálculo asignándole un índice de acuerdo a un punto ubicado en el tamaño de la malla de tres dimensiones como se muestra en la figura 2.1 y la cual queda expresado de la siguiente forma:

$$(x, y, z, t) = (i\triangle x, j\triangle y, k\triangle z, t\triangle n = (i, j, k, n)$$

$$f(x, y) = f(i\triangle x, j\triangle y) = f(i, j)$$

Observámos que los índices i, j, k son enteros y $\triangle x, \triangle y, \triangle z$ son las dimensiones de la celda de espacial y $\triangle t$ es la unidad de paso temporal. La solución del método FDTD esta basada en la serie de potencias de Taylor, cuyo desarrollo para una función finita y continua f(x), queda expresada como:

$$f(x_0 + \triangle x) = f(x_0) + \triangle x f'(x_0) + \frac{1}{2!}(\triangle x)^2 f''(x_0) + \frac{1}{3!}(\triangle x)^3 f'''(x_0) + ... \tag{2.9}$$

$$f(x_0 - \triangle x) = f(x_0) - \triangle x f'(x_0) + \frac{1}{2!}(\triangle x)^2 f''(x_0) - \frac{1}{3!}(\triangle x)^3 f'''(x_0) + ... \tag{2.10}$$

Desarrollando la serie, se obtienen expresiones en diferencias finitas que permiten calcular el valor de una función o sus derivadas en un punto. Consideremos la figura 2.2, la derivada de la función en el punto P del arco PB se estima mediante diferencias finitas como:

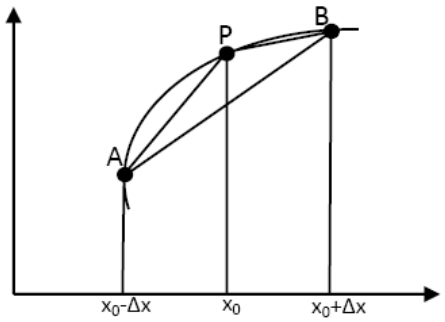


Figura 2.2: Estimación de derivada f(x) en el punto P usando diferencias hacia adelante y hacia atrás

$$f'(x_0 + \triangle x) = \frac{(f(x_0) + \triangle x) - (f(x_0))}{\triangle x} + \frac{1}{2!}(\triangle x)^2 f''(x_0) + \frac{1}{3!}(\triangle x)^3 f'''(x_0) + ... \tag{2.11}$$

Agrupando términos

$$f'(x_0) = \frac{f(x_0 + \triangle x)}{\triangle x} + 0(\triangle x) \tag{2.12}$$

Si no truncamos la serie infinita de Taylor se obtiene una solución exacta del problema. Sin embargo, esto no es realizable, por lo que es necesario truncar la serie a partir del segundo

término, considerando que $\triangle x$ es muy pequeña, introduciendo un error $O(\triangle x)$ en todas las soluciones de diferencia finita. Si no se considera el error, la derivada hacia adelante o por la derecha es:

$$f'(x_0) \cong \frac{f(x_0 + \triangle x) - f(x_0)}{\triangle x} \tag{2.13}$$

Siguiendo un procedimiento similar, la derivada mediante diferencias finitas hacia atrás, en el arco AP es:

$$f'(x_0 - \triangle x) = \frac{f(x_0 - f(x_0 - \triangle x))}{\triangle x} + \frac{1}{2!}(\triangle x)^2 f''(x_0) + ... \tag{2.14}$$

Agrupando términos

$$f'(x_0) = \frac{f(x_0 - f(x_0 - \triangle x))}{\triangle x} + 0(\triangle x) \tag{2.15}$$

Donde $O(\triangle x)$ denota el error introducido por truncar la serie de potencias. Si suponemos que $\triangle x$ es muy pequeño se puede despreciar el término $O(\triangle x)$ de modo que la expresión de diferencia hacia atrás o por la izquierda es:

$$f'(x_0) \cong \frac{f(x_0) - f(x_0 + \triangle x)}{\triangle x} \tag{2.16}$$

Sumando 15 y 16 se puede obtener la derivada centrada del arco AB usando diferencias finitas:

$$f'(x_0) \cong \frac{f(x_0 + \triangle x) - f(x_0 - \triangle x)}{2\triangle x} \tag{2.17}$$

La ecuación (2.17), representa la derivada aproximada del arco AB de la figura 2.2. Las derivadas de una función f(x, y) en un espacio de 2 dimensiones, como el mostrado en la figura (2.1), pueden representarse por medio de diferencias finitas si la región del problema se divide en una malla con celdas de dimensiones $\triangle x$ y $\triangle y$. Las aproximaciones en diferencia central de las derivadas de f en un punto P o nodo (i,j) son:

$$\frac{\partial}{\partial x} f(i,j) \cong \frac{f(i+1,j) - f(i-1,j)}{2\triangle x} \tag{2.18}$$

$$\frac{\partial}{\partial y} f(i,j) \cong \frac{f(i,j+1) - f(i,j-1)}{2\triangle y} \tag{2.19}$$

2.4. Discretización de las leyes de Maxwell

Las ecuaciones rotacionales de Maxwell están expresadas por:

$$\frac{\partial \overrightarrow{E}}{\partial t} = -\frac{1}{\varepsilon}\nabla \times \overrightarrow{H} - \frac{1}{\varepsilon}(\overrightarrow{J}_{fuente}) + \sigma \overrightarrow{H} \tag{2.20}$$

$$\frac{\partial \overrightarrow{H}}{\partial t} = -\frac{1}{\mu}\nabla \times \overrightarrow{E} - \frac{1}{\mu}(\overrightarrow{M}_{fuente}) + \sigma^* \overrightarrow{H} \tag{2.21}$$

Resolviendo el rotacional obtenemos un sistema de 6 ecuaciones en derivadas parciales las cuales se muestran a continuación:

$$\frac{\partial \overrightarrow{H_x}}{\partial t} = -\frac{1}{\mu}\left[\frac{\partial \overrightarrow{E_y}}{\partial z} - \frac{\partial \overrightarrow{E_z}}{\partial y} - (\overrightarrow{M}_{fuentex}) + \sigma^* \overrightarrow{H_x}\right] \tag{2.22}$$

$$\frac{\partial \overrightarrow{H_y}}{\partial t} = -\frac{1}{\mu}\left[\frac{\partial \overrightarrow{E_z}}{\partial x} - \frac{\partial \overrightarrow{E_x}}{\partial z} - (\overrightarrow{M}_{fuentey}) + \sigma^* \overrightarrow{H_y}\right] \tag{2.23}$$

$$\frac{\partial \overrightarrow{H_z}}{\partial t} = -\frac{1}{\mu}\left[\frac{\partial \overrightarrow{E_x}}{\partial y} - \frac{\partial \overrightarrow{E_y}}{\partial x} - (\overrightarrow{M}_{fuentez}) + \sigma^* \overrightarrow{H_z}\right] \tag{2.24}$$

$$\frac{\partial \overrightarrow{E_x}}{\partial t} = -\frac{1}{\varepsilon}\left[\frac{\partial \overrightarrow{H_z}}{\partial y} - \frac{\partial \overrightarrow{H_y}}{\partial z} - (\overrightarrow{J}_{fuentex}) + \sigma \overrightarrow{E_x}\right] \tag{2.25}$$

$$\frac{\partial \overrightarrow{E_y}}{\partial t} = -\frac{1}{\varepsilon}\left[\frac{\partial \overrightarrow{H_x}}{\partial z} - \frac{\partial \overrightarrow{H_z}}{\partial x} - (\overrightarrow{J}_{fuentey}) + \sigma \overrightarrow{E_x}\right] \tag{2.26}$$

$$\frac{\partial \overrightarrow{E_z}}{\partial t} = -\frac{1}{\varepsilon}\left[\frac{\partial \overrightarrow{H_y}}{\partial x} - \frac{\partial \overrightarrow{H_x}}{\partial y} - (\overrightarrow{J}_{fuentez}) + \sigma \overrightarrow{E_x}\right] \tag{2.27}$$

Este es el sistema de ecuaciones que forma la base del algoritmo numérico de diferencias finitas en el dominio del tiempo en tres dimensiones[9].

2.5. Algoritmo de Yee

Cada una de las componentes de campo electromagnético se ubica espacialmente como se muestra en la figura 2.3 de acuerdo al esquema de Yee, de donde se deducen los vectores de posición de cada una de las componentes [10].

$$H_x = (i, j+\frac{1}{2}, k+\frac{1}{2}) \qquad H_y = (i-\frac{1}{2}, j+1, k+\frac{1}{2}) \qquad H_z = (i-\frac{1}{2}, j+\frac{1}{2}, k+1)$$

$$E_x = (i-\frac{1}{2}, j+1, k+1) \qquad E_y = (i, j+\frac{1}{2}, k+1) \qquad E_z = (i, j+1, k+\frac{1}{2})$$

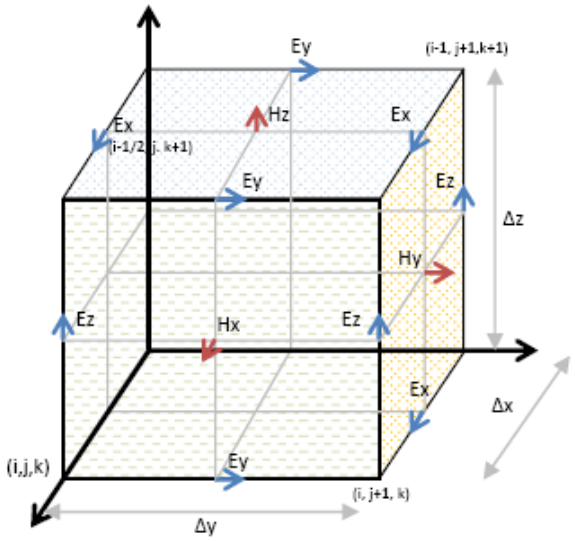


Figura 2.3: Célula unitaria de Yee [11]

Generalidades del algoritmo de Yee

La célula de Yee centra los componentes de $\overrightarrow{E}$ y $\overrightarrow{H}$ en un espacio tridimensional, de tal manera que cada componente $\overrightarrow{H}$ está rodeada de cuatro componentes de $\overrightarrow{E}$ alrededor y viceversa. Además Proporciona una visión tridimensional de un espacio cubierto por contornos entrelazados entre las leyes de Farady y Ampere. Es posible identificar las componentes de $\overrightarrow{H}$ en la célula de Yee asociadas con el flujo magnético que se entrelaza con los lazos de $\overrightarrow{E}$ así como los componentes $\overrightarrow{E}$ asociados con el flujo de desplazamiento de corriente que se entrelazan con los lazos de $\overrightarrow{H}$

El esquema temporal de Yee centra los componentes de $\overrightarrow{E}$ y $\overrightarrow{H}$ en el tiempo, esquema que se conoce como salto de rana. Todos los valores de $\overrightarrow{H}$ en la región son calculados y almacenados en memoria usando los valores de $\overrightarrow{E}$ calculados previamente. Todos los valores de $\overrightarrow{E}$ en la región son calculados y almacenados en memoria usando los valores de $\overrightarrow{H}$ calculados previamente. Este ciclo es recursivo hasta que el escalonado temporal termina[10].

2.6. Ecuaciones de Maxwell expresadas en Diferencias Finitas

El sistema de ecuaciones (2.22-2.27) proporcionan una aproximación numérica de las ecuaciones rotacionales de Maxwell en tres dimensiones. Considérese inicialmente la componente de campo H_x.

$$\frac{\partial \overrightarrow{H_x}}{\partial t} = m\frac{1}{\mu}\left[\frac{\partial \overrightarrow{E_y}}{\partial z} - \frac{\partial \overrightarrow{E_z}}{\partial y} - (\overrightarrow{M}_{fuentex}) + \sigma^* \overrightarrow{H_x}\right] \tag{2.28}$$

Para definir las coordenadas espaciales (i,j,k) de cada componente de campo, se utiliza la célula básica de Yee, figura 2.3. Para la ecuación (2.28) ver la figura 2.4.

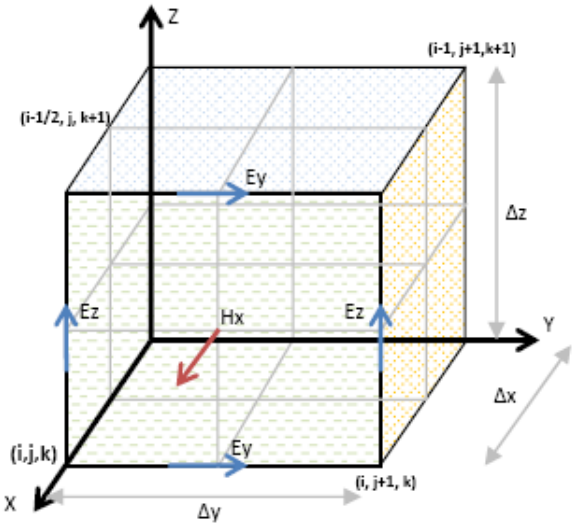


Figura 2.4: La componente H_x que depende de los campos E_y y E_z [11].

Cada derivada parcial se transforma en una ecuación de diferencia central. La transformación de la componente temporal es:

$$\frac{\partial \overrightarrow{H_x}}{\partial t} = \frac{H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle t} \tag{2.29}$$

La transformación de las componentes de campo espaciales es:

$$\frac{\partial \overrightarrow{E_y}}{\partial t} = \frac{E^n_{y(i,j+\frac{1}{2},k+1)} - E^n_{y(i,j+\frac{1}{2},k)}}{\triangle Z} \tag{2.30}$$

$$\frac{\partial \overrightarrow{E_z}}{\partial t} = \frac{E^n_{z(i,j+1,k+\frac{1}{2})} - E^n_{z(i,j,K+\frac{1}{2})}}{\triangle y} \tag{2.31}$$

Sustituyendo (2.29-2.31) en (2.28 se obtiene la transformación de H_x como:

$$\frac{H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle t} = \frac{1}{(\mu_{i,j+\frac{1}{2},k+\frac{1}{2}})}\left[\frac{E^n_{y(i,j+\frac{1}{2},k+1)} - E^n_{y(i,j+\frac{1}{2},k)}}{\triangle Z} - \frac{E^n_{z(i,j+1,k+\frac{1}{2})} - E^n_{z(i,j,K+\frac{1}{2})}}{\triangle y}\right] \tag{2.32}$$

Todos los valores de campo del lado derecho de la ecuación están evaluados en el escalonado temporal n. Se observa que el valor de H_x en $t = n$ no está almacenado en la memoria de la computadora (únicamente los valores de tiempo en el instante $t = n - \frac{1}{2}$, por lo que se necesita estimar dicho término de alguna manera, para ello se establece que los valores de H_x en el intervalo de tiempo n son simplemente el promedio aritmético del valor almacenado de H_x en el intervalo de tiempo $n + \frac{1}{2}$ y del valor aún no calculado de H_x en el intervalo de tiempo $n - \frac{1}{2}$,en la siguiente forma:

$$H^n_{x(i,j+\frac{1}{2},k+\frac{1}{2})} = \left[\frac{H^{n+\frac{1}{1}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} + H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}}{2}\right] \tag{2.33}$$

Sustituyendo en (2.32) y multiplicando ambos lados por $\triangle t$ se tiene:

$$\begin{aligned} &H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} = \\ &\quad \frac{\triangle t}{\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}\left[\frac{E^n_{y(i,j+\frac{1}{2},k+1)} - E^n_{y(i,j+\frac{1}{2},k)}}{\triangle z} - \frac{E^n_{z(i,j+1,k+\frac{1}{2})} - E^n_{z(i,j,k+\frac{1}{2})}}{\triangle y}\right. \\ &\quad \left. - M^n_{fuente(i+j+\frac{1}{2},k+\frac{1}{2})} - \sigma^{*n}_{(i+j+\frac{1}{2},k+\frac{1}{2})} \cdot \left[H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} + H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}\right]\right] \end{aligned} \tag{2.34}$$

Como los términos $H^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}}$ y $H^{n-\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}}$ aparecen en ambos lados de la ecuación (2.34) se agrupan todos términos del mismo tipo y se despeja $H^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}}$ en el lado izquierdo de la expresión, lo que conduce a:

$$\begin{aligned} &H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} = \\ &\quad \frac{\triangle t}{\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}\left[\frac{E^n_{y(i,j+\frac{1}{2},k+1)} - E^n_{y(i,j+\frac{1}{2},k)}}{\triangle z} - \frac{E^n_{z(i,j+1,k+\frac{1}{2})} - E^n_{y(i,j,k+\frac{1}{2})}}{\triangle y}\right. \\ &\quad \left. - M^n_{fuente(i+j+\frac{1}{2},k+\frac{1}{2})} - \sigma^{*n}_{(i+j+\frac{1}{2},k+\frac{1}{2})} \cdot \left[H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})} + H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}\right]\right] \end{aligned} \tag{2.35}$$

$$\left[1+\frac{\sigma^{*n}_{(i+j+\frac{1}{2},k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}\right]\cdot H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}=\left[1-\frac{\sigma^{*}_{(i,j+\frac{1}{2},k+\frac{1}{2})}}{2\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}\right]\cdot H^{n-\frac{1}{2}}x_{i,j+\frac{1}{2},k+\frac{1}{2}}+\frac{\triangle t}{\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}\cdot \left[\frac{E^n_{y(i,j+\frac{1}{2},k+1)}-E^n_{y(i,j+\frac{1}{2},k)}}{\triangle z}-\frac{E^n_{z(i,j+1,k+\frac{1}{2})}-E^n_{z(i,j,k+\frac{1}{2})}}{\triangle y}\right]-M^n_{(fuentei,j+\frac{1}{2},K+\frac{1}{2})} \quad (2.36)$$

Dividiendo ambos lados por $1+\frac{\sigma^{*n}_{(i+j+\frac{1}{2},k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}$ se obtiene la relación para la componente de campo H_x.

$$H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}=\frac{1-\frac{\sigma^{*n}_{(i+j+\frac{1}{2},k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}}{1+\frac{\sigma^{*n}_{(i+j+\frac{1}{2},k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}}\cdot H^{n-\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}+\frac{\frac{\triangle t}{\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}}{1+\frac{\sigma^{*n}_{(i+j+\frac{1}{2},k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i,j+\frac{1}{2},k+\frac{1}{2})}}}$$

$$\left[\frac{E^n_{y(i,j+\frac{1}{2},k+1)}-E^n_{y(i,j+\frac{1}{2},k)}}{\triangle z}-\frac{E^n_{z(i,j+1,k+\frac{1}{2})}-E^n_{z(i,j,k+\frac{1}{2})}}{\triangle y}-M^n_{fuente\ (i,j+\frac{1}{2},k+\frac{1}{2})}\right] \quad (2.37)$$

En la misma forma se obtienen las expresiones en diferencia finita para la componente de campo magnético H_y usando como referencia la figura 2.5

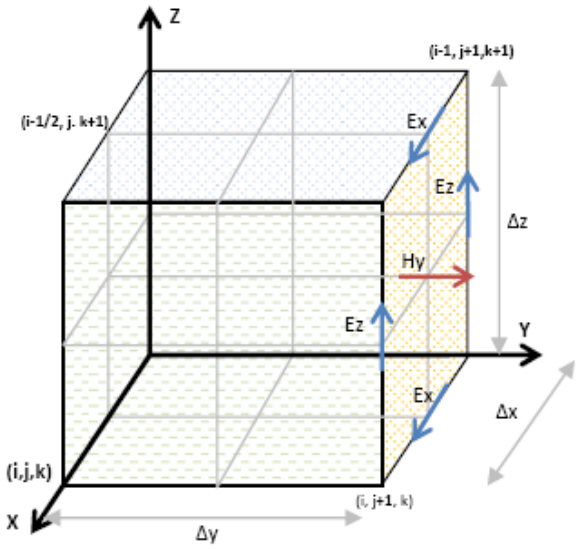


Figura 2.5: Componente H_y que depende de los campos E_x y E_z[11].

$$H^{n+\frac{1}{2}}_{y(i-\frac{1}{2},j+1,k+\frac{1}{2})}=\frac{1-\frac{\sigma^{*n}_{(i-\frac{1}{2},j+1,k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}{1+\frac{\sigma^{*n}_{(i-\frac{1}{2},j+1,k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}\cdot H^{n-\frac{1}{2}}_{y(i-\frac{1}{2},j+1,k+\frac{1}{2})}+\frac{\frac{\triangle t}{\mu_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}{1+\frac{\sigma^{*n}_{(i-\frac{1}{2},j+1,k+\frac{1}{2})\cdot\triangle t}}{2\mu_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}$$

$$\left[\frac{E^n_{x(i-\frac{1}{2},j+1,k+1)} - E^n_{x(i,j+1,k)}}{\triangle z} - \frac{E^n_{z(i,j+1,k+\frac{1}{2})} - E^n_{z(i-1,j+1,k+\frac{1}{2})}}{\triangle x} - M^n_{fuente\ (i,j+\frac{1}{2},k+\frac{1}{2})}\right] \tag{2.38}$$

Usando la figura 2.6 se deduce la expresión para H_z:

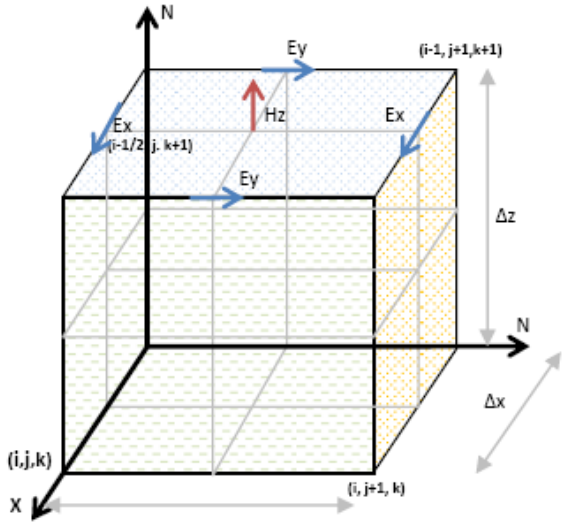


Figura 2.6: Representación de la ecuación (2.39). La componente H_z que depende de los campos E_y y E_x.

$$H^{n+\frac{1}{2}}_{z(i-\frac{1}{2},j+\frac{1}{2},k+1)} = \frac{1 - \frac{\sigma^{*n}_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}\cdot\triangle t}{2\mu_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}}{1 + \frac{\sigma^{*n}_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}\cdot\triangle t}{2\mu_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}} \cdot H^{n-\frac{1}{2}}_{z(i-\frac{1}{2},j+\frac{1}{2},k+1)} + \frac{\frac{\triangle t}{\mu_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}}{1 + \frac{\sigma^{*n}_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}\cdot\triangle t}{2\mu_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}}$$

$$\left[\frac{E^n_{y(i,j+\frac{1}{2},k+1)} - E^n_{y(i-1,j+\frac{1}{2},k+1)}}{\triangle x} - \frac{E^n_{x(i-\frac{1}{2},j+1,k+1)} - E^n_{z(i-\frac{1}{2},j,k+1)}}{\triangle y} - M^n_{fuente\ (i-\frac{1}{2},j+1,k+1)}\right] \tag{2.39}$$

De manera análoga también se obtienen las ecuaciones de diferencias finitas para las componentes de campo E_x, E_y, E_z. Usando la figura (2.7) La expresión para el campo E_x es:

$$E^{n+1}_{x(i-\frac{1}{2},j+1,k+1)} = \frac{1 - \frac{\sigma^{*n}_{(i-\frac{1}{2},j+1,k+1)}\cdot\triangle t}{2\epsilon_{(i-\frac{1}{2},j+1,k+1)}}}{1 + \frac{\sigma^{*n}_{(i-\frac{1}{2},j+1,k+1)}\cdot\triangle t}{2\epsilon_{(i-\frac{1}{2},j+1,k+1)}}} \cdot E^{n-\frac{1}{2}}_{x(i-\frac{1}{2},j+1,k+1)} + \frac{\frac{\triangle t}{\epsilon_{(i-\frac{1}{2},j+1,k+1)}}}{1 + \frac{\sigma^{*n}_{(i-\frac{1}{2},j+1,k+1)}\cdot\triangle t}{2\epsilon_{(i-\frac{1}{2},j+1,k+1)}}}$$

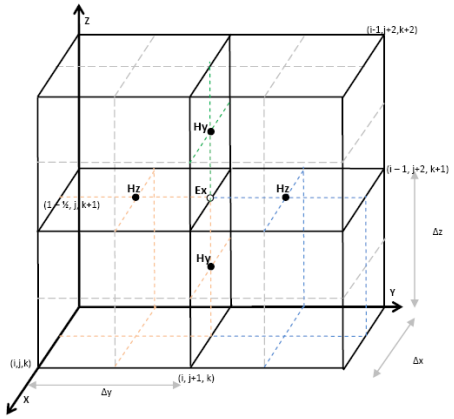


Figura 2.7: La componente E_x que depende de los campos H_y y Hz[11].

$$\left[\frac{H^{n+\frac{1}{2}}_{y(i-\frac{1}{2},j+1,k+\frac{3}{2})} - E^{n+\frac{1}{2}}_{y(i-\frac{1}{2},j+1,k+\frac{1}{2})}}{\triangle z} - \frac{H^{n+\frac{1}{2}}_{z(i-\frac{1}{2},j+\frac{3}{2},k+1)} - E^{n+\frac{1}{2}}_{z(i-\frac{1}{2},j+\frac{1}{2},k+1)}}{\triangle y} - M^{n+\frac{1}{2}}_{fuente\ (i-\frac{1}{2},j+1,k+1)}\right] \tag{2.40}$$

De igual forma se obtiene la ecuación para la componente de campo eléctrico E_y, de acuerdo a la figura 2.8.

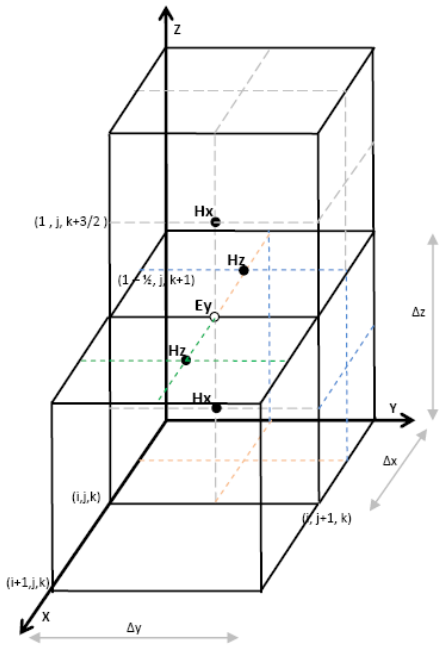


Figura 2.8: Representación de la ecuación (2.41). La componente E_y que depende de los campos H_x y H_z

$$E^{n+1}_{y(i,j+\frac{1}{2},k+1)} = \frac{1 - \frac{\sigma^{*n}_{(i,j+\frac{1}{2},k+1)} \cdot \triangle t}{2\epsilon_{(i,j+\frac{1}{2},k+1)}}}{1 + \frac{\sigma^{*n}_{(i,j+\frac{1}{2},k+1)} \cdot \triangle t}{2\epsilon_{(i,j+\frac{1}{2},k+1}}} \cdot H^{n-\frac{1}{2}}_{y(i,j+\frac{1}{2},k+1)} + \frac{\frac{\triangle t}{\epsilon_{(i,j+\frac{1}{2},k+1)}}}{1 + \frac{\sigma^{*n}_{(i,j+\frac{1}{2},k+1)} \cdot \triangle t}{2\epsilon_{(i,j+\frac{1}{2},k+1)}}}$$

$$\left[\frac{H^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{3}{2})} - E^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle z} - \frac{H^{n+\frac{1}{2}}_{z(i+\frac{1}{2},j+\frac{1}{2},k+1)} - E^{n+\frac{1}{2}}_{z(i-\frac{1}{2},j+\frac{1}{2},k+1)}}{\triangle x} - M^{n+\frac{1}{2}}_{fuente\ (i,j+\frac{1}{2},k+1)}\right] \quad (2.41)$$

Y finalmente la componente de campo eléctrico E_z, tomando como base la figura 2.9.

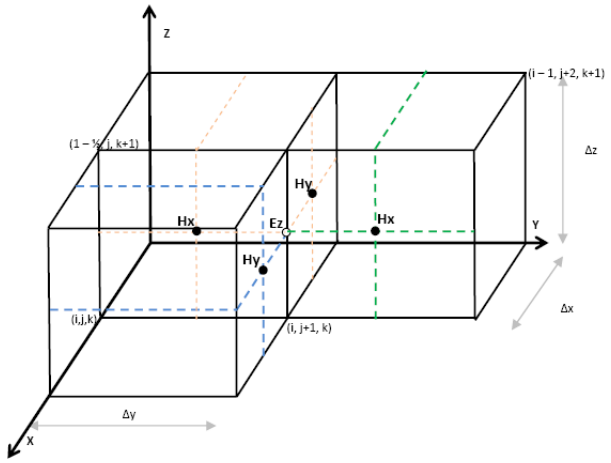


Figura 2.9: Representación de la ecuación (2.42). La componente E_z que depende de los campos H_y y H_x.

$$E^{n+1}_{z(i,j+1,k+\frac{1}{2})} = \frac{1 - \frac{\sigma^{*n}_{(i,j+1,k+\frac{1}{2})} \cdot \triangle t}{2\epsilon_{(i,j+1,k+\frac{1}{2})}}}{1 + \frac{\sigma^{*n}_{(i,j+1,k+\frac{1}{2})} \cdot \triangle t}{2\epsilon_{(i,j+1,k+\frac{1}{2})}}} \cdot E^{n-\frac{1}{2}}_{z(i,j+1,k+\frac{1}{2})} + \frac{\frac{\triangle t}{\epsilon_{(i,j+1,k+\frac{1}{2})}}}{1 + \frac{\sigma^{*n}_{(i,j+1,k+\frac{1}{2})} \cdot \triangle t}{2\epsilon_{(i,j+1,k+\frac{1}{2})}}}$$

$$\left[\frac{H^{n+\frac{1}{2}}_{y(i+\frac{1}{2},j+1,k+\frac{1}{2})} - E^{n+\frac{1}{2}}_{y(i-\frac{1}{2},j+1,k+\frac{1}{2})}}{\triangle x} - \frac{H^{n+\frac{1}{2}}_{x(i,j+\frac{3}{2},k+1)} - E^{n+\frac{1}{2}}_{x(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle y} - M^{n+\frac{1}{2}}_{fuente\ (i-\frac{1}{2},j+1,k+1)}\right] \quad (2.42)$$

Con este sistema de ecuaciones en diferencia finita, un nuevo valor de campo electromagnético en cualquier punto de la malla depende únicamente de su valor previo, de los valores previos de las componentes de los otros vectores de campo en los puntos adyacentes y las fuentes de corriente eléctrica y magnética conocidas.

2.7. Condiciones de frontera.

Es necesario considerar limitaciones en el área de trabajo ya que los equipos de cómputo no pueden almacenar una cantidad ilimitada de datos, por lo tanto, la región de cálculo se debe limitar por lo que debe de establecerse las condiciones de frontera que permitan poder llevar a cabo la simulación. La figura 2.10 muestra un dominio espacial finito Ω. En el interior de Ω aplicamos MDFDT que modela una propagación en todas direcciones. Las condiciones de frontera de la región de cálculo exterior se conocen como condiciones de frontera de absorción (Absobing Boundary Conditions).

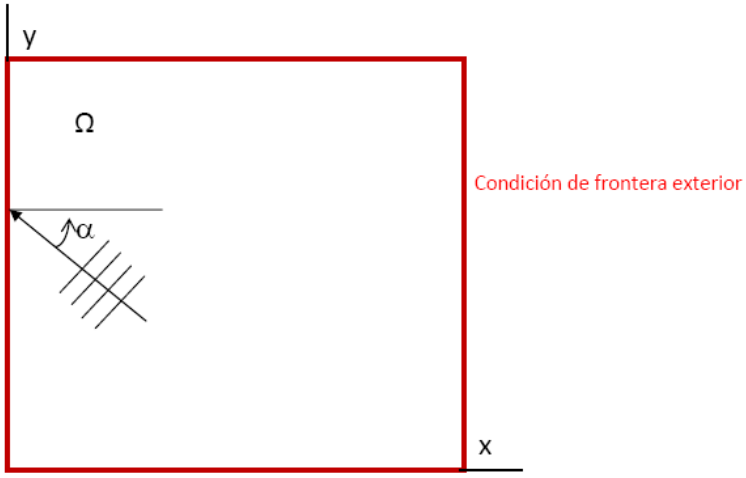


Figura 2.10: Condición de frontera exterior [11].

Las ABCś tienen una gran eficiencia ya que logran anulan las reflexiones espurias. El logro mas significativo es alcanzado en la reducción de las reflexiones hasta de-70 dB.

2.8. Capa Perfectamente Acoplada (PML)

Una propuesta interesante es el uso de los materiales absorbentes. Pero no fué hasta el año de 1994 cuando Jean Pierre Berenger introdujo un método conocido como capa perfectamente acoplada, (PML por sus siglas en inglés). Dicho método de Berenger proponía que las ondas planas de incidencia arbitraria, polarización y frecuencia eran acopladas en la frontera. Para lograr esto Berenger dividió cada componente de vector de campo de las ecuaciones de

Maxwell en dos componentes ortogonales generando así 12 componentes. Con esta propuesta Berenger introduce la construcción de la capa perfectamente acoplada adyacente a la frontera exterior de la región de cálculo para absorber todas las ondas incidentes.

2.9. Ecuaciones PML en dos dimensiones.

El campo eléctrico o magnético, dependiendo del modo de trasmisión (TE o TM) [12], varía transversalmente en el plano xy. Para el modo transversal magnético las ecuaciones de Maxwell se reducen a un sistema de 3 ecuaciones.

$$\varepsilon_0 \frac{\partial E_x}{\partial t} + \sigma E_x = \frac{\partial H_z}{\partial y} \tag{2.43}$$

$$\varepsilon_0 \frac{\partial E_y}{\partial t} + \sigma E_y = \frac{\partial H_z}{\partial x} \tag{2.44}$$

$$\mu_0 \frac{\partial H_z}{\partial t} + \sigma^* H_z = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \tag{2.45}$$

Esto implica dividir cada componente de campo magnético, transversal o normal en 2 subcomponentes[13]: H_{zx} y H_{zy}. Para el caso transversal magnético las 3 ecuaciones (2.43-2.45) son:

$$\varepsilon_0 \frac{\partial E_x}{\partial t} + \sigma_y E_x = \frac{\partial (H_{zx} + H_{zy})}{\partial y} \tag{2.46}$$

$$\varepsilon_0 \frac{\partial E_y}{\partial t} + \sigma_x E_y = \frac{\partial (H_{zx} + H_{zy})}{\partial x} \tag{2.47}$$

$$\mu_0 \frac{\partial H_{zx}}{\partial t} + \sigma_x^* H_{zx} = -\frac{\partial E_y}{\partial x} \tag{2.48}$$

$$\mu_0 \frac{\partial H_{zy}}{\partial t} + \sigma_y^* H_{zy} = \frac{\partial E_x}{\partial y} \tag{2.49}$$

Donde los parámetros $\sigma_x = \sigma_y = \sigma_{*x} = \sigma_{*y} = 0$ son conductividades eléctricas y magnéticas homogéneas. Podemos observar que:

- Si $\sigma_x = \sigma_y$ y $\sigma_x^* = \sigma_y^* = 0$, las ecuaciones anteriores se reducen a las ecuaciones de Maxwell en el vacío.

- Si y $\sigma_x = \sigma_y$ *y* $\sigma_x^* = \sigma_y^* = 0$, reduce las ecuaciones en un medio conductor.
- Si y $\sigma_y = \sigma_y^* = 0$, reduce las ecuaciones a un medio absorbente.

De ahí podemos considerar dos escenarios diferentes con capa PML:

- Si $\sigma_y = \sigma_y^* = 0$ el medio PML puede absorber propagación a lo largo del eje x (Ey, Hzx)
- Si $\sigma_x = \sigma_x^* = 0$ el medio PML puede absorber propagación a lo largo del eje y (Ex, Hzx)

Como se observa existe una estrecha relación entre,$(\sigma_x, \sigma_x^* 0, 0)$ y $(\sigma_y, \sigma_y^*, 0, 0)$, De acuerdo con Berenger si se aplica la condición:

$$\frac{\sigma}{\varepsilon_0} = \frac{\sigma^*}{\mu_0} \tag{2.50}$$

La técnica PML es mostrada en la figura 2.11. La capa PML se termina en un conductor perfecto que refleja la onda disminuida.

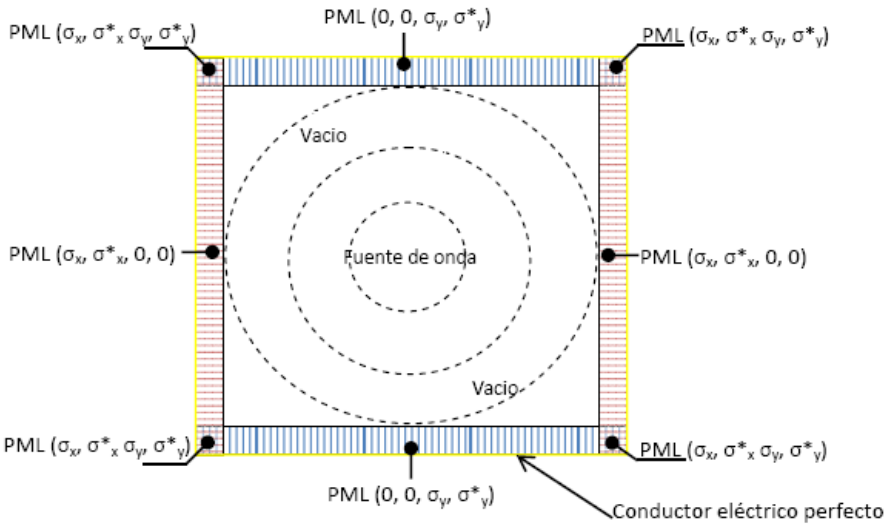


Figura 2.11: PML en dos dimensiones.

2.10. Ecuaciones PML en tres dimensiones

Utilizando el mismo procedimiento a una región de tres dimensiones [14], se obtiene la descomposición de los campos.

$$plano\ \ xy \ \longrightarrow\ \ H_z = H_{zx} + H_{zy}\ y\ E_z = E_{zx} + E_{zy}$$

$$plano\ \ yz \ \longrightarrow\ \ H_x = H_{xy} + H_{xz}\ y\ E_x = E_{xy} + E_{xz}$$

$$plano\ \ zx \ \longrightarrow\ \ H_y = H_{yx} + H_{yz}\ y\ E_y = E_{yx} + E_{yz}$$

En el medio PML se separa cada componete de campo electromagnético en 2 partes. Las seis componentes de campo eleectromagnetico resolviendo el rotacional de las ecuaciones de Maxwell llegan a ser 12 subcomponentes nombradas E_{xy}, E_{xz}, E_{yx}, E_{yz}, E_{zx}, E_{zy}, H_{xy}, H_{xz}, H_{yx}, H_{yz}, H_{zx}, H_{zy} las cuales son intercambiadas por las ecuaciones de Maxwell.

$$\mu\frac{\partial H_{xy}}{\partial t} + \sigma_y^* H_{xy} = \frac{\partial E_{zx} + E_{ZY}}{\partial y} \tag{2.51}$$

$$\mu\frac{\partial H_{zx}}{\partial t} + \sigma_z^* H_{zx} = \frac{\partial (E_{yz} + E_{yx})}{\partial z} \tag{2.52}$$

$$\mu\frac{\partial H_{yx}}{\partial t} + \sigma_x^* H_{yx} = \frac{\partial (E_{zx} + E_{zy})}{\partial x} \tag{2.53}$$

$$\mu\frac{\partial H_{yz}}{\partial t} + \sigma_z^* H_{yz} = -\frac{\partial (E_{xy} + E_{xz})}{\partial z} \tag{2.54}$$

$$\mu\frac{\partial H_{zx}}{\partial t} + \sigma_x^* H_{zx} = -\frac{\partial (E_{yz} + E_{yx})}{\partial x} \tag{2.55}$$

$$\mu\frac{\partial H_{zy}}{\partial t} + \sigma_y^* H_{zy} = \frac{\partial (E_{xy} + E_{xz})}{\partial y} \tag{2.56}$$

$$\varepsilon\frac{\partial E_{xy}}{\partial t} + \sigma_y^* E_{xy} = \frac{\partial (H_{zx} + H_{zy})}{\partial y} \tag{2.57}$$

$$\varepsilon\frac{\partial E_{xz}}{\partial t} + \sigma_z^* E_{xz} = \frac{\partial (H_{yz} + H_{yx})}{\partial z} \tag{2.58}$$

Cada derivada parcial se transforma en una ecuación de diferencia central. Primero se transforma la componente temporal.

$$\varepsilon\frac{\partial E_{yz}}{\partial t} + \sigma_z^* E_{yz} = \frac{\partial (H_{xy} + H_{xz})}{\partial z} \tag{2.59}$$

$$\varepsilon\frac{\partial E_{yx}}{\partial t} + \sigma_x^* E_{yx} = \frac{\partial (H_{zx} + H_{zy})}{\partial x} \tag{2.60}$$

$$\varepsilon\frac{\partial E_{zx}}{\partial t} + \sigma_x^* E_{zx} = \frac{\partial (H_{yz} + H_{yx})}{\partial x} \tag{2.61}$$

$$\varepsilon\frac{\partial E_{zy}}{\partial t} + \sigma_y^* E_{zy} = -\frac{\partial (H_{xy} + H_{xz})}{\partial y} \tag{2.62}$$

Donde los parámetros $, \sigma_x, \sigma_y, \sigma_z, \sigma_x^*, \sigma_y^*, \sigma_z^*$ son conductividades eléctricas y magnéticas homogéneas cuyas propiedades se describen a continuación:

- Si $\sigma_x, \sigma_y, \sigma_z$ y $\sigma_x^*, \sigma_y^*, \sigma_z^* = 0$ las ecuaciones se reducen a las ecuaciones de Maxwell. De esta forma el medio absorbente se define por las 12 ecuaciones anteriores.
- La técnica en tres dimensiones es una generalización de la implementación en 2D. Las ecuaciones de Maxwell se resuelven en el interior de la región de cálculo que es rodeada por una capa absorbente la cual es el agregado de la capa PML.
- En los 6 lados del dominio computacional el medio absorbente es acoplado al medio PML donde las conductividades transversales son iguales a cero.
- En los 12 bordes las conductividades son seleccionadas de tal manera que las conductividades transversales sean iguales en las interfaces localizadas entre el medio del borde y el medio de lado.
- En las 8 esquinas del dominio computacional las conductividades son elegidas iguales a las que se encuentran en los bordes adyacentes, entonces las conductividades transversales son iguales en la interfaz entre la capa del borde y la capa de la esquina.
- Con lo anterior una capa de absorción puede ser construida de tal manera que no existan reflexiones teóricas de cualquiera de las interfaces del dominio computacional. La construcción de la capa PML en tres dimensiones se muestra en la figura 2.12

Son 12 ecuaciones de campo obtenidas las cuales se discretizan para obtener la solución por medio del MDFDT. Se evalúa la diferencia finita central utilizando el punto de evaluación utilizado en la ecuación 2.51, para H_{xy}:

$$\mu \frac{\partial H_{xy}}{\partial t} + \sigma_y^* H_{xy} = \frac{\partial E_{zx} + E_{zy}}{\partial y}$$

De aquí, las derivadas parciales se transforman en una ecuación de diferencia central. Para ello se comienza con la componente temporal.

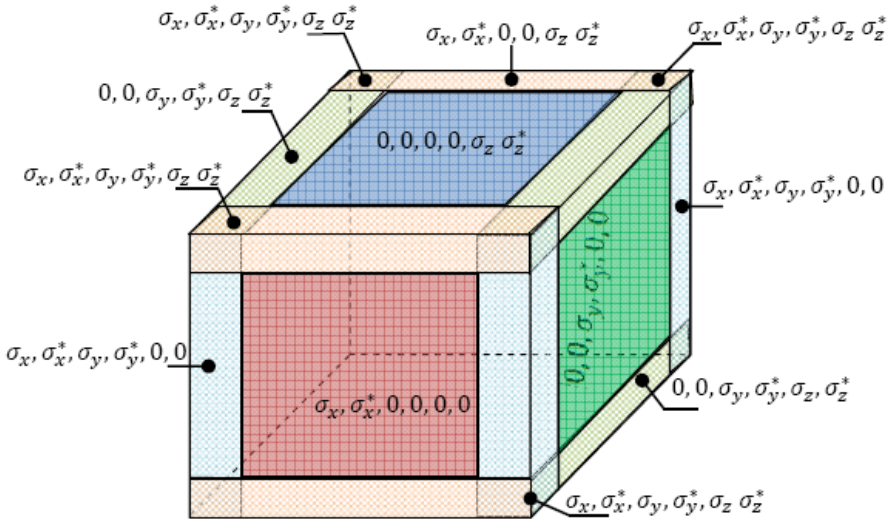


Figura 2.12: PML en tres dimensiones.

$$\frac{\partial H_{xy}}{\partial t} = \frac{H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle t} \tag{2.63}$$

la componente espacial se expresa como:

$$\frac{\partial (H_{zx} + H_{zy})}{\partial y} = \frac{E^n_{zx(i,j+1,k+\frac{1}{2})} + E^n_{zy(i,j+1,k+\frac{1}{2})} - E^n_{zx(i,j,k+\frac{1}{2})} - E^n_{zy(i,j,k+\frac{1}{2})}}{\triangle y} \tag{2.64}$$

$$H_{xy} = \frac{H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}}{2} \tag{2.65}$$

Sustituyendo (2.63-2.65) en (2.51) tenemos:

$$\mu\left(\frac{H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle t}\right) = -\left(\frac{E^n_{zx(i,j+1,k+\frac{1}{2})} + E^n_{zy(i,j+1,k+\frac{1}{2})} - E^n_{zx(i,j,k+\frac{1}{2})} - E^n_{zy(i,j,k+\frac{1}{2})}}{\triangle y}\right) - \sigma_y^*\left(\frac{H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}}{2}\right) \tag{2.66}$$

Dejando en un sólo término $H^{n+\frac{1}{2}}{}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}$

$$H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} - H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} = -\frac{\triangle t}{\mu}\frac{E^n_{zx(i,j+1,k+\frac{1}{2})} + E^n_{zy(i,j+1,k+\frac{1}{2})} - E^n_{zx(i,j,k+\frac{1}{2})} - E^n_{zy(i,j,k+\frac{1}{2})}}{\triangle y}$$

$$-\frac{\triangle t}{\mu}\sigma_y^*\frac{H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}-H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}}{2} \tag{2.67}$$

Considerando únicamente aquellos términos del mismo exponente temporal

$$H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}+\left(\frac{\sigma_y^*\triangle t}{2\mu}\right)H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}=-\frac{\triangle t}{\mu}\frac{E^n_{zx(i,j+1,k+\frac{1}{2})}+E^n_{zy(i,j+1,k+\frac{1}{2})}-E^n_{zx(i,j,k+\frac{1}{2})}-E^n_{zy(i,j,k+\frac{1}{2})}}{\triangle y}$$
$$-\left(\frac{\sigma_y^*\triangle t}{2\mu}\right)H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}+H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} \tag{2.68}$$

Factorizando los términos

$$H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}\cdot\left(1+\frac{\sigma_y^*\triangle t}{2\mu}\right)=H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}\cdot\left(\frac{\sigma_y^*\triangle t}{2\mu}\right)$$
$$-\frac{\triangle t}{\mu}\left(\frac{E^n_{zx(i,j+1,k+\frac{1}{2})}+E^n_{zy(i,j+1,k+\frac{1}{2})}-E^n_{zx(i,j,k+\frac{1}{2})}-E^n_{zy(i,j,k+\frac{1}{2})}}{\triangle y}\right) \tag{2.69}$$

Obtenemos

$$H^{n+\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}=H^{n-\frac{1}{2}}_{xy(i,j+\frac{1}{2},k+\frac{1}{2})}\cdot\frac{\left(1-\frac{\sigma_y^*\triangle t}{2\mu}\right)}{\left(1+\frac{\sigma_y^*\triangle t}{2\mu}\right)}-\frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma_y^*\triangle t}{2\mu}\right)}$$
$$\left(\frac{E^n_{zx(i,j+1,k+\frac{1}{2})}+E^n_{zy(i,j+1,k+\frac{1}{2})}-E^n_{zx(i,j,k+\frac{1}{2})}-E^n_{zy(i,j,k+\frac{1}{2})}}{\triangle y}\right) \tag{2.70}$$

Para obtener un nuevo valor de campo, es necesario el valor del mismo campo en un tiempo anterior y las componentes adyacentes del otro campo. A continuación son mostradas las 5 ecuaciones restantes que describen el campo magnético.

$$H^{n+\frac{1}{2}}_{xz(i,j+\frac{1}{2},k+\frac{1}{2})}=H^{n-\frac{1}{2}}_{xz(i,j+\frac{1}{2},k+\frac{1}{2})}\cdot\frac{\left(1-\frac{\sigma_z^*\triangle t}{2\mu}\right)}{\left(1+\frac{\sigma_z^*\triangle t}{2\mu}\right)}+\frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma_z^*\triangle t}{2\mu}\right)}$$
$$\left(\frac{E^n_{yz(i,j+\frac{1}{2},k+\frac{1}{2})}+E^n_{yx(i,j+\frac{1}{2},k+\frac{1}{2})}-E^n_{yz(i,j+\frac{1}{2},k+\frac{1}{2})}-E^n_{yx(i,j,k+\frac{1}{2})}}{\triangle y}\right) \tag{2.71}$$

$$H^{n+\frac{1}{2}}_{yx(i-\frac{1}{2},j+1,k+\frac{1}{2})} = H^{n-\frac{1}{2}}_{xz(i-\frac{1}{2},j+1,k+\frac{1}{2})} \cdot \frac{\left(1-\frac{\sigma^*_z \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} \left(\frac{E^n_{zx(i,j+1,k+\frac{1}{2})} + E^n_{zy(i,j+1,k+\frac{1}{2})} - E^n_{zx(i-1,j+1,k+\frac{1}{2})} - E^n_{yx(i-1,j+1,k+\frac{1}{2})}}{\triangle y}\right) \quad (2.72)$$

$$H^{n+\frac{1}{2}}_{yz(i-\frac{1}{2},j+1,k+\frac{1}{2})} = H^{n-\frac{1}{2}}_{yz(i-\frac{1}{2},j+1,k+\frac{1}{2})} \cdot \frac{\left(1-\frac{\sigma^*_z \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} \left(\frac{E^n_{xy(i-\frac{1}{2},j+1,k+1)} + E^n_{xz(i-\frac{1}{2},j+1,k+1)} - E^n_{xy(i-\frac{1}{2},j+1,k)} - E^n_{xz(i-\frac{1}{2},j+1,k)}}{\triangle y}\right) \quad (2.73)$$

$$H^{n+\frac{1}{2}}_{zx(i-\frac{1}{2},j+\frac{1}{2},k+1)} = H^{n-\frac{1}{2}}_{zx(i-\frac{1}{2},j+\frac{1}{2},k+1)} \cdot \frac{\left(1-\frac{\sigma^*_x \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_x \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_x \triangle t}{2\mu}\right)} \left(\frac{E^n_{yz(i,j+\frac{1}{2},k+1)} + E^n_{yx(i,j+\frac{1}{2},k+1)} - E^n_{yz(i-1,j+\frac{1}{2},k+1)} - E^n_{yx(i-1,j+\frac{1}{2},k+1)}}{\triangle y}\right) \quad (2.74)$$

$$H^{n+\frac{1}{2}}_{zy(i-\frac{1}{2},j+\frac{1}{2},k+1)} = H^{n-\frac{1}{2}}_{zy(i-\frac{1}{2},j+\frac{1}{2},k+1)} \cdot \frac{\left(1-\frac{\sigma^*_y \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_y \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_y \triangle t}{2\mu}\right)} \left(\frac{E^n_{xy(i-\frac{1}{2},j+1,k+1)} + E^n_{yx(i-\frac{1}{2},j+1,k+1)} - E^n_{xy(i-\frac{1}{2},j,k+1)} - E^n_{xz(i-\frac{1}{2},j,k+1)}}{\triangle y}\right) \quad (2.75)$$

De igual forma se obtienen las ecuaciones para los campos eléctricos

$$E^{n+\frac{1}{2}}_{xy(i-\frac{1}{2},j+1,k+1)} = E^{n-\frac{1}{2}}_{zx(i-\frac{1}{2},j+\frac{1}{2},k+1)} \cdot \frac{\left(1-\frac{\sigma^*_y \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_y \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_y \triangle t}{2\mu}\right)} \left(\frac{H^n_{zx(i-\frac{1}{2},j+\frac{3}{2},k+1)} + H^n_{zy(i-\frac{1}{2},j+\frac{3}{2},k+1)} - H^n_{zx(i-\frac{1}{2},j+\frac{1}{2},k+1)} - H^n_{zy(i-\frac{1}{2},j+\frac{1}{2},k+1)}}{\triangle y}\right) \quad (2.76)$$

$$E^{n+\frac{1}{2}}_{xz(i-\frac{1}{2},j+1,k+1)} = E^{n-\frac{1}{2}}_{xz(i-\frac{1}{2},j+\frac{1}{2},k+1)} \cdot \frac{\left(1-\frac{\sigma^*_z \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} \left(\frac{H^n_{yz(i-\frac{1}{2},j+1,k+\frac{3}{2})} + H^n_{yz(i-\frac{1}{2},j+1,k+\frac{3}{2})} - H^n_{yz(i-\frac{1}{2},j+1,k+\frac{1}{2})} - H^n_{yx(i-\frac{1}{2},j+\frac{1}{2},k+1)}}{\triangle z}\right) \quad (2.77)$$

$$E^{n+\frac{1}{2}}_{yx(i,j+\frac{1}{2},k+1)} = E^{n-\frac{1}{2}}_{xz(i,j+\frac{1}{2},k+1)} \cdot \frac{\left(1-\frac{\sigma^*_x \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_x \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_x \triangle t}{2\mu}\right)} \left(\frac{H^n_{zx(i+\frac{1}{2},j+\frac{1}{2},k+1)} + H^n_{zy(i+\frac{1}{2},j+\frac{1}{2},k+1)} - H^n_{zx(i-\frac{1}{2},j+\frac{1}{2},k+1)} - H^n_{zy(i-\frac{1}{2},j+\frac{1}{2},k+1)}}{\triangle x}\right) \quad (2.78)$$

$$E^{n+\frac{1}{2}}_{yz(i,j+\frac{1}{2},k+1)} = E^{n-\frac{1}{2}}_{yz(i,j+\frac{1}{2},k+1)} \cdot \frac{\left(1-\frac{\sigma^*_z \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_z \triangle t}{2\mu}\right)} \left(\frac{H^n_{xy(i,j+\frac{1}{2},k+\frac{3}{2})} + H^n_{xz(i,j+\frac{1}{2},k+\frac{3}{2})} - H^n_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} - H^n_{xz(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle z}\right) \quad (2.79)$$

$$E^{n+\frac{1}{2}}_{zx(i,j+1,k+\frac{1}{2})} = E^{n-\frac{1}{2}}_{zx(i,j+1,k+\frac{1}{2})} \cdot \frac{\left(1-\frac{\sigma^*_x \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_x \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_x \triangle t}{2\mu}\right)} \left(\frac{H^n_{yz(i+\frac{1}{2},j+1,k+\frac{1}{2})} + H^n_{yx(i+\frac{1}{2},j+1,k+\frac{1}{2})} - H^n_{yz(i-\frac{1}{2},j+1,k+\frac{1}{2})} - H^n_{yx(i-\frac{1}{2},j+1,k+\frac{1}{2})}}{\triangle x}\right) \quad (2.80)$$

$$E^{n+\frac{1}{2}}_{zy(i,j+1,k+\frac{1}{2})} = E^{n-\frac{1}{2}}_{zy(i,j+\frac{1}{2},k1)} \cdot \frac{\left(1-\frac{\sigma^*_y \triangle t}{2\mu}\right)}{\left(1+\frac{\sigma^*_y \triangle t}{2\mu}\right)} + \frac{\frac{\triangle t}{\mu}}{\left(1+\frac{\sigma^*_y \triangle t}{2\mu}\right)} \left(\frac{H^n_{xy(i,j+\frac{3}{2},k+\frac{1}{2})} + H^n_{(i,j+\frac{3}{2},k+\frac{1}{2})} - H^n_{xy(i,j+\frac{1}{2},k+\frac{1}{2})} - H^n_{xz(i,j+\frac{1}{2},k+\frac{1}{2})}}{\triangle x}\right) \quad (2.81)$$

2.10.1. Redondeo de los índices temporales

Las ecuaciones PML en tres dimensiones contienen índices espaciales y temporales fraccionarios por lo que es necesario realizar un ajuste de redondeo a los índices para el procesamiento de los datos ya que no es posible manejar este formato debido al uso del recorrido de matrices. Primero se redondea los pasos temporales, a partir de la ecuación 2.70, considerando un intervalo temporal entre $H^{n+\frac{1}{2}}_{xy}$ y $H^{n-\frac{1}{2}}_{xy}$,con base a la figura 2.13 tenemos:

Se observa que el tiempo $n-\frac{1}{2}$ está antes del cero por lo que toma el valor de n-1. Mientras el valor $n+\frac{1}{2}$ se ubica despues del cero pero antes del 1 y debido al principio del MDFDT se redondea al valor de n, de modo que quedan como H^n y H^{n-1} Como los índices temporales son enteros para la componente de campo eléctrico, no existe cambio que deba aplicarse.

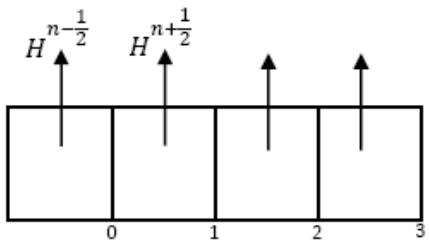


Figura 2.13: Localización del campo magnético en el tiempo

2.10.2. Redondeo de los índices espaciales

Para el redondeo de los subíndices se consideran los puntos centrales de evaluación, de modo que:

$$H_x = \left(i, j+\frac{1}{2}, k+\frac{1}{2}\right) \longrightarrow H_x = (i,j,k) \qquad E_x = \left(i-\frac{1}{2}, j+1, k+1\right) \longrightarrow E_x = (i,j,k)$$

$$H_y = \left(i-\frac{1}{2}, j+1, k+\frac{1}{2}\right) \longrightarrow H_y = (i,j,k) \quad E_y = \left(i, j+\frac{1}{2}, k+1\right) \longrightarrow E_y = (i,j,k)$$

$$H_z = \left(i-\frac{1}{2}, j+\frac{1}{2}, k+1\right) \longrightarrow H_z = (i,j,k) \quad E_z = \left(i, j+1, k+\frac{1}{2}\right) \longrightarrow E_z = (i,j,k)$$

Estos subíndices se encuentran en la primera celda de la región de cálculo y se toman de referencia para hacer el redondeo como se muestra arriba. Observado las ecuaciones anteriores podemos concluir que:

Tabla 2.1: Redondeo de los pasos espaciales

Valor de subíndice	Valor redondeado
$i-\frac{1}{2}, i$	i
$j+\frac{1}{2}, j+1$	j
$k+\frac{1}{2}, k+1$	k

Existe la posibilidad de que índices espaciales que están fuera de los valores de la tabla 2.1. Para ello es necesario realizar un ajuste en el cual se sumará o restará 1, si está antes o después según sea el caso. Las siguientes ecuaciones muestran la técnica del redondeo para la ecuación (2.70).

$$H^n_{xy(i,j,k)} = H^{n-1}_{xy(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{y((i,j,k))} \triangle t}{2\mu_{(i,j,k)}}\right)} - \frac{\frac{\triangle t}{\mu}}{\left(1 + \frac{\sigma^*_{y((i,j,k))} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{E^n_{zx(i,j,k)} + E^n_{zy(i,j.k)} - E^n_{zx(i,j,k-1)} - E^n_{zy(i,j,k-1)}}{\triangle y}\right) \quad (2.82)$$

Finalmente las ecuaciones PML para tres dimensiones en su forma redondeada es mostrada a continuación.

$$H^n_{xz(i,j,k)} = H^{n-1}_{xz(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu(i,j,k)}\right)}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu(i,j,k)}\right)} + \frac{\frac{\triangle t}{\mu(i,j,k)}}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu(i,j,k)}\right)} \left(\frac{E^n_{yz(i,j,k)} + E^n_{yx(i,j,k)} - E^n_{yz(i,j,k-1)} - E^n_{yx(i,j,k-1)}}{\triangle z}\right) \quad (2.83)$$

$$H^n_{yx(i,j,k)} = H^{n-1}_{xz(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu(i,j,k)}\right)}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu(i,j,k)}\right)} + \frac{\frac{\triangle t}{\mu(i,j,k)}}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu(i,j,k)}\right)} \left(\frac{E^n_{zx(i,j,k)} + E^n_{zy(i,j,k)} - E^n_{zx(i-1,j,k)} - E^n_{yx(i-1,j,k)}}{\triangle x}\right) \quad (2.84)$$

$$H^n_{yz(i,j,k)} = H^{n-1}_{yz(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu}\right)}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{E^n_{xy(i,j,k)} + E^n_{xz(i,j,k)} - E^n_{xy(i,j,k)} - E^n_{xz(i,j,k-1)}}{\triangle z}\right) \quad (2.85)$$

$$H^n_{zx(i,j,k)} = H^{n-1}_{zx(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu(i,j,k)}\right)} \left(\frac{E^n_{yz(i,j,k)} + E^n_{yx(i,j,k)} - E^n_{yz(i-1,j,k)} - E^n_{yx(i-1,j,k)}}{\triangle x}\right) \quad (2.86)$$

$$H^n_{zy(i,j,k)} = H^{n-1}_{zy(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{E^n_{xy(i,j,k)} + E^n_{yx(i,j,k)} - E^n_{xy(i,j-1,k)} - E^n_{xz(i,j-1,k)}}{\triangle y}\right) \quad (2.87)$$

$$E^n_{xy(i,j,k)} = E^{n-1}_{zx(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{H^n_{zx(i,j+1,k)} + H^n_{zy(i,j+1,k)} - H^n_{zx(i,j,k)} - H^n_{zy(i,j,k)}}{\triangle y}\right) \tag{2.88}$$

$$E^n_{xz(i,j,k)} = E^{n-1}_{xz(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{H^n_{yz(i,j,k+1)} + H^n_{yz(i,j,k+1)} - H^n_{yz(i,j,k)} - H^n_{yx(i,j,k)}}{\triangle z}\right) \tag{2.89}$$

$$E^n_{yx(i,j,k)} = E^{n-1}_{xz(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{H^n_{zx(i+1,j,k)} + H^n_{zy(i+1,j,k)} - H^n_{zx(i,j,k)} - H^n_{zy(i,j,k)}}{\triangle x}\right) \tag{2.90}$$

$$E^n_{yz(i,j,k)} = E^{n-1}_{yz(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{z(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{H^n_{xy(i,j,k+1)} + H^n_{xz(i,j,k+1)} - H^n_{xy(i,j,k)} - H^n_{xz(i,j,k)}}{\triangle z}\right) \tag{2.91}$$

$$E^n_{zx(i,j,k)} = E^{n-1}_{zx(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{x(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{H^n_{yz(i+1,j,k)} + H^n_{yx(i+1,j,k)} - H^n_{yz(i,j,k)} - H^n_{yx(i,j,k)}}{\triangle x}\right) \tag{2.92}$$

$$E^n_{zy(i,j,k)} = E^{n-1}_{zy(i,j,k)} \cdot \frac{\left(1 - \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)}{\left(1 + \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} + \frac{\frac{\triangle t}{\mu_{(i,j,k)}}}{\left(1 + \frac{\sigma^*_{y(i,j,k)} \triangle t}{2\mu_{(i,j,k)}}\right)} \left(\frac{H^n_{xy(i,j+1,k)} + H^n_{(i,j+1,k)} - H^n_{xy(i,j,k)} - H^n_{xz(i,j,k)}}{\triangle y}\right) \tag{2.93}$$

2.11. Conclusiones

El Método FDTD permite poder analizar los fenómenos electromagnético. Sin embargo es un Método el cual consume bastante recursos computacionales por lo que dentro de las mejoras que se les puede realizar a dicho algoritmo las cuales pueden utilizar diferentes técnicas de segmentación en cada una de las ecuaciones de Maxwell. Dentro de las ventajas del Método es poder cambiar la frecuencia de operación. Otras técnicas pueden ser agregadas reduciendo los tiempos de simulación usando Métodos de procesamiento paralelo. Estos serán discutidos con más detalle en capítulo 3,4 y 5.

Capítulo 3

Técnicas de paralelización y arquitecturas paralelas

El método de Diferencias Finitas en el Dominio del Tiempo emplea un algoritmo que tiene la gran desventaja de consumir bastantes recursos computacionales lo cual lo volvía casi imposible de tratar hasta hace unos años. Por tal razón surge la necesidad de explorar alternativas que permitan dar solución a dicho algoritmo. Hoy en día las supercomputadoras han tomado un papel bastante importante en la solución de algoritmos con complejidad computacional alta. Sin embargo este tipo de dispositivo tiene un costo monetario bastante alto. Es por ello que al buscar alternativas nos encontramos con una buena opción conocida como computación paralela, que constituye como una buena respuesta a todos aquellos algoritmos que consumen bastantes recursos computacionales. Tal es el caso de la programación en paralelo del MDFDT[15].

3.1. Cómputo paralelo

La computación paralela es una forma en que la ejecución de aplicaciones puede reducir el tiempo de ejecución. El principio del paralelismo está basado en que los problemas grandes

se puedan dividir en otros más pequeños y que puedan resolverse en un menor tiempo. Dicha técnica se ha utilizado desde hace muchos años, principalmente en el área de alto rendimiento, pero su uso ha ido en aumento debido a la velocidad de los procesadores. La computación paralela se ha convertido en el paradigma dominante de la arquitectura de computadoras, principalmente en la aparición de los procesadores de múltiples núcleos. Algunas formas de ver el paralelismo son:

- Paralelismo a nivel de bit. Es la forma de ejecutar los bits de un operando directamente a la unidad aritmética y lógica del procesador.
- Paralelismo a nivel de instrucción. Es la forma de ejecutar las instrucciones del lenguaje ensamblador.
- Paralelismo de datos. Se ejecuta código de un programa realizado en lenguaje de medio o alto nivel y puede ser ejecutado como proceso y en forma paralela en cada procesador.
- Paralelismo de tarea. El conjunto de procesos conforman una tarea, la cual puede ser una aplicación o software a ejecutar paralelamente, todo esto a nivel de usuario.

Las computadoras en paralelo pueden ser clasificadas por el nivel de paralelismo, con el hardware y el software que están ocupando o si es compatible con las nuevas arquitecturas para computadoras, y las nuevas tecnologías de red. Existen hoy en día diferentes técnicas para lograr el paralelismo y sin duda esto permitirá tener un mejor procesamiento de información.

3.2. Arquitecturas paralelas

Dentro de la clasificación de las arquitecturas paralelas se han propuesto infinidad de esquemas siendo el más aceptado el de Flyn. Su taxonomía está basada en la clasificación del flujo de datos e instrucciones en un sistema. Un flujo de instrucciones es el conjunto de instrucciones secuenciales que son ejecutadas por un único procesador, y un flujo de datos es el flujo secuencial de datos requeridos por el flujo de instrucciones. Con estas consideraciones, Flynn clasifica los sistemas en cuatro categorías que se describen a continuación.

3.2.1. Flujo único de instrucciones y único flujo de datos

La primera de ellas, de sus siglás en ingles SISD (single instruction stream, single data stream), utiliza el concepto de arquitectura serie de Von Neumann donde, en cualquier momento, sólo se está ejecutando una única instrucción. A menudo a los SISD se les conoce como computadoras series. Todas las máquinas SISD poseen un registro simple que se llama contador de programa que asegura la ejecución en serie del programa. Conforme se van leyendo las instrucciones de la memoria, el contador de programa se actualiza para que apunte a la siguiente instrucción a procesar en serie como se muestra en la figura 3.1.

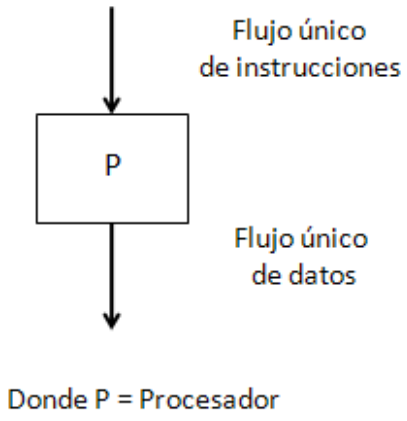


Figura 3.1: Funcionamiento de la arquitectura SISD

3.2.2. Flujo múltiple de instrucciones y único flujo de datos

De sus siglas en inglés MISD (multiple instruction stream, single data stream). Esto significa que varias instrucciones actúan sobre el mismo y único sección de datos. La mejor forma de interpretar a los MISD es a partir de la figura 3.2, en la cual un mismo flujo de datos fluye a través de numerosas unidades procesadoras. Arquitecturas altamente segmentadas, como los procesadores vectoriales, son clasificadas a menudo bajo este tipo de máquinas. Las arquitecturas segmentadas realizan el procesamiento vectorial a través de una serie de etapas, cada uno ejecutando una función particular produciendo un resultado intermedio. La razón por la cual dichas arquitecturas son clasificadas como MISD es que los elementos de un vector pueden ser considerados como pertenecientes al mismo dato, y todas las etapas del cauce representan múltiples instrucciones que son aplicadas sobre ese vector.

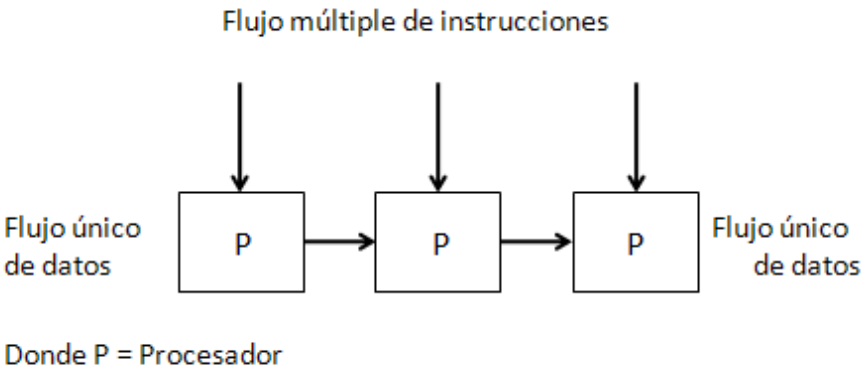


Figura 3.2: Funcionamiento de la arquitectura MISD

3.2.3. Flujo único de instrucción y flujo de datos múltiple

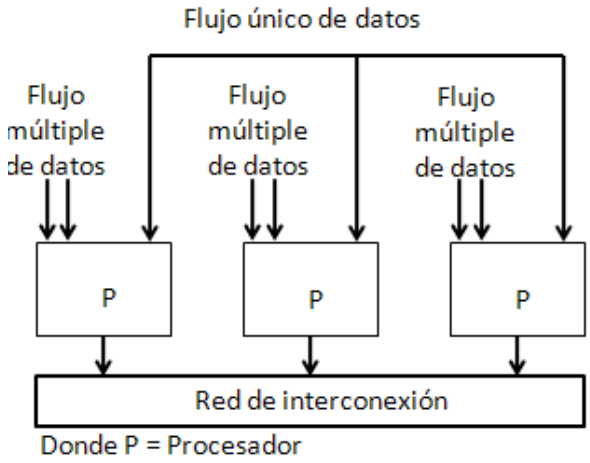


Figura 3.3: Funcionamiento de la arquitectura SIMD

De sus siglas en inglés SIMD (single instruction stream, multiple data stream). Esto significa que una única instrucción es aplicada sobre diferentes datos al mismo tiempo como se muestra en figura 3.3. En las máquinas de este tipo, varias unidades diferentes de procesamiento son invocadas por una única unidad de control. Al igual que las MISD, las SIMD soportan procesamiento vectorial (matricial) asignando cada elemento del vector a una única unidad funcional diferente para procesamiento concurrente. Por ejemplo, el cálculo de la paga para cada trabajador en una empresa, es repetir la misma operación sencilla para cada trabajador; si se dispone de una arquitectura SIMD esto se puede calcular en paralelo para cada trabajador.

3.2.4. Flujo de instrucciones múltiple y múltiple flujo de datos

De sus siglas en inglés MIMD (multiple instruction stream, multiple data stream). Son máquinas que poseen varias unidades de procesamiento en las cuales se pueden realizar múltiples instrucciones sobre diferente flujo de datos de forma simultánea como se muestra en la figura 3.4. Las MIMD son las arquitecturas más complejas, pero son también las que potencialmente ofrecen una mayor eficiencia en la ejecución concurrente o paralela. Aquí la concurrencia implica que no solo hay varios procesadores operando simultáneamente, sino que además hay varios programas (procesos) ejecutándose también al mismo tiempo.

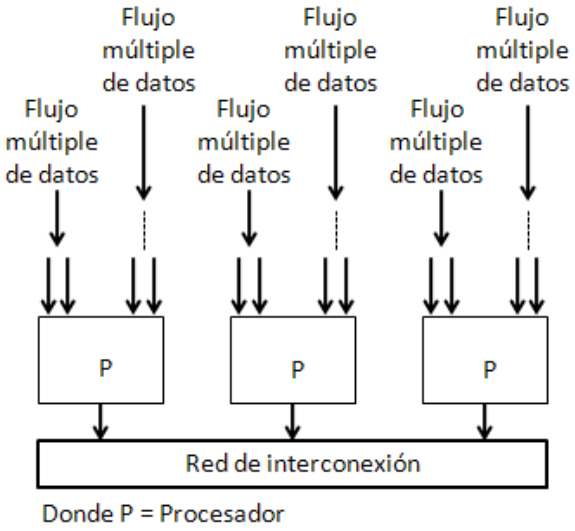


Figura 3.4: Funcionamiento de la arquitectura MIMD

3.3. Esquemas del uso de memoria

3.3.1. Memoria distribuida

La implementación del multiprocesador de memoria distribuida se da a través de lo que conocemos la arquitectura básica de un sistema multiprocesador de paso de mensaje se muestra en la figura 3.5. Un multiprocesador de paso de mensajes consta de nodos que normalmente están conectados mediante enlaces directos a otros nodos.

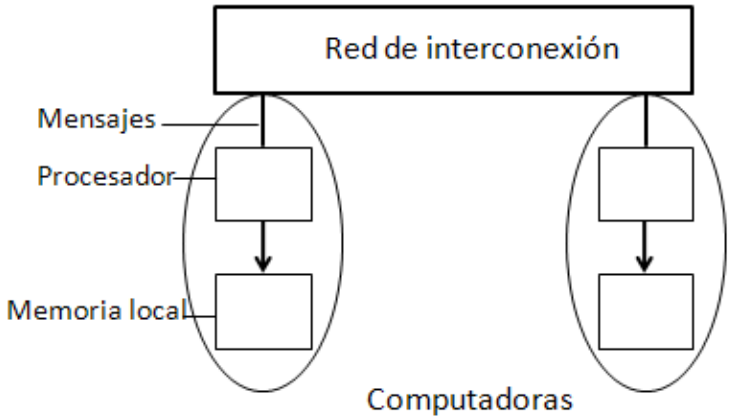


Figura 3.5: Arquitectura memoria distribuida

Cada nodo está compuesto por un procesador junto con una memoria local y canales de comunicación de E/S. No existen localizaciones de memoria global. La memoria local puede usar las mismas direcciones. Dado que cada nodo es una computadora, a los multiprocesadores de paso de mensajes se les suelen denominar multicomputadoras. El número de nodos puede ser tan pequeño como 16 (o menos), o tan grande como varios millares. Sin embargo, la arquitectura de paso de mensajes muestra sus ventajas sobre los sistemas de memoria compartida cuando el número de procesadores es grande. Para sistemas multiprocesadores pequeños, los sistemas de memoria compartida presentarán probablemente un mejor rendimiento y mayor flexibilidad. El número de canales físicos entre nodos suele oscilar entre cuatro y ocho. La principal ventaja de esta arquitectura es que es directamente escalable y presenta un bajo costo para sistemas grandes. La transferencia de datos se realiza a través de la red de interconexión que conecta un subconjunto de procesadores con otro subconjunto. La transferencia de operaciones entre procesadores y otros se realiza por múltiples transacciones entre ellos y son transmitidas a través del diseño que se estableció de la red. Dado que la memoria está distribuida entre los diferentes elementos de proceso, a estos sistemas se les llama distribuidos aunque no hay que olvidar que puede haber sistemas que tengan la memoria distribuida pero que al mismo tiempo tenga la memoria de forma compartida y por lo tanto no sean multicomputadora. Además, y dado que se explota mucho la localidad, a estos sistemas se les llama débilmente acoplados, ya que los módulos funcionan casi independiente unos de otros.

3.3.2. Memoria compartida

El acceso por memoria compartida nos dice que cualquier dirección de memoria es accesible desde cualquier procesador/núcleo: dirección única por posición de memoria. En la figura 3.6 se ilustra a la arquitectura de memoria compartida[16].

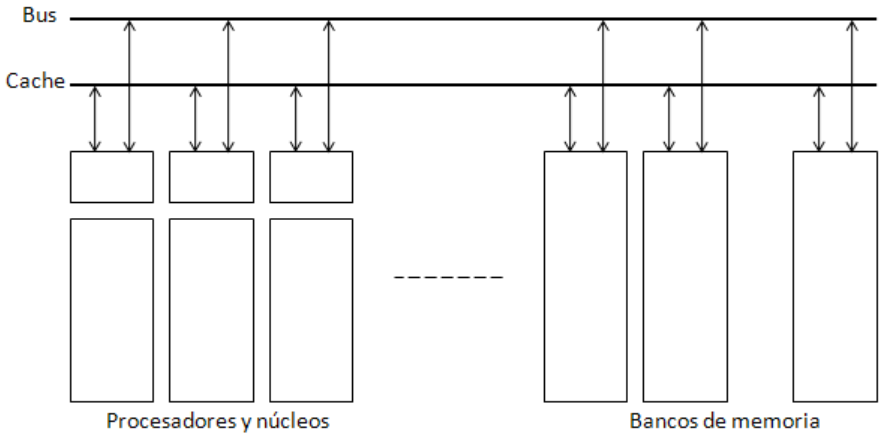


Figura 3.6: Arquitectura de memoria compartida

Los multiprocesadores de memoria compartida presentan algunas desventajas como por ejemplo:

1) Son necesarias técnicas de sincronización para controlar el acceso a las variables compartidas.
2) La contención en la memoria puede reducir significativamente la velocidad del sistema.
3) No son fácilmente ampliables para acomodar un gran número de procesadores.

Un sistema multiprocesador alternativo al sistema de memoria compartida que elimina los problemas arriba indicados es tener únicamente una memoria local por procesador eliminando toda la memoria compartida del sistema. El código para cada procesador se carga en la memoria local al igual que cualquier dato que sea necesario. Los programas todavía están divididos en diferentes partes, como en el sistema de memoria compartida, y dichas partes todavía se ejecutan concurrentemente por procesadores individuales. Cuando los procesadores necesitan acceder a la información de otro procesador, o enviar información a otro procesador, se comunican a otro procesador enviando mensajes. Los datos no están almacenados

globalmente en el sistema; si más de un proceso necesita un dato, éste debe duplicarse y ser enviado a todos los procesadores peticionarios. A estos sistemas se les suele denominar multicomputadoras

3.4. Hilos y procesos

Los hilos pueden ser definidos como una instancia de subtareas divididas para ejecutarse en paralelo, muchas aplicaciones, se dividen en múltiples hilos. En la figura 3.7 se ilustra un proceso.

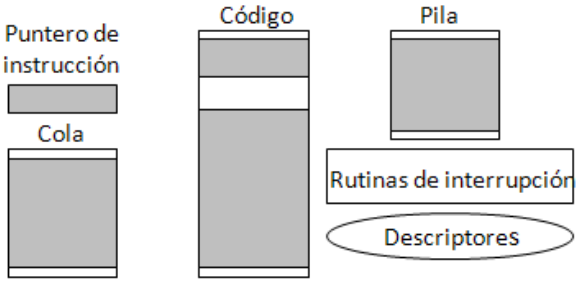


Figura 3.7: Esquema de un proceso

Un puntero de instrucción (IP) apunta la dirección en la siguiente instrucción del hilo para ser ejecutada. Una pila es solicitada por el procedimiento de llamadas así como también un sistema de rutinas y archivos. Como se muestra en la figura 3.8 cada hilo tiene su propio IP apuntando a la siguiente instrucción del hilo que será ejecutada. Cada hilo necesita su propia pila y también una reserva de información en lo que refiere a registros pero comparte el código. Un proceso puede tener más de un hilo.

La creación de un hilo puede tomar tres ordenes de magnitud menos tiempo que la creación del proceso. Además, un hilo inmediatamente tendrá acceso a las variables globales compartidas [16].

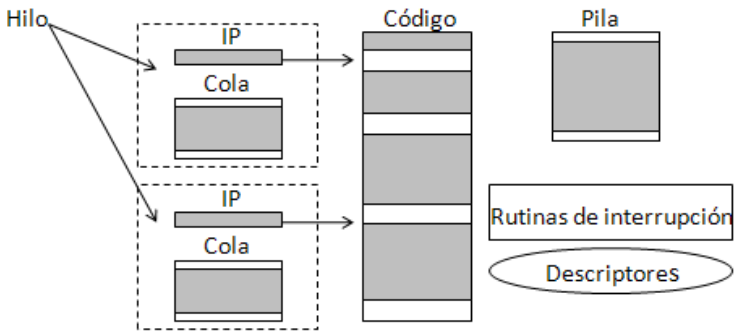


Figura 3.8: Esquema de un Hilo

3.5. Medidas de desempeño

Cuando se ejecuta un proceso es necesario evaluar el desempeño de dicho algoritmo. Factor de velocidad y eficiencia entre otros, muestran el comportamiento en tiempo en la ejecución.

3.5.1. Factor de velocidad

Una medida de desempeño entre un sistema de multiprocesadores y un solo procesador es el factor de velocidad, definido como:

$$s(n) = \frac{ts}{tp} \tag{3.1}$$

Donde:

ts=Es el tiempo de ejecución usando un solo procesador

tp=Es el tiempo utilizando un multiprocesador con n procesadores.

$s(n)$=Da el incremento de velocidad utilizando un multiprocesador.

El factor de velocidad o también conocido como aceleración simple, permite obtener la relación de cuanto se mejora el tiempo de ejecución [17].

3.5.2. Ley de Amdhal

Suponiendo que habrá algunas partes que son ejecutados en un procesador, la situación ideal sería que para todos los procesadores disponibles se pueda operar simultáneamente en diferentes tiempos [17]. Si la fracción del cómputo que no puede ser dividido en tareas concurrentes , y no incurren encabezados cuando el cómputo es dividido dentro de partes concurrentes, el tiempo para representar el cómputo con "n"procesadores se muestra en la siguiente ecuación.

$$TP = Sts + \left(\frac{(1-S)ts}{p}\right) \tag{3.2}$$

Donde:

tp= Tiempo en paralelo

S = es la fracción de trabajo serial intrínseco

ts= es el tiempo de ejecución usando un solo procesador

p= número de núcleos

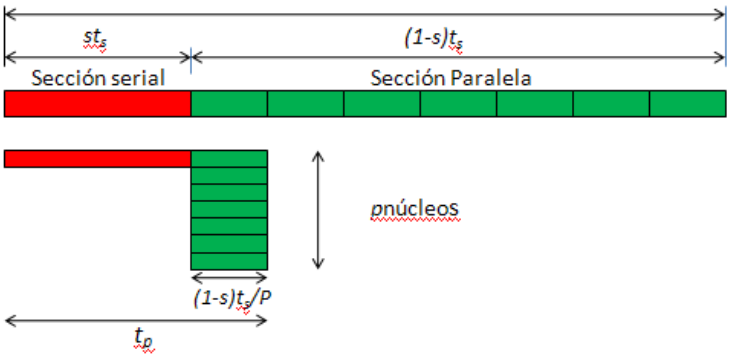


Figura 3.9: Distribución del tiempo en paralelo

3.5.3. Eficiencia

La eficiencia, E, está definida como [17]:

$$E = \frac{ts}{tp \times n} \tag{3.3}$$

Donde:
ts : Tiempo de ejecución usando un procesador

tp : Tiempo de ejecución usando un multiprocesador.

n: número de procesadores La eficiencia computacional nos muestra un porcentaje de los recursos computacionales utilizados cuando se corren los diferentes procesos.

3.6. Técnicas de paralelización

3.6.1. Técnicas basadas en OPENMP

OPENMP es una interfaz de aplicación de programa (API), que nos permite añadir concurrencia a las aplicaciones mediante paralelismo con memoria compartida [18]. Se basa en la creación de hilos de ejecución paralelos compartiendo las variables del proceso padre que los crea. Esta disponible en múltiples plataformas y lenguajes, desde las derivadas de unix hasta las plataformas windows. Existen extensiones para los lenguajes más conocidos como c, c++, fortran.

Modelo de programación

OPENMP está basado sobre la existencia de múltiples hilos en la programación con memoria compartida. De acuerdo a la figura 3.10, un proceso de memoria compartida consiste de múltiples hilos los cuales son generados a partir de un hilo maestro según el modelo de bifurcación unión utilizado por OPENMP [19].

Todos las aplicaciones que utilizan OPENMP comienzan con un solo proceso el cual es el hilo maestro. El hilo maestro ejecuta secuencialmente hasta la primera región paralela construida que es encontrada. El hilo maestro después crea un equipo de hilos paralelos y cuando el equipo de hilos completa las declaraciones en la construcción de la región paralela, sincroniza y termina, dejando sólo el hilo maestro. El paralelismo con OPENMP es especificado a través del uso de directivas en el compilador las cuales son insertadas en

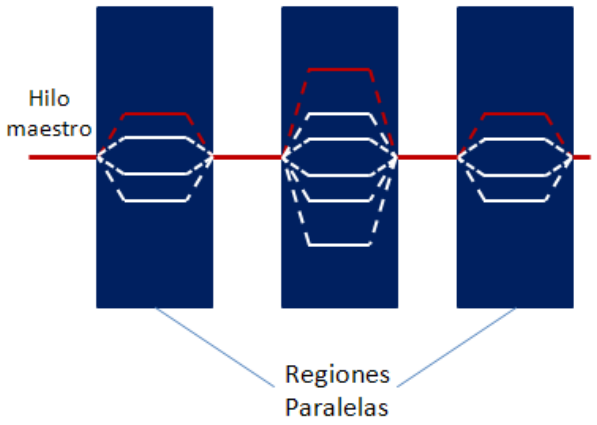


Figura 3.10: Esquema de bifurcación unión

C/C++ o Fortran a través del código. Una directiva (o centinela en Fortran) es la forma de indicarle al compilador alguna instrucción al momento de la compilación [20].

3.6.2. Técnicas Basadas en MPI

MPI de sus siglas en Inglés (Message Passing Interface) es una interfaz de paso de mensaje y es utilizada cuando se trabaja con sistemas con memoria distribuida. Este estándar va permitir poder conectar a computadoras con otras computadoras a través de una red. Utiliza el compilador LAM-MPI o mpich2 a través de procesos en Linux que normalmente están cargados de memoria esperando una señal para habilitar y deshabilitar a estos mismos procesos [21].

Modelo de programación

En el modelo de programación MPI, consta de uno o más procesos comunicados a través de llamadas a rutinas de librerías para mandar (send) y recibir (receive) mensajes hacia y desde otros procesos. En la mayoría de las implementaciones de MPI, se crea un conjunto fijo de procesos al iniciar el programa, y un proceso es creado por cada tarea. La habilidad de MPI para probar mensajes da como resultado el soportar comunicaciones asíncronas. Probablemente una de las características más importantes del MPI es el soporte para la programación modular. Un mecanismo llamado comunicador permite al programador del

MPI definir módulos que encapsulan estructuras internas de comunicación (estos módulos pueden ser combinados secuencialmente o paralelamente).

3.6.3. Técnicas basadas en GPU's

La unidad de procesamiento gráfico o GPU (acrónimo del inglés graphics processing unit) es un procesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos y o aplicaciones 3D interactivas. De esta forma, mientras gran parte de lo relacionado con los gráficos se procesa en la GPU, la CPU puede dedicarse a otro tipo de cálculos (como la inteligencia artificial o los cálculos mecánicos en el caso de los videojuegos). Una GPU implementa ciertas operaciones gráficas llamadas primitivas optimizadas para el procesamiento gráfico. Una de las primitivas más comunes para el procesamiento gráfico en 3D es el antialiasing, que suaviza los bordes de las figuras para darles un aspecto más realista. Adicionalmente existen primitivas para dibujar rectángulos, triángulos, círculos y arcos. Las GPU actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos [22].

CUDA

CUDA es una arquitectura de cálculo paralelo de NVIDIA que aprovecha la gran potencia de la GPU (unidad de procesamiento gráfico) para proporcionar un incremento extraordinario del rendimiento del sistema. Con una base instalada de más de 128 millones de GPUs aptas para CUDA, miles de desarrolladores, científicos e investigadores están encontrando innumerables aplicaciones prácticas para esta tecnología en campos como el procesamiento de video, la astrofísica, la biología y la química computacional, la simulación de mecánica de fluidos, la interferencia electromagnética, la reconstrucción de imágenes, el análisis sísmico o el trazado de rayos entre otras.

Los sistemas informáticos están pasando de realizar *el procesamiento central* en la CPU a realizar *coprocesamiento* repartido entre la CPU y la GPU. Para posibilitar este nuevo paradigma computacional, NVIDIA ha inventado la arquitectura de cálculo paralelo CUDA,

que ahora se incluye en las GPUs GeForce, ION, Quadro y Tesla, lo cual representa una base instalada considerable para los desarrolladores de aplicaciones. En el mercado de consumo, prácticamente todas las aplicaciones de video se han acelerado, o pronto se acelerarán, a través de CUDA, como demuestran diferentes productos de Elemental Technologies, MotionDSP y LoiLo, Inc CUDA ha sido recibida con entusiasmo por la comunidad científica. Por ejemplo, se está utilizando para acelerar AMBER, un simulador de dinámica molecular empleado por más de 60.000 investigadores del ámbito académico y farmacéutico de todo el mundo para acelerar el descubrimiento de nuevos medicamentos. En el mercado financiero, Numerix y Compatibl introdujeron soporte de CUDA para una nueva aplicación de cálculo de riesgo de contraparte y, como resultado, se ha multiplicado por 18 la velocidad de la aplicación. Cerca de 400 instituciones financieras utilizan Numerix en la actualidad. Un buen indicador de la excelente acogida de CUDA es la rápida adopción de la GPU Tesla para aplicaciones de GPU Computing. En la actualidad existen más de 700 clusters de GPUs instalados en compañías Fortune 500 de todo el mundo, lo que incluye empresas como Schlumberger y Chevron en el sector energético o BNP Pariba en el sector bancario. Por otra parte, la inminente llegada de los nuevos sistemas operativos de Microsoft y Apple (Windows 7 y Lion) al mercado convertirá el GPU Computing en una tecnología de uso masivo. En estos nuevos sistemas, la GPU no actuará únicamente como procesador gráfico, sino como procesador paralelo de propósito general accesible para cualquier aplicación.

3.7. Clusters

Las simulaciones en computadora son vitales para el estudio de muchos problemas, desde el diseño en Ingeniería hasta el estudio de procesos complejos en la Naturaleza. Sin embargo, el alcance y la precisión de estas simulaciones están limitados por la potencia computacional de las supercomputadoras más potentes. Estos clusters se utilizan para cualquier tarea que requiera enormes cantidades de cómputo: simulaciones científicas,renderización de gráficos,modelado meteorológico, electromagnetismo computacional, compresión de audio, procesamiento de imágenes entre otras.

Se entiende por cluster a una agrupación de computadoras independientes llamados

también nodos, que tienen componentes de hardware comunes y que utilizan un software de interconexión, que hacen que estos funcionen como una supercomputadora. En los ámbitos científicos, los grupos de clusters consisten en nodos de computadoras, nodos de almacenamiento y uno o más nodos front end (servidores) y en algunos casos hay nodos adicionales dedicadas al monitoreo. Estos nodos pueden estar interconectados por redes diferentes, generalmente redes de alta velocidad, con un ancho de banda muy alto. Esta velocidad solo puede ser alcanzada por tecnologías como Gigabit Ethernet, Myrinet e infiniband, ya que éstos alcanzan los 40 Gigabits/seg. El trabajo de los clusters consiste en:

- Aumentar la disponibilidad y disminuir la probabilidad de fallos, ya que en caso de que un equipo en el cluster falle, existe otro que lo reemplace.
- Facilitar la escalabilidad, puesto que los equipos interconectados permiten que se añadan más de estos equipos a la red.

3.7.1. Ventajas en la construcción de un cluster de computadoras

Construir un cluster puede aportar importantes ventajas en gran variedad de aplicaciones y ambientes:

- Incremento de velocidad de procesamiento ofrecido por los clusters de alto rendimiento.
- Incremento del número de transacciones o velocidad de respuesta ofrecido por los clusters de balanceo de carga.
- Incremento de la confiabilidad y la robustez ofrecido por los clusters de alta disponibilidad.

Por ejemplo, en las investigaciones meteorológicas y pronóstico numérico del estado del tiempo, se requiere el manejo de cantidades masivas de datos y cálculos muy complejos. Al combinar el poder de muchas máquinas del tipo estación de trabajo o servidor, se pueden alcanzar niveles de rendimiento similares a los de las supercomputadoras, pero a menor costo. Otra situación de aplicabilidad de un cluster sería en un sitio web que soporte mucho tráfico.

Si no se cuenta con un plan de alta disponibilidad, cualquier problema menor de una tarjeta de red, puede hacer que un servidor quede completamente inutilizado. Pero al contar con servidores redundantes y servidores de respaldo instantáneos, se puede reparar el problema mientras el sitio sigue funcionando sin suspensión del servicio.

3.7.2. Puntos a considerar en la elección de un cluster

Por sus características especiales, hay varias cuestiones particulares asociadas a esta tecnología que deben ser tenidas en cuenta. Uno de los principales problemas a los que hay que hacer frente cuando se construye un cluster es buscar y eliminar los puntos de fallo únicos (single points of failure). Si se trabaja en un cluster de supercomputación que depende de un servidor central para repartir las tareas, si este servidor cae, todo el cluster quedará inservible. Igualmente, si se trata de un cluster de balanceo de carga o de alta disponibilidad, se deben establecer garantías de que los servidores seguirán funcionando, pero si estos servidores están conectados a una red corporativa o a Internet, mediante una sola interfaz, un fallo en ella dejaría aislado al sistema. Es importante lograr la redundancia para evitar que el fallo de un sólo componente hardware (recordemos que en un cluster van a integrarse gran número de elementos con lo que la probabilidad de fallo crece) anule la funcionalidad de todo el sistema. Otra cuestión importante, es elegir correctamente la tecnología que vamos a utilizar en función de nuestras necesidades. Mantener un cluster sobre una red Ethernet de 10 Mb, puede resultar una buena decisión si el cluster sólo tiene unos cuantos nodos, pero en el momento en que se inserten más nodos, la red va a convertirse en un cuello de botella que obligará los servidores a estar desocupados en espera de los datos durante, quizá demasiado tiempo

3.7.3. Cluster tipo Beowulf

Thomas Sterling y Don Becker, en 1994, trabajando en el CESDIS (Center of Excellence in Space Data and Information Sciences NASA) bajo la tutela del proyecto ESS (Earth and Space Sciences), construyeron un cluster computacional consistente en procesadores de tipo x86 comerciales conectados por una red Ethernet de 10Mb. Llamaron a su máquina

Beowulf, nombre de un héroe de la mitología danesa relatado en el libro La Era de las Fábulas, del autor norteamericano Thomas Bulfinch (Beowulf derrotó al monstruo gigante Grendel). Inmediatamente, aquello fue un éxito y pronto se difundió a través de la NASA y las comunidades académicas y de investigación. Los clusters Beowulf están hoy reconocidos como un tipo de clusters dentro de los HPC(High Performance Computer)[23].

Un Beowulf es una clase de computadora de altas prestaciones principalmente construida a base de un cluster de componentes hardware estándard. Un Beowulf ejecuta un sistema operativo de libre distribución como Linux o FreeBSD, y se interconecta mediante una red privada de gran velocidad. Generalmente se compone de un grupo de PCs o estaciones de trabajo dedicados a ejecutar tareas que precisan una alta capacidad de cálculo. Los nodos en el cluster de computadoras no se hayan en los puestos de trabajo de los usuarios, sino que están totalmente dedicados a las tareas asignadas al cluster. Generalmente, el cluster se haya conectado al mundo exterior por un solo nodo. El software puede ejecutarse más rápido en un Beowulf si se dedica algún tiempo a reestructurar los programas. En general es necesario partirlos en tareas paralelas que se comunican usando alguna librería como MPI o PVM, o sockets o SysV IPC. Figura 3.11. muestra la estructura básica de un cluster tipo Beowulf. Algunas aplicaciones en las que se utiliza este tipo de cluster se encuentra electromagnetismo computacional [24], fotónica [25], aplicaciones en tiempo real [26], redes neuronales [27], etc.

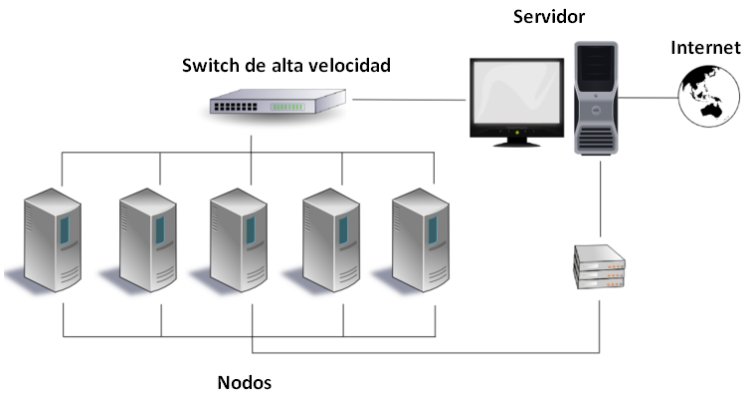


Figura 3.11: Cluster Beowulf

3.7.4. Cluster tipo MOSIX

Mosix fue desarrollado en la universidad hebrea de Jerusalén por Mnon Barack y Amnon Siloh, como una alternativa de conexión de red. Mosix es una herramienta creada para balancear la carga (software en ejecución), de manera que los usuarios no se den cuenta que los procesos del software en ejecución estén trabajando en segundo orden; al final los nodos del cluster se comportan como una sola máquina, y de esta forma se incrementa el aprovechamiento de los programas que corren paralelamente. Mosix fue diseñado para correr en plataformas x86, que incluyen a computadoras personles, estaciones de trabajo y servidores. Todos estos hosts pueden ser conectados a través de una red LAN o en una red de interconexión muy rápida [28]. Dependiendo del tipo de aplicaciones, Mosix puede ser utilizado para la configuración de un pequeño cluster de computadoras personales que utilizan un estándar de comunicación como: Fast Ethernet(100 Mbps), Giga Ethernet (1000 Mbps) o Mirynet (10 Gps), etc. En un sistema Mosix, los programadores y los usuarios pueden crear aplicaciones y ejecutarlas en un cluster Mosix, el cual crea múltiples procesos y en forma transparente pueden migrar estos procesos a los diferentes nodos conectados a la red.

Algunas características del kernel de Mosix son:

- Proveer una imagen única de sistema (SSI). La transparencia de los recursos lo realiza mediante un algoritmo interno (de colas) dando a cada nodo la información y la disponibilidad para ejecutar procesos migrados. Mosix trabaja con procesos Linux y Mosix. Los procesos Linux trabajan normalmente en modo nativo y no son afectados por el trabajo del kernel de Mosix y no pueden ser migrados. Los procesos de Mosix son aplicaciones de usuario que se favorecen al ser migrados de nodo a nodo.
- Provee un ambiente de ejecución seguro para ejecutar procesos remotos. En el cluster Mosix solamente se trabajan con procesos conocidos a nivel de kernel de Mosix y no se toman en cuenta procesos que no sean Mosix, creando así un ambiente de ejecución seguro.
- Trabajo de cola. Se incorpora una sola dinámica que permite trabajar con un número

de tareas, que se puedan ejecutar con procedimientos FIFO(First in, First out, Primero en entrar, primero en salir).

- Soporte de procesos batch, puntos de recuperación y restauración. Los archivos batch que se realicen en cualquier nodo, se pueden guardar y posteriormente ejecutar en otro nodo o en el nodo principal, y además Mosix tiene la capacidad de guardar las imágenes de los procesos en ejecución, creando así un punto de restauración, almacenando la información en un archivo batch para que posteriormente pueda ser restaurado.

- Cuenta con un monitor, que provee de información sobre los recursos que se están ejecutando en el cluster y además nos puede mostrar la velocidad del CPU, carga del CPU, memoria libre, memoria usada, espacio del swap, etc.

- Procesos de migración preferentes. Cuando se quieren migrar procesos a nivel de usuario, Mosix busca a los nodos disponibles para ejecutar estos procesos. Los procesos de migración incluyen costos fijos, y con esto los procesos pueden ser migrados a nodos disponibles.

- Balanceo de carga dinámica. La continua intención de reducir la carga entre diferentes nodos, se realiza mediante la migración de procesos, Mosix cuenta con algoritmos que se encargan de balancear estos procesos en cada uno de los nodos del cluster. Para que esto se pueda llevar a cabo es necesario que estos procesos tengan características de sobrecarga, aunque esto también depende del número de nodos.

- Compartición de memoria. Otra característica de Mosix, es la prevención en el agotamiento de la memoria del cluster, ya que generalmente la memoria RAM de cada uno de los nodos, se hacen uno solo y actúa como una sola memoria RAM de red (UMA, Acceso de Memoria Uniforme). Mosix cuenta con un algoritmo que hace funcionar a la memoria cuando siente que en uno de los nodos hay espacio de memoria insuficiente.

En la actualidad si trabajamos con el cluster con optimización de balanceo de carga basado en el kernel mosix, esperamos que este realice trabajos de alto desempeño y de alto nivel de complejidad, el número de aplicaciones típicas de esta solución abarca las simulaciones dinámicas moleculares [29], la generación de nuevos medicamentos [30], la

compresión de archivos mp3 [31], la renderización de películas con gráficos de alta resolución [32], el electromagnetismo computacional [33], la nanotecnología, fotónica, etc.

3.8. Conclusiones

El avance de las arquitecturas paralelas son mostradas tanto para memoria compartida así como memoria distribuida. Se muestra de manera general como se puede optimizar el tiempo de ejecución determinada por la Ley de Amdhal. Asimismo se muestran diferentes tecnologías en las cuales a través de diferentes configuraciones se pueden aplicar diferentes tipos de paralelización. Tal es el caso tanto de Cluster Beowulf como Cluster Mosix. La diferencia radica principalmente si se desea paralelizar el código como es el caso del Cluster Beowulf o si se desea que todo se haga a nivel de hardware. Es por ello que ambos son muy buenas opciones para la implementación del Método FDTD los cuales serán utilizados para la implementación tanto en dos y tres dimensiones.

Capítulo 4

Implementación Optimizada del Algoritmo de Yee en Cómputo Paralelo

---

Con el fin de exponer el flujo completo de la metodología utilizada, utilizaremos los lenguajes C, Fortran y Matlab como caso de prueba para la implementación del los algoritmos. El algoritmo de Yee muestra una complejidad computacional de n^3 para el caso de dos dimensiones y n^4 para el caso del de tres dimensiones [34]. Para su análisis fué necesario realizar algunas consideraciones tal como es el caso de los recursos computacionales, específicamente lo que respecta al tipo de arquitectura y memoria. El análisis para la construcción del algoritmo serial se realizó a dos niveles. Se implementó tanto para dos dimensiones así como para tres dimensiones. Los resultados de tiempo de ejecución al paralelizar la ejecución de cada uno de los algoritmos nos darán una medida de la efectividad de este proyecto.

4.1. Análisis y construcción del algoritmo serial MDFDT

Para la implementación del MDFDT en dos dimensiones se consideró tanto el modo transversal magnético como el modo transversal eléctrico. El algoritmo se muestra de manera general de tal manera que cual quiera que sea el modo, sólo es necesario seguir el algoritmo. Para la implementación del algoritmo partimos de las ecuaciones en diferencias finitas que se obtuvieron en el capítulo II para el caso de dos dimensiones. El algoritmo del MDFDT en dos dimensiones quedó implementado en la Tabla 4.1.

Tabla 4.1: Algoritmo FDTD en dos dimensiones

Algoritmo: MDFDT en Dos-Dimensiones Modo-Serial

Realizar una vez el trabajo de inicialización: Inicializar campos, aplicar condiciones iniciales
para t=1 a tmax **hacer**
para i,j a imax,jmax **hacer**
Actualizar campo eléctrico usando campo magnético
Actualizar campo magnético utilizando campo eléctrico
Inicialización de la fuente en el punto deseado
Actualización de los campos en la frontera y aplicar condiciones de frontera

Este algoritmo como se había mencionado anteriormente, tiene una complejidad n3, por lo que se necesitará considerar los recursos de cómputo necesarios para poder programar este método con regiones de cálculo muy grandes.

El código en dos dimensiones se encuentra en el anexo B

4.1.1. Algoritmo serial del MDFDT en tres dimensiones

El algoritmo serial en tres dimensiones involucra el manejo de las seis ecuaciones de Maxwell como ya se describió en el capítulo II. El cálculo de las ecuaciones permite ver el comportamiento de los campos tanto eléctrico como magnético en cada una de las diferentes direcciones.

Tabla 4.2: Algoritmo FDTD modo serial en tres dimensiones

Algoritmo: MDFDT en Tres-Dimensiones Modo-Serial

Realizar una vez el trabajo de inicialización: Inicializar campos, aplicar condiciones iniciales
para t=1 a tmax **hacer**
para i=0 a imax **hacer**
para j=0 a jmax **hacer**
para k=0 a kmax **hacer**
Actualizar campo eléctrico($E_{xy}, E_{xz}, E_{yx}, E_{yz}, E_{zx}, E_{zy}$) usando campo magnético
Actualizar campo magnético ($H_{xy}, H_{xz}, H_{yx}, H_{yz}, H_{zx}, H_{zy}$) utilizando campo eléctrico
Inicialización de la fuente en el región de cálculo deseado
Obtención de los cálculos ($E_x, E_y, E_z, H_x, H_y y H_z$)
Actualización de los campos en la frontera y aplicar condiciones de frontera

El algoritmo MDFDT en tres dimensiones quedó implementado de la siguiente manera [35]:

El código en tres dimensiones se encuentra en el anexo C

4.2. Determinación de la arquitectura

4.2.1. Tipos de Arquitecturas

Con base al tipo de arquitecturas utlizadas para la construcción de clusters ya mencionadas en el capítulo número dos, se encuentran los clusters Beuwulf , Mosix, etc. Sin embargo, de acuerdo a las características del equipo en existencia, los cuales involucran 2 servidores de cuatro núcleo cada uno, se optó por un cluster tipo Mosix pensando en no necesitar hacer modificaciones a nivel software. Utiliza sistema operativo Ubuntu linux de 64 bits, con kernel Linux 2.6.37.1 y la última versión de Mosix 2.29.0.2.

4.2.2. Tipo de Usuario

El cluster va dirigido a un usuario general, es decir, puede o no trabajar con cómputo paralelo por lo que debe ser una arquitectura paralela que ofrezca ventajas de ejecutar programas sin la obligación de que dichos códigos tengan que alterarse para correr en paralelo y que al mismo tiempo se tengan ventajas en tiempo que es lo que se busca con esta implementación. Va dirigido a usuarios con o sin experiencia en área del paralelismo. Este tipo de usuario es gente que trabaja con algoritmos computacionales con complejidad alta y por lo tanto el uso de grandes recursos de cómputo son necesarios, pero que desconoce de la programación paralela.

4.2.3. Finalidad del cluster

La finalidad del cluster es poder tener una herramienta que sea capaz de ejecutar, simular y monitorizar métodos numéricos que resuelvan algún tipo de problema de carácter electromagnético, el cual requiere de numerosos recursos computacionales que en máquinas con pocos recursos podría ser imposible su ejecución.

4.3. Determinación del lenguaje de programación

Para la elección del lenguaje de programación fue necesario observar cuales lenguajes manejaban las mejores técnicas de paralelización usadas hoy en día, considerando sólo aquellas que pudieran involucrar herramientas para poder desarrollar cómputo paralelo. Siendo las 4 mejores propuestas las siguientes:

- MPI
- GPUMAT
- CUDA
- OPENMP

Para analizar que plataforma es más conveniente, fué necesario considerar la arquitectura en la cual deseamos trabajar. En nuestro caso, se cuentan con dos servidores quad-core de Intel, con procesador de 64 bits, memoria de 8Gb y 500 Gb de espacio en disco. Es por ello que MPI no puede ser candidato ya que MPI utiliza el esquema de memoria distribuida y al tener sólo dos equipos no se tendrían las mejores ventajas. Este tipo de esquema tiene grandes beneficios cuando se cuenta con una mayor cantidad de nodos. Una de sus desventajas más notables es la programación ya que involucra cierto conocimiento de las librerías y manejo de paso de mensajes. Sin embargo a medida que el cluster implementado bajo este esquema crece, la ganancia es bastante considerable. Por otro lado, GPUMAT es una de las mejores propuestas hoy en día sin lugar a duda en el mercado junto con CUDA, ya que los GPU's son considerados una de las mejores herramientas para desarrollar paralelismo debido a que puede integrar dentro de este procesador de video muchos núcleos lo que lo hace mucho más rápido. Para ello es necesario tener un módulo externo de GPU's o en su defecto una tarjeta de video para poder realizar este tipo de paralelización. Al igual que MPI, en CUDA el nivel de programación es bastante alto y quizás para el tipo de usuario para el cual va dirigido el cluster no es muy viable, por lo que lo vuelve no oportuno para este caso en específico, ya que el nivel de programación aumenta considerablemente. Finalmente tenemos el caso de OPENMP siendo esta la mejor opción por las características del equipo con el que se cuenta, la facilidad en la programación y por el manejo de diferentes lenguajes de programación así como de sistemas operativos. Openmp maneja el uso de memoria compartida y en el paralelismo hace uso de multihilos, de tal manera que reparte la carga de trabajo y el tiempo de ejecución se ve disminuído.

Es por ello, que la solución final, de acuerdo a la arquitectura con la que se cuenta (procesadores de cuatro núcleos), al número de equipos (sólo dos servidores) y al nivel intermedio de programación (OPENMP a través del uso de directivas de paralelización), OPENMP y cluster Mosix es la mejor alternativa y la que usaremos en gran parte de este trabajo.

4.4. Método de paralelización

Después de conocer el lenguaje de programación con el que se va a trabajar, es necesario considerar el escenario en cuanto a la técnica de paralelización que se usará. Dicho escenario parte de un algoritmo que es muy demandante en cuanto a recursos de cómputo se refiere. Asimismo considerar el tipo de arquitectura con el que vamos a trabajar así como el lenguaje de programacíon. Teniendo en cuenta lo anterior, para cada algoritmo fué necesario adoptar una técnica diferente principalmente por los problemas que surgieron ante ciertas condiciones de trabajo. Dentro de la metodología usada se presentaron dos problemas principalmente. En primer lugar se necesitaba acelerar el tiempo de procesamiento pero la complejidad del algoritmo impedía que fuera algo óptimo. Aunando el consumo de memoria que el algorimo requería, la paralelizacón se realizó utilizando dos técnicas para poder obtener los resultados deseados. A continuación se describen dichas técnicas utlizadas para cada algoritmo.

4.4.1. Paralelización del MDFDT en dos dimensiones

Paralelización del MDFDT en dos dimensiones consideró adoptar la siguiente forma de paralelización. Únicamente se paralelizó el algoritmo principal del método y no todo el programa. La razón fundamental es el hecho de que antes del algoritmo principal el cual es un ciclo, solo se realiza una vez por lo que el tiempo de sincronización se elevaría contra las ganancias en tiempo que se pudieran tener lo cual sería no funcional. Es por ello que se paralelizó únicamente el ciclo principal del algoritmo FDTD. La implementación del algoritmo se muestra en la siguiente tabla [36]:

En seguida es necesario establecer algunas variables de entorno las cuales van definir ciertas condiciones de trabajo como por ejemplo en cuantos procesadores se desea que corra el proceso. Dicha variable de entorno controla el número de procesadores en los que se desea correr las aplicaciones. Cabe señalar que cada uno de los núcleos del procesador comparten una memoria cache de nivel 2 por lo que el acceso a memoria en datos que no están dentro de la memoria caché, retardan un poco la parte de la paralelización junto con la campartición de la memoria cache. Existen dos maneras de establecer las variables de entorno. Para nuestro

Tabla 4.3: Algoritmo FDTD paralelo en dos dimensiones

Algoritmo: MDFDT en Dos-dimensiones Modo-Paralelo

Realizar una vez el trabajo de inicialización: Inicializar campos, aplicar condiciones iniciales
Hacer para cada hilo disponible
para t=1 a tmax **hacer**
para i=0 a i=imax **hacer**
para j=0 a j=jmax **hacer**
Actualizar campo eléctrico usando campo magnético
Actualizar campo magnético utilizando campo eléctrico
Inicialización de la fuente en el punto deseado
Actualización de los campos en la frontera y aplicar condiciones de frontera

caso en específico se establecieron desde el código pero también se puede implementar desde la terminal de ejecución. Para establecer dichas condiciones se utilizan los siguientes centinelas:

OMP_get_thread_num

!$call OMP_SET_NUM_THREADS (8)

El compilador utilizado fue Fortran Build Environment Intel para aplicaciones que corren a 64/32 bits. Dicho compilador permite agregar ciertas características a la compilación las cuales optimizan la manera en como se compila. La manera de compilación para este algoritmo quedó de la siguiente manera:

mosixub0: $ifort F100000000 FDTD2D.f90 -o FDTD

Se genera un ejecutable el cual desde la misma terminal tecleamos:

mosixub0: $./FDTD

4.4.2. Paralelización del MDFDT en tres dimensiones

A diferencia del MDFDT en dos dimensiones, el de tres dimensiones utilizó una técnica de paralelización diferente. En lugar de tomar sólo el ciclo principal, se toma cada uno de los campos a los cuales se les asocia una directiva de paralelización. [36]

Tabla 4.4: Algoritmo FDTD paralelizado en tres dimensiones

Algoritmo: MDFDT en Tres-dimensiones Modo-Paralelo

Realizar una vez el trabajo de inicialización: Inicializar campos, aplicar condiciones iniciales
para t=1 a tmax **hacer**
Para cada hilo disponible
para i=0 a imax **hacer**
para j=0 a jmax **hacer**
para k=0 a kmax **hacer**
Actualizar campo eléctrico($E_{xy}, E_{xz}, E_{yx}, E_{yz}, E_{zx}, E_{zy}$) usando campo magnético
Actualizar campo magnético ($Hxy, Hxz, Hyx, Hyz, Hzx, Hzy$) utilizando campo eléctrico
Termina asignación a cada hilo disponible
Se realiza para un sólo hilo
Inicialización de la fuente en el región de cálculo deseado
Termina asignación para un sólo hilo
Para cada uno de los hilos disponibles
Obtención de los cálculos mediante una reducción de parámetros ($E_x, E_y, E_z, H_x, H_y y H_z$)
Barrera de sincronización

De acuerdo a la Tabla 4.4, se realiza en primer lugar, el cálculo del campo, en la cual se debe indicar el tipo de variables. Para este caso las variables que consideramos privadas son los índices del ciclo porque necesitamos tener control sobre cada iteración, ya que buscamos que cada uno de los hilos participen en el cálculo del campo.

Establecemos una fuente puntual senoidal para cada uno de los campos en los puntos de interés $H(i, j, k)$. Dicha centinela tiene la función específica de que sólo un hilo va a ejecutar esta parte. Esto debido a que no es un ciclo, es decir, sólo es un acceso a un punto de la

matriz. En seguida sumamos para obtener el campo total. Esta directiva suma cada uno de los elementos con cada uno de los hilos pero con la diferencia de que en este caso utiliza una optimización 02 la cual implica suma de elementos del mismo tamaño y proporciona un mejor tiempo de ejecución.

Las mismas centinelas son aplicadas a los campos H_y, H_z, E_x, E_y, E_z con su respectiva fuente puntual. Finalmente necesitamos utilizar una directiva más que nos permita controlar la iteración. Dicha centinela tiene la función de un semáforo de tal manera que no pasará a la siguiente iteración del ciclo principal hasta que todos los procesadores hayan terminado de realizar su tarea.

4.5. Cluster

El tipo de cluster que se eligió por las características ya mencionadas, fue un cluster tipo Mosix con equipo existente en la ESIME en la sección de posgrado de la maestría en Telecomunicaciones. Dicho equipo consiste de las siguientes características:

- 2 Servidores, cada uno cuenta con un procesador Intel Xeon quad core a 2.4 Ghz
- 16 GB de memoria
- 1 TB de almacenamiento
- switch de alta velocidad gigabit ethernet linksys
- tarjetas de red de alta velocidad
- Medio de comunicación cable UPT-CAT6
- Sistema operativo Linux, en su distribución Ubuntu 10.04LTS
- Utiliza sistema operativo Ubuntu linux de 64 bits, con kernel 2.6.25.20 y mosix 2.28.2.
- Uso de compiladores gcc, Intel para C/C++/FORTRAN con librerías que dan soporte tanto para OPENMP y MPI

La descripción de la instalación del sistema operativo, actualización del kernel,instalación del cluster mosix, configuración de red, instalación de compiladores, instalación y configuración de monitores del sistema se encuentran en el anexo A.

4.6. Conclusiones

En este capítulo se ha mostrado como arreglos involucran a las ecuaciones de Maxwell las cuales simulan el Método FDTD en propagación electromagnética en espacio libre. Dichas ecuaciones requieren altos recursos computacionales. Se ha implementado tanto modo serial como modo paralelo tanto para dos y tres dimensiones considerando la administración de la memoria tanto para hilos como para códigos. La implementación requirío el uso de memoria compartida así como memoria distribuida para el caso de tres dimensiones. La optimización del Método FDTD a través de técnicas de paralelización son eficientes en el paralelismo pero sufren de un tiempo muerto de sincronización de comunicación. Este puede ser significativamente mejorado aplicando un segundo nivel de paralelismo mejorando el tiempo de ejecución.

Capítulo 5

Resultados

5.1. Resultados y Pruebas de desempeño en procesadores multinúcleo

Este capítulo describe los resultados obtenidos con el Método FDTD. Para analizar la ganancia en el rendimiento total de simulación del algoritmo del MDFDT en la propagación electromagnética se realizó en dos niveles. El primero de ellos modela la propagación electromagnética en espacio libre tanto en dos y tres dimensiones y en el segundo modela la propagación electromagnética un recinto cerrado en el cual se involucran los diferentes materiales como son libreros, escritorios, ventanas, etc. Se analizan los efectos ocasionados considerando una fuente puntual a 2.4 GHz. El porque de esta frecuencia tiene que ver directamente con un punto de acceso (acces point), localizado en las instalaciones que se desean simular. Dentro de la simulación se considera una fuente puntual que simula dicho punto de acceso localizado en el área de trabajo y que transmite a la misma frecuencia de 2.4 GHz. Finalmente este trabajo es una optimización de un trabajo realizado en el seno de la Maestría [1], en el cual se realizan simulaciones reales con un analizador de espectros, cuyas medidas son comparadas, verificando un correcto funcionamiento en los resultados obtenidos en este trabajo y muestra de ello se encuentran algunas publicaciones en revistas internacionales. En este proyecto se analizan los resultados de la metodología utilizada en el proyecto actual (speed up, o aceleración del tiempo de simulación), se hizo una estimación del tiempo en paralelo de cada proceso y se comparó contra el tiempo total que en un principio le tomaba al algoritmo ejecutarse de manera serial.

5.1.1. Características de la región de cálculo

Las características de la simulación de propagación electromagnética en espacio libre o sin obstáculos que se utilizaron para la ejecución para las diferentes arquitecturas en prueba se muestran en la Tabla 5.1.

Tabla 5.1: Medidas y características de la región de cálculo para el MDFDT en dos dimensiones

Tamaño	$Imax = 7000, Jmax = 7000$
Frecuencia de operación	2.4 GHz
Capa PML	12
Grado del polinomio	6
Número de iteraciones	1000

Dichas características varían de acuerdo a como responde la arquitectura.

5.1.2. Características de la máquina de dos núcleos

En primera instancia se consideró la simulación en una máquina con un procesador de dos núcleos para comprobar que la técnica de programación en paralelo estuviera funcionando correctamente. En la siguiente tabla se muestra las características del equipo utilizado:

Tabla 5.2: Características del equipo utilizado

Procesador	Dos núcleos
Frecuencia de operación	2.13 GHz
Memoria	2 GB
Disco Duro	160 GB

Resultados obtenidos

Los resultados obtenidos en el MDFDT en dos dimensiones son mostrados en las siguientes dos figuras. En primer lugar se tiene la comparación del incremento de velocidad o mejor

conocida como aceleración. Figura 5.1 realiza una comparación al ejecutar el programa tanto de forma paralela como de manera serial.

Se observan un incremento en la aceleración por encima de la ejecución en el monoprocesador. La aceleración se incrementa en un factor de 1.6x aproximadamente debido a que tiene ciertas variaciones conforme va avanzando las simulación.

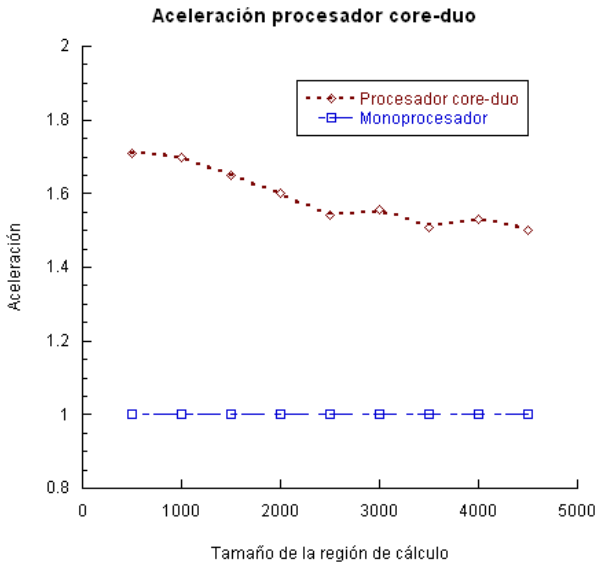


Figura 5.1: Comparación de la aceleración del MDFDT en dos dimensiones

Por otro lado se evalúa la eficiencia computacional que gasta el algoritmo en la arquitectura en cuestión y de acuerdo a la gráfica anterior hace suponer que tiene un mejor comportamiento el caso paralelizado. En la figura 5.2 se observa la eficiencia computacional en un procesador de dos núcleos cuando se ejecuta el MDFDT en dos dimensiones observándose claramente que se utilizan mejor los recursos de cómputo y además el tiempo que se están utilizando los recursos de cómputo se ve decrementado a favor de la técnica de paralelización.

5.1.3. Características de la máquina de cuatro núcleos

Cambiando de arquitectura por una máquina quad core de 64 bits, se realizó la misma simulación con lo cual se pretendía mejorar los resultados.

Las características del servidor con procesador de cuatro núcleos son las siguientes:

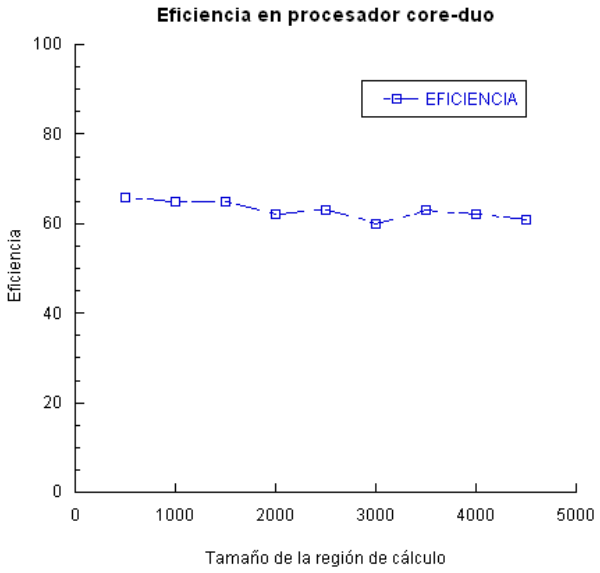


Figura 5.2: Comparación de la eficiencia computacional del MDFDT en dos dimensiones

Tabla 5.3: Características del equipo utilizado

Procesador	Intel Xeon Quad Core E5405
Frecuencia de operación	2.13 GHz
Memoria	8 GB
Disco Duro	500 GB

Resultados en procesador de cuatro núcleos

A continuación se muestran los resultados de la simulación del MDFDT en dos dimensiones. En primera instancia se compara el tiempo de ejecución de manera serial contra el tiempo de ejecución en paralelo.

En figura 5.3 se observa un mejor tiempo de ejecución en modo paralelo a diferencia del modo serial.

En la figura 5.4 se muestra la eficiencia entre ambas modalidades de igual manera y se puede observar que de modo paralelo se ocupan los recursos computacionales un menor tiempo además de aprovechar todos los recursos y no sólo una parte como es el caso del modo serial.

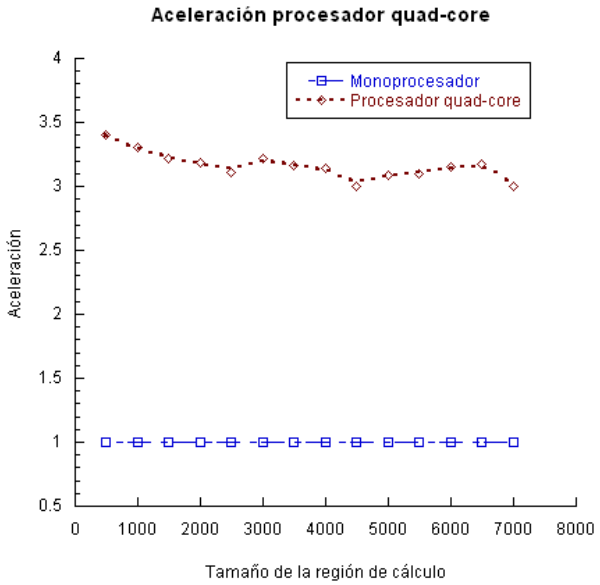


Figura 5.3: Comparación de la speedup del MDFDT en dos dimensiones

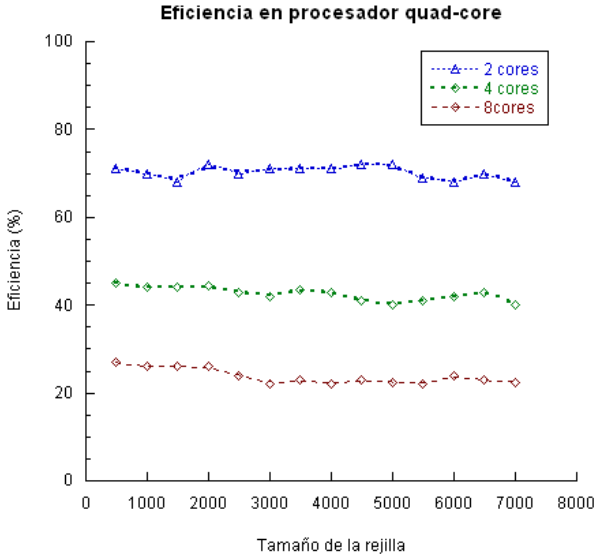


Figura 5.4: Comparación de la eficiencia del MDFDT en dos dimensiones

5.1.4. MDFDT en dos dimensiones en el Cluster-Mosix

Se ejecutó la aplicación del MDFDT en dos dimensiones en el Cluster-Mosix en el cual se considera las mismas características de la región de cálculo que se ejecutó en la máquina de cuatro núcleos. Dichos resultados se muestran a continuación.

En la figura 5.5 se compara la aceleración en el Cluster-Mosix. Podemos observar

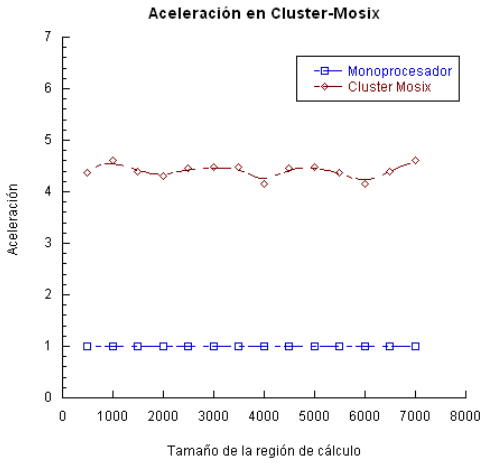


Figura 5.5: Comparación de aceleración en el Cluster Mosix

claramente como el proceso que se corre en el Cluster tiene un mejor desempeño en comparación con el proceso que se corre de manera serial.

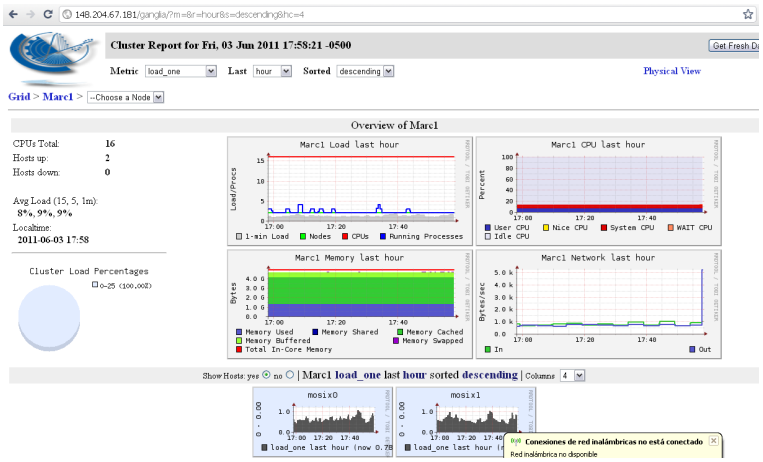


Figura 5.6: Monitor de procesos Ganglia

En la figura 5.6 se tiene el monitor del sistema. Este monitor de sistema permite visualizar el comportamiento del cluster cuando se está ejecutando un proceso. El monitor permite observar cada uno de los hosts e identifica el número de núcleos en cada host.

Hace ver el comportamiento de memoria, de espacio en disco, el número de elementos recibidos, transmitidos , el comportamiento de la red en general. En dicha figura se puede apreciar, la carga en cada uno de los hosts.

5.1.5. MDFDT en tres dimensiones en la máquina Alebrije

El MDFDT en tres dimensiones se consideró para propagación electromagnética en un recinto cerrado es mostrado en la figura 5.7.

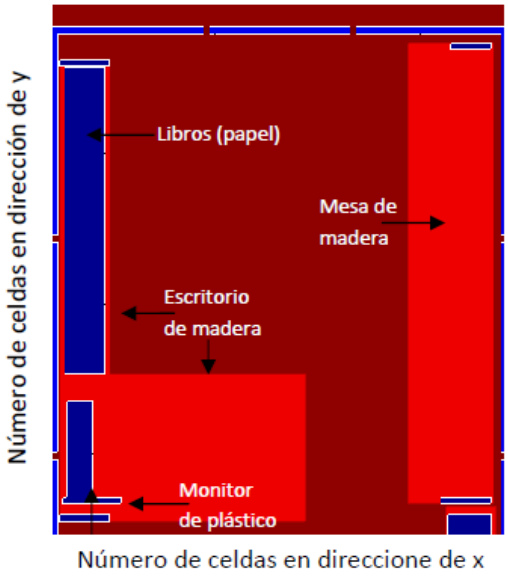


Figura 5.7: Modelo de propagación electromagnética en un salon

Dicho recinto cerrado esta dentro de la Maestría en telecomunicaciones en la SEPI de ESIME y se realizaron algunas pruebas ya que el costo computacional en cuanto a memoria se refiere sobrepasaba los límites con el que se contaban. Es por ello que se dividió en una prueba pequeña que para este caso es un salón para ver el funcionamiento del algoritmo. Figura 5.7 muestra la región simulada en la cual se consideran los diferentes materiales de cada uno de las cosas que se encontraban en dicha oficina.

Para ello la región de cálculo quedó asignada de la siguiente manera:

Por lo tanto, considerando una región de cálculo de $Imax = 468, Jmax = 597, Kmax = 449$ es necesario considerar la siguiente memoria:

$$Memoria = 468 * 597 * 449 = 125448804 * 4 * 1000/1024 * 30 = 14701031718.75B \quad (5.1)$$

Tabla 5.4: Medidas y características de la región de cálculo en un salón

Tamaño	$Imax = 468, Jmax = 597, Kmax = 449$
Frecuencia de operación	2.4 GHz
Capa PML	12
Grado del polinomio	6
Número de iteraciones	6000

Para poder utilizar la memoria necesaria se hizo uso de un equipo ubicado en la Unidad de Supercómputo de la UNAM con las siguientes características:

Tabla 5.5: Características del equipo utilizado (Alebrije UNAM)

Procesador	24 Procesadores Intel Itanium 2 SGI Altix 350
Frecuencia de operación	2.13 GHz
Memoria	64 GB
Disco Duro	2.4 TB

Supercomputadora paralela de memoria compartida. Contiene 24 procesadores Intel Itanium 2, 24 Gigabytes de memoria y 2.4 Terabytes de almacenamiento. Finalmente los resultados obtenidos de la simulación son los siguientes:

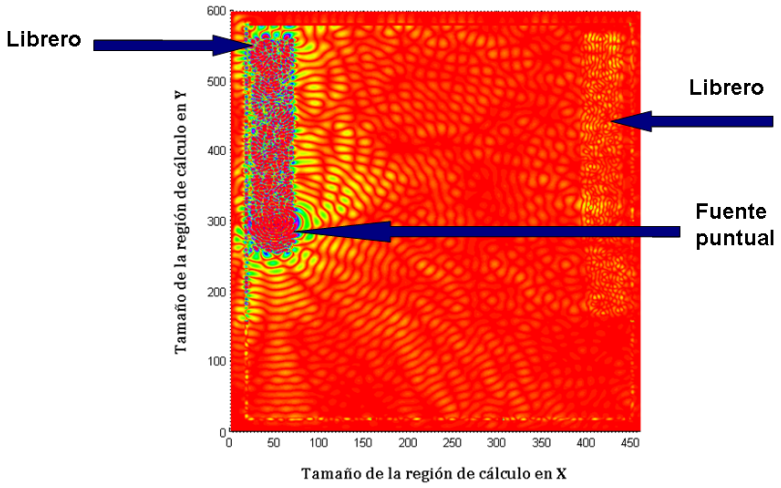


Figura 5.8: Propagación electromagnética en un salón

En la figura 5.8 se puede observar como la fuente que esta radiando choca con el obstáculo en este caso sobre el librero de madera, apreciandose algunas reflexiones y difracciones así como la aparición de multitrayectorias de propagación. Sin embargo cuando la onda llega al escritorio de la esquina ya no llega con la misma intensidad por lo que los efectos ya no son tan visibles como el caso del librero del inicio.

5.1.6. Modelo de Propagación en Tres salones

En la parte central de la figura 5.9 se puede apreciar la segunda oficina, la onda inmediatamente incide sobre el librero de madera, observándose claramente reflexiones y difracciones visibles dentro del librero. El comportamiento en el salón de en medio es el que muestra la mayor concentración de potencia a diferencia de los salones que están a lado. El MDFDT en tres salones se muestra en la siguiente figura 5.9.

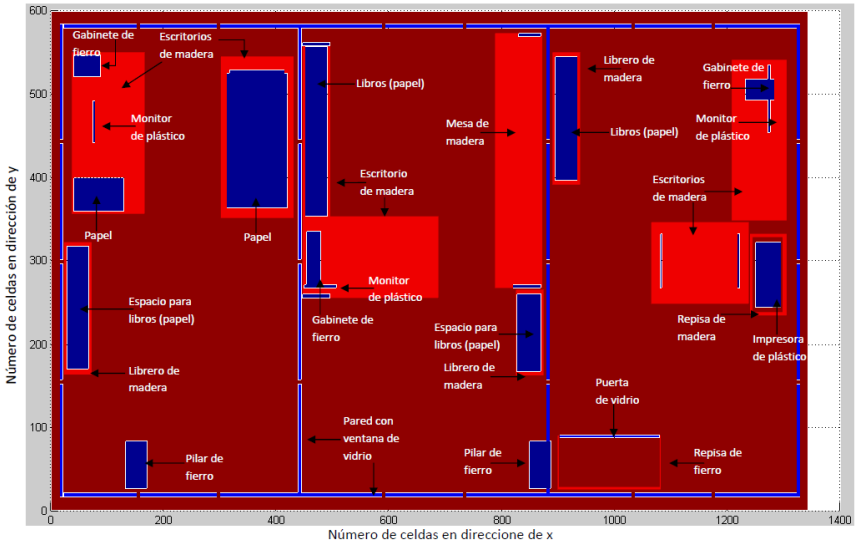


Figura 5.9: Región simulada en la propagación electromagnética en tres salones

Las características de la región de cálculo para la simulación en tres salones, se realizó de la misma manera en la UNAM, en la computadora Alebrije, y las características de la simulación se muestran en la siguiente tabla:

Contemplando dichas características se realizó el cálculo de memoria necesario para la simulación. Para ello consideramos el tamaño de la matriz, el tipo de dato y el número de

Tabla 5.6: Medidas y características de la región de cálculo en tres salones

Tamaño	$Imax = 1341, Jmax = 597, Kmax = 450$
Frecuencia de operación	2.4 GHz
Capa PML	12
Grado del polinomio	6
Número de iteraciones	6000

matrices que se utilizan en el programa. Dicho cálculo se realiza a continuación:

$$Memoria = 1341 * 597 * 450 = 360259650 * 4 * 1000/1024 * 36 = 50039302343.75B \quad (5.2)$$

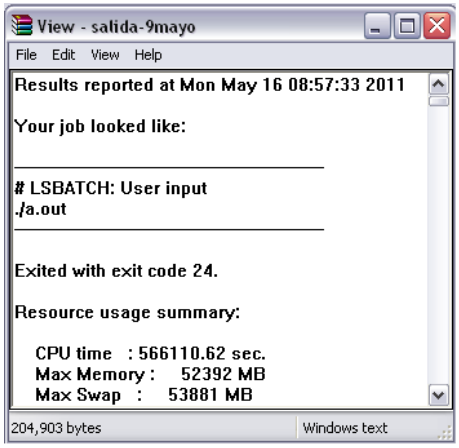


Figura 5.10: Archivo generado por la cola de procesos de la máquina Alebrije

La figura 5.10, muestra el archivo generado por la cola de procesos de la máquina Alebrije. Comparando con la memoria calculada observamos que la memoria utilizada por el proceso es ligeramente mayor a la calculada pero esta dentro de un margen considerable ya que dentro del proceso se ven involucrada 3 tipos de memoria que son memoria de código, memoria de hilos y memoria de datos. Es por ello que la memoria aumenta ligeramente.

Finalmente la propagación en dichos recintos se presenta a continuación con la finalidad de comprobar que la optimización sea la correcta y que dicho algoritmo este funcionando de manera adecuada.

La figura 5.11 muestra la simulación de las tres oficinas en los cuales se está propagando la fuente de onda puntual. El modelado se llevó a cabo considerando escenarios reales,

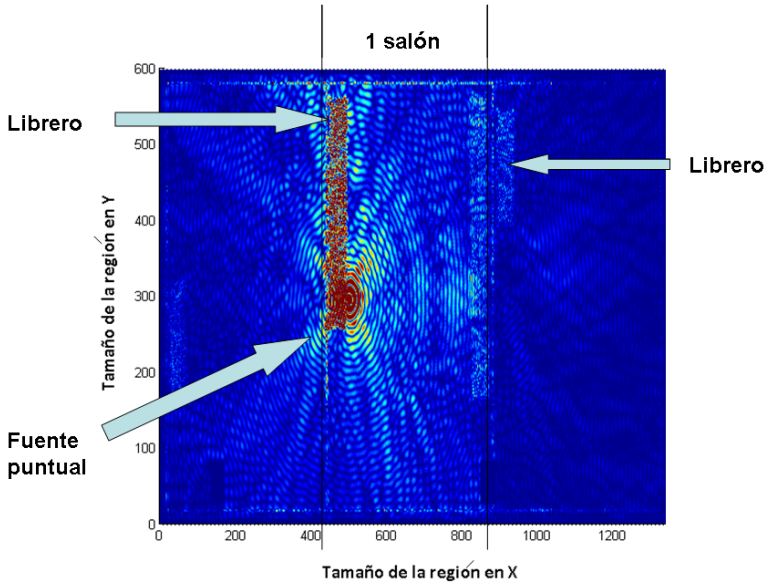


Figura 5.11: Propagación electromagnética en tres salones

con disposición y tipo inmobiliario existente en tres cubículos en las instalaciones del programa de Maestría. En el primer rectángulo de la figura se puede observar como la onda electromagnética está inside sobre un librero de madera y sobre la pared de vidrio, que es la línea que se observa en el extremo izquierdo de ambas figuras. En el espacio libre la onda sufre ciertas deformaciones producto de los efectos de reflexión, difracción y fenómenos de multitrayectoria presentes en toda la región de cálculo.

En la parte central de la figura 5.11 se puede apreciar la segunda oficina, la onda inmediatamente incide sobre el librero de madera, observándose claramente reflexiones y difracciones visibles dentro del librero. El comportamiento en el salón de en medio es el que muestra la mayor concentración de potencia a diferencia de las áreas de a lado las cuales representas los salones. En la tercer sección de la figura se encuentra un librero y arriba de éste la pared de vidrio que delimita la oficina donde se observa el cambio de velocidad de la onda y como se propaga a través del vidrio.

5.2. Comparación de resultados analíticos, simulados y mediciones

Para comprobar el funcionamiento del cluster y de los resultados obtenidos a través de la técnica de paralelización utilizada, se realizan ahora comparaciones de los resultados obtenidos mediante parámetros de aceleración y eficiencia computacional entre máquina y cluster así como cluster mosix contra el cluster de la UNAM corriendo un mismo proceso.

Comparación de la aceleración en diferentes arquitecturas

Tabla 5.7: Medidas y características de la región de cálculo para el MDFDT en dos dimensiones

Tamaño	$Imax = 7000, Jmax = 7000$
Frecuencia de operación	2.4 GHz
Capa PML	12
Grado del polinomio	6
Número de iteraciones	4000

Considerando un caso en particular en el cual se compara el MDFDT en dos dimensiones incluyendo las diferentes arquitecturas a lo largo de este trabajo. Las características de simulación quedan expresados en la tabla (5.7):

Dichas características varían de acuerdo a como responde la arquitectura. En figura 5.12, se muestra la aceleración en Máquina Alebrije, Cluster Mosix, Procesador de cuatro núcleos y monoprocesador.

Los tiempos de interés que son el caso del Cluster Mosix y la mína Alebrije son mostradas en la 5.8. Se aprecia un menor tiempo de ejecución en la máquina Alebrije.

Tabla 5.8: Tiempo de ejecución del MDFDT en dos dimensiones

Máquina	Tiempo de ejecución
Cluster Mosix	$240Hrs$
Alebrije	$197.5Hrs$

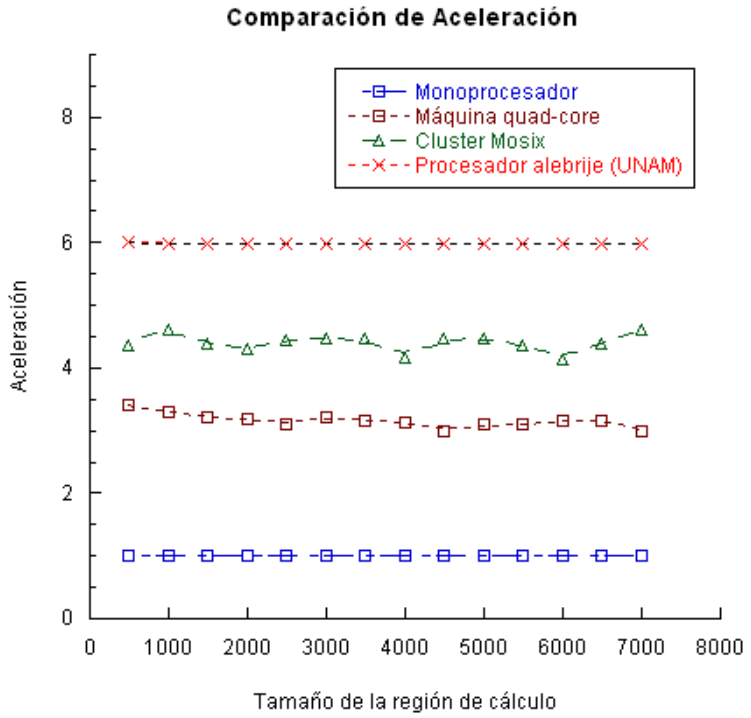


Figura 5.12: Comparación modo paralelo y modo serial

5.3. Conclusiones

Como se muestra a lo largo del capítulo, el tiempo total de simulación depende de los recursos computacionales. La más grande simulación físicamente hablando puede resolverse con dichas técnicas de paralelización dependiendo de la complejidad del algoritmo. Especiales computadoras pueden ser construidas para simular el FDTD en 3D teniendo la gran ventaja que dichas computadoras pueden utilizarse para otros tipos de Métodos. Es por ello que la paralelización del MDFDT redujo extremadamente el tiempo de ejecución. Esas largas simulaciones pueden ser resueltas empleando dichos cluster de computadoras como se discutió en el capítulo 3. Para el caso de este capítulo se ha modelado toda una región de Estudios de posgrado en el cual se analizan fenómenos de propagación multitrayectoria. Se modela a una frecuencia de 2.4 GHz debido al uso comercial de la frecuencia de un punto de acceso (Acces point) permitiendo tener un caso real el cual se compara en otros trabajos, en el cual fué comprobado con mediciones puntuales los cuales fueron resultado de otro trabajo de Tesis [1]. Para la ejecución del algoritmo se requirió el uso de diferentes computadoras. La máquina Alebrije muestra un mejor desempeño ya que ejecuta 6 veces más rápido en

comparación con el monoprocesador. Por otro lado tenemos el caso del Cluster Mosix en el que su aceleración muestra un desempeño arriba de 4 veces más rápida y el caso del quad-core abajo de 3.5 veces más rápido en promedio en comparación con el modo serial. Sin embargo el hecho de tener dos procesadores de cuatro núcleo en el Cluster-Mosix incrementa el desempeño lo que sugiere aumentar el número de nodos para tener mejores resultados.

Capítulo 6

Conclusiones y trabajos a futuro

El aporte de este proyecto de Tesis es un indicador de que la simulación paralela es ciertamente una estrategia efectiva para obtener tiempos más cortos de verificación funcional en algoritmos con complejidad computacional alta, eliminar las limitaciones de memoria impuestas por una simulación en un sistema monoprocesador y minimizar los costos inherentes a una ejecución de procesos de forma distribuida. Se llega a la conclusión de que la aceleración de la simulación es posible si se sabe explotar la información sobre arquitecturas, jerarquía, carga de comunicaciones y paralelismo propio de las tareas a realizar . Las principales contribuciones de esta Tesis son:

- Se comprobó que uno de los más recientes estándares como es el caso de OPENMP puede ser usado para modelar métodos matemáticos disponible para diferentes lenguajes de programación, con la particularidad de poder obtener ejecutables del diseño.
- Se diseñó e implementó un cluster Mosix con la particularidad de incluir compilador tanto para C, Fortran, java, Linux script, Matlab, Office (Open Office), Monitor de Sistema Ganglia y Monitor de Red gkrellm. Directivas OPENMP así como biblioteca MPI para el paso de mensajes de una máquina a otra.

- Se implementó un primer Algoritmo del MDFDT paralelizado en dos dimensiones, donde la paralelización es por ciclo y cada en cada ciclo se calculan los campos para una iteración, obteniendo un mejor tiempo de ejecución de forma paralelizada. Hace uso de memoria compartida
- Se implementó un segundo Algoritmo del MDFDT en tres dimensiones paralelizando cada uno de los campos en donde la técnica de paralelización utilizada porporciona un mejor tiempo de ejecución en comparación con un monoprocesador. Utiliza memoria compartida.
- Se implementó un tercer algoritmo del MDFDT en tres dimensiones el cual simula la propagación electromagnética en un recinto cerrado en el cual la técnica de paralelización se realiza en lenguaje C y maneja el uso de Paso de mensajes mediante MPI. Usa memoria distribuida.
- Se diseño una pagina web para el cluster el cual permite visualizar monitores asociados, misma que arrojó la métrica sobre número de transiciones por cada señal monitoreada y tener así una idea del costo en la carga de comunicaciones ante diferente cutsize.

Los condiciones de aceleración de la simulación aprendidas durante el estudio del estado del arte sobre simulación paralela y distribuida fueron confirmadas por los experimentos realizados, y se concluye que con el fin de minimizar el tráfico de mensajes entre las múltiples particiones resultantes del circuito, el tiempo muerto de sincronización debe ser minimizado, y que con el fin de obtener una ganancia máxima en tiempo de simulación, el algoritmo de particionamiento debe considerar un balance en las cargas de trabajo al generar las particiones.

Para tener una verdadera medida de la aceleración en la simulación, es preciso crear un agente maestro encargado de repartir los procesos en estaciones distribuidas de trabajo conectadas en red y estar administrando las colas de los procesos para enviar y recibir los datos que entre dichos procesos es necesario intercambiar.

Hasta ahora hemos trabajado en un esquema que aprovecha la información intrínseca del diseño para particionarlo en bloques donde cada sección representa un submódu-

lo que es parte de la funcionalidad de los procesos siguiendo un modelo de repartición de la carga de trabajo. Sin embargo, en un algoritmo posterior pretendemos explotar la simulación electromagnética de toda la unidad de posgrado en Telecomunicaciones ubicada en ESIME Zacatenco mediante el paso de mensajes mediante MPI. El objetivo de esta segunda alternativa de paralelización es formar un conjunto de bloques donde cada uno estará constituido por nodos cuyo rango de carga de trabajo estén dentro de un umbral aún a definir dependiendo de los recursos de cómputos dados. De este trabajo en proceso se deriva la necesidad por robustecer el cluster mucho más adelante agregando nodos al cluster Mosix para poder seguir trabajando con este tipo de métodos computacionales y tener las herrmientas para poder llevar a cabo su ejecución.

Por otro lado, un mecanismo de sincronización vía TCP/IP usando sockets como colas de entrada y salida de datos en los procesos particionados, impactaría considerablemente al reducir la latencia de comunicación así como de equipos de alta velocidad debido a que entre más nodos sean agregados, aumentaría considerablemente la latencia y el tiempo muerto de sincronización de ejecución.

Anexo A: Instalación y configuración del Cluster Mosix en Ubuntu.

Instalando y configurando Ubuntu

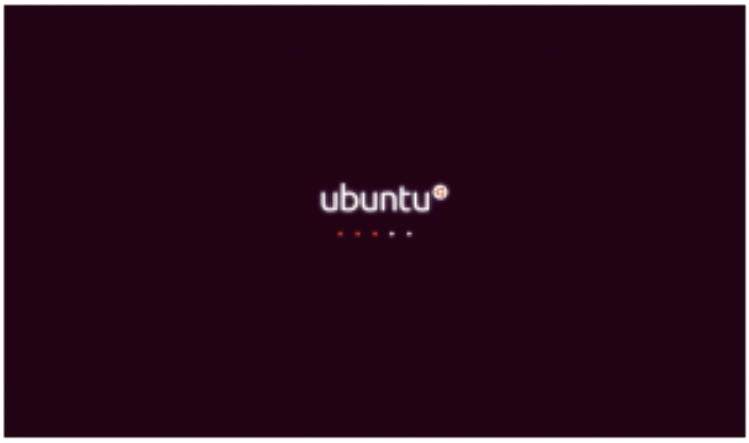


Figura 6.1: Imagen de inicio en Ubuntu 10.04 LTS

Lo primero que hay que realizar es descargar Ubuntu 10.04 LTS la cual la podemos encontrar en la siguiente liga:

http://www.ubuntu.com/getubuntu/download

Para su instalación se puede seguir la siguiente liga, en la cual se muestra paso a paso la instalación de dicha distribución de Linux:

http://sliceoflinux.com/2010/04/29/instalar-ubuntu-10-04-paso-a-paso/

Cosas que hacer despúes de instalar Ubuntu

Enseguida abrimos una terminal con:

Aplicaciones menu – Accesorios – Terminal

Teclear lo siguiente para crear la contraseña de administrador del sistema (root):

mosixub0: $sudo passwd root
mosixub0: $Introduzca la contraseña del usuario:
mosixub0: $Introduzca la contraseña Unix:
mosixub0: $Confirme la contraseña Unix:

Esta contraseña es necesaria para realizar todas las tareas administrativas del sistema. Para iniciar como superusuario o administrador es necesario teclear lo siguiente:

mosixub0: $su
mosixub0: $Introduzca la contraseña de superusuario:

Ya que estamos como super usuario es necesario actualizar el sistema. Para hacerlo tecleamos la siguiente instrucción desde la terminal:

mosixub0: $ sudo apt-get update

Para obtener una versión mas reciente del sistema operativo

mosixub0: $ sudo apt-get upgrade

Configuración de red

Configuración del dispositivo de red eth0

Para la configuración de red es necesario abrir una terminal e iniciar sesión como administrador del sistema (root). Una vez que ya se inició como root es necesario abrir el archivo intefaces con un editor de texto para cambiar los parámetros de la configuración de red. La siguiente línea abre dicho archivo:

```
mosixub0: $pico/etc/network/interfaces
```

En la siguiente tabla se muestra la edición de dicho fichero:

Tabla 6.1: Configuración de red modo permanenete

Edición del archivo pico/etc/network/interfaces
auto eth0
iface eth0 inet static
address 200.0.0.1
netmask 255.250.255.0
hostname: nodo0 200.0.0.1
hostname: nodo1 200.0.0.2

2. Finalmente reiniciamos el servicio de red mediante la siguiente ruta en una terminal:

```
mosixub0: $/etc/init.d/networking restart
```

Configuración del dispositivo de red eth1

La configuración de la red pública sólo se realiza en el nodo maestro. La manera en la que se se implementó fue de manera temporal abriendo una terminal y ejecutando las siguientes

instrucciones:

Tabla 6.2: Configuración de la red temporal desde una terminal

Edición del archivo pico/etc/network/interfaces
ifconfig eth1 148.204.67.181 up
ifconfig eth1 148.204.67.254
route add default gw 148.204.67.254
echo nameserver 148.204.103.2 /etc/resolv.conf
echo nameserver 148.204.102.3 /etc/resolv.conf

Instalación y configuración de MOSIX

Para la instalación y configuración primero es necesario definir los hostsnames en el archivo pico/etc/hosts accediendo con editor de texto, en la tabla siguiente se muestra dicha edición:

Tabla 6.3: Configuración de los hosts pertenecientes al cluster

Edición del archivo pico/etc/hosts
auto eth0
127.0.0.1 localhost
200.0.0.1 mosix 0
200.0.0.2 mosix 1

Instalación de algunos paquetes necesarios

Instalación de NFS

NFS permite crear un folder en el nodo maestro y tener registrado en todos los nodos. Este folder puede ser utilizado para almacenar programas. Para instalar NFS solo hay que corer lo siguiente en la terminal

mosixub0: $sudo apt-get install nfs-kernel-server

Instalación de SSH server

Correr el siguiente commando en cada uno de los nodos para instalar OpenSSH Server:

```
mosixub0: $sudo apt-get install openssh-server
```

Para probar SSH es necesario correr en la terminal:

```
mosixub0: $ssh ub1 hostname
```

Debe regresar el hostname sin preguntar contraseña

Instalación de builds-essential

Build essential contiene todos los compiladores como son el caso de C,C++,Fortran. Para instalar el paquete, correr el siguiente comando:

```
mosixub0: $sudo apt-get install build-essential
```

Instalación of Patch

```
mosixub0: $sudo apt-get install patch
```

Instalación of Make

```
mosixub0: $sudo apt-get install make
```

Instalación de Ncurses-Devel

```
mosixub0: $sudo apt-get install Ncurses-Devel
```

Instalación de Start-up manager

```
mosixub0: $sudo apt-get install start-up manager
```

Instalación y configuración ganglia

Para la instalación y configuración de ganglia en ubuntu, es necesario considerar tres partes:

1)Instalación de Ganglia
2)Configuración de Ganglia: en el host(gmetad)
3) Configuración de Ganglia en los nodos cliente(gmond)

Paquetes soportados

Ganglia utiliza RDDTool para generar y desplegar sus gráficos. Sólo es necesario correr en la terminal la siguiente línea:

```
mosixub0: $ apt-get install rrdtool librrds-perl librrd2-dev
```

Con la finalidad de levantar páginas en php, es necesario correr un paquete adicional de la siguiente manera:

```
mosixub0: $ apt-get install php5-gd
```

Instalación de ganglia

A excepción de frontend, Ganglia es un repositorio de Debian que es fácil de instalar. En el servidor web es necesario contar con los siguientes paquetes:

```
mosixub0: $ apt-get install ganglia-monitor gmetad
```

En todos los nodos necesitarás tener ganglia-monitor

Obteniendo Ganglia Frontend

Desafortunadamente, el frontend se instala independientemente de Ganglia ya que no está incluido en los repositorios de Debian y sólo necesitaras construirlo una sola vez. Visitar la página web y copiar la liga de la versión mas reciente. Para descargar el paquete lo realizamos de la siguiente manera:

wget http://downloads.sourceforge.net/ganglia/ganglia-3.0.7***.tar.gzmodtime=12065***&$big\_mirror = 0$

Después necesitamos descomprimir el archivo mediante la siguiente liga:

mosixub0: $ tar xvzf ganglia*.tar.gz

Accedemos al directorio mediante el comando cd. Ganglia utiliza la forma tradicional de instalación de código fuente. Para su instalación es necesario ejecutar la siguiente línea en una terminal:

mosixub0: $./configure –prefix=/opt/ganglia –enable-gexec –with-gmetad

Cuando termina tu puedes ver una pantalla como la que aparece en Fig. 6.2:

Figura 6.2: Imagen de inicio de Ganglia

Enseguida instalaremos con make y make install como se muestra en la siguiente tabla. Finalmente, cuando esto es hecho, creamos un directorio ganglia y lo copiamos todo a /web

Apache

Para un mejor desempeño, debes actualizar los servicios de Apache para redirigir cualquier petición HTML al directorio de administrador (por ejemplo, yourserver.yourdomain.com)

```
mosixub0: $make
mosixub0: $make install
mosixub0: $mkdir /var/www/ganglia
mosixub0: $cp web/ /var/www/ganglia
```

para ir directamente a ganglia. Abrir /etc/apache2/sites-enabled/000-default and encontrar el bloque de código para el directorio /var/www y agregar una redirección. El bloque queda de la siguiente manera:

```
Directory /var/www/
Options FollowSymLinks MultiViews
AllowOverride None
Order allow,deny
 RedirectMatch ^/$ /ganglia/
/Directory
```

Después de esto, es necesario reiniciar los servicios de apache

```
mosixub0: $/etc/init.d/apache2 restart
```

Ganglia: Host Configuration

Hasta este punto, tu deberías ya tener instalado el demonio de ganglia meta daemonen el nodo maestro.

/etc/gmetad.conf

La base de control para gmetad en Debian es /etc/gmetad.conf. Existen algunos valores a configurar así que es necesario abrirlo con un editor de textos.

Cambios requeridos

Acontinuación se muestran los más importantes:

authority -Establecer tu host (por ejemplo, 148.204.67.181/ganglia).
trusted_hosts - Si tu servidor web tiene múltiples nombres de dominio, ellos deben ser enlistados aqui

Cambios Opcionales

gridname - Usado para cambiar el nombre del "Grid"por default. Es necesario enlistar aquí

Reiniciando Ganglia

Antes de cualquier cambio, gmetad necesitará ser reiniciado. Para hacer esto es necesario correr la siguiente línea en una terminal:

```
mosixub0: $/etc/init.d/gmetad restart
```

Configurando Gmon

El host que corre gmetad esta probablemente corriendo también gmon. Si se quiere monitorizar este host será necesario configurarlo como un nodo cliente también.

Configuración del nodo

El archivo responsable para conectar cada nodo con el servidor Ganglia es precisamente /etc/gmond.conf.

Este archivo necesita ser editado propiamente para cada nodo. Esto puede ser hecho individualmente, o un archivo puede ser creado en un nodo y puede ser copiado en cada uno de los nodos. Los siguientes valores necesitan ser editados:

Edición del archivo pico/etc/gmond.conf

***name* - Este es el nombre del nodo asociado con el cluster**

***owner* - Diferentes dueños serán usados para separar diferentes clusters dentro de dominios administrativos.**

***mcast_if* - Si el nodo tiene múltiples interfaces, el primero debe ser usado para conectar al host el cual debe ser especificado en este punto.**

***num_nodes* - El número de nodos en el Cluster.**

Reiniciando monitoreo Ganglia

Después de hacer los cambios, gmond necesita ser reiniciado en el nodo. Para realizar esto, es necesario ejecutar en una terminal lo siguiente:

```
mosixub0: $/etc/init.d/ganglia-monitor restart
```

Tu necesitas esperar alrededor de 10 minutos para que los cambios hagan efecto. Figuras 6.3 muestra el funcionamiento del monitor.

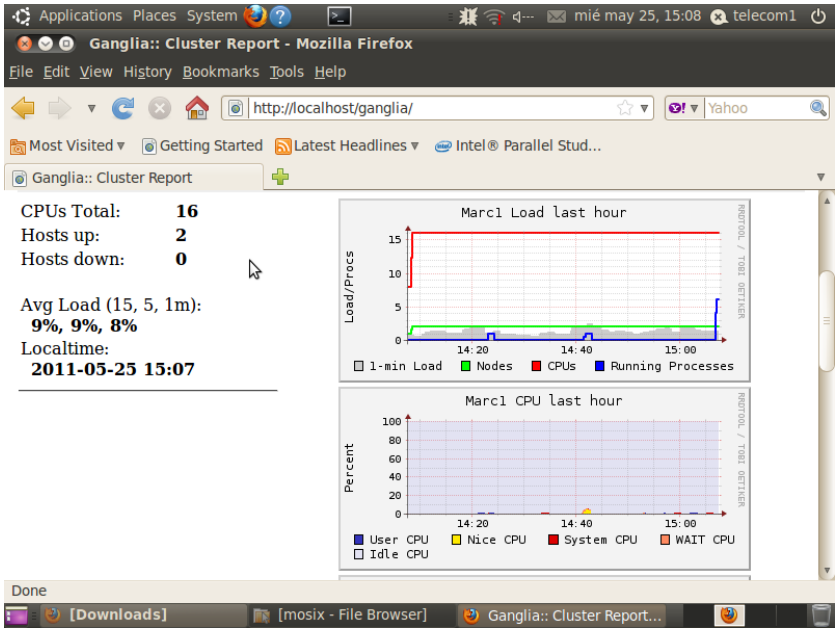


Figura 6.3: Imagen del monitor Ganglia

Anexo B: Publicaciones.

Aceptación de Revistas

- Indoor 2.4 GHz microwave propagation study using 3D FDTD approach,J. Sosa, S. Coss A. Rodrguez, L. Rodrguez, M. Galaz, E. Ramírez, and M. Enciso, Instituto Politecnico Nacional, ESIME-SEPI-Telecomunicaciones, Unidad Profesional Zacatenco, Mexico D.F.,Revista Electronics Letters, 24 November 2011 – Volume 47, Issue 24, p.13081309

Internacionales

- *Performance of The Three Dimensional Parallel FDTD Method on Different Architectures*, presentado en la Vigésima Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición Industrial IEEE. Diciembre 2011. Acapulco, México.
- *Parallel algorithm for solving the three-dimensional FDTD Method*, presentado en VI International Conference on Electromechanics and Systems Engineering (VI CIIES)el International . Marzo 2011. México, D.F.
- *Three Dimensional parallel FDTD Method based on Multicore processors*, presentado en el segundo International Supercomputing Conference in México en el Centro Internacional de Supercómputo: More than research. Marzo 2011. San Luis Potosí, México.
- *Parallel Performance of FDTD Based On Linux using Multicore Processors*, presentado en la Vigésima Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición Industrial IEEE. Diciembre 2010. Acapulco, México.

Nacionales

- *Optimization and parallelization of FDTD using Multicore Processors*, presentado en el Vigésimo Segundo Congreso Nacional de Ingeniería Electromecánica y de Sistemas. Noviembre 2010. México, D.F.
- *Implementación de un cluster para el uso de aplicaciones paralelas a través de computadoras multinúcleo*, presentado en el 11vo Congreso Nacional de Ingeniería Electromecánica y de Sistemas. Noviembre 2009. México, D.F.

Anexo C: ParalelizaciÓN del MDFDT en tres dimensiones.

Código del algoritmo principal del MDFDT en Fortran utilizando directivas de OPENMP

$**********INICIA\ FDTD\ EN\ 3\ DIMENSIONES**********$

OMP END PARALLEL DO

OMP PARALLEL

$Ciclo_Tiempo : DOTime = 1, IterMax$

$PRINT'(1X, A21, I7, A14, I7)', "IteracionesTotales : ", IterMax, "Progreso : ", Time$

$**********Hx**********$

OMP PARALLEL DO DEFAULT(NONE)

OMP SHARED(C9,Hxz,C10,Eyx,Eyz,DeltaEsp,C7,Hxy,C8,Ezx,Ezy,Imax,Jmax,Kmax)

$OMPPRIVATE(I, J, K)$

$DOK = 0, Kmax$

$DOJ = 0, Jmax$

$DOI = 1, Imax$

$Hxz(I, J, K) = C9(I, J, K) * Hxz(I, J, K) + C10(I, J, K) * ((Eyx(I, J, K + 1)$

$+Eyz(I, J, K + 1) - Eyx(I, J, K) - Eyz(I, J, K)) * (1/DeltaEsp))$

$Hxy(I, J, K) = C7(I, J, K) * Hxy(I, J, K) - C8(I, J, K) * ((Ezx(I, J + 1, K)$

$+Ezy(I, J + 1, K) - Ezx(I, J, K) - Ezy(I, J, K)) * (1/DeltaEsp))$

$ENDDO$

$ENDDO$

$ENDDO$

$OMPENDPARALLELDO$

!$Fuentepuntualsenoidalpara Hxz$

$Hxz(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

!$Fuentepuntualsenoidalpara Hxy$

$Hxy(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

OMP PARALLEL BARRIER

$Hx = Hxy + Hxz$

!$**********Hy**********$

OMP PARALLEL DO DEFAULT(NONE)

OMPSHARED(C11,Hyx,C12,Ezx,Ezy,DeltaEsp,C9,Hyz,C10,Exy,Exz,Imax,Jmax,Kmax)

OMP PRIVATE(I,J,K)

$DOK = 0, Kmax$

$DOJ = 1, Jmax$

$DOI = 0, Imax$

$Hyx(I, J, K) = C11(I, J, K) * Hyx(I, J, K) + C12(I, J, K) * ((Ezx(I + 1, J, K)$

$+Ezy(I + 1, J, K) - Ezx(I, J, K) - Ezy(I, J, K)) * (1/DeltaEsp))$

$Hyz(I, J, K) = C9(I, J, K) * Hyz(I, J, K) - C10(I, J, K) * ((Exy(I, J, K + 1)$

$+Exz(I, J, K + 1) - Exy(I, J, K) - Exz(I, J, K)) * (1/DeltaEsp))$

$ENDDO$

$ENDDO$

$ENDDO$

OMP END PARALLEL DO

!$Fuente\ puntual\ senoidal\ para\ Hyx$

$Hyx(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

!$Fuente\ puntual\ senoidal\ para\ Hyz$

$Hyz(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

OMP PARALLEL BARRIER

$Hy = Hyx + Hyz$

$**********Hz**********$

OMP PARALLEL DO DEFAULT(NONE)

OMP SHARED(C7,Hzy,C8,Exy,Exz,DeltaEsp,C11,Hzx,C12,Eyx,Eyz,Imax,Jmax,Kmax)

OMP PRIVATE(I,J,K)

$DOK = 1, Kmax$

$DOJ = 0, Jmax$

$DOI = 0, Imax$

$Hzy(I,J,K) = C7(I,J,K) * Hzy(I,J,K) + C8(I,J,K) * ((Exy(I,J+1,K)$

$+Exz(I,J+1,K) - Exy(I,J,K) - Exz(I,J,K)) * (1/DeltaEsp))$

$Hzx(I,J,K) = C11(I,J,K) * Hzx(I,J,K) - C12(I,J,K) * ((Eyx(I+1,J,K)$

$+Eyz(I+1,J,K) - Eyx(I,J,K) - Eyz(I,J,K)) * (1/DeltaEsp))$

$ENDDO$

$ENDDO$

$ENDDO$

$OMPENDPARALLELDO$

!$Fuente\ puntual\ senoidal\ para\ Hzy$

$Hzy(500,297,222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

!$Fuente\ puntual\ senoidal\ para\ Hzx$

$Hzx(500,297,222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

OMP PARALLEL BARRIER

$Hz = Hzx + Hzy$

$**********Ex**********$

OMP PARALLEL DO DEFAULT(NONE)

OMP SHARED(C1,Exy,C2,Hzx,Hzy,DeltaEsp,C3,Exz,C4,Hyx,Hyz,Imax,Jmax,Kmax)

OMP PRIVATE(I,J,K)

DO K=1,Kmax

DO J=1,Jmax

DO I=0,Imax

$Exy(I,J,K) = C1(I,J,K) * Exy(I,J,K) + C2(I,J,K) * ((Hzx(I,J,K)$

$+Hzy(I,J,K) - Hzx(I,J-1,K) - Hzy(I,J-1,K)) * (1/DeltaEsp))$

$Exz(I,J,K) = C3(I,J,K) * Exz(I,J,K) - C4(I,J,K) * ((Hyx(I,J,K)$

$+Hyz(I,J,K) - Hyx(I,J,K-1) - Hyz(I,J,K-1)) * (1/DeltaEsp))$

$ENDDO$

$ENDDO$

$ENDDO$

$OMPENDPARALLELDO$

!$Fuente$

$Exy(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

$Exz(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

!$PEC\ en\ direccion\ Y$

$Exy(0 : Imax, 0, 0 : Kmax + 1) = 0.0$

$Exy(0 : Imax, Jmax + 1, 0 : Kmax + 1) = 0.0$

!$PECendireccinZ$

$Exy(0 : Imax, 0 : Jmax + 1, 0) = 0.0$

$Exy(0 : Imax, 0 : Jmax + 1, Kmax + 1) = 0.0$

!$PEC\ en\ direccion\ Y$

$Exz(0 : Imax, 0, 0 : Kmax + 1) = 0.0$

$Exz(0 : Imax, Jmax + 1, 0 : Kmax + 1) = 0.0$

!$PEC\ en\ direccion\ Z$

$Exz(0 : Imax, 0 : Jmax + 1, 0) = 0.0$

$Exz(0 : Imax, 0 : Jmax + 1, Kmax + 1) = 0.0$

OMP PARALLEL BARRIER

$Ex = Exy + Exz$

$**********Ey**********$

OMP PARALLEL DO DEFAULT(NONE)

OMP SHARED(C3,Eyz,C4,Hxy,Hxz,DeltaEsp,C5,Eyx,C6,Hzy,Hzx,Imax,Jmax,Kmax)

OMP PRIVATE(I,J,K)

DO K=1,Kmax

DO J=0,Jmax

DO I=1,Imax

$Eyz(I, J, K) = C3(I, J, K) * Eyz(I, J, K) + C4(I, J, K) * ((Hxy(I, J, K)$

$+Hxz(I, J, K) - Hxy(I, J, K - 1) - Hxz(I, J, K - 1)) * (1/DeltaEsp))$

$Eyx(I, J, K) = C5(I, J, K) * Eyx(I, J, K) - C6(I, J, K) * ((Hzx(I, J, K)$

$+Hzy(I, J, K) - Hzx(I - 1, J, K) - Hzy(I - 1, J, K)) * (1/DeltaEsp))$

END DO

END DO

END DO

OMP END PARALLEL DO

$!FUENTE$

$Eyz(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

$Eyx(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

$!PEC\ en\ direccion\ X$

$Eyz(0, 0 : Jmax, 0 : Kmax + 1) = 0.0$

$Eyz(Imax + 1, 0 : Jmax, 0 : Kmax + 1) = 0.0$

$!PEC\ en\ direccion\ Z$

$Eyz(0 : Imax + 1, 0 : Jmax, 0) = 0.0$

$Eyz(0 : Imax + 1, 0 : Jmax, Kmax + 1) = 0.0$

$!PEC\ en\ direccion\ X$

$Eyx(0, 0 : Jmax, 0 : Kmax + 1) = 0.0$

$Eyx(Imax + 1, 0 : Jmax, 0 : Kmax + 1) = 0.0$

$!PEC\ en\ direccion\ Z$

$Eyx(0 : Imax + 1, 0 : Jmax, 0) = 0.0$

$Eyx(0 : Imax + 1, 0 : Jmax, Kmax + 1) = 0.0$

OMP PARALLEL BARRIER

$Ey = Eyx + Eyz$

$**********Ez**********$

OMP PARALLEL DO DEFAULT(NONE)

OMP SHARED(C5,Ezx,C6,Hyx,Hyz,DeltaEsp,C1,Ezy,C2,Hxy,Hxz,Imax,Jmax,Kmax)

OMP PRIVATE(I,J,K)

DO K=0,Kmax

DO J=1,Jmax

DO I=1,Imax

$Ezx(I, J, K) = C5(I, J, K) * Ezx(I, J, K) + C6(I, J, K) * ((Hyx(I, J, K)$

$+Hyz(I, J, K) - Hyx(I - 1, J, K) - Hyz(I - 1, J, K)) * (1/DeltaEsp))$

$Ezy(I, J, K) = C1(I, J, K) * Ezy(I, J, K) - C2(I, J, K) * ((Hxy(I, J, K)$

$+Hxz(I, J, K) - Hxy(I, J - 1, K) - Hxz(I, J - 1, K)) * (1/DeltaEsp))$

END DO

END DO

END DO

OMP END PARALLEL DO

!$Fuente$

$Ezx(500, 297, 222) = 5 * sin(Frad * DeltaTemp * REAL(Time))$

Ezy(500,297,222) = 5*sin(Frad*DeltaTemp*REAL(Time))

!$PEC\ en\ direccion\ X$

$Ezx(0, 0 : Jmax + 1, 0 : Kmax) = 0.0$

$Ezx(Imax + 1, 0 : Jmax + 1, 0 : Kmax) = 0.0$

!$PEC\ en\ direccion\ Y$

$Ezx(0 : Imax + 1, 0, 0 : Kmax) = 0.0$

$Ezx(0 : Imax + 1, Jmax + 1, 0 : Kmax) = 0.0$

!$PEC\ en\ direccion\ X$

$Ezy(0, 0 : Jmax + 1, 0 : Kmax) = 0.0$

$Ezy(Imax + 1, 0 : Jmax + 1, 0 : Kmax) = 0.0$

!$PEC\ en\ direccion\ Y$

$Ezy(0 : Imax + 1, 0, 0 : Kmax) = 0.0$

$Ezy(0 : Imax + 1, Jmax + 1, 0 : Kmax) = 0.0$

OMP PARALLEL BARRIER

$Ez = Ezx + Ezy$

OMP PARALLEL BARRIER

END DO Ciclo Tiempo

$OMPENDCRITICAL$

$EndprogramPMLv15$

Bibliografía

[1] Tesis: *Propagación electromagnética en un recinto cerrado utilizando diferentes materiales*,Salvador Coss Domínguez, SEPI ESIME Zacatenco, México D.F., 2011

[2] *Metodología de monitoreo para validación de circuitos VLSI*,Jaime Ismael Rangel Martínez, pages 10-27, Diciembre de 2009

[3] *On the dispersion in TLM and FDTD*,Krumphoiz M. Russer P,IEEE Transactions on MTT, vol 42, no.7 pages 1277-1279, Julio 1994

[4] *International Journal of Numerical Modelling*Meliani H, de Cogan D, Johns PB, 1998, PAGES 221-238

[5] *Field Analysis Software based on the transmission-line modelling method*Cristopolous C.Advances in Engineering Software, Springer-Verlag, pp 135-148

[6] *The transmission line matrix(TLM)method*Numerical techniques in microwave and milimetre wave passive devices,Wiley, 1981, pp 496-591

[7] *Equivalence of propagation characteristics for the TLM and FDTD*, Simpson N y Bridges E. IEEE transactions on MTT, vol. 39 pages 354.357, Feb 1991

[8] *Field Computations by Moment Methods*,Harrington, MacMillan, 1968

[9] *Computational Electrodynamics: The Finite Difference Time Domain Method*A. Taflove, Artech House,2000.

[10] *Numerical solution of initial boundary value problem involving Maxwells equations in isotropic media*K. S. Yee, IEEE Transactions on antennas and Propagation, vol. 14, May 1966, pp. 302-307.

[11] *Modelo de propagacin electromagntica en recintos cerrados con obstculos de diferentes materiales*Salvador Coss Dominguez, SEPI ESIME IPN, 2011

[12] *Estudio de propagacin electromagntica en medios guiados no homogneos usando el Mtodo de Diferencias Finitas en el Dominio del Tiempo*Jorge Sosa Pedroza, Mauro Enciso Aguilar, Jos Lus Lpez Bonilla, Jos Ricardo Garca Olivo, II Congreso Internacional de Electromagnetismo Aplicado, Santiago de Cuba, Junio 2007

[13] *A perfectly matched layer for the Absorption of electromagnetic waves*J. Berenger, Journal of computation physics 111, 185 200, 1994.

[14] *Three-Dimensional Perfectly Matched Layer for the Absorption of Electromagnetic Wave*J, Berenger,Journal of computational physics 127, 363.379 (1996)

[15] *Parallel Finite-Difference Time-Domain Method*,Wenhua Yu, Raj Mittra, Tao Su, Artech House Electromagnetic Analysis Ser.71:230-231, (1947).

[16] *High-level parallel programming models and supportive environments*, Frank Mueller, Springer W. Shockley, M. Sparks, G. Teal, Physical Review 83:151-162, (1951).

[17] *Parallel programming: Techniques and aplications using networked workstations and parallel computers*, Barry Wilkinson, Michael Allen, Prentice Hall

[18] *Barbara Chapman, Gabriele Jost and Ruud van der Pas, ,Using OpenMP Portable Shared Memory Parallel Programming, 93-110*

[19] *Programación sobre sistemas de memoria compartida: programación con OPENMP*, María J. Martín, Universidad de A. Coruña

[20] *Linux Bible 2008: Boot up to Ubuntu, Fedora, KNOPPIX, Debian, openSUSE, and 11 Other Distributions, Vicki Stanfield*

[21] *Parallel programming in C with MPI and OPENMP*,Michael J. Quinn, McGraw-Hill, 2004.

[22] *Technical News from The Portland Group, October 2009*

[23] *A Parallel FDTD Algorithm Using the MPI Library Laboratoire ART (Antennas, Radar & Telecommunications*,C. Guiffaut and K. Mahdjoubi , IEEE Antennas and Propagation Magazine, Vol. 43, NO. 2, France April 2001

[24] *How to Build a Beowulf: A Guide to the Implementation and Application of PC, Clusters*, Thomas L. Sterling, John Salmon, Donald J. Becker and Daniel F. Savarese,MIT Press, 1999

[25] *Parallel 3D FDTD Simulator for photonic crystals*,JASON S. AYUBI-MOAK, STEPHEN M. GOODNICK, GIL SPEYER, AND DANIEL C. STANZIONE, , IEEE computer society 2007

[26] *A Soft Real-time Concurrent Graphics Platform*, Andri F. B. Silva, Felipe B. Alves, Florianpolis, Olinto J. V. Furtado,Computer society IEEE , Brazil 2012

[27] *Parallel protein secondary structure prediction based on neural networks*, Zhong, W.; Altun, G.; Tian, X.; Harrison, R.; Tai, P.C.; Pan, Y.;Engineering in Medicine and Biology Society, 2004. 26th Annual International Conference of the IEEE,Volume: 2, 2004 , Page(s): 2968 - 2971

[28] *MOSIX: Implementation, Trend and Benchmark in Malaysia, M. Hakim, J. Jais, S. Salwa*, College Of Infonnation Technology, Universiti Tenaga Nasional, IEEE 2008

[29] *Impact of multicores on large-scale moleculardynamics simulations*Alam, S.R.; Agarwal, P.K.; Hampton, S.S.; Hong Ong; Vetter, J.S.; Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on 2008 , Page(s): 1 - 7,

[30] *Computer aided drug design strategies for thedevelopment of novel beta lactam analogs targeting penicillin binding proteins*, Suresh, M.X.;, Trendz in Information Sciences & Computing (TISC), 2010 , Page(s): 88 - 93

[31] *Parallel compression of correlated files*,Meiri, E.; Barak, A.;Cluster Computing, 2007 IEEE International Conference on, 2007 , Page(s): 285 - 292

[32] *Study on the hybrid software architecture of distributed parallel rendering system for virtual design* Gao Xin; Jia Qing-xuan; Sun Han-xu; Song Jing-zhou; Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on,2009 , Page(s): 911 - 916

[33] *A robust parallel conformal FDTD algorithm and its application toelectrically large antenna array simulation* Wenhua Yu; Yongiun Liu; Tao Su; Neng-Tien Huang; Mittra,

R.;Antennas and Propagation Society International Symposium, 2004. IEEE Volume: 1, 2004 , Page(s): 1030 - 1033

[34] *A Novel FDTD Application Featuring OpenMP-MPI Hybrid Parallelization*, Mehmet F. Suc, Ihab El-Kady, David A. Baderí, Shawn-Yu Lin, February 14, 2004

[35] *A parallel FDTD algorithm using the MPI library*, C.Guiffaut and K. Mahdjoubi , April 2001, IEEE Antenas and propagation Magazine, Vol 43, No2,

[36] *Estimation of Parallel FDTD- based lectromagnetic field solver on Pc Cluster with Multicore CPU's*, Electrical Design of Advanced Packaging and bvSystems Symposium