Mecánica Computacional Vol. XXIV
A. Larreteguy (Editor)
Buenos Aires, Argentina, Noviembre 2005

COMPUTACIÓN DE ALTO RENDIMIENTO UTILIZANDO GRID COMPUTING EN UN ENTORNO MULTIPLATAFORMA

Pablo Turjanski[*], Diego Fernández Slezak[*], Juan Pablo Suarez[*], Alejandro Panelli[*], Alejandro Soba[*], Santiago Peña[†], Guillermo Marshall[*]

[*] Laboratorio de Sistemas Complejos, Departamento de Computación,
Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina.
e-mail: marshallg@mail.retina.ar

[†] Center for Computation & Technology, Louisiana State University, USA

Key words: Grid Computing, Cálculo Distribuido, Globus Toolkit, Multiplataforma, Fractal.

Grid Computing (GRID) es un entorno persistente que permite la integración de recursos informáticos administrados por diversas organizaciones geográficamente alejadas, asumiendo la ausencia de una ubicación y control central. Resulta evidente que este entorno no puede estar limitado a un sistema operativo o a un lenguaje de programación específico. Así, surgen iniciativas multiplataforma para el uso distribuido de recursos. En este trabajo nos abocamos al estudio de una herramienta para crear un entorno GRID multiplataforma de modo de compartir diversos recursos sin importar el sistema operativo ni la arquitectura subyacente, enfatizando el cálculo paralelo. Se implementó una aplicación distribuida de generación de fractales basada en tecnología GRID.

1. INTRODUCCIÓN

Durante las décadas pasadas hemos sido testigos de avances impensados en tecnología de la información. Microprocesadores, capacidad de almacenamiento y velocidad de las telecomunicaciones han contribuido a crear una poderosa infraestructura para satisfacer el cada día más acelerado mercado de los negocios, y la demanda cada vez más elevada de poder de cómputo para aplicaciones científicas y comerciales.

Como ejemplo podemos mencionar las computadoras personales. En los últimos años han evolucionado de tal manera que hoy en día se han convertido en una herramienta de gran capacidad de cómputo y almacenamiento de información. Una simple máquina de escritorio ha aumentado muchísimo su poder de cómputo y nos provee una capacidad similar a la de una supercomputadora en 1990. Aún a pesar de este desarrollo las necesidades del ambiente académico y comercial no se ven satisfechas y requieren una mayor capacidad. Como solución a esta necesidad surgen alternativas basadas en la interconexión de computadoras (Clusters) y sus diversas variantes, aplicados a la resolución de problemas altamente costosos en necesidad de cómputo. Hoy en día más del 50% del ranking de las 500 computadoras más eficientes del mundo (TOP500[1]) está ocupado por clusters de máquinas personales; situación impensada en el comienzo de los ‘90.

Sin embargo ello aún no es suficiente para los problemas que actualmente se tratan de resolver, como el análisis de estructuras de macromoléculas, procesamiento de 10^{25} bytes de información del acelerador CERN’S Large Hadron Collider (LHC) o simulación del funcionamiento del cerebro humano, entre otros problemas en donde el requerimiento de procesamiento de datos y el tamaño de los mismos ha aumentado en tres o cuatro órdenes de magnitud.

Estos problemas aún no resueltos han demandado cada vez más respuestas adecuadas en el área de telecomunicaciones y de programación que aprovechen en todo su potencial las nuevas tecnologías. Cluster Computing, Peer-to-peer Computing, soluciones distribuidas a través de Internet, entre otras, fueron respuestas a esas demandas en alguno o varios de sus aspectos fundamentales.

De este modo nos hallamos con un punto de quiebre en tres aspectos fundamentales:

1. El desarrollo de la tecnología informática es tan acelerado que no da tiempo a construir una supercomputadora con la máxima velocidad disponible de procesadores que, al terminar de hacerlo, se encuentra compitiendo con procesadores del doble de capacidad de cómputo. Este fenómeno comienza a observarse también en los clusters de PC ya que para dar respuesta a la demanda de cómputo de ciertas aplicaciones científicas, se requieren una excesiva cantidad de nodos y su renovación continua.
2. La mayoría de las empresas y centros educativos cuentan con una enorme capacidad latente de cómputo no utilizada. La estadística dice que se utilizan menos de un 5% de los recursos disponibles en computadoras personales, y el 15% a 20% de uso en los servidores basados en Unix[2,3].
3. Las telecomunicaciones especialmente en los países desarrollados presentan ancho

de banda desaprovechados y a un costo sumamente bajo, que se compatibiliza con la creciente necesidad de acceso a datos distribuidos geográficamente alrededor del planeta.

Desde mediados de los '90 se viene desarrollando una respuesta a todas esas demandas unificadas: Grid Computing (ver Figura 1).

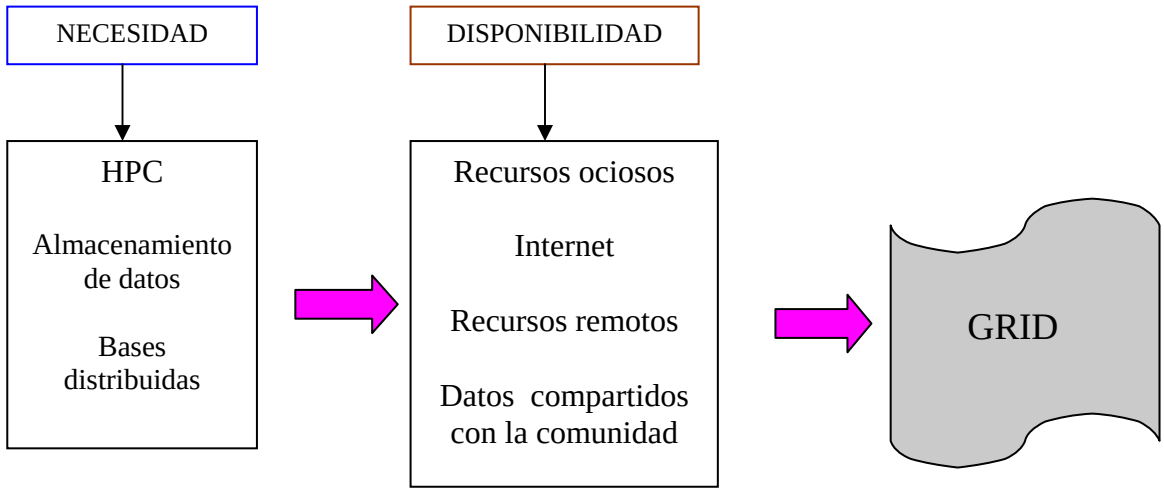

Figura 1: Convergencia en el desarrollo de GRID Computing.

1.1. Definición de GRID

Grid Computing posee muchas acepciones, una de ellas, quizás la más aceptada, es aquélla que lo considera un tipo de sistema paralelo o distribuido que permite compartir, seleccionar, y agregar recursos distribuidos a través de múltiples dominios administrativos basados en su disponibilidad, capacidad, rendimiento, costo, y requerimientos de calidad de servicios[2]. Otra acepción más restringida, pero común en el medio académico, es que GRID es una manera de utilizar muchas computadoras conectadas a través de una red simultáneamente para resolver un problema científico o técnico; en los casos más comunes estos problemas requieren substancial poder de cómputo y producen o requieren acceso a una masiva cantidad de datos [3].

En síntesis podemos llamar GRID a cualquier entorno o sistema en el que se cumplen las siguientes condiciones[15]:

1- Coordinación de recursos que no están sujetos a control centralizado. Esto implica la integración de recursos y usuarios que pertenecen a diferentes dominios de control. La coordinación de recursos no se reduce solo a las funciones de un simple administrador de tareas (scheduler), sino que también implica el control y gestión sobre membresía, seguridad, establecimiento de políticas, etc.

2 - Desarrollo e implementación de protocolos e interfaces estándar que se encargan de las tareas básicas de autenticación, autorización y descubrimiento de recursos para su posterior

utilización.

3 - Obtención de resultados no triviales. En otras palabras esta condición se cumplirá cuando el uso combinado de los recursos de un sistema es considerablemente superior a la suma de las partes que lo componen

Dentro de esta nueva tecnología comienzan a surgir nuevas formas de colaboración. Distintas personas u organizaciones, geográficamente alejadas, se agrupan y asocian para compartir recursos comunes. En este contexto llamamos recurso a cualquier tecnología que pueda utilizarse a distancia. Por ejemplo, una computadora personal, un servidor, un cluster o una supercomputadora, como también dispositivos de almacenamiento, sensores, cámaras, o ancho de banda sin utilizar.

Estas agrupaciones de personas o instituciones dan lugar a lo que llamamos Organización Virtual. La Organización virtual puede estar compuesta desde una pequeña corporación con algunos departamentos en el mismo edificio a un enorme grupo de personas de diferentes organizaciones diseminadas por todo el planeta, cada una con diferentes políticas de uso y requerimientos pero todas compartiendo sus recursos, constituidos no solo por archivos o datos sino también por equipos, software, licencias, etc. Estos recursos son tratados virtualmente para darles una mayor interoperabilidad a través de la heterogénea y enorme cantidad de participantes de GRID.

La ausencia de localización geográfica y de control centralizado implica que los recursos en GRID no requieren una posición específica para su utilización.

1.2. Historia y evolución de Grid Computing

Desde mediados de los ‘90 surgieron numerosos proyectos de investigación focalizados en desarrollar cálculo distribuido intentando obtener rendimientos cercanos al de una supercomputadora. De este modo en el IEEE/ACM de 1995 se obtuvo la primer concesión de 17 sitios con una red de alta velocidad, para crear la primer “meta computadora” en donde se ejecutaron numerosas aplicaciones en un entorno distribuido. A raíz de esa exitosa demostración comenzó el proyecto Globus[9]. Este proyecto comenzó a ser referenciado como Grid Computing en un intento por utilizar la analogía con la red de potencia eléctrica.

GRID, que comenzó como un proyecto en ámbitos educativos y académicos, también movilizó a partir del 2000 a empresas que comenzaron a interesarse en sus posibilidades comerciales. Hoy en día corporaciones como IBM, Sun Microsystems, Intel y Hewlett Packard, comenzaron a desarrollar productos basados en esta tecnología tanto en aplicaciones de negocios como académicas.

Actualmente existen diversas soluciones GRID en el mercado. Básicamente esta constituido por un middleware desarrollado para distintas plataformas. Algunos ejemplos de ello son:

NORDUGRID[4]: el middleware NorduGrid (o *Advanced Resource Connector*, *ARC*) es un software open source distribuido bajo licencia GPL. Desde su primera versión en mayo del 2002 este middleware esta siendo utilizado en entornos de producción con énfasis en la escalabilidad, estabilidad y performance. ARC provee todos los servicios GRID fundamentales como servicios de información, descubrimiento de recursos y monitoreo de los

mismos, envío y manejo de trabajos, manejo de datos y de recursos. El middleware está construido sobre soluciones open source como OpenLDAP, OpenSSL, SASL y librerías Globus Toolkit 2 (GT2).

TERAGRID[5] TeraGrid es una solución que conecta básicamente grupos científicos distribuidos geográficamente que requieren de HPC y acceso a grandes cantidades de datos, instrumentos remotos, etc. También provee de una distribución compartida de fuentes de almacenamiento en forma transparente para todos los "socios" adheridos. Organizado por Texas Advances Computing Center (TACC) en la Universidad de Austin, uno de los actualmente 8 centros adheridos que provee recursos para computar, visualizar y almacenar datos así como acceder a software provisto por la TeraGrid.

EUROGRID[6] este proyecto (actualmente concluido) fue un desarrollo de investigación y tecnología que financió la Comunidad Europea entre los años 2000 y 2004. El proyecto demostró el uso de GRID entre centros científicos e industrias seleccionadas aplicado a cuatro grandes áreas de trabajo: Biomedicina, Meteorología, Industria Aeroespacial y HPC, con sus cuatro proyectos BioGRID, MeteoGRID, CAE GRID y HPC research GRID.

GRIDBUS[7]: este proyecto es una herramienta GRID que provee especificaciones de código abierto e implementaciones GRID para el desarrollo de servicios y tecnologías de computabilidad dedicadas a aplicación en e-science y e-business. El proyecto es financiado por: Australian Research Council, Storage Technology Corporation, Sun Microsystems, VPAC, IBM, y Singapore Computer Systems. GRIDBUS propone crear la nueva generación de cómputo en **GRID** y tecnologías en **BUS**iness, que posibiliten la creación de aplicaciones que exploten el potencial de dichos servicios en la industria y el comercio.

LSF[8]: Es un sistema que se utilizaba inicialmente para manejar Clusters de Computadoras. Constituía un manejador de recursos, administrándolos y realizando el balance de carga de los sistemas. LSF 4.0 es la versión ampliada de este proyecto que propone el trabajo con una red de Cluster utilizando tecnología GRID para posibilitar el trabajo con cluster de clusters, facilitando el acceso remoto, la unificación de los recursos distribuidos y deslocalizados, además de las acostumbradas tareas de un manejador de recursos como balance de carga, checkpointing, etc.

GLOBUS TOOLKIT[9]: es un conjunto de servicios y librerías de software, de arquitectura abierta y código abierto para aplicaciones en plataforma GRID. Este "toolkit" proporciona las funcionalidades de seguridad, descubrimiento de recursos, comunicación, manejo de datos, detección de fallas y portabilidad. Debido a que este paquete de utilidades es código abierto, muy flexible y uno de los más utilizados en el mundo, fue el seleccionado para la instalación de un laboratorio basado en la tecnología GRID.

2. PRIMERAS EXPERIENCIAS EN GRID COMPUTING

La implementación de un proyecto de GRID debe satisfacer condicionamientos de muy diversa índole tanto de hardware como de software, requiriendo una infraestructura que en general excede o se encuentra en los límites de los presupuestos universitarios argentinos. Es por ello que antes de tomar una decisión del esquema definitivo de GRID a utilizar es necesario estudiar distintas opciones desarrollando diseños preliminares que ayuden a la

elección óptima. Una herramienta que facilita enormemente los diseños preliminares es la construcción de redes virtuales mediante el uso de máquinas virtuales.

Por este motivo, el trabajo se separó en dos etapas. Por un lado, durante el 2004 se realizó un diseño preliminar de un laboratorio GRID basado en máquinas virtuales[10]. Luego, una vez probada y analizada la tecnología, se pasó al desarrollo real de un laboratorio de estas características.

2.1. Primera Etapa

En el 2004, se realizó un diseño preliminar de GRID denominado GRID Virtual basado en máquinas virtuales utilizando, Virtual PC® bajo Windows XP®, Linux y Globus Toolkit 3.2®. El desarrollo del laboratorio de GRID Virtual que se realizó se dividió en distintas etapas. En una primera etapa se creó una máquina virtual en la cual se instaló un nodo Gris y se analizaron distintas configuraciones de instalación. Una vez comprendida la técnica de instalación de un nodo, se pasó a la segunda etapa: instalación del laboratorio virtual conformado por nueve máquinas virtuales, cada una con un nodo similar (ver Figura 2).

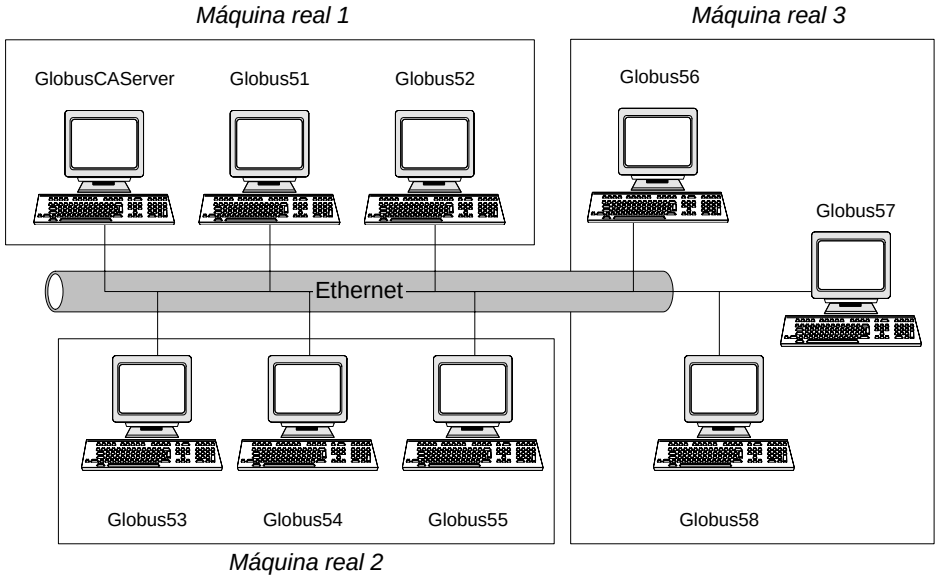

Figura 2: Esquema de laboratorio de GRID Virtual UBA

El Globus Toolkit 3.2 consiste en una serie de programas que implementan la arquitectura GRID basado en la especificación Open Grid Service Infraestructure (OGSI)[11]. El Open Grid Service Architecture (OGSA) define una arquitectura común, estándar, abierta y basada en servicios para aplicaciones de GRID. El objetivo de OGSA es estandarizar prácticamente todos los servicios que se pueden encontrar en estas aplicaciones (servicios de manejo de recursos, servicios de seguridad, etc.) especificando un conjunto de interfaces para estos servicios. En esta arquitectura un servicio de GRID es simplemente una extensión de un servicio web. El OGSI da una especificación formal y técnica de estos servicios.

La distribución de Globus Toolkit 3.2 utilizada en el laboratorio virtual fue reemplazada

durante los primeros meses del 2005 por una nueva versión: Globus Toolkit 4.0 (GT4). Esta nueva implementación se basa en un nuevo estándar para servicios de arquitecturas GRID: Web Services Resource Framework (WSRF). Esta implementación fue desarrollada por numerosos grupos, tanto del área de código abierto como empresas comerciales.

2.2. Segunda Etapa

En los primeros meses del 2005, la Globus Alliance anunció la llegada de la nueva versión del paquete de GRID: Globus Toolkit 4.0 (GT4). Como ya dijimos, esta distribución implementa la nueva especificación llamada WSRF.

Esta implementación se basa en la especificación de servicios web y le agrega funcionalidades para mantener estados de programas. Recordemos que un servicio web es un componente de software que puede auto describirse y provee cierta funcionalidad a otras aplicaciones, a través de una conexión de Internet. Esas aplicaciones, acceden los servicios web vía protocolos web y formatos de datos estándares, tales como HTTP y XML, sin tener en cuenta en absoluto cómo los servicios web están implementados o sobre qué plataforma corren[9]. Los servicios web tienen una interfaz bien definida en un formato llamado WSDL (lenguaje basado en XML). Otros sistemas interactúan con el servicio web utilizando mensajes SOAP los cuales se encuentran establecidos previamente[12]. El estándar de servicios web no contempla aplicaciones con persistencia de estado. Por tal motivo se decidió ampliar dicho estándar agregándole este tipo de funcionalidades (WSRF)

Una de las características más importantes de esta nueva implementación es que no es compatible con las versiones anteriores (GT3.2 y anteriores). Si bien esto puede parecer una gran desventaja, esta implementación surge de numerosos acuerdos entre los más experimentados usuarios y desarrolladores de la arquitectura GRID. En la Figura 3 se muestra la relación entre OGSA, WSRF, GT4 y servicios web.

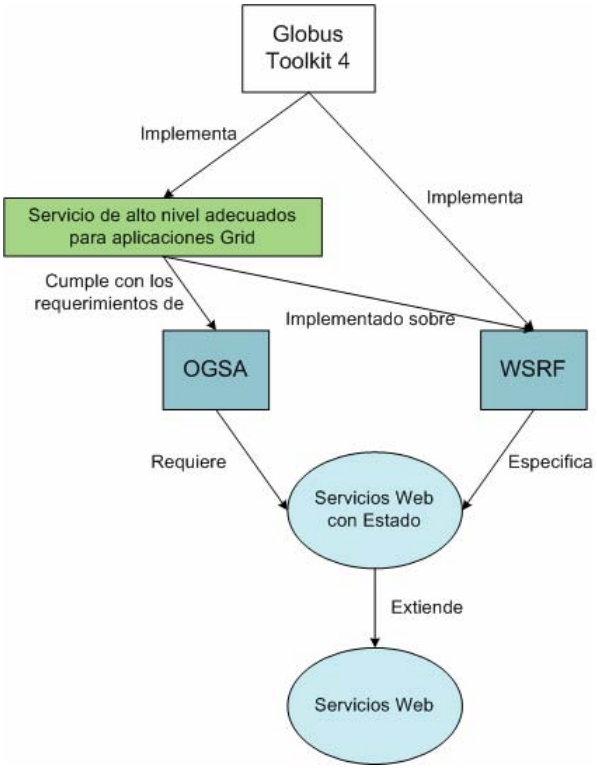

Figura 3: Relación entre OGSA, GT4, WSRF y servicios web.

Utilizando como punto de partida el diseño preliminar del laboratorio de GRID virtual, se procedió a instalar y configurar GT4 en una máquina real del laboratorio. El objetivo principal de esta máquina sería el de proveer los archivos comunes a las instalaciones en los nodos que formen parte del GRID, firmar y entregar los certificados para el uso del GRID y mantener la lista actualizada de los usuarios autorizados a utilizar los distintos recursos disponibles. Además, se agregaron otros nodos al GRID con sistemas operativos totalmente distintos, una máquina basada en Microsoft Windows y otra basada en Mac OS X, con el objetivo de configurar y probar el funcionamiento de un GRID heterogéneo con varios sistemas operativos.

Adicionalmente se estableció una colaboración entre el Center for Computation & Technology (CCT), Louisiana State University (LSU), EEUU y el Laboratorio de Sistemas Complejos (LSC), Departamento de computación, FCEyN, UBA. Como resultado de esta colaboración varios integrantes del LSC fueron invitados por el CCT para la construcción de un GRID internacional como comienzo de actividades en conjunto entre ambos centros, incorporando un nodo bajo plataforma MAC OS X al GRID.

3. INSTALACIÓN Y CONFIGURACIÓN

Para la instalación de la nueva versión de Globus, se procedió a instalar la primera máquina teniendo en cuenta las pruebas realizadas en el laboratorio virtual. Dicha máquina

fue instalada con el sistema operativo Linux Debian (versión Sarge) y GT4. Los certificados de usuarios y máquinas fueron administrados utilizando el paquete especializado para este fin provisto por GT4: simpleCA.

Una vez instalado el primer nodo, que se encargaría, entre otras cosas, de firmar los certificados, se procedió a instalar los nodos adicionales. Un esquema del laboratorio se puede ver en la Tabla 1.

Nombre	Sistema Operativo	Descripción
lsc.dc.uba.ar	Linux Debian (Sarge)	Máquina encargada del firmado de certificados y repositorio de archivos de instalación.
CCT-LSU	Apple Mac OS X	Máquina ubicada en EEUU, proveedora y cliente de servicios.
mani.lsc.dc.uba.ar	Microsoft Windows	Máquina ubicada en el Laboratorio de Sistemas Complejos proveedora y cliente de servicios.
jp.lsc.dc.uba.ar	Microsoft Windows	Máquina ubicada en el Departamento de Computación, sólo cliente de servicios
rocky.lsc.dc.uba.ar	Linux Rocks Clusters	Cluster del laboratorio, proveedora y cliente de servicios.

Tabla 1: Descripción de nodos incluidos en el GRID

Durante el proceso de Instalación nos encontramos con diversos problemas que hemos ido resolviendo a medida que se suscitaban. Entre los más destacados, podemos mencionar:

- *LINUX – Servidor de Certificados.* Al instalar el programa certificador *simpleCA*. que viene con el paquete GT4, ingresamos el nombre del servidor sin tener en cuenta que debía ser *full qualified*. Esto nos trajo algunos inconvenientes a posteriori ya que, al acceder desde otro nodo, el DNS respondía con el nombre *full qualified*, generando una confusión en los nombres. Para solucionar este inconveniente optamos por reinstalar el *simpleCA* con el nombre *full qualified* y luego certificamos nuevamente el resto de los nodos.
- *WINDOWS - variables de entorno*: A diferencia del entorno Linux, GT4 para plataforma Windows necesita variables de entorno adicionales que deben ser configuradas y no están incluidas dentro de los manuales, ya que estos vienen destinados principalmente a usuarios Linux. Estas variables de entorno se fueron descubriendo y resolviendo a medida que fueron necesarias.
- *WINDOWS – Certificado*. El simpleCA incluido en GT4 no posee un archivo de instalación de nodos adicionales para plataforma Windows. Tal como propone la bibliografía, se generaron dichos certificados en un entorno Linux y luego se movieron al nodo Windows.
- *MAC – Inconvenientes durante la compilación*. Debido a que las fuentes de GT4 no vienen pre-compiladas para el entorno *MAC*, hubo que compilarlas para dicho entorno, lo que trajo bastantes problemas de dependencias de bibliotecas que se

fueron solucionando hasta obtener la compilación completa.

- *Todos los entornos - Red en General – Firewall*. En general tuvimos problemas con la habilitación de los puertos necesarios para comunicarnos entre máquinas y con las reglas puestas en el firewall ya que éste filtra algunos paquetes de GT4. Por lo tanto tuvimos que montar nuestro GRID en una red experimental que posee el departamento para realizar las pruebas sin restricciones de ningún tipo.

Como contrapartida a todos los problemas antes mencionados, podemos mencionar que dentro del entorno *LINUX Debian*, la instalación se realizó en no más de 15 minutos por reloj (tarea que en GT3.2 tomaba horas). Esto es debido a que GT4 cuenta con una versión pre-compilada para el Sistema Operativo *LINUX Debian*.

Una vez realizada la instalación y la certificación de todos los nodos de nuestro GRID, se decidió ejecutar algunos ejemplos de prueba. El primero consistió en realizar un GridFTP. Esta prueba se realizó con éxito en y entre todos los entornos. Esto permitió verificar que la conectividad y seguridad entra máquinas estaba funcionando correctamente.

Luego, se decidió ejecutar un servicio de manera remota. Para ello, es importante destacar que se tuvo que estudiar el nuevo enfoque de GT4 en cuanto a la nueva manera de proveer un servicio. Como se mencionó anteriormente, esta difiere con respecto a GT3.2 (versión que se utilizó en la etapa de diseño preliminar).

Debido a nuestro conocimiento acerca de GT3.2 adquirido en la primer etapa del proyecto y debido a que GT4 para Linux Debian preserva parte de la funcionalidad de GT3.2 (sólo es necesario configurarlo), pudimos ejecutar desde todos los entornos un programa cliente (globusjob-run) que utiliza un servicio remoto corriendo en el entorno Linux Debian. En contrapartida, el entorno Windows no cuenta con la funcionalidad que permite que GT4 funcione como servidor de servicios de manera similar a GT3.2, por lo tanto este entorno fue utilizado sólo como cliente

Como segundo ejemplo, decidimos ejecutar un servicio sencillo utilizando la nueva tecnología de servicios web de GT4. El ejemplo sencillo de servicio web que se implementó es el *Math Web Service*, que menciona el manual de GT4[13]. Este servicio fue seleccionado por su simplicidad y porque utiliza WSRF para mantener el estado de la información. Este servicio permite al usuario ejecutar las operaciones de suma y resta. Además permite tener acceso a los siguiente recursos: valor actual del elemento almacenado y última operación que se realizó (suma o resta).

Este ejemplo pudo ser implementado en todos los entornos de GT4 utilizados (Linux, Mac y Windows) y además pudo ser invocado desde cada uno de ellos indistintamente el entorno servidor.

4. DESARROLLO DE APLICACIÓN

Una vez instalado, configurado y testeado el laboratorio GRID, se procedió al desarrollo de una aplicación real para ser resuelta con esta tecnología. Se eligió una aplicación sencilla, capaz de ser dividida en procesos totalmente independientes entre sí, para poder ejecutarla en forma distribuida pasándole a cada instancia los parámetros que le corresponden.

Dicha aplicación genera fractales utilizando el método de Newton[14], el cual consiste en un

algoritmo para hallar raíces de polinomios en forma iterativa, a partir de una semilla inicial. El algoritmo converge o no, según el punto inicial y el polinomio sobre el que se realiza la búsqueda, la cual culmina cuando el error es menor que el especificado, o cuando se supera cierta cota de iteraciones.

Para generar el fractal se deben conocer las raíces del polinomio estudiado. A partir de una porción del plano complejo se aplica el método de Newton a cada punto, analizando su convergencia a alguna de las raíces, o el caso particular de la divergencia. Luego, a cada punto del plano se le asigna un color que corresponde a la raíz a la cual converge (asignando otro color para la divergencia), obteniendo así la representación gráfica del fractal.

El lenguaje utilizado para implementar este algoritmo fue Java, ya que este nos permite poder ejecutar dicha aplicación en distintas plataformas sin tener que modificar el código fuente. Tal como se mencionó anteriormente, dicha aplicación se utiliza a través del mecanismo de servicio web. El servicio web recibe los parámetros que indican la sección del fractal que se desea construir en cada instancia y las raíces del polinomio. Cada una de las instancias genera una imagen de la sección indicada, que es guardada en un archivo. De esta manera, se logra hacer una paralelización a nivel de parámetros en un entorno multiplataforma (Linux, Mac y Windows).

Una vez que se ejecutan todos los procesos, se transfieren todas las imágenes generadas utilizando el protocolo GridFTP a un mismo nodo, donde se agrupan todas las imágenes para formar el fractal completo.

Para la creación del servicio web en GT4 es necesario realizar los siguientes pasos:

1. Definir la interfaz del servicio en el Web Service Defined Lenguage (WSDL)
2. Implementar el servicio en Java
3. Configuración para la publicación del servicio
4. Compilar y generar el archivo GAR
5. Registrar el servicio web en Globus Toolkit

Como primer paso es necesario definir la interfaz del servicio que estará disponible para los clientes, sin tener en cuenta los algoritmos utilizados. Dicha especificación se describe mediante el lenguaje WSDL. La interfaz implementada se puede ver en la Figura 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="FractalService" …. >
<wsdl:import namespace=… >

<!-- TYPES -->

<types>
<xsd:schematargetNamespace="http://…. /core/FractalService" >

    <xsd:element name="generateFractal">
          <xsd:complexType>
          <xsd:sequence>
                <xsd:element name="out"   type="xsd:string" />
                <xsd:element name="cx"    type="xsd:double"  />
                <xsd:element name="cy"    type="xsd:double"  />
                <xsd:element name="x"     type="xsd:double"  />
```

```
                <xsd:element name="y"     type="xsd:double"  />
                <xsd:element name="rx"    type="xsd:int"  />
                <xsd:element name="ry"    type="xsd:int"  />
                <xsd:element name="nr"    type="xsd:int"  />
                <xsd:element name="roots" type="xsd:string" />
          </xsd:sequence>
          </xsd:complexType>
    </xsd:element>
    <xsd:element name="generateResponse">
          <xsd:complexType/>
    </xsd:element>
</types>

<!-- PORTTYPE -->

<portType name="FractalPortType" >

    <operation name="generateFractal">
          <input message="tns:GenerateInputMessage"/>
          <output message="tns:GenerateOutputMessage"/>
    </operation>

</portType>

<!-- MESSAGES -->
```

Figura 4: Código de la Interfaz del servicio de fractal en leguaje WDSL

Una vez definida la interfaz, es necesario implementar el servicio. Dentro de la implementación se encuentra el algoritmo con el cual se creará el fractal. Con este fin se creó la clase “FractalService”, que implementa dicho algoritmo y que respeta la especificación de la interfaz definida en el punto anterior.

Una vez que se poseen la implementación y la interfaz del servicio (los componentes más importantes en la creación de un servicio web), es necesario configurar datos sobre la publicación del servicio, por ejemplo la dirección URI (Dirección Única de Identificación del servicio). Esta configuración es llevada a cabo con dos archivos: Web Service Deployment Descriptor (WSDD) y el JNDI (Java Naming and Directory Interface).

Luego de realizar los pasos anteriores, contamos con todos los componentes necesarios para el servicio web. En el cuarto paso, incluimos todos estos componentes en un único archivo contenedor, denominado archivo GAR. Este contenedor incluye el archivo WSDL, la implementación del servicio en Java, y los archivos WSDD y JNDI.

Un esquema del archivo GAR se muestra en la Figura 5.

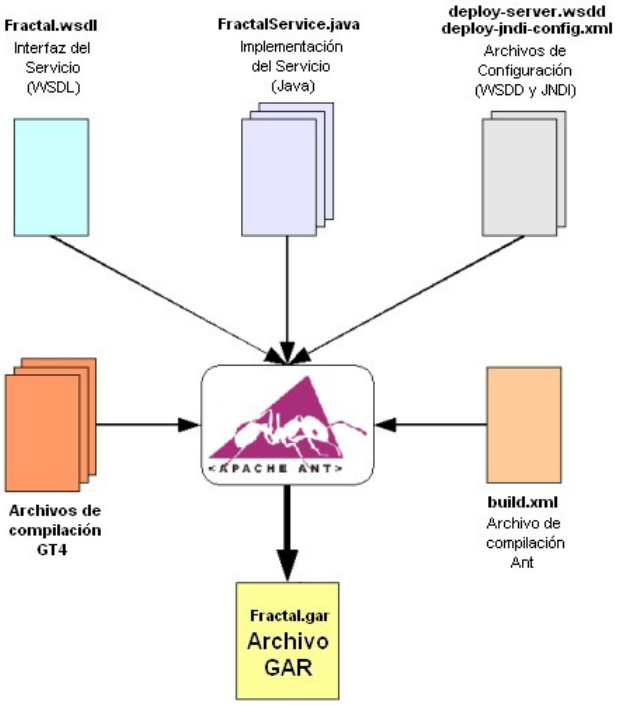

Figura 5: Esquema de archivo GAR

Para generar el archivo GAR se ejecuta un script (que en plataforma Windows utiliza el interprete Python; en plataforma Linux y Mac, shell script), y el compilador ant (requerido para instalación de GT4).

Por último, una vez obtenido el archivo GAR (en nuestro caso “fractal.gar”), solo se debe agregar el mismo a la lista de servicios disponibles. Para ello GT4 provee la herramienta globus-deploy-gar, que recibe como parámetro el archivo GAR a incluir (“fractal.gar”). Este comando registra el servicio web, que se activará en el nodo cuando el usuario inicie el contenedor de servicios (globus-start-container), como se muestra en la Figura 6.

```
Starting SOAP server at: http://10.1.100.2:8080/wsrf/services/
With the following services:

[1]: http://10.1.100.2:8080/wsrf/services/Version
[2]: http://10.1.100.2:8080/wsrf/services/NotificationConsumerService
[3]: http://10.1.100.2:8080/wsrf/services/NotificationTestService
[4]: http://10.1.100.2:8080/wsrf/services/SecureCounterService
[5]: http://10.1.100.2:8080/wsrf/services/PersistenceTestSubscriptionManager
[6]: http://10.1.100.2:8080/wsrf/services/gsi/AuthenticationService
[7]: http://10.1.100.2:8080/wsrf/services/TestRPCService
[8]: http://10.1.100.2:8080/wsrf/services/SubscriptionManagerService
[9]: http://10.1.100.2:8080/wsrf/services/ManagementService
[10]: http://10.1.100.2:8080/wsrf/services/TestServiceWrongWSDL
[11]: http://10.1.100.2:8080/wsrf/services/WidgetService
[12]: http://10.1.100.2:8080/wsrf/services/SampleAuthzService
[13]: http://10.1.100.2:8080/wsrf/services/examples/core/first/MathService
[14]: http://10.1.100.2:8080/wsrf/services/AuthzCalloutTestService
[15]: http://10.1.100.2:8080/wsrf/services/WidgetNotificationService
[16]: http://10.1.100.2:8080/wsrf/services/AdminService
```

```
[17]: http://10.1.100.2:8080/wsrf/services/ShutdownService
[18]: http://10.1.100.2:8080/wsrf/services/ContainerRegistryService
[19]: http://10.1.100.2:8080/wsrf/services/CounterService
[20]: http://10.1.100.2:8080/wsrf/services/TestService
[21]: http://10.1.100.2:8080/wsrf/services/FractalService
[22]: http://10.1.100.2:8080/wsrf/services/TestAuthzService
[23]: http://10.1.100.2:8080/wsrf/services/SecurityTestService
[24]: http://10.1.100.2:8080/wsrf/services/ContainerRegistryEntryService
[25]: http://10.1.100.2:8080/wsrf/services/NotificationConsumerFactoryService
[26]: http://10.1.100.2:8080/wsrf/services/TestServiceRequest
```

Figura 6: lista de servicios web disponibles luego de incluir fractal.gar

Una vez cumplimentados estos cinco pasos, el servicio se encuentra disponible y listo para ser llamado por los clientes.

5. RESULTADOS

Luego de realizar un diseño preliminar de un laboratorio GRID basado en máquinas virtuales, se llevó a cabo la instalación y configuración del laboratorio definitivo en máquinas reales con diversos sistemas operativos utilizando GT4. La disposición final de dicho laboratorio se puede observar en la Figura 7.

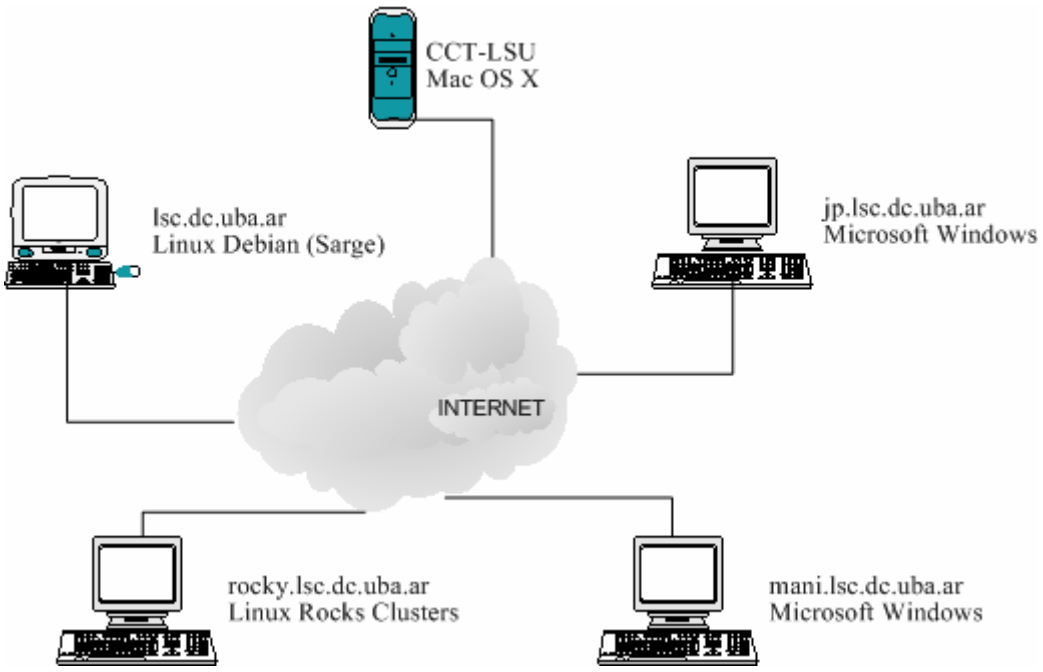

Figura 7: Esquema de laboratorio GRID

A su vez, se desarrolló una aplicación sencilla de generación de fractales basado en la nueva especificación (WSRF), como se explicó en los puntos anteriores.

El polinomio seleccionado como ejemplo corresponde a las raíces cúbicas de la unidad, analizando el intervalo bidimensional determinado por: $(-2 \leq x \leq +2)$ $(-i2 \leq y \leq +i2)$.

Como mencionamos anteriormente, el sistema permite distribuir la tarea en la cantidad de instancias definidas, cada una resolviendo una subdivisión del dominio del fractal. En la Figura 8 se puede observar una división en 16 tareas (4x4): “lsc.dc.uba.ar” 3 instancias, “CCT-LSU” 3 instancias, “mani.lsc.dc.uba.ar” 3 instancias, “jp.lsc.dc.uba.ar” 3 instancias y “rocky.dc.uba.ar” 4 instancias.

Cada instancia generó un sector de 1x1 (500x500 píxeles) y almacenó el resultado en una imagen en formato bitmap, transferida posteriormente al nodo cliente utilizando el protocolo GridFTP. Una vez terminado el trabajo de todas las instancias, se agruparon las imágenes para formar el fractal completo con una resolución total de 2000x2000 píxeles.

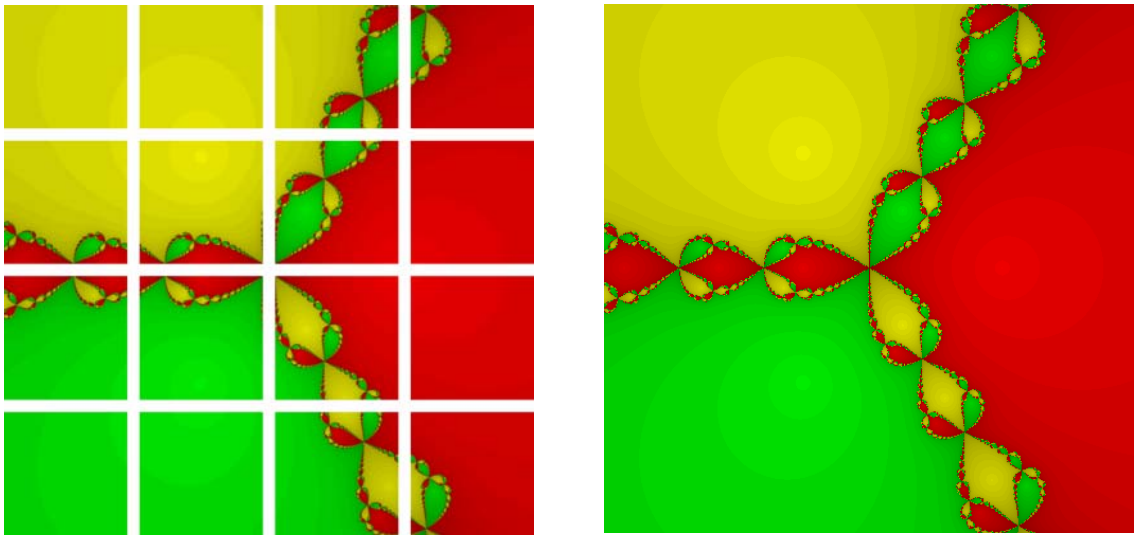

Figura 8: Resultado parcial y complete del fractal generado.

Podemos observar como se puede reducir la capacidad de cómputo requerida por una tarea específica, al ejecutarla en forma distribuida. Esto se ve claramente al comparar los 4.000.000 de puntos que se hubieran recorrido en un programa serial, frente a los 250.000 recorridos por cada nodo al distribuir la tarea.

6. CONCLUSIONES

Este trabajo concluye una primera etapa de estudio y verificación de una tecnología emergente en nuestro país. Luego de un estudio preliminar en máquinas virtuales, se logró instalar y configurar la plataforma GRID en un laboratorio con máquinas reales. Como resultado de este proceso se han documentado las tareas llevadas a cabo.

El sistema instalado para el desarrollo de este laboratorio fue Globus Toolkit 4.0 que introduce el estándar definido mediante servicios web, el cual hace el desarrollo más transparente y facilita la distribución y configuración del mismo entre los nodos sin importar el sistema operativo de los mismos. Dicha instalación fue realizada en las plataformas Windows, Linux y MacOS.

Por otro lado, se desarrolló una aplicación de generación de fractales basada en la especificación WSRF, lo que permitió distribuir el procesamiento de la generación del fractal en las distintas máquinas del laboratorio, en forma independiente del sistema operativo sobre el que trabaja.

En particular, nuestra experiencia al utilizar el lenguaje Java, nos permitió ejecutar este servicio en tres plataformas distintas: Linux, MacOS y Windows, con el mismo archivo GAR, sin tener la necesidad de realizar ningún tipo de compilación adicional para cada sistema.

7. BIBLIOGRAFÌA

[1] URL: http://www.top500.org/.

[2] Foster, C. Kesselman, J. Nick, S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Global Grid Forum (2002).

[3] Ahmar Abbas, *GRID COMPUTING: A Practical Guide to Technology and Applications*, Networking Series. Charles river media (2003).

[4] URL: http://www.nordugrid.org/

[5] URL: http://www.teragrid.org/

[6] URL: http://www.eurogrid.org/

[7] URL: http://www.gridbus.org/

[8] URL: http://www.platform.com/

[9] URL: http://www.globus.org/

[10] D. Fernández Slezak, P. Turjanski, M. Silva, J. Monetti, E. Mocskos, C. García Garino y G. Marshall, *Construcción de un Laboratorio de GRID virtual para HPC*, VI Simposio Argentino de Tecnología en Computación, JAIIO (2005).

[11] S. Tuecke, K. Czajkowski, I. Foster, *Open Grid Services Infrastructure (OGSI) Version1.0*, Global Grid Forum Draft Recommendation (2003).

[12] URL: http://www.w3.org/TR/soap/

[13] URL: http://gdp.globus.org/gt4-tutorial/

[14] J. Orlando Freitas, J. Sousa Ramos, *Computer Experiments with Newton's Method*, Colloque Européen ITEM, (2003).

[15] URL: http://www.gridtoday.com/02/0722/100136.html