ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

INFORME DE MATERIA DE GRADUACIÓN

"DESARROLLO DE UNA HERRAMIENTA PARA LA CREACIÓN Y ADMINISTRACIÓN DE CLÚSTERES COMPUTACIONALES PARA SIMULACIONES FDTD (FINITE-DIFFERENCE TIME-DOMAIN) CON EL PAQUETE MEEP, SOBRE EL SERVICIO ELASTIC COMPUTE CLOUD (EC2) DE LOS AMAZON WEB SERVICES (AWS)."

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN
ESPECIALIZACION SISTEMAS INFORMACIÓN

Presentada por:

MARCO ANTONIO CALDERÓN ARGUELLO

CARLOS FERNANDO CORRAL ESPINOZA

Guayaquil - Ecuador
2010

AGRADECIMIENTO

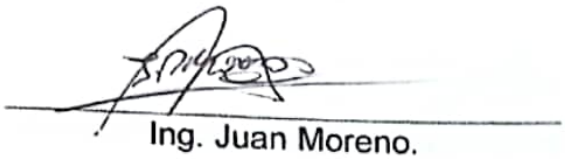
A Dios, todopoderoso y eterno.
A nuestras familias que nos han apoyando siempre en el camino de nuestra vida,
a todas las personas que nos han guiado y que sin su apoyo no hubiésemos podido lograr el presente trabajo.

DEDICATORIA

A todos aquellos que luchan por sus sueños e ideales, que no se dejan vencer fácilmente ante la adversidad y, que verdaderamente son participes de un cambio y mejora continua de nuestra sociedad

TRIBUNAL DE SUSTENTACIÓN

PROFESOR DE LA MATERIA DE GRADUACIÓN



Ing. Juan Moreno.

PROFESOR DELEGADO POR EL DECANO

Ing. Germán Vargas

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Marco Antonio Calderón Arguello

Carlos Fernando Corral Espinoza

RESUMEN

Actualmente, realizar simulaciones electromagnéticas constituye muchas veces un problema con un alto grado de complejidad, esto es, por la gran cantidad de procesamiento que se realiza para generar la solución de un problema. Una buena alternativa para efectuar un procesamiento eficiente y reducir el tiempo del mismo, está basada en la plataforma de cloud computing y sus herramientas.

En este trabajo se presenta la implementación de una herramienta para la creación y administración de clústeres computacionales para realizar simulaciones FDTD sobre el servicio EC2 de Amazon Web Services.

En el Capítulo primero, se describe los detalles del problema como el análisis de los antecedentes, los objetivos alcanzados, y la justificación de porque se realiza la implementación de la herramienta.

En el Capítulo segundo, se describe el marco teórico utilizado tales como la plataforma de cloud computing, el método FDTD de electromagnetismo computacional, el concepto y arquitectura de paralelismo.

En el Capítulo tercero, se describe las herramientas utilizadas tanto para el manejo eficiente del clúster como para las simulaciones electromagnéticas, dentro de este campo se utilizó la herramienta StarCluster, el paquete Parallel Meep, el Framework GWT y Ganglia.

En el Capítulo cuarto, se detalla la arquitectura e implementación de la herramienta final, y aspectos tales como la instalación, configuración y desarrollo de cada uno de sus componentes.

En el Capítulo quinto, se detallan las pruebas realizadas, comparaciones de rendimiento de la plataforma de cloud computing, y el manejo eficiente de los recursos disponibles.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

FDTD	Finite-Difference Time-Domain
EC2	Amazon Elastic Compute Cloud
AWS	Amazon Web Services
S3	Amazon Simple Storage Service
GWT	Google Web Toolkit
ESPOL	Escuela Superior Politécnica Del Litoral.
NFS	Network File System (Sistema de archivos de red)
HDF5	Hierarchical Data Format
MPI	Message Passing Interface o Interfaz de Paso de Mensajes
SGE	Sun Gride Engine
RRD	Round-Robin Database
XML	Extensible Markup Language Llenguaje de Marcas Extensible)
XDR	eXternal Data Representation HTTP HyperText Transfer Protocol
HPC	High-performance computing (Computación de alto rendimiento).
SSH	Secure Shell (Intérprete de Comandos Seguro)
I/O	Internet Protocol (Protocolo de Internet).
PHP	PHP Hypertext Pre-processor
AMI	Amazon Machine Image
IDE	Integrated Development Environment
CTL	Control Type Language
MIT	Massachusetts Institute of Technology

INTRODUCCIÓN

En la actualidad, para resolver problemas de simulaciones electromagnéticas existe un método estandarizado denominado FDTD. Este método se basa en resolver Ecuaciones Matemáticas (Ecuaciones de Maxwell) mediante un sistema de ecuaciones de diferencias finitas.

A nivel de software, existen aplicaciones que implementan este algoritmo **[1]**; y permiten realizar estas simulaciones, además de, ofrecer una visualización de sus respectivos resultados. Una de las dificultades presentes está en que existen simulaciones muy complejas de realizar debido a la gran cantidad de cálculos que el mismo posee. Además, investigadores indican la necesidad de ejecutar varias simulaciones que requieren parametrización de valores dentro de la estructura de un problema con el objetivo de poder realizar comparaciones de resultados en diferentes escenarios.

El problema está en que al realizar esto en una máquina local puede llegar a consumir muchos recursos y tiempo por escenario; lo más óptimo es poder realizarlo en un clúster de alto rendimiento.

Una buena alternativa para efectuar un procesamiento eficiente es reducir costos y tiempo utilizando la plataforma de cloud computing y sus respectivas herramientas.

En este trabajo se presenta la implementación de una herramienta para la creación y administración de clústeres computacionales para realizar simulaciones FDTD sobre el servicio EC2 de AWS.

CAPÍTULO 1

1. Antecedentes y Justificación.

En este capítulo se provee información acerca de los antecedentes sobre los cuáles se plantearon los objetivos, una explicación profunda del problema y la justificación del presente trabajo.

1.1. Antecedentes

Maxwell formula cerca del año 1870 las ecuaciones diferenciales parciales de electrodinámica. La misma, representa la unificación fundamental de los campos eléctricos y magnéticos, prediciendo el fenómeno de las ondas electromagnéticas. A lo cual Nobel y Feynman llamaron el más excepcional logro de la ciencia en el siglo XIX **[2].**

El método de diferencias finitas en el dominio del tiempo (FDTD) resuelve las ecuaciones de Maxwell, modelando directamente la propagación de las ondas electromagnéticas dentro de un volumen. Este método fue presentado en 1966 por Kane Yee, que es una técnica de modelado numérico electrodinámico.

Al principio era casi imposible implementar este método computacionalmente, probablemente debido a la falta de recursos

informáticos. Sin embargo, con la llegada de equipos más poderosos, modernos, con mayor accesibilidad para adquirirlos y además de las mejoras en el algoritmo, el método FDTD se ha convertido en una herramienta estándar para resolver problemas de este tipo. Hoy en día, podemos decir que este método agrupa a un conjunto de técnicas numéricas que permiten la resolución de las ecuaciones de Maxwell en el dominio del tiempo y que permiten el análisis electromagnético de una amplia gama de problemas **[3]**. Ahora los científicos e ingenieros usan las computadoras para obtener soluciones de estas ecuaciones con el propósito de investigar estos campos electromagnéticos.

1.2. Definición del Problema

En la actualidad, el método o algoritmo FDTD es una de las técnicas más utilizadas para cálculos en el campo electromagnético computacional. Existe una implementación de esta técnica en Linux llamado Meep. Sin embargo una de los principales problemas que tiene este método es el tiempo y utilización de recursos que requiere para resolver un problema, especialmente si las simulaciones involucran estructuras en tercera dimensión. Esto se debe al gran procesamiento que hace al momento de generar la solución.

Hoy en día, estos tipos de problemas son resueltos con mayor facilidad, con la aparición de lo que hoy llamamos Clúster, un conjunto de ordenadores potentes que funcionan como un solo sistema que permiten mejorar el rendimiento en cuanto a procesamiento masivo [4].

El problema de usar los mismos es el costo que implica en obtener uno y la difícil administración y configuración para su funcionamiento.

El “Sistema de creación y administración de clústeres computacionales para simulaciones FDTD con el paquete Meep sobre el servicio de Elastic Compute Cloud (EC2)”, ha sido creado con la finalidad de facilitar la administración y ejecución de clústeres computacionales sobre el servicio de cloud computing de Amazon EC2 para resolver problemas FDTD, mejorar su rendimiento y optimizar el tiempo que lleva generar la solución, utilizando los servicios de cloud computing que brinda la plataforma de Amazon EC2.

1.3. Objetivos

El “Sistema de creación y administración de clústeres computacionales para simulaciones FDTD con el paquete Meep sobre el servicio de Elastic Compute Cloud (EC2)”, fue planteado para proporcionar una herramienta de monitoreo y ejecución de múltiples simulaciones FDTD. Con la finalidad de lograr esto se trazaron los siguientes objetivos:

- Proveer un AMI pública que permita la administración y creación de clústeres computacionales para simulaciones FDTD aplicando procesamiento distribuido.
- Integrar una herramienta Web que ofrezca un monitoreo de recursos utilizados por los clústeres con gráficos de los resultados.
- Implementar una interfaz Web para la administración del AMI pública.

1.4. Justificación

La justificación principal para el desarrollo del “Sistema de creación y administración de clústeres computacionales para simulaciones FDTD con el paquete Meep sobre el servicio de Elastic Compute Cloud (EC2)” es disminuir el tiempo que toma en realizar las simulaciones FDTD junto con un rendimiento mas óptimo, y monitorear los recursos utilizados durante la ejecución de múltiples simulaciones FDTD que se realizan paralelamente. De esta manera, el usuario puede verificar el estado de sus trabajos mientras estos son resueltos a través del paquete Meep.

CAPÍTULO 2

2. Análisis Conceptual

2.1. Cloud Computing

Es una tecnología que permite ofrecer servicios informáticos a través de la plataforma del Internet y pagar según lo que consuma. Su funcionamiento consiste en basar aplicaciones en servicios alojados de forma externa dentro de la web**. [5]**

Gracias a este tipo de tecnología, todo lo que puede ofrecer un sistema informático, se ofrece como servicio, de manera que los usuarios puedan acceder a los mismos disponibles "en la nube" sin ser expertos en la gestión de los recursos a utilizar.

No hay necesidad de conocer la infraestructura, es una nube donde aplicaciones y servicios pueden ser fácilmente escalables, eficientes, sin conocer los detalles de su funcionamiento e instalación.

Como ejemplos destacan Amazon*EC2, Google App Engine, eyeOS y Microsoft Azure.*

2.1.1. Amazon EC2

Para la implementación de nuestra herramienta utilizamos el servicio de Cloud Computing ofrecido por Amazon EC2, un servicio web que ofrece determinada capacidad de cómputo con opción de crecimiento en la nube, de acuerdo al requerimiento del usuario. Está diseñado para facilitar la escabilidad web computacional a los desarrolladores.

2.1.2. Funcionalidad

EC2 presenta un entorno computacional de manera virtual, permitiéndonos utilizar las interfaces de servicios web para solicitar clústeres para su uso, se carga nuestro propio entorno aplicativo, administramos los permisos y accesos de red y ejecutamos ésta imagen utilizando tantos sistemas como se requiera.

Está diseñado para usar con otros Servicios de Amazon en conjunto así como Amazon S3, Amazon EBS, Amazon SimpleDB y Amazon SQS, para proveer una completa solución computacional.

Nos provee seguridad, ya que posee numerosos mecanismos de seguridad como firewall, accesos, configuraciones de red, etc.

2.2. Método FDTD

El método FDTD permite simular la evolución temporal del campo electromagnético en una región de interés, además de poder realizar alteraciones en su estructura. La formulación original del método propuesto por Yee estudia el comportamiento del campo electromagnético en el vacío. Así mismo, permite definir algunas condiciones de contorno sencillas, como las de pared eléctrica y pared magnética, siendo posible aplicar el método al estudio de problemas resonantes.

Para este método se utilizan las Ecuaciones de Maxwell que describen la evolución en el tiempo y en el espacio de los campos magnéticos ***B*** y eléctricos ***E.***

Estas ecuaciones en derivadas parciales son reemplazadas por un sistema de ecuaciones en diferencias finitas. Las diferencias finitas es una expresión matemática que permite realizar una aproximación a soluciones de ecuaciones diferenciales.

La técnica FDTD se basa en la división en varias partes, tanto espacial como temporal, de los campos electromagnéticos y la aproximación de las derivadas parciales que aparecen en las ecuaciones de Maxwell del rotacional expresadas en el dominio del tiempo por cocientes de diferencias finitas. Como resultado, se obtiene un problema algebraico de tipo explícito que permite ir calculando, en instantes sucesivos, el valor del campo eléctrico (magnético) en cada punto del espacio a partir del valor del campo eléctrico (magnético) del mismo punto en el instante de tiempo anterior y de los valores del campo magnético (eléctrico) en sus nudos adyacentes y en el instante de tiempo anterior.

Transcurrieron nueve años hasta que el método FDTD original fue convenientemente modificado para la resolución de un problema de scattering **[6]**.

2.3. Paralelismo Computacional

La computación paralela es el uso simultáneo de múltiples recursos de cómputo para resolver un problema computacional.

Un problema se divide en partes distintas que se pueden resolver al mismo tiempo. Cada parte se desglosan en una serie de instrucciones y cada instrucción se ejecutan de manera simultánea en diferentes CPU's.

El cálculo de los recursos puede incluir un solo equipo con varios procesadores, o un número arbitrario de ordenadores conectados por una red o una combinación de las anteriores.

El problema computacional suele mostrar características como la capacidad de ser dividida en piezas discretas de trabajo que pueden ser resueltos al mismo tiempo, ejecutar múltiples instrucciones del programa en cualquier momento en el tiempo y resolver en el menor tiempo con múltiples recursos de cómputo.

El paralelismo ha sido empleado durante muchos años, sobre todo para la Computación de alto rendimiento. **[7]**

Computación de alto rendimiento (HPC), son aquellos que utilizan superordenadores y clústeres de ordenadores para resolver problemas de

cálculos avanzado. Por lo general utilizados para investigaciones científicas. **[8]**

2.3.1. Arquitectura de memoria de Memoria Distribuida

Un sistema de memoria distribuida está compuesto por varios procesadores independientes con una memoria local, y conectados a través de una red. Es decir, que cada procesador dispone de su propia memoria al momento de realizar algún proceso. Este tipo de memoria brinda la posibilidad de fácilmente poder escalar, pero también conlleva a que debe existir un medio de comunicación entre los nodos para que exista una sincronización. Esta tarea de comunicación está implementada, en el caso de nuestro proyecto, la misma que está a cargo del paquete Meep.

Este tipo de arquitectura trae muchas ventajas como las siguientes:

- Cada procesador puede acceder directamente a su memoria sin interferir y sin sobrecargar a los demás.
- Ofrece escalabilidad con respecto al número de procesadores para conectarlos, y únicamente depende de la red.
- No hay problemas de coherencia de cache, ya que cada procesador contiene sus propios datos, y no tiene que preocuparse por las copias locales.

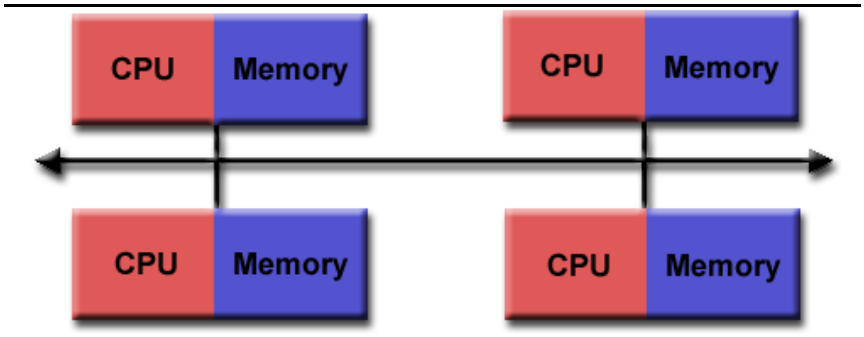


Figura1: Arquitectura de Memoria Distribuida

La principal dificultad que presenta esta arquitectura es el método de comunicación entre los procesadores, ya que si un procesador requiere información de otro esta debe ser enviada a través de mensajes. De la cual se destaca dos aspectos de sobrecarga: el tiempo de construir y enviar un mensaje de un procesador a otro, y la interrupción de un procesador receptor para manejar los mensajes enviados por otros procesadores. **[7]**

2.4. Modelos De Programación Paralela

Hemos ya mencionado acerca de la Arquitectura de Memoria Paralela que usamos en nuestro proyecto. Ahora en esta sección del documento procederemos a realizar una explicación del Modelo de Programación paralela que hemos implementado.

Generalmente un modelo de programación paralela está basada en alguna arquitectura de memoria, en nuestro caso, para el desarrollo de nuestra aplicación hemos usado la Arquitectura de Memoria Distribuida.

Un modelo de Programación paralela no es más que un conjunto de algoritmos, procedimientos o herramientas de software que permiten la creación de Aplicaciones Paralelas y sistemas de comunicación de I/O.

Para la implementación de aplicaciones distribuidas, los desarrolladores deben saber escoger un modelo de programación paralela apropiado y en algunos casos, una combinación de ellos, que se acople al tipo de problema que se desea resolver.

Nuestra herramienta incluye e implementa una aplicación que está basada en el modelo de **Interfaz de Paso de Mensajes.**

2.4.1. MPI (Interfaz de Paso de Mensajes)

Es una técnica empleada en programación paralela para el intercambio de información a través de una comunicación, basada en recepción y envío de mensajes. El envío de paso de mensajes puede ser de manera síncrona o asíncrona. Cuando el proceso espera recibir el mensaje para continuar su ejecución decimos que es síncrona mientras que cuando el proceso que envía, no espera que el mensaje sea recibido y continúa su ejecución decimos que es asíncrona.

La transferencia de información requiere sincronización entre procesos para mejorar su rendimiento. Su principal característica radica en que utiliza su propia memoria local durante el proceso de ejecución y no de memoria compartida.

MPI se ha convertido en un estándar para la comunicación entre los nodos que ejecutan un problema en particular dentro de un Sistema Distribuido.

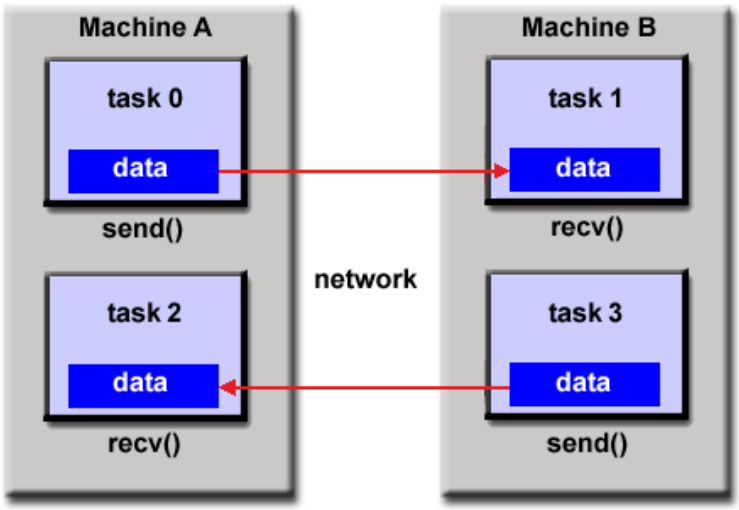


Figura 2: Envío de paso mensajes a través MPI entre dos Computadoras

CAPÍTULO 3.

3. Herramientas de Implementación

Para este capítulo veremos los detalles acerca de las herramientas que hemos utilizado para el desarrollo del proyecto, explicaremos acerca de sus conceptos y características importantes que provee cada una de éstas.

3.1. StarCluster

StarCluster es una utilidad que permite la creación, administración y monitoreo de clústeres computacionales que se encuentran alojados en el servicio de Amazon Elastic Compute Cloud (EC2) todo a través de una instancia máster. Su objetivo principal es reducir al mínimo la administración asociada a la configuración y manejo de clústeres computacionales utilizados en laboratorios de investigación o aplicaciones en general que utilizan computación distribuida.

Para utilizar esta herramienta se crea un archivo de configuración dónde se detalla información sobre la cuenta en Amazon Web Services, el tipo de AMI'S a utilizar y características adicionales que deseemos configurar en el clúster. Posteriormente, se podrá realizar la ejecución de la herramienta mediante el uso de comandos**. [10]**

3.1.1. Características de StarCluster.

- Uso a través de comandos para realizar las tareas de creación, administración y monitoreo de uno o más clústeres sobre EC2.
- Provee un AMI pública previamente configurada con todo lo necesario para instalación de StarCluster.
- Soporte para servicios de almacenamiento en la nube tales como Elastic Block Storage (EBS) y Simple Storage Service (S3) ofrecidos también por Amazon.
- El AMI pública incluye herramientas como OpenMPI, ATLAS, Lapack, NumPy, y SciPy.
- Todos los nodos del Clúster se configuran automáticamente con los servicios de NFS, SGE y OpenMPI.

Para nuestro proyecto usaremos los servicios NFS y OpenMPI, por lo que detallaremos los conceptos de cada uno a continuación:

3.1.2. Network File System (Sistema de archivos de red), o NFS

Permite que distintos clientes conectados en una misma red accedan a archivos remotos compartidos, como si fueran parte de su sistema de archivo local. Este protocolo de aplicación trabaja en entorno cliente/servidor, dónde el servidor indica los directorios que desean compartir y los clientes montan estos directorios es su sistema de archivos.

Este puede ser utilizado en un entorno distribuido para realizar procesamiento de memoria compartida.

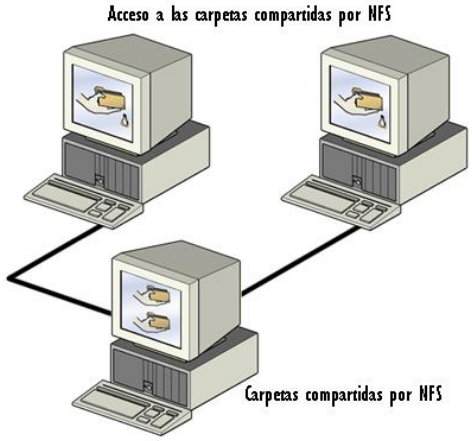


Figura 3: Acceso a archivos compartidos a través de NFS

3.1.3. OPENMPI

Es un proyecto que combina las tecnologías y recursos de otros proyectos (FT-MPI, LA-MPI, LAM/MPI, y PACX-MPI) para construir una librería MPI. Open MPI es una implementación de código abierto de los estándares MPI-1 y MPI-2. **[11]**

3.2. Paralell Meep

Meep es un paquete de Software de simulación desarrollado para modelar sistemas electromagnéticos. Meep implementa el algoritmo de tiempo en diferencias finitas de dominio (FDTD), método de electromagnetismo computacional. Este algoritmo consiste en dividir el espacio en una malla y, ver como los campos evolucionan en el tiempo, utilizando los pasos de tiempo, se vuelve más aproximado la solución para las ecuaciones

continuas, de ésta manera, se simulan muchos problemas esencialmente prácticos.

El paquete paralelo de Meep brinda soporte para realizar paralelismo con memoria distribuida, y trabajar en problemas muy grandes (problemas en espacio 3D) para que puedan ser resueltos de manera distribuida.

El problema debe ser lo suficientemente grande para poder beneficiarse de muchos procesadores. **[12]**

Para lograr esto, el paquete paralelo de Meep divide la celda computacional de la simulación en “chunks” que son asignados entre los procesadores. Cada “chunk” es puesto en pasos de tiempo, y los procesadores se encargan de comunicarse los valores usando MPI.

3.3. GWT- Google Web Toolkit

3.3.1.Introducción

GWT o Google Web Toolkit es un framework creado por Google que permite facilitar el uso de la tecnología AJAX. Permite resolver el gran problema de compatibilidad de código cliente (HTM, java script) entre navegadores, facilitando al usuario el desarrollo de alguna aplicación sin tener la necesidad de testearlos en varios navegadores. El concepto de Google Web Toolkit es bastante sencillo, básicamente lo que se debe hacer es crear el código en Java usando cualquier entorno de desarrollo (IDE) de Java y el compilador lo traducirá a HTML y Java Script.

3.3.2. Plataforma GWT

GWT tiene cuatro componentes principales: un compilador Java-a-Java Script, un navegador web "hosted", y dos librerías de clases:

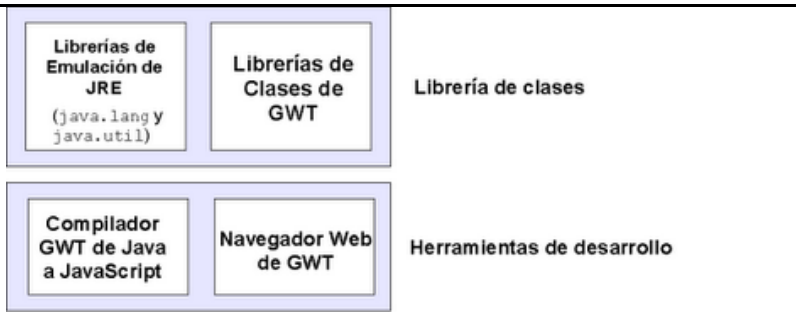


Figura 4: Componentes de GWT

- **GWT Java-to-JavaScript Compiler:** La función de este componente es traducir el código desarrollado en Java al lenguaje JavaScript.
- **Hosted Web Browser:** Este componente ejecuta la aplicación Java sin traducirla a JavaScript, en modo host usando la máquina virtual de Java.
- **JRE Emulation Library:** Contiene implementaciones en JavaScript de las librerías de clases más usadas en Java como java.lang, java.util, etc
- **GWT Web UI Class Library:** Contiene un conjunto de elementos de interfaz de usuario que permite la creación de objetos tales como textos, cajas de texto, imágenes y botones.**[13]**

3.4. Ganglia

3.4.1. Introducción

El monitoreo de un clúster computacional requiere una correcta administración de recursos así el administrador puede invertir menos tiempo en detectar, investigar, solucionar fallos que sucedan y además con esta información es posible plantear un plan de contingencia.

Ganglia es un sistema escalable y distribuido para el monitoreo de clústeres y Grids computacionales en tiempo real. Es una implementación robusta que ha sido adaptada a muchos sistemas operativos y distintos arquitecturas de computadores, y es actualmente usada en miles de clústeres alrededor del mundo como universidades, laboratorios de investigación comerciales y gubernamentales.

Ganglia fue inicialmente desarrollado por la Universidad de Berkeley en el departamento de ciencias computacionales para enlazar los clústeres del campus, actualmente está a cargo de SourceForge.

Está basado en un esquema jerárquico de clústeres y es configurado mediante archivos XML y XDR en cada nodo que permite tener extensibilidad y portabilidad. Es completamente Open-Source y no contiene ningún componente propietario. Ganglia enlaza líneas de clústeres y computación distribuida por lo que se conoce como “Clúster to Clúster” **[14]**.

3.4.2. Funcionamiento

Ganglia está definido en un esquema jerárquico. Se basa en una comunicación a través de un protocolo multicast de envío/recepción para controlar el estado del clúster y utiliza un árbol de conexiones punto a punto entre los niveles de nodos del clúster para reportar su estado. Ganglia usa mensajes de estado en un entorno multicast, como lo básico para un protocolo de comunicación. Para mantener la comunicación cada nodo envía su estado en un intervalo de tiempo, de ésta manera da a conocer que se encuentra activo, en el momento en que deje de enviar ese nodo deja de ser participe en el monitoreo.

Además, cada nodo monitorea sus recursos locales y envía paquetes multicast con la información de su estado cada vez que ocurra una actualización. Todos los nodos de un mismo clúster siempre tienen una vista aproximada del estado completo del clúster, y este estado es fácilmente reconstruido si sucede un colapso.

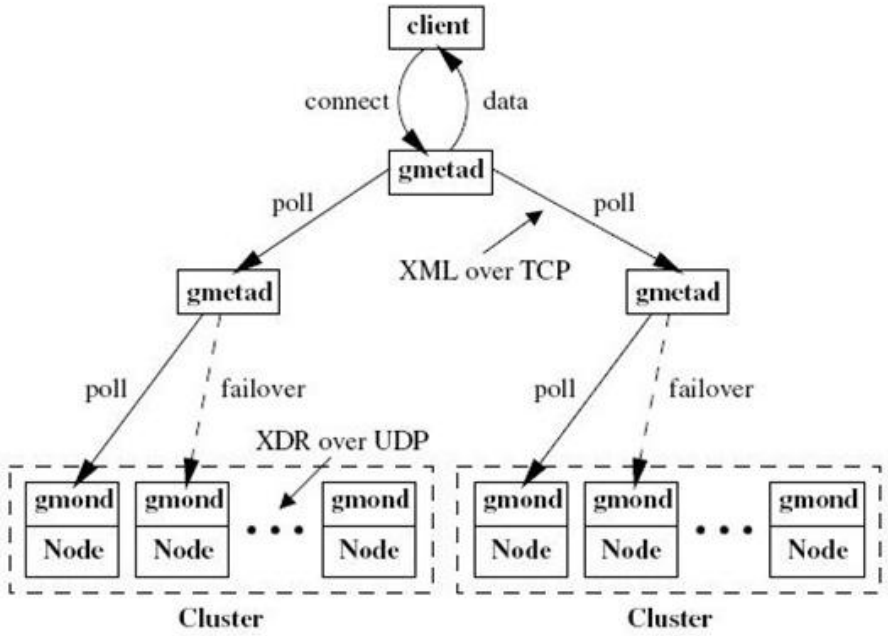


Figura 5: Esquema de Funcionamiento de Ganglia

3.4.3. Arquitectura Ganglia

Los principales componentes de Ganglia, son dos demonios *gmond* y *gmetad*.

El Ganglia Monitoring Daemon (**gmond**), es el pilar fundamental de la herramienta, es un demonio multi-hilo el cual corre en cada uno de los nodos del clúster que se desea monitorear. Su instalación es muy fácil. No hay necesidad de tener un sistema de archivo NFS en común ni una base de datos. Gmond tiene su propia base de datos distribuida y su propia redundancia.

El Ganglia Meta Daemon (**gmetad**), Este demonio permite obtener la información vía XML en intervalos regulares desde los nodos, el gmetad toma la información y la guarda en una base de datos Round-Robin (RRD) y concatena los XML de los nodos para compartir la información con el servidor web u otro Front-end que corra el demonio gmetad.

Otro de los componentes principales es la **aplicación web** integrada en la herramienta, Ganglia usa una base de datos round robin (RRD) para almacenar y consultar la información histórica para el clúster, presenta métricas basadas en el tiempo gradualmente. RRDTool es un popular sistema para almacenar y graficar datos en series del tiempo, la cual usa forma compacta especialmente diseñada para el almacenamiento de datos en series del tiempo. RRDtool genera gráficos los cuáles muestran la tendencia de las métricas versus el tiempo. Estos gráficos luego son exportados para ser desplegados en el Front-end.

La plantilla del Front-end puede ser personalizada o utilizar la que está por defecto, Ganglia realiza una separación entre el contenido y la presentación, el contenido se trata de un archivo XML que si se desea puede ser accedido directamente para alguna otra aplicación. **[14]**

CAPÍTULO 4.

4. Arquitectura e Implementación

4.1. Arquitectura

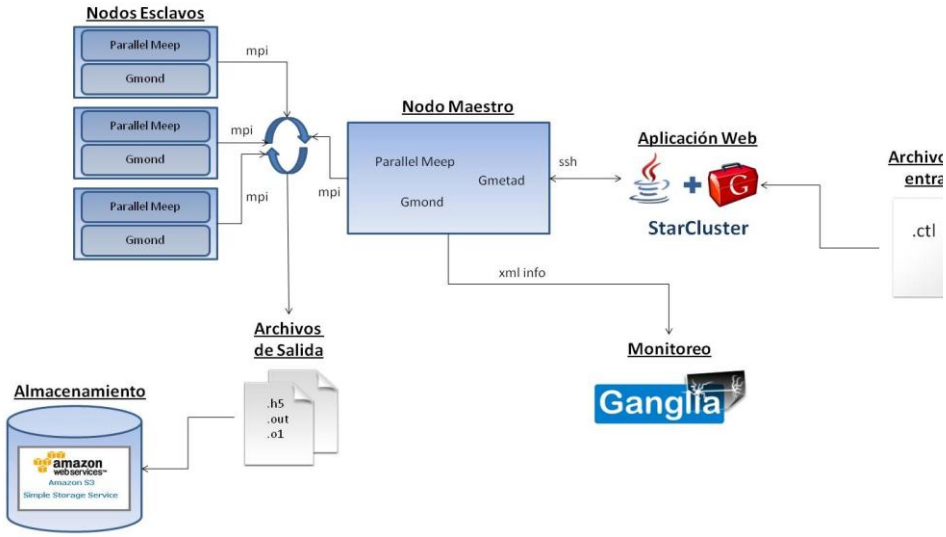


Figura 6: Diseño de Arquitectura de StarMeep

4.1.1. Archivos de Entrada

Los archivos de entrada para la aplicación son los de tipo CTL (Control Type Language) que son los que utiliza el paquete Meep para poder realizar las simulaciones. Un archivo CTL especifica la geometría del problema, las fuentes a utilizar, las salidas, y todo lo demás que sea necesario para poder realizar el cálculo. Este archivo es escrito en forma de un lenguaje de scripting y nos permite definir la estructura de un problema como una secuencia.

El archivo CTL es parte de la librería libctl que es un conjunto de herramientas basadas en el lenguaje *Scheme.* El archivo CTL puede ser escrito en cualquiera de estas tres formas:

- Scheme, que es un lenguaje de programación desarrollado por MIT. Este lenguaje cumple con la forma (*function arguments...*) y el cual puede ser ejecutado bajo un intérprete GNU Guile.
- Libctl, es una librería para el compilador Guile, la cual simplifica la comunicación entre Scheme y el software de computación científica. Libctl define la interfaz básica y un sinnúmero de funciones útiles.
- Meep, se puede escribir un CTL basado en meep mismo. Este define todas las interfaces específicas al cálculo de problemas FDTD.

4.1.2. Aplicación Web "StarMeep"

La implementación de la aplicación Web "StarMeep" permite mejorar la administración del clúster computacional para un usuario, ya que permite tener beneficios tales como:

- Disponibilidad a través de un navegador Web.
- Realizar configuraciones con formularios en lugar de hacerlo a través una consola.
- Visualización del procesamiento de cálculo de una simulación electromagnética.
- Acceso a los archivos de resultados de las simulaciones.

El desarrollo está basado en el lenguaje de programación Java junto con el framework GWT para el *front-end* de la aplicación. GWT, nos permite crear una interfaz simple para el usuario y además de poder realizar requerimientos al servidor vía la tecnología AJAX. En el desarrollo del *back-end* el punto principal fue el acceso y envío de comandos a las instancias o nodos del clúster. Esto se realizó por medio del protocolo SSH. Por lo cual se utilizó la librería JSch, que permite crear distintos tipo de conexiones tales como SSH, SCP y otras más desde Java.

4.1.3. Monitoreo

El monitoreo de recursos utilizados por parte de los nodos del clúster es realizado mediante la herramienta Ganglia. La información de utilización es enviada desde el clúster a través archivos XML y son procesados por la herramienta para su visualización.

Ganglia provee una vista de información de forma gráfica en tiempo real a través de su Web Front-End, a los administradores y usuarios del clúster acerca de la utilización de recursos consumidos.

4.1.4. Nodo Maestro y nodos esclavos

Dentro del procesamiento para el cálculo de las simulaciones FDTD, los nodos del clúster se encargan de procesar el archivo de entrada en forma paralela mediante el paquete Meep. Este divide equitativamente la simulación en tareas y cada tarea es entregada hacia un respectivo nodo para que el mismo lo procese. Cada uno de los nodos realiza las tareas de forma sincronizada comunicándose con los demás a través del protocolo de comunicación MPI.

Una vez que las tareas son realizadas por los nodos, los resultados son escritos mediante la librería HDF5 y son ubicados en un directorio NFS compartido del clúster. Este directorio puede estar dentro de la misma instancia o puede utilizarse un almacenamiento como el S3 de Amazon.

La información de la utilización de los recursos dentro del clúster es recompilada mediante un demonio *Gmond* que se encuentra dentro de cada uno de los nodos. Los nodos que serán monitoreados son especificados dentro del demonio *Gmetad* que se encuentra en el Nodo Maestro.

Finalmente, los resultados de la simulación como la información de los recursos son accedidos a través del Nodo maestro.

4.1.5. Archivos de Salida y Almacenamiento

Los archivos de salida generados por la simulación tienen el formato HDF5 que es un formato estándar científico utilizado por muchas herramientas de visualización como Matlab, GNU Octave, y otros más. Aparte del HDF5, también encontraremos archivos que nos indican si hubo algún problema en tiempo de ejecución y conocer el estado de la simulación a través de los pasos de tiempo.

Los resultados generados por la simulación pueden ser almacenados en la misma instancia, mediante un bloque EBS o almacenamiento S3 de AWS.

4.2. Creación y Configuración de un AMI Publico

Esta sección explicaremos la instalación y configuración de las distintas aplicaciones que usa nuestra herramienta. El detalle de estas aplicaciones fueron explicadas profundamente en el capítulo anterior.

4.2.1. Instalación de StarCluster

Para la instalación de esta herramienta usamos como base el AMI que provee la comunidad de StarCluster, la misma provee algunos servicios y dependencias, que ayudan a su rápida instalación. Visitamos la página oficial de StarCluster, donde se encuentra especificado el identificador de la AMI.A través de éste realizamos el levantamiento de una respectiva instancia. La misma posee las siguientes características:

- 1.7 GB Memoria RAM
- 2 Núcleos virtuales de 2.5 Ghz.
- 350 GB de disco duro
- Ubuntu 9.04 de 32 bits

Antes de la instalación de StarCluster, describiremos cuáles son sus algunas de sus dependencias más importantes.

- **Python (2.4**)

  Lenguaje de programación interpretado que permite dividir un programa en módulos para ser reutilizados en otros programas.

- **Boto(1.9b+)**

  Es un módulo integrado de python para el manejo de servicios de infraestructura actuales y futuros ofrecidos por AWS.

- **Paramiko (1.7.6+)**

  Es otro módulo de python que implementa el protocolo SSH2 (encriptación y autenticación) para conexiones a equipos remotos

En la instalación de Starcluster hemos descargado la última versión en desarrollo desde el repositorio GIT (Software para manejo de versiones), y luego compilado e instalado a través de python, los siguientes comandos describen lo explicado:

- Descarga el Instalador el StarCluster desde el repositorio

```
root@tmp:~#
git clone git://github.com/jtriley/StarCluster.git
```

- Una vez descargada nos movemos al directorio creado de Starcluster

```
root@tmp:~# cd StarCluster/
```

- Y con Python compilamos e instalamos el Starcluster.

```
root@tmp:~/StarCluster# sudo python setup.py install
```

- Una vez terminada la instalación, podemos comprobar ejecutando

```
root@tmp:~# starcluster start
```

Al ejecutar este comando, starcluster pedirá un archivo de configuración, el mismo no será creado desde aquí sino a través de la Aplicación Web StarMeep que más adelante hablaremos.

4.2.2. Instalación del Paquete Meep-OpenMPI

Previo a la instalación del Paquete Meep, revisaremos las principales dependencias que requiere este software.

- **Guile-1.8-libs**

  Guile es una implementación de scheme (lenguaje de Programación Funcional) diseñada para la programación.

- **Libctl3**

  Es la implementación de una librería libre basado en Guile, utilizado para simulaciones científicas.

- **Libhdf5**

  Librería que le permite a Meep imprimir sus salidas en formato HDF5 para luego poder ser procesado según lo que requiera el usuario.

- **Libmeep-openmpi2**

  Librería que permite a Meep resolver problemas FDTD en forma paralela usando OPENMPI.

Para la instalación de MEEP-OPENMPI utilizamos el paquete que ofrece Python-Meep. A continuación detallaremos la instalación que se realizó:

- Primero debemos agregar en el repositorio de instalación de Ubuntu dos nuevas direcciones de dónde se descargará el paquete Meep-OpenMPI, se modificó el archivo de repositorios que se encuentra en***"/etc/apt/source.list"*** con algún editor y al final agregamos:

```
deb-src http://ppa.launchpad.net/python-meep/ppa/ubuntu jaunty main
deb http://ppa.launchpad.net/python-meep/ppa/ubuntu jaunty main
```

- Luego actualizamos el repositorio de UBUNTU de la siguiente manera:

```
root@tmp:~# apt-get update
```

- Una vez terminada la actualización del repositorio procedimos a instalar dos paquetes adicionales para Meep.

```
root@tmp:~# apt-get –force-yes install liblapack3gf
libblas3gf
```

- Al final procederemos a instalar el paquete MEEP-OPENMPI con el comando:

```
root@tmp:~#apt-get –force-yes install meep-openmpi
```

4.2.3. Instalación de Apache

Este servidor contiene la aplicación web de Ganglia para monitorear los recursos de hardware de los nodos que estén ejecutándose.

Para su instalación debemos de realizar lo siguiente:

- Primero instalamos PHP versión 5 con sus respectivos módulos para que Apache pueda soportar aplicaciones escritas en este lenguaje. En el terminal debemos ejecutar lo siguiente:

```
root@tmp:~#apt-get –force-yes php5 libapache2-mod-php5
```

- Luego se realizó la instalación del Apache al ejecutar:

```
root@tmp:~#apt-get –force-yes install apache2
```

- Por lo general si se desea realizar algún cambio en la configuración del servidor web lo puede realizar en **"*/etc/apache2/apache2.conf*"**. Nosotros dejamos la configuración por defecto.

4.2.4. Instalación de Apache-Tomcat.

El servidor apache-tomcat permite soportar aplicaciones web que están escritas en Java. Este permite contener la aplicación StarMeep que ayuda con la administración y ejecución de trabajos FDTD.

La instalación contiene los siguientes procedimientos:

- Descargamos el instalador del apache-tomcat desde la página oficial, abrimos el terminal y ejecutamos:

```
root@tmp:~#wget
http://www.poolsaboveground.com/apache/tomcat/tomcat-
6/v6.0.26/src/apache-tomcat-6.0.26-src.tar.gz
```

- Luego descomprimimos el paquete descargado para generar el directorio del apache tomcat a través del siguiente comando

```
root@tmp:~#tar -xvf apache-tomcat-6.0.26.tar.gz
```

- Después movimos este directorio generado al directorio de instalaciones. En este caso nosotros elegimos colocarla en ***"/usr/local"***.

```
root@tmp:~#mv apache-tomcat-6.0.26 /usr/local/
```

- Al final se configuró un archivo ejecutable **"*tomcat.sh*"** que permite levantar o parar el servidor y se colocó en **"*/etc/init.d/*"**de manera que pueda levantarse el apache-tomcat cada vez que se inicie un nodo.

4.2.5. Instalación y Configuración de Ganglia

Para la instalación y configuración del Monitor Ganglia procedimos a realizar los siguientes pasos.

- Primero instalamos las dependencias del paquete que permitirán el correcto funcionamiento Ganglia, en un terminal ejecutamos

```
root@tmp:~#apt-get –force-yes install build-essential
          librrd2-dev libapr1-dev  libconfuse-dev
          libexpat1-dev  python-dev  rrdtool
```

- Una vez instaladas las dependencias, nos descargamos el instalador de ganglia de la página oficial, entonces ejecutamos:

```
root@tmp:~#wget
http://sourceforge.net/projects/ganglia/files/ganglia%20
monitoring%20core/3.1.7/ganglia-3.1.7.tar.gz/download
```

- Después de descargar el instalador observamos que estaba comprimido, entonces procedimos a la descompresión ejecutando:

```
root@tmp:~# tar xzf ganglia-3.1.7.tar.gz
```

- Al descomprimir se generó automáticamente un directorio con el nombre de ***"ganglia-3.1.7"***, nos cambiamos al mismo y ejecutamos el comando para configurar un instalador.

```
root@tmp:~# cd ganglia-3.1.7/
root@tmp:~# ./configure --with-gmetad –
sysconfdir=/etc/ganglia
```

- Podemos observar que al configurar el instalador del Ganglia le enviamos como parámetros ***"–with-gmetad"*** esto permite que se instale el demonio gmetad adicionalmente. También enviamos como parámetro la dirección en dónde queremos que se cree el directorio de instalación, en este caso ***"/etc/ganglia"***.

  Ahora para generar el instalador con los parámetros configurados y ejecutar el mismo, realizamos lo siguiente:

```
root@tmp:~# make
root@tmp:~# make install
```

- Adicionalmente ganglia provee también una aplicación para monitorear recursos desde el web. Para esto, debemos mover el directorio "web" del Paquete del Ganglia hacia el directorio ***"/var/www"*** del Servidor Apache. Luego simplemente cambiamos el nombre de la carpeta de web a ganglia para que podamos llamar a la aplicación con ese nombre. Ejecutamos los siguientes comandos para realizar estas acciones:

```
root@tmp:~# cp -r web/ /var/www/
root@tmp:~# mv /var/www/web /var/www/ganglia
```

- Una vez terminada la instalación realizamos las configuraciones del los demonios gmetad y gmond. Para el gmond generamos el archivo de configuración por defecto con el siguiente comando:

```
root@tmp:~# sudo gmond --default_config >
/etc/ganglia/gmond.conf
```

- Por recomendación solo se necesita cambiar la información en la sección de "*clúster*" en el archivo de configuración del gmond.

```
cluster {
  name = "StarMeep"
  owner = "FIEC-ESPOL"
  latlong = "unspecified"
  url = "www.fiec.espol.edu.ec"
}
```

- Ahora pasaremos a la configuración del demonio gmetad, también usamos una plantilla guía que está en el paquete de instalación de Ganglia y luego la movimos al directorio de instalación de Ganglia.

```
root@tmp:~# cp /root/ganglia-3.1.7/gmetad/gmetad.conf
/etc/ganglia/gmetad.conf
```

- En el archivo de configuración del gmetad es dónde agregamos el nombre de aquellos nodos que deseamos monitorear. Este archivo es editado por la Aplicación Web StarMeep, ya que, éste mismo administra el levantamiento y ejecución de los nodos.
- Por último tuvimos que definir el directorio dónde se van a crear los archivos RRD que son necesarios para el Monitor Web del ganglia además de cambiar el propietario del usuario a "ganglia" para que los pueda leer.

```
sudo mkdir -p /var/lib/ganglia/rrds/
sudo chown -R ganglia:ganglia /var/lib/ganglia/rrds/
```

CAPÍTULO 5

5. Pruebas y Análisis de Resultados

5.1. Prueba de Eficacia

Para realizar esta prueba hemos tomado un ejemplo de un problema FDTD para ser resuelto con Meep. Al ejecutarlo en una sola máquina nos muestra el siguiente error:

```
Initializing structure...
Working in 3D dimensions.
terminate called after throwing an instance of
'std::bad_alloc'
```

Esto ocurre cuando necesitamos resolver un problema con un alto grado de procesamiento, como estructuras de gran tamaño o de imágenes con mayor resolución que implica gran uso de memoria. Esto limita mucho a los usuarios puesto que siempre depende de la característica, velocidad y memoria de una sola máquina.

Gracias a nuestra herramienta, estos tipos de problemas son resueltos debido a que al dividir el problema en pequeños trabajos entre los nodos, minimiza el uso de procesamiento y memoria que se requiere.

5.2. Pruebas de eficiencia

Los problemas que se detallan a continuación fueron resueltos sobre la plataforma de Amazon EC2, usando diferentes números de nodos, de esta manera vamos a demostrar la mejora que ofrece nuestra herramienta.

Hemos probado tres diferentes tipos de problemas que detallamos a continuación:

5.2.1. Resonador de anillo

Una de las tareas comunes que se realiza en las simulaciones FDTD es examinar el comportamiento del campo electromagnético de un objeto cuando es afectado por una fuente de energía. Al resolver estos tipos de problemas se crea datos de salida con los que se realiza un post procesamiento para generar un gráfico dónde podemos ver cómo se comporta el campo magnético.

El objetivo del siguiente ejercicio es realizar lo anteriormente mencionado en un "**resonador de anillo"** que no es más que una guía de onda doblada de manera circular. El resultado del mismo es una imagen animada dónde se puede apreciar las resonancias del campo electromagnético.

El código del ejercicio lo podemos consultar en el **anexo a1.1.**

La estructura del ejemplo de Resonador de anillos está dada por los siguientes pasos:

- Definir los parámetros que se van a utilizar en el problema
- Dibujar la geometría del problema, es decir el anillo el cuál se hace resonar, para esto se realizó dos objetos cilindros uno definido como material dieléctrico y el otro definido de material aire de manera que formen un anillo.
- Definir el pulso Gaussiano que permitirá resonar el anillo y con esto alterar su campo electromagnético.
- Por último ejecutamos la simulación, la idea básica del problema es ejecutar hasta que las fuentes hayan terminado y luego adicionar un tiempo desde el cuál vamos a calcular las señales y a obtener imágenes de cómo se comporta el campo electromagnético.

La **Tabla 1.1** muestra los datos de la estructura Geométrica.

Índice de Guía de Onda	3.4
Ancho de Guía de Onda (micrón)	1
Radio interior del anillo (micrón)	1
Espacio entre la Guía de Onda y la capa Pml (micrón)	4
Grosor del PML (micrón)	2
Ancho del Pulso	0,15
Frecuencia del Pulso	0,1
Resolución	40

Tabla 1:Datos de la Estructura Geométrica de Resonador de Anillo

Una vez finalizada la ejecución del problema el resultado obtenido es procesado a través de los siguientes comandos:

```
root@tmp:~#h5topng -RZc dkbluered -C ring-eps-
000000.00.h5
root@tmp:~#ring-ez-*.h5convert ring-ez-*.png ring-ez-
0.118.gif
```

Al ejecutar con diferentes números de nodos el problema se obtuvo cinco resultados, se realizó la respectiva comparación para verificar la confiablidad de los mismos y se comprobó efectivamente que todos generaban exactamente lo mismo. De esta manera se da por hecho que no importa el número de nodos a utilizarse siempre el resultado es el mismo.

La **Figura 7** muestra el resultado final del ejercicio:

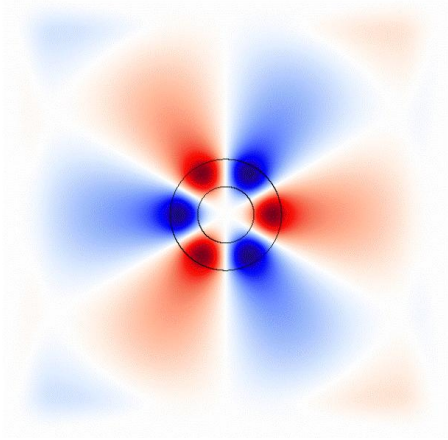


Figura 7: Resultado del post-procesamiento del Problema de Resonancia de Anillo (figura animada .gif)

Lo siguiente que realizaremos es observar el tiempo que se llevó por cada nodo completar el ejercicio, para esto se realizó el gráfico siguiente:

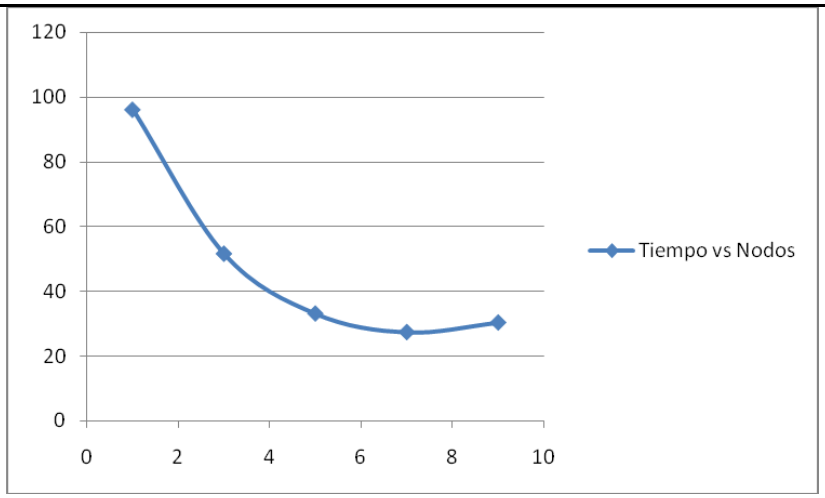


Figura 8: Gráfico de Nodos vs Tiempo (Minutos) del Ejercicio de Resonancia de Anillo

Como podemos observar en la **Figura 8** hemos ejecutado el problema cinco veces con diferentes números de nodos y podemos darnos cuenta de la mejora en tiempo que existe cuando resuelve el ejercicio con mayor número de nodos. Esto sucede hasta cierto punto, en este caso 9 nodos que el tiempo vuelve a subir.

5.2.2. Simulación 3D de un Sistema de Anillo Óptico Resonante para obtener el espectro de transmisión.

Otro de los problemas más comunes que se realizan para simulaciones FDTD es estudiar el espectro de transmisión del flujo electromagnético.

En el siguiente ejercicio mostraremos cómo se calcula la resonancia mediante el espectro de transmisión en un sistema de dos guías de ondas y un anillo.

Una guía es afectada por una fuente de energía y éste transmite energía hacia el anillo debido a la resonancia, que a su vez permite transmitir energía hacia la otra guía de onda.

Para realizar esto hay que ejecutar el ejercicio dos veces, la primera vez se lo realiza con presencia del anillo y la segunda vez sin el anillo. Luego se procede a realizar el procesamiento de los resultados.

La **Figura 9** ilustra el sistema anteriormente mencionado.

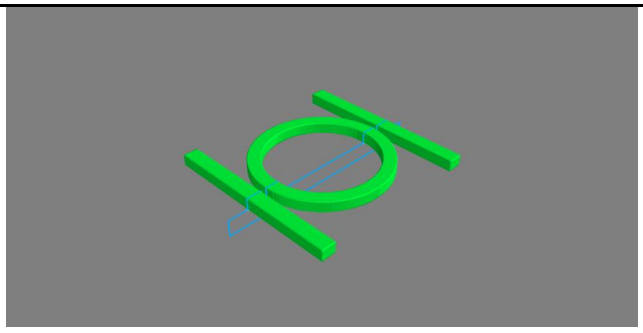


Figura 9: Gráfico de la Simulación de Anillo de Transmisión

El resultado del problema son tres gráficos que muestra el comportamiento del espectro de transmisión electromagnético.

El código del ejercicio lo podemos consultar en el **anexo a1.2.**

La estructura del código del ejercicio está dada por lo siguiente:

- Definir los parámetros o variables que se van a utilizar en el problema
- Dibujar la geometría del problema, es decir el anillo el cuál se hace resonar, para esto se realizó dos guías de ondas definido como material dieléctrico y un anillo resonador dieléctrico en caso de que se lo requiera.
- Definir la fuente de pulso Gaussiana que permitirá que el anillo entrar en resonancia.
- Escribir la condición de salida en este caso cuando el pulso se apague y esperando 150 intervalos de tiempo, para luego definir lo que queremos en este caso el flujo de espectro.

La **Tabla 2** muestra los datos de la estructura Geométrica.

Índice de Refracción	3.03
Índice de Refracción del Sustrato	1.67
Radio interior del anillo (micrones)	2
Radio exterior del anillo (micrones)	2.5
Ancho de la Guía de Onda (micrones)	0.55
Largo de la Guía de Onda (micrones)	0.405
Espacio entre la Guías de Onda y el anillo (micrones)	0.2
Distancia del Sustrato con respecto a la guía (micrones)	0.75
Ancho del Sustrato (micrones)	0.6
Grosor del PML (micrones)	1

Tabla 2: Datos de la Estructura Geométrica Anillo Óptico Resonador

Como habíamos mencionado se debe ejecutar dos veces el problema, una vez realizado esto cada uno genera un archivo de salida, tomamos todos los resultados que contengan estas líneas que se muestran a continuación:

```
flux1:; 0,36; 1,87190375615421e-8; 5,33633135330676e-9;
2,25302194395918e-10

flux1:; 0,360200100050025; 1,91956223787719e-8;
5,74839347857691e-9; 2,32817150153588e-10
```

Luego tabulamos los resultados que genera el problema para el post-procesamiento, como se muestra en la **Tabla 3**.

	FLUJO ESPECTRAL		
Frecuencia	**Puerto de Entrada**	**Puerto de Paso**	**Puerto de Extracción**
0,36	1,87190375615421e-8	5,33633135330676e-9	2,25302194395918e-10
0,360200100050025	1,91956223787719e-8	5,74839347857691e-9	2,32817150153588e-10

Tabla 3: Ejemplo de la Tabulación de los Resultados de Anillo Óptico Resonador

Como son dos ejecuciones al final vamos obtener dos tablas como la mostrada anteriormente, lo que hicimos luego fue dividir los datos correspondientes de cada columna excepto la de frecuencia entre el resultado de la ejecución con anillo para el resultado de la ejecución sin anillo.

Al final obtenemos una sola tabla con el cuál procederemos a graficar la Frecuencia contra cada flujo espectral como se muestra a continuación.

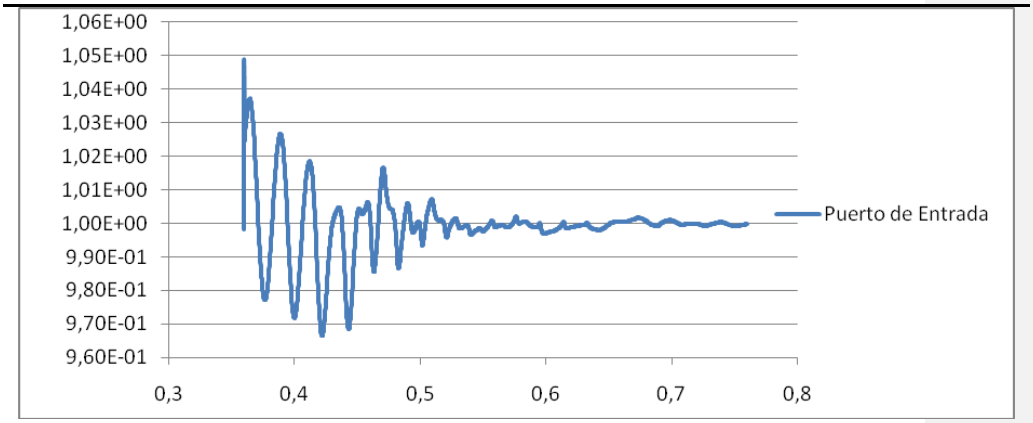


Figura 10: Gráfico Frecuencia vs Espectro(Entrada)

La **Figura 10** podemos observar el comportamiento del espectro de transmisión entre la fuente y la primera guía de onda. El gráfico indica que la mayor parte de fluctuaciones se da alrededor del valor unitario entre 0,3 y 0,5.

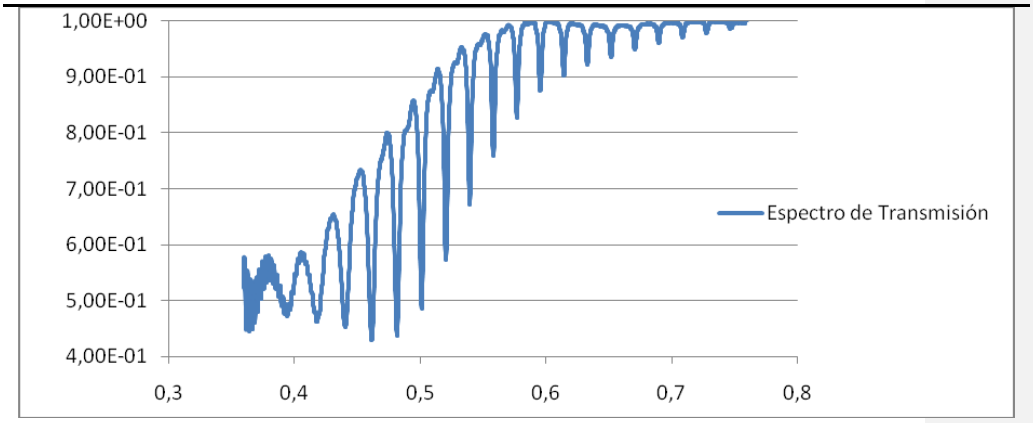


Figura 11: Gráfico Frecuencia vs Espectro(Paso)

La **Figura 11** podemos observar el comportamiento del espectro durante el paso entre la primera guía de onda y el anillo, los cuáles los valles que podemos observar son debido a la resonancia del anillo, debemos tomar en cuenta los errores de cálculos, para este caso, entre el rango de 0.3 y 0.4, podemos ver que aparecen varias fluctuaciones en la curva de respuesta.

Comentario [r1]: Aparecen varias fluctuaciones en la curva de respuesta.

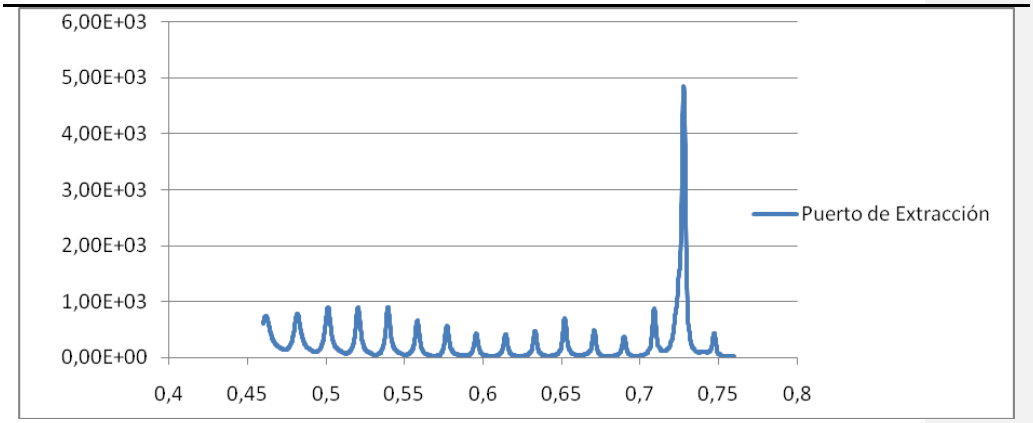


Figura 12: Gráfico Frecuencia vs Espectro(Extracción)

La **Figura 12** podemos observar el comportamiento del espectro durante la extracción de energía. Debido a que la fuente al final no genera energía, la misma comienza a ser extraída del anillo.

Al igual que el problema anterior al ejecutar con diferentes números de nodos el problema, se obtuvo cinco resultados, los cuáles todos generaron exactamente los mismos valores.

Finalmente veremos los tiempos en que se completó el problema, como la ejecución es en dos pasos, obtendremos dos gráficas mostradas a continuación.

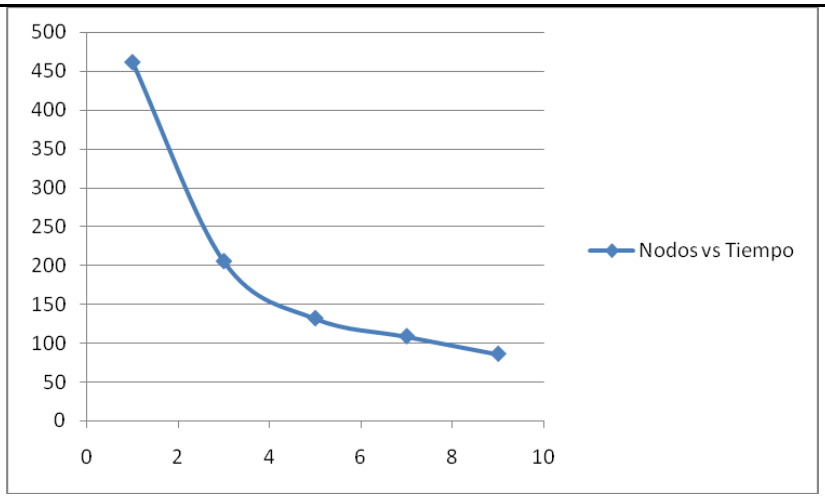


Figura 13*: Gráfico Estadístico de Nodos vs Tiempo (Minutos) del Ejercicio de Transmisión con Anillo*

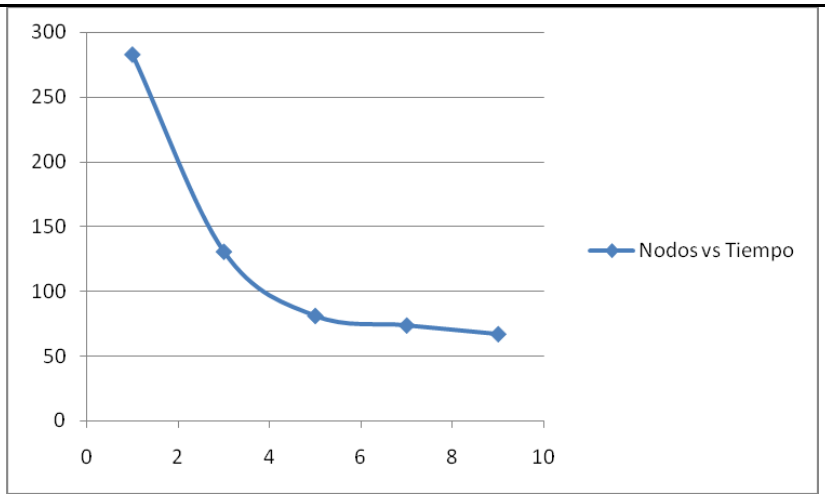


Figura 14*:Gráfico Nodos vs Tiempo (Minutos) del Ejercicio de Transmisión sin Anillo*

Como podemos observar en la ***Figura 13 y 14*** hemos ejecutado cada problema cinco veces con diferentes números de nodos, podemos denotar primero que la ejecución del problema con el anillo demora un poco más que la ejecución del problema sin anillo, sin embargo, lo interesante radica en que la proporción en que disminuye el tiempo en ambos ejercicios al ejecutarlo con más nodos son muy aproximados.

Generar la salida del Campo Electromagnético

Si queremos adicionalmente observar el comportamiento del campo electromagnético tal como se lo hizo en el primer ejercicio durante toda la simulación, se lo puede realizar simplemente agregando la siguiente línea

(to-appended "ez" (at-every 0.5 output-hfield-z))

Esto quiere decir que va imprimir el campo electromagnético Ez cada 0.5 unidades de tiempo

Ahora veamos el tiempo en que se demora cuando lo simulamos en varios nodos

.

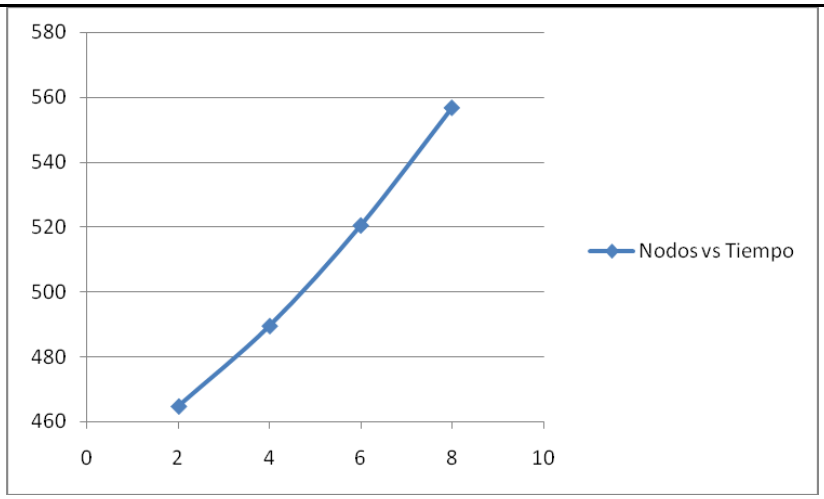


Figura 15*: Gráfico Nodos vs Tiempo (Minutos) del Ejercicio de Transmisión con Anillo generando el campo electromagnético durante toda la simulación*

Como podemos ver en la **Figura 15**, pasa algo diferente a los anteriores casos, en lugar de disminuir el tiempo con el uso de más nodos, el mismo aumenta. ¿A qué se debe este comportamiento?

Esto se debe a que este problema en especial genera muchos datos (20 GB); pues al realizar los nodos la escritura en disco de lo que ocurre en la simulación, este demanda tiempo, debido al lento acceso que tienen los nodos al disco duro.

Para demostrar esto vamos a consultar el archivo de salida del ejercicio, el mismo se muestra a continuación.

```
on time step 1 (time=0.0125), 54.5664 s/step
on time step 7 (time=0.0875), 0.686385 s/step
on time step 13 (time=0.1625), 0.672188 s/step
on time step 20 (time=0.25), 0.65516 s/step
on time step 27 (time=0.3375), 0.655888 s/step
on time step 34 (time=0.425), 0.658867 s/step
on time step 40 (time=0.5), 1.42186 s/step
on time step 41 (time=0.5125), 114.796 s/step
on time step 48 (time=0.6), 0.652713 s/step
on time step 55 (time=0.6875), 0.65162 s/step
on time step 61 (time=0.7625), 0.69854 s/step
on time step 68 (time=0.85), 0.646802 s/step
on time step 75 (time=0.9375), 0.658464 s/step
on time step 80 (time=1), 1.59971 s/step
on time step 81 (time=1.0125), 110.083 s/step
```

Si analizamos esto, las líneas marcadas indican los pasos que demandan más tiempo en el problema, y si recordamos nosotros configuramos el mismo para que genere la salida de lo que ocurre en la simulación cada 0.5unidades de tiempo. Si observamos el time justamente es el paso después de 0.5, y eso se repite en todo el archivo. Por lo tanto ese paso corresponde a la escritura en disco que hacen los nodos. En los problemas anteriores todo el proceso era cargado en Memoria RAM haciendo más eficiente la simulación

5.2.3. Cálculo de Frecuencia de Resonancia y factor de Calidad de un anillo resonante en 3D

Luego de haber visto cómo se comporta el campo electromagnético a través de un gráfico y calcular el espectro de transmisión para encontrar las resonancias, es importante también encontrar valores numéricos para identificar las frecuencias y las tasas de atenuación, así como el factor de calidad, esto lo podemos realizar a través del método de inversión armónica utilizando el paquete Harminv.

El objetivo del siguiente ejercicio es procesar las señales electromagnéticas para calcular las frecuencias y factor de calidad en el sistema de resonancia usado anteriormente.

Para este problema también estudiaremos los dos casos existentes: el sistema con anillo y el sistema sin anillo.

El resultado del problema muestra varios valores que nos ayudan a indicar la resonancia, la frecuencia, Q factor de calidad, amplitudes y el margen de error.

El código del ejercicio lo podemos consultar en el **anexo a1.3.**

La estructura del código y los datos son los mismos que el ejercicio anterior, lo único que variamos es que al final en lugar de obtener el flujo de espectros, vamos a procesar las señales, para ello Harminv recibe cuatro parámetros que son el Componente de Campo Eléctrico Ez, la posición donde vamos analizar, y el rango de frecuencias.

En ambos caso del problema (con anillo y sin anillo), al terminar de ejecutarse genera un archivo de salida dónde muestra todos los pasos realizados en cuanto al procesamiento y el resultado final (los valores de Harminv). Al final se tabula el resultado final**, la Tabla 4**, muestra cómo queda los datos tabulados, esto se realiza con el resultado en ambas ejecuciones.

	Freq. Real	**Freq. Imaginaria**	**Q**	**|Amp|**	**Amplitud**	**Error**
harminv0:	0,4621	1,76E-04	-1329,10	0,002	-0,0095-0,0012i	2,69E-04
harminv0:	0,4935	-0,0016	149,42	0,049	0,017+0,04874i	3,63E-05
harminv0:	0,5065	-5,20E-04	490,13	0,065	-0,037-0,05496i	1,40E-05
harminv0:	0,5189	-0,0027	94,93	0,059	0,0519+0,013851i	1,15E-04
harminv0:	0,5225	-3,66E-04	723,34	0,134	0,06928+0,11025i	2,31E-05

Tabla 4: Ejemplo de la Tabulación de los Resultados de Harminv

Para procesar el resultado final de los datos tabulados, queremos encontrar las frecuencias dónde hay resonancia, pero no todos los valores deben ser considerados pues como podemos fijarnos existe un margen de error que se debe considerar. Para saber si los valores por cada frecuencia son correctos se realiza una comparación. La frecuencia imaginaria en valor absoluto debe ser mayor al margen de error. En la **Tabla 5** señalamos cuáles son los candidatos a ser valores correctos.

	Freq, Real	Freq. Imaginaria	Q	|Amp|	Amplitud	Error
harminv0:	0,46	1,76E-04	-1329,10	0,002	-0,0095-0,0012i	2,69E-04
harminv0:	0,49	-0,0016	149,42	0,049	0,017+0,04874i	3,63E-05
harminv0:	0,50	-5,20E-04	490,13	0,065	-0,037-0,05496i	1,40E-05
harminv0:	0,51	-0,0027	94,93	0,059	0,0519+0,013851i	1,15E-04
harminv0:	0,52	-3,66E-04	723,34	0,134	0,06928+0,11025i	2,31E-05

Tabla 5: Ejemplo de la Tabulación de los Resultados de Harminv

Comentario [r2]: Ingresar los valores con 4 dígitos significativos

Revisando los resultados que generaron los problemas de acuerdo al número de nodos usados, se puede verificar que los valores obtenidos son los mismos.

Por último observemos los tiempos en que se completó el problema, como el mismo consta de dos partes (una con anillo y otra sin anillo), obtendremos dos gráficas.

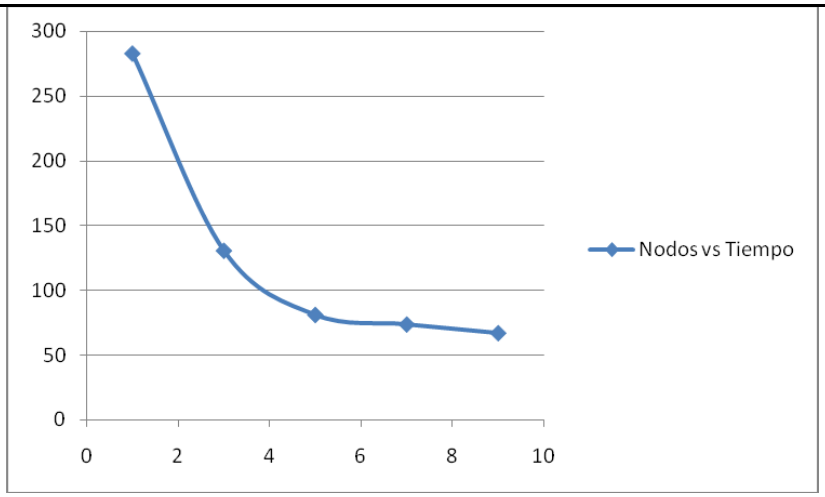


Figura 16*: Gráfico de Nodos vs Tiempo (Minutos) del Ejercicio de Transmisión con Anillo usando Harminv*

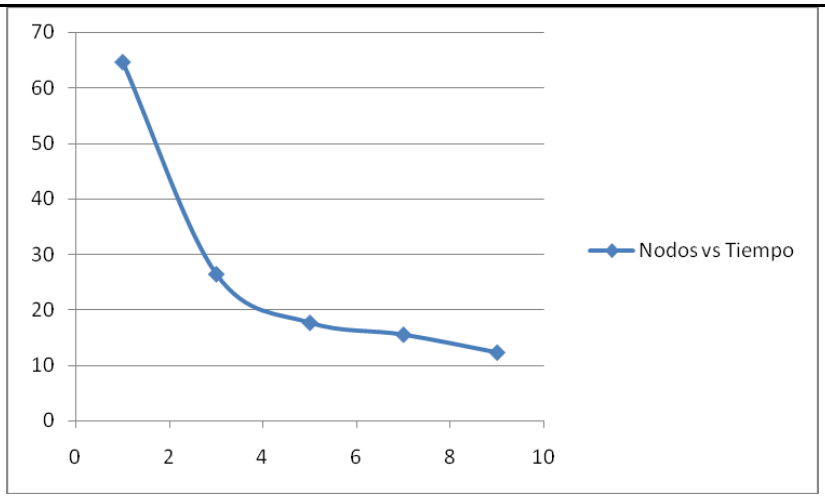


Figura 17*: Gráfico Nodos vs Tiempo (Minutos) del Ejercicio de Transmisión sin Anillo usando Harminv*

Como podemos observar en la ***Figura 16 y Figura 17*** al ejecutar cada problema cinco veces con varios números de nodos, notamos que disminuye el tiempo en cada uno de los casos a medida que usamos más nodos.

5.3. Análisis de los resultados

En cada gráfico se puede observar el comportamiento del rendimiento de StarMeep bajo tres condiciones de recursos. De manera general, se observa que a mayor número de nodos, se tiene un menor tiempo de procesamiento. Sin embargo, el correcto aprovechamiento del número de nodos depende del tipo de problema que se resuelve y los archivos de salida que se generan como resultado.

En el caso del tipo de problema existen varios factores que influyen de los cuáles los más importantes tenemos la resolución con la que se desee que se generen los archivos de salida y el tamaño de la estructura que se desea realizar la simulación.

Mientras que para los archivos de salida, influye la cantidad de información en gigabytes, que el problema requiera, debido a que la escritura en disco por lo general demanda mucho tiempo.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1.-Debido a la fácil administración que se provee, usuarios pueden levantar grandes clústeres computacionales en poco tiempo sin la necesidad de ser un especialista en el manejo de éstos.

2.-Gracias a los servicios que ofrece EC2 y a la creación de esta herramienta presentada en este proyecto, usuarios que necesiten resolver grandes problemas FDTD tienen fácil acceso a clústeres computacionales con mínimos costos.

3.-Debido a que nuestra herramienta soporta la asignación de recursos de procesamiento de acuerdo al problema, facilita a que los usuarios pueden resolver las mismas, con diferentes valores, sin la necesidad de esperar que termine algún otro problema que se esté ejecutando.

4.-Con el Monitoreo de Recursos se puede verificar el estado de cada nodo del clúster, así, el usuario puede conocer el uso de procesamiento y memoria además de si existe alguna clase de sobrecarga que pueda conllevar algún error.

RECOMENDACIONES

1.-El sistema mejora notablemente los tiempos utilizando instancias High-CPU, pero el rendimiento puede ser aun mejor con el anuncio de Amazon de un nuevo tipo de instancia High Performance Computing (HPC), que de acuerdo a las especificaciones ha sido diseñada para el procesamiento de grandes cantidades de cálculos. Y también destaca el alto rendimiento en red por su baja latencia, lo cual es un gran beneficio para aplicaciones que utilizan el protocolo de paso de mensajes MPI.

2.-Actualmente la escritura de los resultados se los realiza con la librería HDF5, hemos demostrado que esto crea un retardo en comparación al tiempo de procesamiento, y se acentúa más cuando en la simulación se escribe continuamente. Una posible solución para este problema es compilar desde el código fuente más sus dependencias la librería HDF5 paralela, por lo tanto es posible que si esta versión es mejorada podría mejorar ese tiempo de retardo en la escritura

3.-El paquete de Meep ofrece una librería hdf5utils para realizar post-procesamiento a los archivos de salida pero tiene sus limitaciones y descargar los resultados puede ser más dificultoso debido al gran tamaño de algunas simulaciones. Por lo cual una herramienta más completa seria lo mas óptimo tales como Octave, que es de software libre, aunque también se puede optar por opciones comerciales como Matlab.

Anexo a1

Resonancia de un Anillo óptico (http://ab-initio.mit.edu/wiki/index.php)

```
; ; Calculating 2d ring-resonator modes, from the Meep tutorial.

(define-param n 8.4) ; index of waveguide
(define-param w 6) ; width of waveguide
(define-param r 6) ; inner radius of ring

(define-param pad 4) ; padding between waveguide and edge of PML
(define-param dpml 2) ; thickness of PML

(define sxy (* 2 (+ r w pad dpml))) ; cell size
(set! geometry-lattice (make lattice (size sxy sxy no-size)))

; Create a ring waveguide by two overlapping cylinders - later objects
; take precedence over earlier objects, so we put the outer cylinder first.
; and the inner (air) cylinder second.
(set! geometry (list
                (make cylinder (center 0 0) (height infinity)
                     (radius (+ r w)) (material (make dielectric (index n))))
                (make cylinder (center 0 0) (height infinity)
                     (radius r) (material air))))

(set! pml-layers (list (make pml (thickness dpml))))
(set-param! resolution 40)

; If we don't want to excite a specific mode symmetry, we can just
; put a single point source at some arbitrary place, pointing in some
; arbitrary direction.  We will only look for TM modes (E out of the plane).

(define-param fcen 0.15) ; pulse center frequency
(define-param df 0.1)  ; pulse width (in frequency)
(set! sources (list
           (make source
             (src (make gaussian-src (frequency fcen) (fwidth df)))
             (component Ez) (center (+ r 0.1) 0))))

; exploit the mirror symmetry in structure+source:
(set! symmetries (list (make mirror-sym (direction Y))))

(run-sources+ 300
              (at-beginning output-epsilon)
              (after-sources (harminv Ez (vector3 (+ r 0.1)) fcen df)))

; Output fields for one period at the end.  (If we output
; at a single time, we might accidentally catch the Ez field when it is
; almost zero and get a distorted view.)
(run-until (/ 1 fcen) (at-every (/ 1 fcen 20) output-efield-z))
```

Anexo a2

Simulación 3D de un Sistema de Transmisión de energía entre Guías de Ondas a través de un Anillo para calcular Flujos de Espectro. (Ing. German Vargas)

```
;Simulation of a 3D ring resonator in add-drop configuration
;OUTPUT: Transmission flux from thru port and drop port
;TM MODE

;General parameters
(define-param nc  3.03) ;core refractive index
(define-param ns  1.67) ;substrate refractive index
(define-param r1  2) ;inner radius of ring in microns
(define-param r2  2.5) ;outer radius of ring
(define-param w  0.55) ;waveguide width in microns
(define-param gap  0.2) ; gap between ring and straight waveguide in microns
(define-param pad  0.75) ;padding distance in microns
(define-param h 0.405) ; height of waveguides in microns
(define-param hs  0.6) ;height of substrate
(define-param dpml  1) ;pml thickness

;Cell structure definition
(define sx (* 2 (+ r2 gap w pad dpml))) ; X direction cell size
(define sy (* 2 (+ r2 pad ))) ; Y direction cell size
(define sz (* 2 (+ hs dpml))) ; Z direction cell size
(set! geometry-lattice (make lattice (size sx sy sz)))

;FLAG to denote waveguides only = true, complete structure = false
(define-param no-ring? true) ;default simulate waveguides only

;Construct waveguide and ring resonator
(define subs-width (* 2 (+ r2 gap w pad)))
(define subs-length (* 2 (+ r2 pad)))

(if no-ring?
  (begin
    (set! geometry (list
                ;substrate
                (make block
                    (center 0 0 (* -1 (/ hs 2))) ; center of substrate
                    (size  subs-width  subs-length hs)
                    (material (make dielectric (index ns))))
                ;waveguides
                (make block
                    (center (+ r2 gap (/ w 2)) 0 (/ h 2)) ;right waveguide
                    ;(size  w (+ subs-length dpml) h)
                    (size  w subs-length h)
                    (material (make dielectric (index nc))))
                (make block
                    (center  (* -1 (+ r2 gap (/ w 2))) 0 (/ h 2))
                    ;(size w (+ subs-length dpml) h)
                    (size  w subs-length h)
                    (material (make dielectric (index nc))))
```

```
                )))
  ;else condition
  (begin
    (set! geometry (list
                ;substrate
                (make block
                    (center 0 0 (* -1 (/ hs 2))) ; center of substrate
                    (size  subs-width  subs-length hs)
                    (material (make dielectric (index ns))))
                ;waveguides
                (make block
                    (center (+ r2 gap (/ w 2)) 0  (/ h 2)) ;right waveguide
                    ;(size  w (+ subs-length dpml)h h)
                    (size  w subs-length h)
                    (material (make dielectric (index nc))))
                (make block
                    (center  (* -1 (+ r2 gap (/ w 2))) 0 (/ h 2))
                    ;(size w (+ subs-length dpml) h)
                    (size  w subs-length h)
                    (material (make dielectric (index nc))))
                ;ring resonator structure here
                (make cylinder
                    (center 0 0 (/ h 2) ) (height h)
                    (radius r2) (material (make dielectric (index nc))))
                (make cylinder
                    (center 0 0 (/ h 2) ) (height h)
                    (radius r1) (material air))
                )))
  )

;PML setup
(set! pml-layers (list (make pml (thickness dpml))))
;RESOLUTION
(set-param! resolution 30) ; use resolution, 40 for a 3.8 GB Computer
;Source definition
(define-param fcen 0.56) ; 0.56 center frequency at 170 Thz
(define-param df 0.4) ; 0.4 pulse width 120 Thz
(define-param nfreq 2000) ; 2000 number of freq elements to compute flux
(set! sources (list
            (make source
                (src (make gaussian-src (frequency fcen) (fwidth df)))
                (component Ez) ; TM excitation
                (center (* -1 (+ r2 gap (/ w 2))) r2  (/ h 2))
                (size w 0 h )))) ;point source

;Define flux computation Region
(define transmission_in ;input transmission
 (add-flux fcen df nfreq
        (make flux-region
            (center (* -1 (+ r2 gap (/ w 2))) r1 (/ h 2))
            (size (* w 1.0) 0 (* h 1.0))
            (weight -1.0))))
(define transmission_thru ;transmitted flux thru port
 (add-flux fcen df nfreq
        (make flux-region
            (center (* -1 (+ r2 gap (/ w 2))) (* -1 r1) (/ h 2))
            (size (* w 1.0) 0 (* h 1.0))
            (weight -1.0)))) ; because flux is outward the waveguide (negative axis)
(define transmission_drop ;transmitted flux drop port
 (add-flux fcen df nfreq
```

```
        (make flux-region
             (center (* +1 (+ r2 gap (/ w 2))) r1 (/ h 2))
             (size (* w 1.0) 0 (* h 1.0))
             (weight +1.0)))) ; because flux is outward the waveguide (positive axis)

;Run command
(run-sources+
 (stop-when-fields-decayed 150 Ez
                      (vector3 (* -1 (+ r2 gap (/ w 2))) (* -1 r1) (/ h 2))
                      1e-4)
 (at-beginning output-epsilon)
 )
(display-fluxes transmission_in transmission_thru transmission_drop) ; print out flux spectrum
```

Anexo a3

Simulación 3D de un Sistema de Transmisión entre Guías de Ondas a través de un Anillo Resonante mediante Harminv
(Ing. German Vargas)

```
;Simulation of a 3D ring resonator in add-drop configuration
;OUTPUT: RESONANT MODES OBTAINED BY HARMINV
;General parameters
(define-param nc  3.03) ;core refractive index
(define-param ns  1.67) ;substrate refractive index
(define-param r1  2) ;inner radius of ring in microns
(define-param r2  2.5) ;outer radius of ring
(define-param w  0.55) ;waveguide width in microns
(define-param gap  0.2) ; gap between ring and straight waveguide in microns
(define-param pad  0.75) ;padding distance in microns
(define-param h 0.405) ; height of waveguides in microns
(define-param hs  0.6) ;height of substrate
(define-param dpml  1) ;pml thickness

;Cell structure definition
(define sx (* 2 (+ r2 gap w pad dpml))) ; X direction cell size
(define sy (* 2 (+ r2 pad dpml))) ; Y direction cell size
(define sz (* 2 (+ hs dpml))) ; Z direction cell size
(set! geometry-lattice (make lattice (size sx sy sz)))

;FLAG to denote complete structure = false, waveguide only = true
(define-param no-ring? false) ;default simulate complete ring + waveguides

;Construct waveguide and ring resonator
(define subs-width (* 2 (+ r2 gap w pad)))
(define subs-length (* 2 (+ r2 pad)))

(if no-ring?
   (begin
    (set! geometry (list
                ;substrate
                (make block
                     (center 0 0 (* -1 (/ hs 2))) ; center of substrate
                     (size subs-width subs-length hs) ; dimension of substrate is (wxLxh) =
        2*(r2+gap+w+pad) x 2*(r2+pad) x hs
                     (material (make dielectric (index ns))))
                ;waveguides
                 (make block
                      (center (+ r2 gap (/ w 2)) 0 (/ h 2)) ;right waveguide
                      (size  w subs-length h)
                      (material (make dielectric (index nc))))
                  (make block
                       (center  (* -1 (+ r2 gap (/ w 2))) 0 (/ h 2))
                       (size w subs-length h)
                       (material (make dielectric (index nc))))
                  )))
```

```
   ;else condition
   (begin
     (set! geometry (list
                  ;substrate
                  (make block
                       (center 0 0 (* -1 (/ hs 2))) ; center of substrate
                       (size subs-width subs-length hs) ; dimension of substrate is (wxLxh) =
          2*(r2+gap+w+pad) x 2*(r2+pad) x hs
                       (material (make dielectric (index ns))))
                  ;waveguides
                   (make block
                       (center (+ r2 gap (/ w 2)) 0  (/ h 2)) ;right waveguide
                       (size  w subs-length h)
                       (material (make dielectric (index nc))))
                    (make block
                       (center  (* -1 (+ r2 gap (/ w 2))) 0 (/ h 2))
                       (size w subs-length h)
                       (material (make dielectric (index nc))))
                    ;ring resonator structure here
                    (make cylinder
                       (center 0 0 (/ h 2) ) (height h)
                       (radius r2) (material (make dielectric (index nc))))
                    (make cylinder
                       (center 0 0 (/ h 2) ) (height h)
                       (radius r1) (material air))
                     )))
   )

;PML setup
(set! pml-layers (list (make pml (thickness dpml))))
;RESOLUTION
(set-param! resolution 20); NOTE: CHANGED RESOLUTION FROM 40 to 20 DUE TO LIMITED
          MEMORY RESOURCES

;Source definition
(define-param fcen 0.56) ;center frequency at 170 Thz
(define-param df 0.2) ; broadband pulse width 60 Thz

;Excite with a gaussian pulse broadband
(set! sources (list
             (make source
                  (src (make gaussian-src (frequency fcen) (fwidth df)))
                  (component Hz) ; TE excitation
                  (center (* -1 (+ r2 gap (/ w 2))) (+ r2 (/ pad 2)) (/ h 2))
                  (size 0 0 0 )))) ;point source

;Run command
(run-sources+ 200
        (at-beginning output-epsilon)
        (after-sources (harminv Hz(vector3 0 (+ r1 (/ w 2)) (/ h 2)) fcen df)))
```

REFERENCIAS BIBLIOGRÁFICAS

[1] Colaboradores de Wikipedia, Finite-difference time-domain method *Wikipedia, La enciclopedia libre*, http://en.wikipedia.org/w/index.php?title=Finite-difference_time-domain_method&oldid=370208130, [14 de Julio del 2010]

[2] Taflove Allen, Advances in Computational Electrodynamics – The Finite Difference Time Domain Method, Norwood, MA: Artech House Inc., 1998. 15 de Abril del 2010

[3] Mario Omar Calla Salcedo, FDTD al cálculo de Magnitudes Electromagnéticas, http://fittelecomunicaciones.blogspot.com/2009/09/fdtd-al-calculo-de-magnitudes.html, [fecha de consulta: 1 de julio del 2010]

[4] Colaboradores de Wikipedia. *Clúster* [en línea]. Wikipedia, La enciclopedia libre. http://es.wikipedia.org/w/index.php?title=Cluster_(inform%C3%A1tica)&oldid=38185002. 20 de junio del 2010

[5] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. *Above the Clouds: A Berkeley View of CloudComputing*. Technical Report No. UCB/EECS 2009-28, University of California at Berkley, USA. 10 de Febrero del 2009

[6] Dr. Oscar González Rodríguez. Extensión del Método de las Diferencias Finitas en el Dominio del Tiempo para el Estudio de Estructuras Híbridas de Microondas Incluyendo Circuitos Concentrados Activos y Pasivos.

Tesis Doctoral, Universidad de Cantabria, España, Septiembre, 2008.

[7] Blaise Barney, Lawrence Livermore, Introduction to Parallel Computing. From: https://computing.llnl.gov/tutorials/parallel_comp/, 15 de Julio del 2010

[8] Kevin Dowd, High performance computing, O'Reilly & Associates, Inc., Sebastopol, CA, 1993

[10] StarCluster. Massachusetts Institute of Technology. http://web.mit.edu/stardev/cluster/. 17 de Julio del 2010

[11] The Open MPI Project, Open Source High Performance Computing. http://www.open-mpi.org/, 2004. Marzo del 2010

[12] Parallel Meep. http://ab-initio.mit.edu/wiki/index.php/Parallel_Meep. Enero del 2010

[13] Google Web Toolkit. http://code.google.com/webtoolkit/. Mayo del 2010

[14] Brent Chun, Matt Massie. Ganglia Cluster Toolkit. http://ganglia.sourceforge.net/, Julio del 2010