Implementación de un Mini-Grid de Cómputo

Gabriela A. Gangotena, Marco D. Bonilla e Iván M. Bernal
Escuela Politécnica Nacional

Resumen*—El desarrollo del *middleware Globus Toolkit* que integra tecnologías como los sistemas distribuidos, el Internet, los servicios web, seguridad, la virtualización y las redes *peer-to-peer* ha permitido la conformación de *Grids*. Por otro lado, basándose en la tecnología de *clusters, en* Matlab se han desarrollado herramientas para computación distribuida como *Matlab Parallel Computing Toolbox* y *Matlab Distributed Computing Server.

Este artículo inicia con una visión general de la computación *Grid* y el uso del *middleware Globus Toolkit (GT),* el cual permite realizar la implementación de una infraestructura de mini-grid conformada por recursos heterogéneos y distribuidos en un ambiente de laboratorio. Entre los recursos heterogéneos, se emplean computadoras con Sistema Operativo Linux e integradas al mini-grid con los servicios que ofrece el *GT4*; por otra parte, un *cluster* construido en base a las herramientas de computación distribuida de Matlab en computadoras con Sistema Operativo Windows, se integra al mini-grid en base a un servicio web desarrollado en Java, cumpliendo con especificaciones de seguridad. Se recicla una aplicación de elementos finitos para el cálculo de la distribución de campo eléctrico en una guía de onda de longitud infinita y se la modifica en base a las herramientas de paralelización de Matlab para ser ejecutada en el *cluster.* El servicio web que invoca a la aplicación del *cluster* de Matlab es desplegado en Windows en el contenedor de *Globus Toolkit* para ser consumido en el mini-grid.

***Términos para indexación*—*cluster, Grid, Globus Toolkit,* Sistema Operativo.**

Este proyecto se realizó con el financiamiento de la Escuela Politécnica Nacional (EPN), en el Departamento de Electrónica, Telecomunicaciones y Redes de la Información (DETRI).

M. D. Bonilla participó en el proyecto por la Escuela Politécnica Nacional (e-mail: mbonilla2109@gmail.com).

G. A. Gangotena participó en el proyecto por la Escuela Politécnica Nacional (e-mail: ggangotena@gmail.com).

I. M. Bernal Profesor a tiempo completo de la Escuela Politécnica Nacional en el Departamento de Electrónica y Redes de la Información (e-mail: imbernal@mailfie.epn.edu.ec).

***Abstract*-** The development of the *middleware Globus Toolkit* that integrates technologies such as distributed systems, Internet, web services, security, virtualization and peer-to-peer has enabled the creation of Grids. On the other hand, based on clusters technologies, in Matlab distributed computing tools like *MATLAB Parallel Computing Toolbox* and *Matlab Distributed Computing Server* have been developed.

This paper begins with an overview of Grid computing and the use of *Globus Toolkit (GT) middleware*, which enables the implementation of a mini-grid infrastructure which consists of heterogeneous and distributed resources in a laboratory environment. Amongst the heterogeneous resources are computers running Linux and integrated into the mini-grid by using the services offered by GT4. Besides, a cluster built upon the distributed computing tools of Matlab running on PCs with the Windows OS, is integrated into the mini-grid by consuming a Java web service developed in compliance with safety specifications. An application of finite elements method for calculating the electric field distribution in a waveguide of infinite length is chosen and modified using the parallelization tools for Matlab and is run on the cluster. The Web service that invokes the application on the Matlab cluster is deployed in Windows in the Globus Toolkit container in order to be consumed in the mini-grid.

I. Introducción a la Computación *Grid*

Al conjunto de recursos computacionales heterogéneos y distribuidos, pertenecientes a diferentes organizaciones se le denomina *Grid*. Este conjunto de computadoras se conectan a través de una *WAN* o del Internet y las computadoras pueden encontrarse distribuidas geográficamente por todo el mundo [1].

En la Fig. 1 se muestra la topología de un sistema *Grid* típico, en la que se pueden distinguir los diferentes tipos de recursos heterogéneos distribuidos geográficamente e interconectados a través del Internet.

Fig. 1 Recursos distribuidos y heterogéneos.

La computación *Grid* se presenta como una colección de servicios comunes que proporcionan seguridad, administración de datos, programación de la ejecución de aplicaciones, notificación de eventos y monitoreo del sistema; actualmente estos servicios son implementados como servicios web y muchas de las aplicaciones de *Grids* pueden ser construidas por la composición de servicios básicos y componentes [4].

II. TIPOS DE *GRID*

Tomado de las referencias [1][2][3][4].

La computación *Grid* provee la integración de recursos computacionales heterogéneos de cualquier tipo: unidades de procesamiento, unidades de almacenamiento, unidades de comunicación.

A. Grid de Cómputo (Computational Grid)

Tiene la capacidad de procesamiento como el principal recurso para ser compartido entre los nodos. Es usado en computación de alto rendimiento para abordar procesamiento de tareas bajo demanda.

B. Grid de Datos (Data Grid)

Tiene la capacidad de almacenamiento de datos como el principal recurso para ser compartido entre los nodos. Es responsable por almacenar y proveer el acceso a los datos a través de múltiples organizaciones a los usuarios, siendo un proceso transparente para el usuario que solo accede a los datos sin tener conocimiento de la ubicación de los mismos.

C. Grid en Red (Network Grid)

En una red corporativa, cada máquina, servidor o computadora tiene subutilizado el ancho de banda, lo que puede ser considerado como un recurso disponible. Por ejemplo, un servidor de transferencia de archivos no permite más de dos o tres conexiones simultáneas desde el mismo solicitante; desde el punto de vista del servidor, para satisfacer las solicitudes de varios usuarios se limita el ancho de banda por conexión; desde el punto de vista del cliente, al tener una conexión limitada se desperdicia el recurso de ancho de banda al permanecer ocioso. En este ejemplo, el cliente podría aprovechar los recursos disponibles de ancho de banda con los que dispone y conectarse a otro servidor dentro de la misma infraestructura para simultáneamente descargarse otra porción del mismo archivo, lo que permitiría que el cliente optimice el uso de su interfaz de red aprovechando efectivamente el ancho de banda disponible.

Este tipo de *Grid* se aplica en el lado del servidor, ya que se requiere un duplicado de los archivos en cada servidor, teniendo a los servidores distribuidos geográficamente en un ambiente de área extendida.

D. Scavenging

Es comúnmente utilizado con gran número de computadoras personales cuando tienen ciclos de CPU y otros recursos disponibles; por ejemplo, durante la noche, las computadoras se encuentran sin utilizar, lo que permite que se integren al *Grid*, para esto los propietarios dan el control de estas máquinas cuando se encuentren libres. Las computadoras pueden integrarse a diferentes organizaciones virtuales aunque pertenezcan a una misma organización y de igual manera pueden integrarse a una organización virtual aunque pertenezcan a diferentes organizaciones.

E. Grid Multipropósito

Es la implementación más común en el futuro de la computación *Grid*. La infraestructura del *Grid* Multipropósito deberá ser lo suficientemente capaz de proveer cualquiera de los modelos de *Grid* presentados anteriormente. Puede ser implementado como un *meta-Grid* con la habilidad para dirigir las peticiones hacia el *Grid* que soporta el modelo adecuado.

III. ARQUITECTURA DE UN GRID

En la Fig. 2 se muestra el Modelo de Arquitectura de un *Grid* que está relacionado estrechamente con la Arquitectura de Protocolos de Internet. A continuación se describe brevemente cada una de las capas.

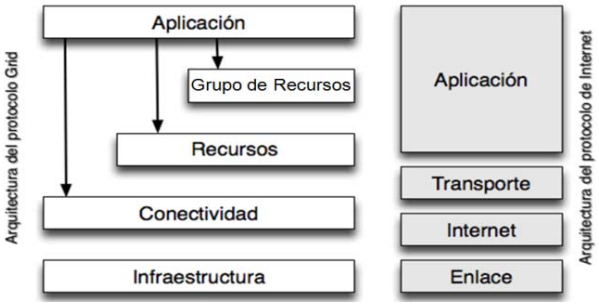

Fig. 1 Arquitectura de Protocolos de un *Grid.*

A. *Infraestructura: Interfaces para control local*

Se refiere a los recursos que en realidad van a ser compartidos mediante los protocolos del *Grid.* Un recurso puede ser una entidad lógica, como un sistema distribuido de archivos, una computadora clúster o un sistema de computación distribuida; en algunos casos, la implementación de un recurso involucra protocolos internos que no están comprendidos en el modelo de arquitectura *Grid* [3].

B. *Conectividad: Comunicación fácil y segura*

Los recursos computacionales necesitan ser capaces de comunicarse entre ellos. La capa de conectividad se refiere a todos los protocolos que permiten a los recursos comunicarse, fundamentalmente Protocolos de Internet como TCP/IP, HTTP, DNS, etc. Sin embargo, un Sistema *Grid* requiere de comunicaciones seguras, por lo que esta capa incluye protocolos que han surgido en respuesta a problemas de seguridad [3].

C. *Recursos: Compartición de Recursos Simples*

Se refiere a todos los servicios y protocolos que permiten la administración individual de los recursos. La administración debe incluir tareas como la inicialización de los recursos, monitoreo y registro. Esta capa no se preocupa de la interacción global entre los recursos [3].

D. *Grupo de Recursos: Coordinación de múltiples recursos*

Se refiere a los servicios y protocolos que tratan con la administración de múltiples recursos. Trabajando a partir de los servicios provistos en la Capa de Recursos, esta capa permite tomar una colección de recursos y hacer que estos trabajen juntos para resolver una tarea común [3].

E. *Aplicación*

La capa de aplicación se refiere a las aplicaciones que en realidad correrán en el Sistema *Grid*. Estas aplicaciones no requieren interactuar directamente con el servicio de Grupo de Recursos y están en libertad de acceder a los Servicios de Recursos y Conectividad directamente cuando lo requieran. Las aplicaciones son construidas en términos de servicios definidos en cualquier capa, debido a que en cada capa se tienen bien definidos los protocolos que facilitan el acceso a servicios útiles como gestión de recursos, acceso a los datos, descubrimiento de recursos, etc. En cada capa, las *APIs*[1] pueden estar definidas implementando protocolos de intercambio de mensajes con el servicio apropiado para realizar las acciones deseadas [3].

[1]*API (Application Programming Interface)*: Interfaz de programación de aplicaciones

IV. *GLOBUS TOOLKIT*

Globus Toolkit (GT) es una herramienta *middleware* para la construcción de *Grids* desarrollada por *Globus Alliance*, centrada en la simplificación y la heterogeneidad, mediante interfaces estándares para interactuar con diferentes sistemas sin perder la autonomía local [4].

En un sistema *Grid*, el *middleware* es usado para transparentar la naturaleza heterogénea y proveer a los usuarios y aplicaciones un ambiente homogéneo y sin vacíos, proporcionando un conjunto de interfaces normalizadas para una variedad de servicios.

GT4 está organizado como un grupo de componentes, que consiste de servicios, librerías de programación y herramientas de desarrollo para aplicaciones basadas en *Grid* [1][4].

A. *Seguridad*

Las herramientas de seguridad son las encargadas de establecer la identidad de los usuarios y los servicios, asegurar la integridad y privacidad de la comunicación, también genera *logs* para verificación de políticas de usuarios, así como de la autenticación.

Mediante la utilización de *Grid Security Infrastructure* (*GSI*), se realiza la comunicación segura y la aplicación de políticas uniformes en distintos sistemas, empleando el protocolo SSL *(Secure Socket Layer)*, encriptación de clave pública, certificados X.509[2] y certificados *proxy* [4].

B. *Administración de Datos*

Provee los mecanismos para descubrir, transferir y acceder a gran cantidad de datos [4].

C. *Administración de la Ejecución*

Despliega, planifica y monitorea programas ejecutables referidos como trabajos *(jobs*) [4].

D. *Servicios de Información*

Conocido como *Monitoring and Discovery System (MDS)*, monitorea y descubre recursos en una VO (organización virtual) mediante acceso a información dinámica y estática de los recursos, y permite el acceso a los datos históricos para la planificación futura [4].

E. *Entorno de Ejecución Común*

El contenedor provee el entorno para la gestión y *hosting* de servicios web, para lo cual *GT4* incorpora un conjunto fundamental de librerías y herramientas para el desarrollo de servicios web en lenguajes como *C, Phyton y Java*. El contenedor también conocido como "*standalone container*" está basado en *Apache Axis* [4].

[2]Certificado X.509, Dado por *Engineering Task Force (IETF)*: pueden ser compartidos con otros programas basados en clave pública incluyendo web *browsers de Microsoft* y *Netscape*.

V. IMPLEMENTACIÓN DEL MINI-GRID DE CÓMPUTO

En la Fig. 3, se muestra la infraestructura de red del mini-grid está basada en la serie de protocolos de TCP/IP y el *middleware Globus Toolkit 4*.

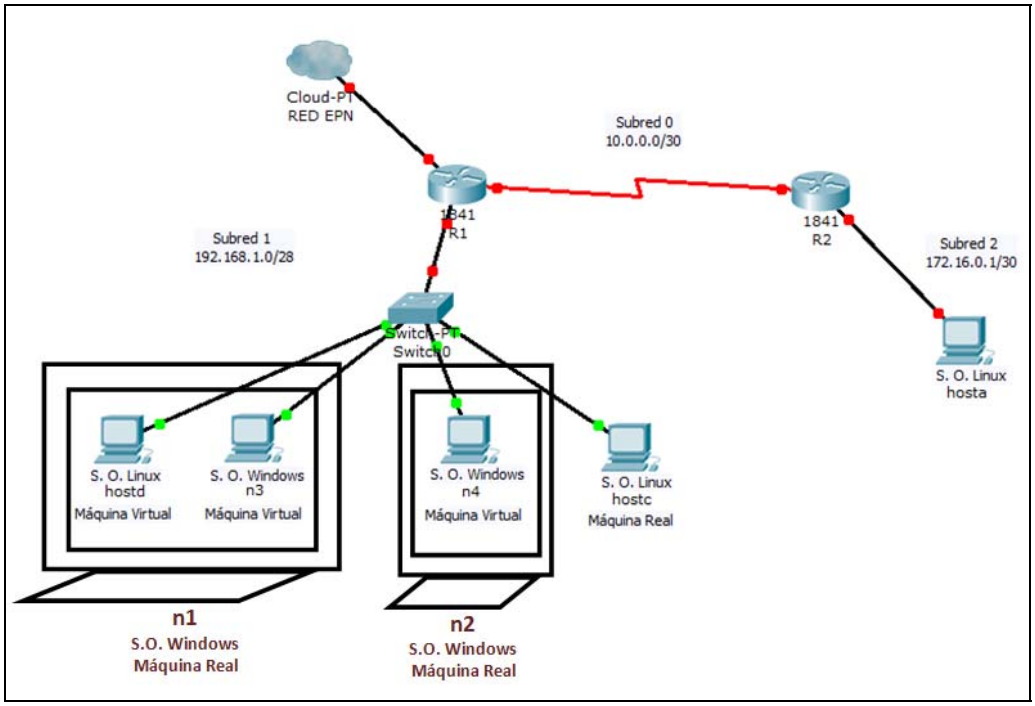

Fig. 3 Diagrama de Red del Mini-Grid.

A. *Planeamiento Físico*

Haciendo referencia a la Fig.3, el modelo implementado en un ambiente de laboratorio utiliza dos ruteadores para separar tres subredes. La subred 0, pretende simular un ambiente *WAN*, siendo el enlace entre ruteadores. La subred 1 y subred 2 son redes remotas conformadas por cuatro computadoras personales.

El diagrama de red de la Fig. 3 indica la posibilidad de comunicarse entre centros remotos a través del Internet con el uso de direcciones IP públicas y empleando TCP/IP y *Globus Toolkit* para el desarrollo de un sistema distribuido en un área geográfica global. En la Fig. 3 se puede observar las máquinas virtuales que se crearon en la máquina real n1 y n2.

B. *Esquema de Direccionamiento*

El esquema de direccionamiento se lo realiza utilizando direcciones IP estáticas para identificar a los recursos debido a que varios de los servicios son en base a la dirección IP.

C. *Enrutamiento*

Para compartir información y recursos, en un ambiente geográficamente distribuido, se requiere conectividad entre múltiples redes. Se utiliza enrutamiento estático para conectar las redes remotas.

D. *Seguridad en el mini-grid*

La seguridad en el mini-grid se implementa por medio de SimpleCA (Autoridad Certificadora) que es distribuida por el *Globus Toolkit*.

1) Seguridad en la Primera Máquina

Se configura el SimpleCA que genera un paquete que provee una autoridad certificadora simplificada en un nodo del mini-grid.

2) Seguridad en el resto de Máquinas

Para la configuración de seguridad en el resto de nodos del mini-grid se necesita de certificados de *host* por cada nodo (máquina) y certificados de *user* por cada usuario en el nodo. En cada nodo del mini-grid se debe instalar el paquete CA generado en la primera máquina, que es la Autoridad Certificadora.

VI. HERRAMIENTAS DE MATLAB PARA COMPUTACIÓN DISTRIBUIDA

A. *Matlab Parallel Computing Toolbox*

Parallel Computing Toolbox de Matlab proporciona las herramientas y el lenguaje necesario para la

programación de aplicaciones de forma paralela, así como los mecanismos para el envió de trabajos para su ejecución. La sesión de Matlab ejecuta la Máquina Virtual de Java *(JVM)* para correr el software de *Parallel Computing Toolbox* [5].

B. Matlab Distributed Computing Server (MDCS)

Servidor de MATLAB que permite distribuir el trabajo a varias computadoras, de esta forma optimizando el procesamiento de la información. Debe estar instalado en cada una de las máquinas que van a ejecutar el proceso computacional. Permite que los ejecutables y aplicaciones generadas con el lenguaje dado por el *Parallel Computing Toolbox* sean procesados en coordinación con un administrador de trabajos [5].

Para una sesión MDCS son necesarios los siguientes pasos:

- Creación del planificador (*scheduler)*
- Creación del trabajo (*job*)
- Creación de las tareas (*tasks*)
- Ejecutar el trabajo utilizando el administrador
- Obtención de resultados

C. Administrador de Trabajos

El administrador (*scheduler*) es el encargado de la interacción de los diferentes componentes del *cluster*. Se tiene a disposición tres diferentes tipos de administradores: Administrador Local, Administrador de Trabajos *(MathWorks Job Manager)* y Administrador de *third-party* [5].

VII. IMPLEMENTACIÓN DEL *CLUSTER* DE MATLAB

La implementación realizada está basada en la utilización del administrador de trabajos propio de Matlab (*MathWorks Job Manager*).

A. Ejecución de Trabajos en el cluster

El archivo principal (ver Fig.4) tiene la función de crear los elementos necesarios para ejecutar una aplicación que emplea el método de elementos finitos [4] para electromagnetismo, considerando una guía de onda con condiciones de borde.

Al momento de ejecutar el archivo principal *FEM2DL_Box_mdce.m* (Fig. 5), se invoca a la función denominada *calcula.m* que hace uso de MDCS mediante el uso de *Parallel Computing Toolbox*. Esta función es el cliente del *cluster* de Matlab debido a que se encarga de encontrar el administrador de trabajos, creación del objeto trabajo, creación de las tareas y envío del trabajo.

Se crea la función llamada *area.m* que se encarga del cálculo del campo eléctrico de cada elemento finito y es la tarea que va a ser distribuida entre los trabajadores del *cluster (Fig. 4).*

Finalmente, los trabajadores devuelven el resultado de sus tareas, con lo que a su vez se obtiene el resultado del objeto trabajo.

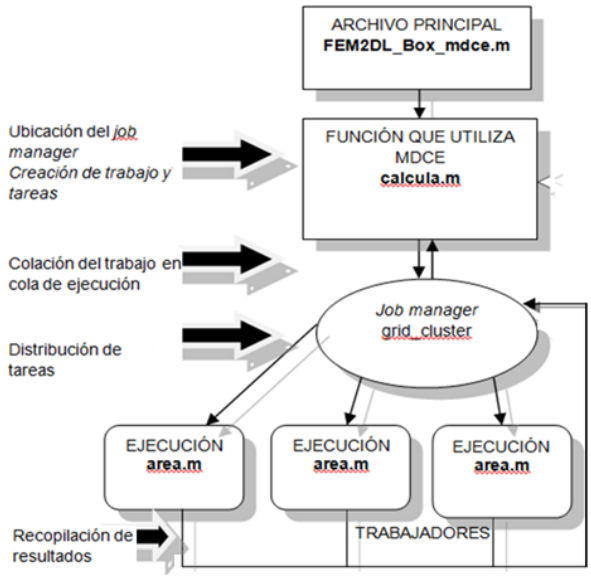

Fig. 4 Ejecución de trabajos en el *cluster*.

VIII. INTEGRACIÓN DEL *CLUSTER* AL MINI-GRID

Se despliega un servicio web simple, utilizando las herramientas del *Globus Toolkit* para la interacción entre Windows y Linux.

Los servicios web desplegados en el contenedor del *Globus Toolkit* permiten la interoperabilidad entre diferentes aplicaciones de software para la interacción máquina-máquina; en el caso del mini-grid permite la interoperabilidad entre una aplicación realizada en Matlab sobre una máquina con Sistema Operativo Windows y máquinas con Sistema Operativo Linux.

A continuación se presenta el procedimiento para desplegar el servicio web en el contenedor del mini-grid desde una máquina con sistema operativo Windows para permitir que cualquier máquina dentro del mini-grid lo consuma.

A. Definición de la Interfaz en *WSDL*

El primer paso antes de implementar un servicio web es definir la interfaz del servicio. *WSDL* (Lenguaje de Descripción de Servicios Web) es el lenguaje usado para especificar qué operaciones va a ofrecer el servicio web.

La interfaz Java requerida para la integración del *cluster* de Matlab al mini-grid especifica lo que el servicio va a ofrecer, es decir, se define el puerto de conexión del servicio web que contiene la función que

va a estar disponible para que los usuarios puedan invocar la aplicación del *cluster* [4].

B. Implementación del servicio en Java

Después de haber definido la interfaz del servicio, el siguiente paso es implementar esta interfaz, es decir, cómo el servicio va a hacer lo que dice que hace. La implementación del servicio web en Java consiste de una clase simple en Java con código para ambos: el servicio y el recurso [4].

C. Definición de los parámetros para el despliegue

El siguiente paso es tomar las piezas escritas anteriormente y hacerlas disponibles a través del contenedor de servicios web de *Globus*, lo que se conoce como el despliegue del servicio web.

El archivo escrito en formato *WSDD (Web Service Deployment Descriptor)* describe al contenedor de servicios web cómo debería publicar el servicio web.

La importancia del archivo JNDI es mínima en el servicio web que se está desplegando debido a que más parámetros de éste son necesarios cuando se va a desarrollar servicios web más complejos [4].

D. Compilación y generación del archivo GAR

El archivo *GAR* (Grid Archive) es un archivo simple que contiene absolutamente todos los archivos y la información que el contenedor de servicios necesita para desplegar el servicio y hacerlo disponible a los usuarios del mini-grid. La generación del archivo *GAR* se obtiene con Ant[3]; esto permite a los programadores olvidarse de cada paso involucrado en la obtención de un ejecutable desde los archivos fuente. Los pasos están descritos en el archivo *buildfile*, el cual indica a Ant lo que debe compilar, cómo y en qué orden, facilitando el proceso considerablemente. De lo que hay que preocuparse es de la escritura de la interfaz de servicio, la implementación del servicio y el descriptor de despliegue [4].

E. Despliegue del servicio

El archivo *GAR* mencionado anteriormente contiene los archivos y la información que el servidor web necesita para desplegar el servicio web. El despliegue es realizado en base a la herramienta del *Globus Toolkit 4* que desempaqueta el archivo *GAR* y copia los archivos dentro de ubicaciones clave en el árbol de directorios del *Globus Toolkit* [4].

F. El cliente

El objetivo de crear el servicio web descrito es la integración del *cluster* de Matlab al mini-grid, pero necesita un cliente que consuma el servicio web que ejecuta la aplicación desarrollada en Matlab y permita obtener los resultados en las máquinas del mini-grid.

El cliente es desarrollado en Java y debe ser compilado y ejecutado en Windows o Linux para que todas las máquinas del mini-grid puedan consumir el servicio web sin importar el Sistema Operativo que utilicen.

G. Ejecución de la aplicación en el Mini-Grid

Al consumir el servicio web que está disponible en el contenedor de *Globus* (ver Fig. 5) se crea un proceso correspondiente a la aplicación del cluster mediante el archivo principal creado en Matlab, el proceso tiene la capacidad de enviar los resultados a cada uno de los nodos del mini-grid mediante el uso del componente GridFTP, para lo cual se modifica el archivo principal *FEM2DL_Box_mdce.m.*

El resultado final es una gráfica de la distribución del potencial eléctrico en una guía de onda rectangular (Fig. 6).

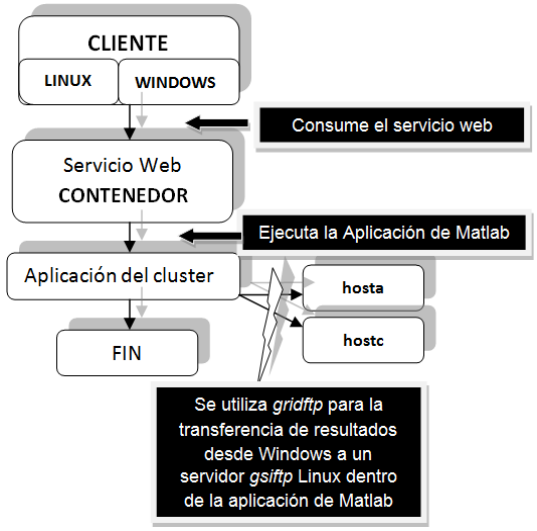

Fig. 5 Ejecución de la aplicación en el mini-grid.

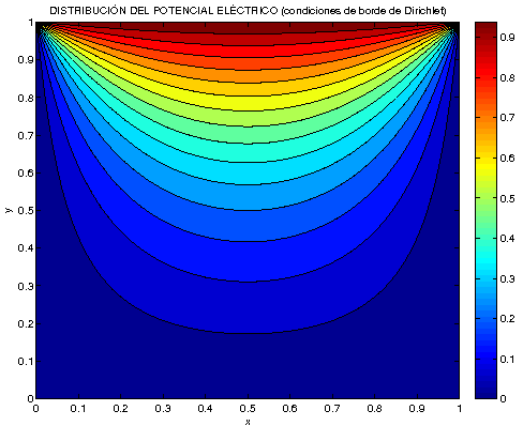

Fig. 6 Gráfico resultante de la aplicación.

[3] Apache ANT: herramienta que permite la realización de tareas mecánicas y repetitivas, especialmente durante la fase de compilación y construcción (*build*).

Para visualizar de mejor forma las ventajas del uso de este servicio web y del empleo del cluster, se realizó la comparación entre los tiempos de ejecución del trabajo al utilizar un solo trabajador (una máquina) y dos trabajadores (dos máquinas). Para lo cual se utiliza la función *plottask.m* desarrollada para Matlab.

En la Fig. 7 se observa que el tiempo de ejecución total para un trabajador es de 56 segundos. En la Fig. 8 se observa la disminución del tiempo de ejecución total a 30 segundos al utilizar dos máquinas.

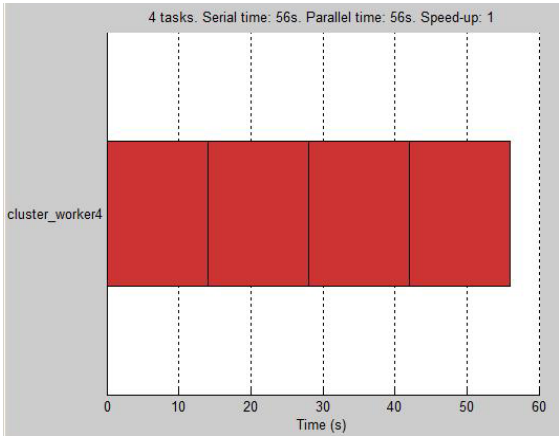

Fig. 7 Duración de tareas utilizando un trabajador.

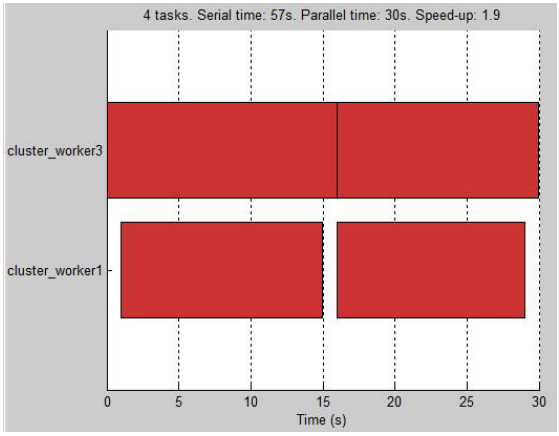

Fig. 8 Duración de tareas utilizando dos máquinas.

I. Conclusiones

La instalación de un Grid depende de la utilización de un middleware, en el presente trabajo se utilizó *Globus Toolkit* 4 (GT4) para integrar máquinas de características diferentes y sistemas operativos Windows o Linux.

El mini-grid se implementa en base al *midleware* GT4 y se utilizan las herramientas de Matlab para la conformación de un *cluster* computacional para el desarrollo de aplicaciones de gran carga computacional y, finalmente, la aplicación desarrollada en Matlab es presentada al mini-grid en base al consumo de un servicio web.

El mini-grid desarrollado permite la realización y pruebas de servicios web mediante el contenedor propio de *Globus Toolkit* como herramienta fundamental para la heterogeneidad del *Grid*. Además, involucra la programación específica para uso dentro de Matlab para la computación distribuida.

Referencias

[1] J. Bart, L. Ferreira, N. Bieberstein, C. Gilzean, R. Strachowski, Enabling Applications for Grid Computing with Globus. IBM Redbooks, Primera Edición, 2003.

[2] F. Magoules; J. Pan, K. Tan; A. Kumart. Introduction to Grid Computing. Chapman & Hall CRC, Primera Edición, Estados Unidos, 2009

[3] F. Berman; A. Hey; G. Fox. GRID COMPUTING: Making the Global Infrastructure a Reality. Wiley, Primera Edición, Inglaterra, 2003

[4] L. Childers, B. Sotomayor, "Globus Toolkit 4: Programming Java Services" Morgan Kaufmann, Primera Edición, 2006.

[5] Mathworks Install Support Team, "Installing Parallel Computing Toolbox™ 3.3 on Windows® Operating Systems", Estados Unidos, 2008.

[6] A. Polycarpou, "Introduction to the Finite Element Method in Electromagnetics", Morgan & Claypool, 2006.

[7] Página oficial de *Globus Toolkit*http://www.globus.org/

Gabriela A. Gangotena, nació en Quito el 5 de Abril de 1986. Realizó sus estudios secundarios en el Colegio Experimental "24 de Mayo" y los finalizó en el año 2004. En el mismo año ingresó a la Escuela Politécnica Nacional, donde obtuvo el título de Ingeniera en Electrónica y Telecomunicaciones en el año 2010.

Marco D. Bonilla, nació en Quito el 21 de Septiembre de 1986. Realizó sus estudios superiores en el Colegio San Gabriel y los finalizó en el año 2004. En el mismo año ingresó a la Escuela Politécnica Nacional, donde obtuvo el título de Ingeniero en Electrónica y Telecomunicaciones en el año 2010.

Iván M. Bernal, graduado del Instituto Nacional Mejía. Obtuvo el título de Ingeniero en Electrónica y Telecomunicaciones en la Escuela Politécnica Nacional en 1992. Obtuvo los títulos de M.Sc. (1997) y Ph.D. (2002) en *Computer Engineering* en *Syracuse University*, NY, USA. Ha realizado cursos especializados en varios países europeos, latinoamericanos, Estados Unidos y en Corea del Sur. Actualmente trabaja en la EPN, en el Departamento de Electrónica, Telecomunicaciones y Redes de Información.