Universidad Central “Marta Abreu” de Las Villas
Facultad de Matemática, Física y Computación

Trabajo para optar por Título de
Licenciado en Ciencia de la Computación

Conformación de un mini-grid para el Centro de Estudios de Informática (CEI).

Autor

Ali Saeidy.

Tutor

MSc. Leonardo Flavio del Toro Melgarejo.

Santa Clara

2013

Hago constar que el presente trabajo fue realizado en la Universidad Central Marta Abreu de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencias de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor Firma del jefe del Seminario

Dedicatoria

A mi papá, que desde el cielo me bendice.

A mi familia, por su paciencia y amor incondicional.

A mi amor, Liannet, por ser la verdadera razón de mi vida.

Agradecimientos

A mi mamá, por su infinito amor y sus oraciones.

A mis hermanos, porque a pesar de todo siempre me apoyan.

A mi hermanita, por su sinceridad, amor y amistad.

A mi familia en Cuba, por tratarme amorosamente.

A Liannet, por ser la luz que ilumina mi vida.

A Leonardo, por su dedicación y apoyo, ya que sin él nada hubiera sido posible.

A mis amigos cubanos y sirios, por compartir juntos todos estos años.

A todos mis profesores.

Pensamiento

La ciencia no es más que muchas repuestas fáciles a preguntas difíciles.

Resumen

El incremento en las necesidades de cómputo requeridos por los proyectos científicos presenta un reto para la comunidad de investigadores, lo cual, exige explorar nuevas tecnologías de computación distribuidas para buscar soluciones a sus demandas. La computación *Grid* propone una solución fiable y eficiente para conectar recursos heterogéneos y distribuidos entre diferentes organizaciones virtuales utilizando algún tipo de redes de comunicación. Actualmente, existen diversas herramientas que proveen una plataforma para desarrollar sistemas *Grid*, de ellas, se destaca *Globus Toolkit*, por su gran aceptación a nivel mundial.

La Universidad Central "Marta Abreu" de Las Villas, carece de un sistema capaz de agrupar todas las potencias de cómputos existentes, por lo cual, este trabajo propone explorar la tecnología *Grid* y como extenderla en la UCLV, mediante el diseño e implementación de una infraestructura *Grid*, simulada virtualmente, para el Centro de Estudios de Informática (CEI).

Para la conformación del mini-grid virtual, resultó preciso la instalación y configuración de la herramienta *Globus Toolkit* 5.2.2, además, un conjunto de software necesario para el funcionamiento del *Grid*. La integración entre administradores de recursos locales y el componente *GRAM*, que ofrece *Globus*, es sumamente necesaria, ya que este último no tiene la capacidad de ejecutar trabajos en paralelo por sí mismo, sino que lo hace a través del administrador de recursos locales.

Las pruebas de funcionamiento llevadas a cabo en el mini-grid mostraron: el correcto funcionamiento del sistema y proporcionaron resultados similares a los obtenidos mediante la ejecución de las mismas tareas empleando comandos *PBS* o de la biblioteca *MPI*.

Abstract

The increased computing needs required by scientific projects presents a challenge for the research community, which requires exploring new distributed computing technologies to find solutions to their demands. Grid computing offers a reliable and efficient solution for connecting heterogeneous and distributed resources between different virtual organizations using any type of communication networks.

Currently, there are various tools that provide a platform for developing Grid systems, including Globus Toolkit stands out for its wide acceptance worldwide.
The Central University "Marta Abreu" of Las Villas, lacks a system to group all installed power computations, therefore, this paper will explore the Grid technology and how to extent it to the UCLV through the design and implementation of a Grid infrastructure, virtually simulated, for the Centre on Informatics Studies (CEI).

For the implementation of the virtual mini-grid, was necessary the installation and configuration of the tool Globus Toolkit 5.2.2 also a set of software needed to operate the Grid. The integration between local resource managers and the GRAM component that offers Globus, it is extremely necessary since the latter does not have the ability to run parallel jobs by itself, but does so through the local resource manager.

Performance tests carried out on the mini-grid showed: the proper functioning of the system, obtained results similar to those obtained by performing the same tasks, using PBS or MPI library commands.

Tabla de contenido

Lista de figuras

Lista de tablas

Introducción

La tecnología actual, y principalmente la computación, ha contribuido de forma única a la resolución de muchos problemas en diferentes ámbitos y disciplinas, constituyendo hoy día una fuente de recursos imprescindible.

Desde sus orígenes, la computación ha visto la luz de su evolución en las actividades científicas, y precisamente en sus necesidades de almacenamiento y procesamiento de datos. Si bien en la mayoría de los casos la ciencia y otra variedad de disciplinas han visto satisfechos sus requerimientos, aún quedan desafíos abordables que esperan ser atendidos.

El incremento de las necesidades de cómputo para el desarrollo de diversos procesos académicos y científicos, demanda cada día mayores recursos computacionales que los disponibles en las redes locales de las diversas instituciones.

Los proyectos investigativos frecuentemente involucran a diversas universidades u organizaciones, que pueden sumar sus recursos individuales para satisfacer los requerimientos originados de todos los actores involucrados. En sus redes, coexisten computadores dedicados o de uso compartido con diferentes características y sistemas operativos.

Conseguir una mayor potencia de cálculo, de almacenamiento y aprovechamiento de recursos, combinando los recursos computacionales de estas organizaciones, constituye un nuevo paradigma de programación distribuida conocida como "Computación *Grid*" propuesta por Ian Foster y Carl Kesselman en 1997(I Foster, *et al.*, 1998).

La computación *Grid* es un término referente al uso coordinado de todo tipo de recursos (cómputo, almacenamiento y aplicaciones), de múltiples dominios administrativos para alcanzar un objetivo común. Esto quiere decir que los recursos empleados no tienen por qué estar sujetos a un control centralizado.

Esto supone una nueva forma de computación distribuida, en la cual, los recursos pueden ser heterogéneos (diferentes arquitecturas, supercomputadores y clústeres) y se encuentran conectados mediante redes de área extensa (por ejemplo, Internet).

Antecedentes y planteamiento del problema

Actualmente existen varios software que permiten desarrollar sistemas *Grid*, tales como: *Globus*, *Avaki*, *Entropia*, *Fura*, entre otros. Estos, proveen la base para construir sistemas *Grid*, pero necesitan integrarse con otros componentes, tales como: planificadores, administradores de recursos locales y protocolos de seguridad (Beristes, *et al.*, 2003).

Una de las herramientas de software más importante y comúnmente empleada es *Globus Toolkit*. Esta constituye una herramienta de código abierto que proporciona un conjunto de servicios para acceder de forma segura a múltiples dominios administrativos y compartir recursos distribuidos (Beristes, *et al.*, 2003; Martín Llorente, 2007). Esta herramienta incluye varios componentes que permiten la administración de datos, trabajos o tareas y aspectos referentes a la seguridad en su operación.

En el Centro de Estudios de Informática (CEI) existe un clúster de cómputo dedicado (nombrado *heaven*), además de otras máquinas disponibles. En la Universidad Central "Marta Abreu" de Las Villas (UCLV) existen, de manera similar, otros clústeres dedicados. Estos recursos se encuentran aislados unos de otros y apenas cumplen con las demandas locales de cada centro. Uniendo las capacidades de estos clústeres y de los computadores distribuidos en el entorno universitario, se podría ofrecer una potencia de cómputo nada despreciable.

Actualmente, la universidad no cuenta con una infraestructura para compartir todos los recursos disponibles, lo que implica desperdiciar capacidades de cómputo. Aunque este trabajo no se focaliza en implementar dicha infraestructura, se propone un diseño para implementar un sistema *Grid* universitario.

Utilizando la herramienta *Globus Toolkit*, la cual se ha convertido en el estándar de facto para construir sistemas y proyectos *Grid* a escala mundial, el clúster del CEI pudiera dotarse de una mayor capacidad de cómputo, que satisfaga las necesidades de los proyectos investigativos que se desarrollan actualmente.

De ahí la necesidad de que se explore esta tecnología, pues hasta el momento no se encuentran trabajos en la UCLV referidos al tema. De lo cual se derivan las siguientes preguntas de investigación:

1. ¿Cómo configurar la herramienta *Globus Toolkit* para que se integre con el administrador de recursos *Torque* ya instalado en el clúster del CEI?

2. ¿Es posible integrar un clúster dedicado y otros computadores a través de una infraestructura *Grid* implementada sobre *Globus*?

Objetivo general:

Se propone como objetivo general:
Diseñar e implementar una infraestructura de *Grid* para el Centro de Estudio de Informática (CEI), que permita flexibilizar el empleo de los recursos computacionales del clúster existente y de las computadoras del centro.

Para dar cumplimiento a este objetivo general se proponen los siguientes objetivos específicos:

1. Describir las características, componentes y herramientas necesarias para implementar una infraestructura de *Grid*.
2. Proponer un diseño de *Grid* para el *CEI* considerando los componentes de la red de comunicación y los servicios necesarios.
3. Implementar este diseño empleando la herramienta *Globus ToolKit* 5.2.2 en un entorno de prueba virtual.
4. Evaluar el rendimiento del diseño propuesto.
5. Realizar una propuesta que permita extender la tecnología *Grid* al entorno universitario.

Hipótesis:

Teniendo en cuenta los elementos teóricos antes expuestos, así como las interrogantes existentes, se plantea la siguiente hipótesis general de investigación:
La implementación con *Globus* de una infraestructura *Grid*, permite la ejecución de aplicaciones (paralelas o secuenciales), ofreciendo resultados similares a los obtenidos al ejecutar dichas aplicaciones sobre un clúster dedicado.

Estructura del trabajo:

El trabajo se estructura esencialmente en tres capítulos.

- El capítulo 1 está dedicado, en su primera parte, a la descripción de los diferentes tipos de computación distribuida, y una breve historia del surgimiento de la computación *Grid*, para luego centrarse en esta última, explorando y explicando su arquitectura, sus

principales componentes y las capacidades de esta tecnología. Finalmente, se presentan las principales herramientas para la implementación de la computación *Grid*, destacándose el *Globus Toolkit*.

- En el capítulo 2 se describen los principales servicios y componentes de software, necesarios para el diseño de una infraestructura de un mini-grid virtual; la segunda parte del capítulo está dedicada al proceso de instalación y configuración de la herramienta *Globus Toolkit5*.2.2.
- En el capítulo 3 se discuten los resultados obtenidos al ejecutar una aplicación paralela, para comprobar el correcto funcionamiento del mini-grid. Además, se realizan dos propuestas para extender el *Grid* al entorno universitario.

Al final de cada capítulo se muestran conclusiones parciales sobre los temas abordados. Finalmente, se presentan las conclusiones generales, las recomendaciones, las referencias bibliográficas y los anexos, dedicados a describir el proceso de instalación y configuración de la herramienta *Globus Toolkit*.

Capítulo 1

Fundamentos teóricos de la computación *Grid.*

1 FUNDAMENTOS TEÓRICOS DE LA COMPUTACIÓN *GRID*.

Este capítulo está dedicado a ilustrar la teoría de la computación *Grid*, partiendo de la computación distribuida. Luego se describen los principales conceptos del *Grid* y las capacidades que ofrece un sistema *Grid* a sus integrantes, además sus diferentes tipos y topologías. Después se exploran las principales herramientas utilizadas para diseñar sistemas *Grid*, enfocando específicamente en el *Globus Toolkit*.

1.1 Elementos de la computación distribuida.

1.1.1 Evolución de la computación distribuida

La necesidad de comunicación, intercambio de datos, compartir recursos y resolver problemas complejos no es una meta actual, sino que ha visto la luz desde el surgimiento de la computación. En los años cuarenta, del pasado siglo, surgieron los primeros intentos para desarrollar tecnologías basadas en computación de alto rendimiento. Estos intentos se reflejaron en el proyecto *Manhattan*. En este proyecto se pretendía desplegar capacidades de cómputo avanzada (Silva, 2006).

Con el paso del tiempo y específicamente en el año 1958, comenzó el uso de circuitos transistores en las computadoras, en lugar de válvulas al vacío. Este hecho revolucionó la era de la computación, debido a su menor tamaño y su bajo consumo de energía (Mejía Mesa, 2004).

En 1962, J.C.R. Licklider, escribió un ensayo sobre el concepto de Red Intergaláctica, donde todo el mundo estaba interconectado para acceder a programas y datos desde cualquier lugar del planeta (Mejía Mesa, 2004). Después de la dominada "crisis de los misiles de Cuba", el departamento de defensa de los Estados Unidos notó la necesidad de una forma de comunicación más segura, y como solución a este problema, fue necesario construir una red que proporcionara una comunicación fiable. Con este objetivo se creó en el año 1969 la red *ARPANET*[1] (*Advanced Research Projects Agency NetWork*)(Mejía Mesa, 2004).

[1] Todos los términos provenientes del idioma inglés están escritos en cursivas.

Las décadas de los setenta y de los ochenta, se caracterizaron por grandes éxitos en el campo de la computación: como fue la creación del Ethernet, el surgimiento del protocolo *TCP/IP*, la creación de dos nuevas redes: *PRNET* por *Packet Radio*, de la Universidad de Hawái, diseñada por Norm Abramson y capaz de conectar siete computadoras en cuatro islas, y *SATNET*, una red conectada vía satélite que enlazaba dos naciones: Noruega e Inglaterra. Además, se fabricaron computadoras personales y se evolucionó en la capacidad del Hardware (Mejía Mesa, 2004). Todo esto, junto a la creación de la Internet, condujo a las grandes empresas de software y a los científicos a pensar en nuevos paradigmas de computación que fueran capaces de compartir, explotar y manejar los recursos no aprovechados en cada organización que desea conectarse con otras organizaciones a través de una red. Dicho paradigma se denomina: computación distribuida. Desde la década de los noventa y hasta el presente, la computación distribuida desempeña un papel importante en los grandes logros científicos de diversos campos.

La computación distribuida se define según Liu (Liu, 2004) como: "conjunto de computadores independientes, interconectados a través de una red y que son capaces de colaborar para realizar una tarea".

1.1.2 Tipos de computación distribuida

Se puede clasificar la computación distribuida, según Silva (Silva, 2006), en tres ramas: punto a punto (*peer to peer*), que utiliza la comunicación sin servidor; computación en Internet (computación de ciclos redundantes), que trata de aprovechar los ciclos ociosos de las diferentes computadoras conectadas a la red, y por último, la computación *Grid*, para cerrar la brecha existente entre el cliente/servidor y los servicios web.

1.1.2.1 Computación en Internet

Es un modelo de la computación distribuida que surgió gracias a la creación de la Internet. Se trata de aprovechar el poder de cómputo de todo tipo de dispositivos conectados a la red global, voluntariamente, cuando los usuarios no utilizan sus máquinas, tanto: ordenadores, portátiles, así como otros dispositivos. El objetivo general es dividir un trabajo complejo en fragmentos pequeños y remitirlo a las máquinas de los usuarios conectados para procesarlo, y

posteriormente enviar los resultados a los servidores distribuidos en todo el planeta (Silva, 2006).

Este modelo consiste en el uso de muchas máquinas alrededor del mundo, aunque algunas de ellas tienen un nivel de procesamiento bajo, eso no afecta la capacidad de procesamiento general esperado, debido a que todas las máquinas actúan conjuntamente, para dar potencia de cálculo igual o superior a una supercomputadora.

El modelo tiene la ventaja de ser de bajo costo, además, posee una enorme capacidad de procesamiento. Su desventaja consiste en ser público, lo cual trae consecuencias no deseadas, tales como: ataques maliciosos y la falsificación en los datos que reciben los usuarios. Además, presenta otros aspectos negativos como la escasez de características avanzadas de agrupamiento, tales como: el tiempo compartido, servicios de información, capacidad de búsqueda y gestión de recursos. La computación en Internet, sin embargo, ha tenido éxito a través del proyecto *SETI@home* (*Search for ExtraTerrestrial Intelligence*) (Silva, 2006). Este proyecto consiste en la búsqueda de señales extraterrestres inteligentes. Después de su lanzamiento creció potencialmente y se convirtió en la red distribuida más grande y potente del mundo, con millones de usuarios (Silva, 2006).

El problema de la seguridad, dio lugar a un grupo de la Universidad de California, con sede en Berkeley, para que desarrollara *BOINC* (*Berkeley Open Infrastructure for Network Computing*) como una plataforma de seguridad. Gracias al uso de *BOINC*, el proyecto *SETI@home* dispone de dos *PetaFLOPS*[2], casi el doble de la potencia del mayor superordenador del mundo. Además, se lanzaron varios proyectos bajo la misma plataforma en diversos campos.

1.1.2.2 Punto a Punto

Este modelo conocido como (*P2P*) consiste en una red de computadoras conectadas entre sí, donde cada computadora (nodo) es dominado "par", debido a que actúa como servidor y cliente al mismo tiempo, o sea, en este tipo de redes no hay servidores centralizados. Aunque en la práctica, no es del todo cierto. Estas redes toman ventajas de los recursos disponibles en

[2] 1 petaflops=10^15 bytes.

los nodos conectados como: ciclos, recursos, contenidos y otras. El acceso a estas redes se produce en base de direcciones *IP* (Silva, 2006).

Las redes basadas en el modelo *P2P* aprovechan el ancho de banda global de todas las máquinas conectadas, y mientras más computadoras están conectadas a la red, mayor es el ancho de banda y la descarga será mucho más rápida.

Este modelo presenta características favorables que la distingue de las redes normales, entre estas cabe mencionar las siguientes:

- Escalabilidad: en este tipo de redes, a mayor cantidad de usuarios conectados, mejor rendimiento y mayor potencial de recursos. En cambio, en las redes normales, bajo la arquitectura de servidor-cliente, los servidores distribuyen los recursos y el ancho de banda entre los usuarios, lo cual significa lentitud en la transferencia de datos a medida que crece la totalidad de clientes.
- Robustez: Las redes *P2P* aumentan la robustez en casos de fallos, debido a su naturaleza distribuida.
- Descentralización: Estas redes, por definición, son descentralizadas y todos los nodos son iguales. No existen nodos con funciones especiales, y por tanto, ningún nodo es imprescindible para el funcionamiento de la red.
- Anonimato. Se prefiere que en estas redes quede anónimo: el autor de un contenido, el editor, el lector, el servidor que lo alberga, y la petición para encontrarlo siempre que así lo necesiten los usuarios. Muchas veces el derecho al anonimato y los derechos de autor son incompatibles entre sí.

Las redes *P2P* se clasifican en tres ramas, según Silva (Silva, 2006):

- Redes *P2P* centralizadas: Tienen una arquitectura que consiste en un único servidor de punto de enlace entre los nodos, en él se almacenan los nodos que contienen el contenido, pero presentan limitación en la escalabilidad, robustez y la privacidad de los clientes; lo cual viola las principales características de las redes *P2P*. *Napster* y *Audiogalaxy* son algunas de las aplicaciones que se basan en este tipo de redes.
- Redes *P2P* híbridas, semi-centralizadas o mixtas: De igual forma, tienen un servidor central que administra los recursos y el ancho de banda, pero no guardan información acerca de las identidades de los nodos. Estas redes suelen funcionar de manera que se puedan agregar nuevos servidores, que a su vez gestionan los recursos. En caso de fallo

de los servidores, los nodos pueden seguir conectados entre sí. De las aplicaciones que siguen esta filosofía se destacan: *eDonkey* y *BitTorrnet*.

- Redes *P2P* "puras" o totalmente descentralizadas: Estas redes son las más comunes y deseables en la comunidad de compartición de recursos, puesto que no se necesitan servidores para gestionar o enrutar las direcciones entre los nodos. Ellos se comunican directamente y actúan como servidor y cliente. Las aplicaciones de este modelo son reconocidas a nivel mundial, entre ellas se encuentran: *Freenet*, *Ares Galaxy*, *Kademlia*, y otras (Silva, 2006).

La seguridad y el derecho de autor son los principales aspectos que ponen las redes *P2P* en debate, donde sus defensores desean aumentar la seguridad y defienden la filosofía de compartir los recursos (archivos, música, videos, etc.), los oponentes argumentan que mientras los sistemas *P2P* anónimos pueden apoyar la protección de discursos impopulares, protegen actividades ilegales no protegidas bajo el derecho a la libertad, por lo tanto, opinan que las desventajas son mayores que las ventajas que ofrecen.

1.2 Principios de la computación *Grid*.

1.2.1 Surgimiento de la computación Grid

La computación *Grid* no surgió de una forma espontánea, sino que tiene sus raíces desde el nacimiento de la Internet; las máquinas conectadas a la red dejaban capacidades de cómputo inutilizadas. Las capacidades de cómputo requeridas en ambiciosos proyectos científicos, la simulación a gran escala y la toma de decisiones a partir de un gran volumen de datos; no han encontrado una solución total en las herramientas disponibles de las tecnologías actuales, donde ni las grandes supercomputadoras pueden satisfacer sus necesidades de procesamientos. Además, en ocasiones, no solo se busca la capacidad de procesamiento, sino que, también se espera lograr un nivel de conectividad y cooperación entre proyectos científicos a gran escala. Con estas ideas y problemáticas en mente, los científicos percibieron que al unir estos recursos, podría lograrse una fuente infinita de cómputo, capaz de sustituir o al menos competir con las supercomputadoras, que son poseídas únicamente por los gobiernos, empresas multinacionales y universidades reconocidas.

Los antecedentes de la computación *Grid* vieron la luz en la década de los ochenta. Se destacaron las tecnologías basadas en *MPI*[3] (*Message Passing Interface*), *HPF* (*High-Performance Fortron*) y *PVM* (*Parallel Virtual Machine*). Esta última es una biblioteca compatible con C estándar y Fortran, donde un determinado programa se prepara para ser ejecutado en paralelo. Además, tiene cierta modularidad, en la que cada máquina realiza una tarea específica; pero en su totalidad forman o cumplen el cometido del programa, dar una solución al problema que se plantea (Fernández, 2012).

Al inicio de la década de los noventa, el enfoque cambió hacia la coordinación, distribución y colaboración, constituyendo conceptos fundamentales de la computación *Grid*, y se desarrollaron investigaciones. Como resultado, surgió lo que se conoce como "*metacomputing*", que fue utilizado para describir los esfuerzos y conectar los centros de supercomputación de EE.UU (Café, 2012). Posteriormente se lanzaron dos proyectos de "*metacomputing*" en 1995, que influyeron en la evolución de las principales tecnologías *Grid*, estos son: *FAFNER* (*Factoring via Network-Enabled Recursion*), proyecto que aspiraba a calcular el factorial de los números muy grandes e *I-WAY* (*Information Wide Area Year*) que buscaba enlazar supercomputadores utilizando las redes existentes (Café, 2012).

Finalmente, en septiembre del año 1997 Ian Foster y Carl Kesselman, "los padres del *Grid*", dieron a conocer desde el Laboratorio Nacional Argonne (*Argonne National Laboratory*), el nacimiento de la computación *Grid*, en un taller denominado "Construyendo una *Grid* Computacional". El dúo Foster-Kesselman publicó en ese mismo año un documento llamado "*Globus*: Herramientas para la Infraestructura a *Metacomputing*" (*Globus*: a *Metacomputing Infrastructure Toolkit*) y en 1998, publicaron "La *Grid*: *Anteproyecto para una nueva Infraestructura Computacional*" (The *Grid*: *Blueprint for a New Computing Infrastructure*), conocido como "la biblia *Grid*" (I Foster, *et al.*, 1998).

Existes diversos conceptos sobre la computación *Grid*, uno de ellos elaborado por los "padres del *Grid*", Foster, Kesselman y Tuecke, en el cual plantean la existencia de organizaciones virtuales (OV) como punto de partida de este enfoque. Definen la organización virtual como "un conjunto de individuos y/o instituciones definidas por reglas que controlan el modo en que

[3] Una vez que se pone el nombre completo de un término proveniente del inglés, aparecerá abreviado en el resto del trabajo.

comparten sus recursos" (I Foster, *et al.*, 2001). Básicamente, son organizaciones unidas para lograr objetivos comunes. Ellos proponen el *Grid* como un modelo de trabajo para "compartir recursos en forma coordinada y resolver problemas en organizaciones virtuales multi-institucionales de forma dinámica" (I Foster, *et al.*, 2001). De esta manera, varias instituciones pueden formar distintas OV e incluso formar parte de más de una al mismo tiempo, realizando diferentes roles e integrando distintos recursos.

Otro concepto de la computación *Grid* fue elaborado por una de las asociaciones dedicadas al desarrollo de esta tecnología (*Grid Computing Information Center*), donde definen el *Grid* como: "un tipo de sistema paralelo y distribuido que permite compartir, seleccionar y reunir recursos autónomos y geográficamente distribuidos en forma dinámica en tiempo de ejecución, dependiendo de su disponibilidad, capacidad, desempeño, costo y calidad de servicio requerida por sus usuarios" (Buyya, 2004). Se puede decir que el *Grid* es un nuevo modelo de computación distribuida, coherente al uso compartido de todo tipo de recursos, que pueden ser homogéneos o heterogéneos, entre organizaciones virtuales a través de redes.

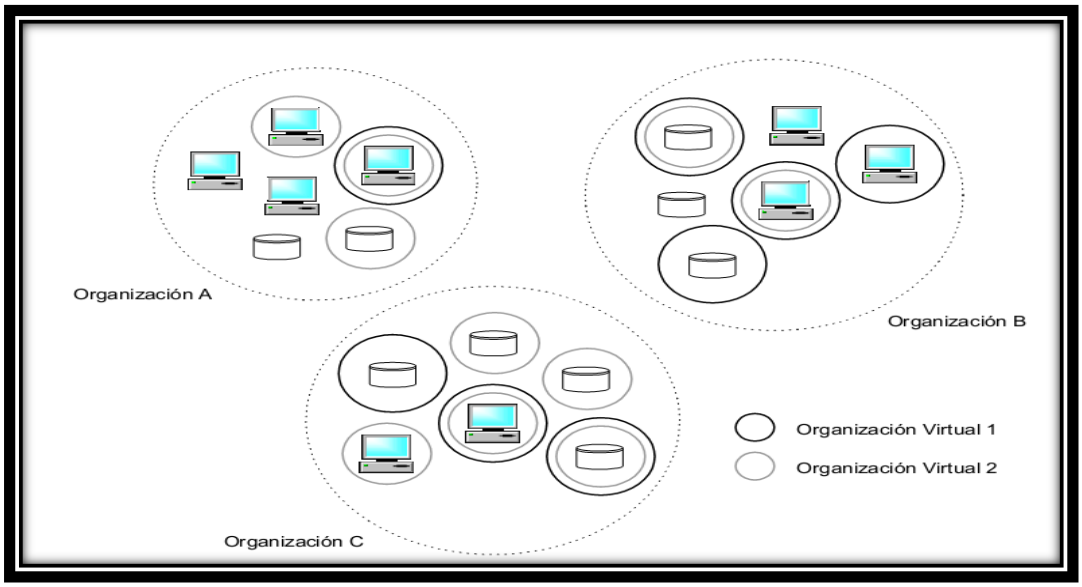


Figura 1-1: Organizaciones virtuales.

Se resalta el papel esencial de la computación paralela, *PVM*, los sistemas operativos distribuidos y el proyecto *SETI@home*, como elementos fundamentales que dieron los primeros pasos para el nacimiento del *Grid*; el cual tiene una arquitectura que lo diferencia de

otros modelos de computación y a su vez necesita estándares y servicios para dar complemento al correcto funcionamiento.

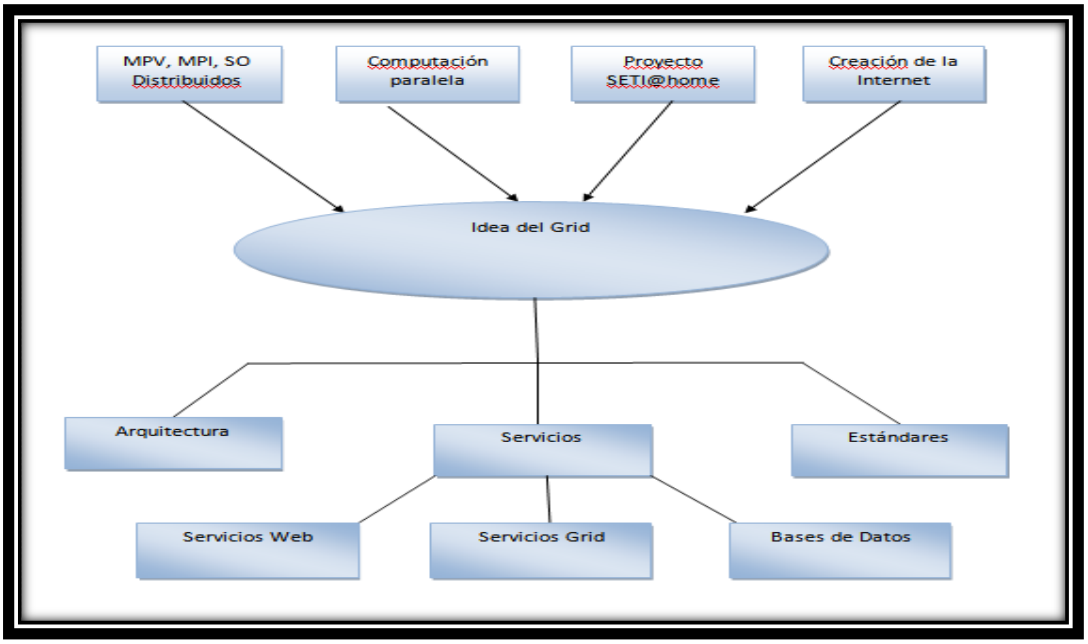


Figura 1-2 Idea general del *Grid*.

1.2.2 Conceptos y Componentes del Grid

La computación *Grid* cuenta con numerosos componentes para llevar a cabo su diseño y funcionamiento (Beristes, *et al.*, 2003).

1.2.2.1 Tipos de recursos

Un *Grid* es una colección de máquinas, a veces llamados: "nodos", "recursos", "miembros", "servidores", "clientes", "organizadores", y otros. Todos ellos contribuyen con un grupo de recursos al *Grid*. Estos recursos pueden ser globales, usados por todos los usuarios del *Grid*, mientras que otros pueden tener restricciones específicas.

1.2.2.2 Computación

La computación *Grid* cuenta con los ciclos de cómputo proporcionados por los procesadores, que se diferencian en velocidad, arquitectura y plataforma; como su principal recurso. Los ciclos de cómputo son aprovechados de tres maneras diferentes:

- Ejecutar aplicaciones existentes en un nodo disponible en el *Grid*, en lugar de ejecutarlo localmente. Existen dos prerrequisitos para ese escenario, la aplicación debe

ejecutarse remotamente y sin ocasionar gastos elevados e inesperados de recursos, así como la máquina remota debe encontrar cualquier hardware o software especial impuesto por la aplicación.

- Dividir un trabajo en sub-trabajos, elaborado por una aplicación diseñada para este propósito, de manera que ellos son ejecutados en paralelo en diferentes procesadores.
- Ejecutar una aplicación varias veces en diferentes nodos del *Grid* para lograr confiabilidad en los resultados obtenidos y escalabilidad, que es una medida de cómo se utilizan eficazmente los procesadores.

La escalabilidad se expone en sistemas *Grid* de dos maneras: absolutamente escalable, cuando los procesadores ejecutan dos veces una aplicación completa en la mitad del tiempo; y escalabilidad limitada, que se manifiesta en dos casos: el primero ocurre cuando las aplicaciones son divididas en un número limitado de partes ejecutables, y el segundo sucede si las partes experimentan alguna otra contención para recursos de almacenamiento.

1.2.2.3 Almacenamiento

El almacenamiento de datos es un recurso esencial en los sistemas *Grid*, ya sea, como memoria unida al procesador, o como almacenamiento secundario. La memoria unida al procesador mantiene un rápido acceso, aunque es inconstante. Por lo que es mejor utilizarla con el fin de guardar temporalmente datos y servicios para las aplicaciones ejecutadas. Mientras, el almacenamiento secundario, incrementa la calidad, capacidad, fiabilidad y permite compartir los datos.

Los sistemas *Grid* mayormente usan sistemas de archivos de red montables: *AFS* (*Andrew File System*), *GPFS* (*General Parallel File System*) y *NFS* (*Network File System*), entre otros.

Con un sistema de archivo unificado se incrementa la capacidad mediante el uso del almacenamiento en múltiples máquinas, eso implica que las bases de datos pueden expandirse a varios dispositivos de almacenamiento, borrando así las restricciones de máxima capacidad impuestas por los sistemas de archivos de los sistemas operativos.

De igual manera se mantiene un único espacio uniforme para el almacenamiento, lo que facilita referenciar los datos que se alojan en un sistema *Grid*, sin considerar las ubicaciones exactas de los datos. Existen sistemas de archivos más avanzados que duplican conjuntos de datos, para proveer la redundancia y el aumento del rendimiento y la fiabilidad.

1.2.2.4 Comunicaciones

La evolución de las redes de comunicaciones y el incremento del ancho de banda hace que la comunicación de datos, entre las máquinas de un sistema *Grid*, sea un recurso importante. Tales comunicaciones pueden ser internas y externas.

La comunicación interna se refiere a todo tipo de intercambio de datos y comunicación entre las máquinas dentro del sistema *Grid*; se utiliza normalmente para enviar trabajos y sus datos hacia algunos puntos dentro del *Grid*. También se emplea, cuando algún trabajo exige procesar una cantidad enorme de datos que no residen en la misma máquina donde se pretende ejecutar el trabajo. En cambio, la comunicación externa se refiere al uso de Internet por parte de algunas máquinas. Habitualmente, se usa cuando se quiere construir motores de búsqueda. En algunos casos, se deben utilizar redes de altas prestaciones para satisfacer demandas de trabajos que transfieren cantidades muy grandes de datos.

1.2.2.5 Planificación, Reservación, y Barrido

En un sistema *Grid* existen dos tipos de planificación. En sistemas simples, el usuario selecciona la máquina donde se ejecutará su trabajo y luego envía el trabajo a dicha máquina; y en los sistemas avanzados, un planificador es el encargado de seleccionar la máquina más apropiada para ejecutar el trabajo.

En un barrido "*scavenging*" de un sistema *Grid*, cualquier máquina que se vuelve ociosa, sin trabajo local que ejecutar, informa su estado al nodo de administración del *Grid*. Este nodo de administración asigna a esta máquina el próximo trabajo que se satisface por los recursos de la misma.

Los recursos del *Grid* pueden "reservarse" por adelantado para un conjunto de trabajos designados. Esto se hace para reunir fechas tope y garantizar la calidad del servicio. Cuando las políticas lo permitan, los recursos, reservados de antemano, podrían ser recogidos para ejecutar trabajos de menor prioridad.

1.2.2.6 El Software y las Licencias

Un sistema *Grid* no significa solamente potencia y rendimiento, sino también ahorro para las organizaciones que lo utilizan. Esto se refleja cuando el *Grid* necesita un determinado software costoso. Existen dos maneras de manejar el software, instalarlo en máquinas

particulares y enviar los trabajos, ejecutados en otras máquinas que necesitan este software, a la máquina que lo posee. Además, se puede hacer algún tipo de arreglo de licencia de software que permita instalarlo en todos los nodos, pero esto puede limitar el uso simultáneo del mismo.

1.2.2.7 Los Trabajos y las Aplicaciones

Habitualmente se usa el término "aplicación" como el nivel más alto de una porción de trabajo en el *Grid*. Sin embargo, a veces el término "trabajo" se usa equivalentemente.
Las aplicaciones pueden dividirse en cualquier número de trabajos individuales, a su vez, pueden dividirse en sub-trabajos.

1.2.3 Arquitectura Grid

El objetivo de describir la arquitectura *Grid* no es proporcionar una enumeración completa de todos los protocolos requeridos, servicios, *API* (*Application Programming Interface*) y *SDK* (*Software Development Kit*), sino para identificar los requisitos para las clases generales de componentes. Esta arquitectura es abierta y extensible, organizada en capas y sigue el modelo "reloj de arena" (I Foster, *et al.*, 2001), donde en el centro del reloj, "el cuello", se definen los protocolos (*TCP/IP* , *HTTP*), hacia donde pueden trazarse muchas conductas diferentes de alto nivel (la cima del reloj de arena); y que ellos pueden trazarse hacia muchas tecnologías subyacentes diferentes (la base del reloj de arena). Por la definición, el número de protocolos definidos al cuello debe ser pequeño.
La arquitectura *Grid* está designada a direccionar las necesidades de utilidad de cómputo, es decir, que permite utilizar los recursos de procesamientos propios de la compañía, sin tener que invertir más en adquirir capacidad extra de procesamiento. Principalmente, la arquitectura propuesta es una arquitectura de protocolos que definen los mecanismos básicos que permiten a los usuarios y a los recursos negociar, establecer, gestionar y utilizar la compartición de recursos.
Una arquitectura abierta, basada en un estándar, facilita la extensibilidad, la interoperabilidad, la portabilidad y la compartición de código (I Foster, *et al.*, 2001).

1.2.3.1 Capa de Infraestructura (Fabric)

Esta capa proporciona los recursos compartidos mediados por los protocolos *Grid*, estos recursos pueden ser: recursos computacionales, sistemas de almacenamiento y recursos de redes. Donde dichos recursos pueden ser entidades lógicas, tales como: sistema de fichero distribuido, clústeres, y otras. Los componentes de esta capa implementan las operaciones locales específicas de cada recurso. Existe de este modo una ajustada y sutil dependencia mutua, entre las funciones implementadas en dicha capa y las operaciones soportadas para compartir recursos.

Se puede agregar funcionalidad a esta capa, lo cual posibilita el soporte de operaciones en el nivel superior de manera más sofisticada, pero al mantener simple esta capa, se logra que el desarrollo de la infraestructura *Grid* sea menos complejo.

1.2.3.2 Capa de Conectividad (Connectivity)

En esta capa se implementan todo tipo de protocolos de comunicación, tales como: *TCP/IP, SSL* (*Secure Sockets Layer*) y *DNS* (*Domain Name System*), ya que los recursos necesitan comunicarse entre sí. Esta capa ofrece soporte de comunicación en la arquitectura *Grid*. También, debe ofrecer protocolos de seguridad como respuesta a la naturaleza distribuida y compartida de la computación *Grid*.

Se incluyen protocolos para identificar usuarios y recursos, pero cualquiera que sea la solución referente a la seguridad, deberá contar con las siguientes características: la delegación, la integración con varias clases de solución de seguridad local y el SSO (*Single Sign On*) que es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación,

1.2.3.3 Capa de Recurso (Resource)

En la capa de recurso se hallan dos tipos de protocolos, que se implementan sobre un recurso en particular. El primero de estos protocolos permite obtener información de este recurso. Dicha información, puede ser sobre la carga actual, características técnicas, y otras. El segundo protocolo se especializa en el control y la administración del recurso, y sus tareas son: el acceso al mismo, el arranque de procesos, la gestión, la parada, la monitorización, la contabilidad de uso y la auditoria del recurso. Esta capa se construye sobre los protocolos de la

capa anterior y la implementación de sus protocolos hace llamadas a la capa de infraestructura para acceder y controlar los recursos locales.

1.2.3.4 Capa de Recursos o Colectiva (Collective)

A diferencia de la capa anterior, esta gestiona y administra un conjunto de recursos para que ellos trabajen conjuntamente en una tarea común. Los servicios y protocolos que se encuentran en esta capa son los encargados de realizar la gestión y la administración. Los servicios de directorio permiten localizar los recursos de interés y la contabilidad, que permite calcular el costo de la utilización de varios recursos heterogéneos, y el acceso a datos distribuidos, que gestiona la replicación de datos; los planificadores (*Schedulers*) distribuidos, permiten asignar las tareas a cada recurso, la monitorización y diagnóstico de la ejecución en que se distribuye la ejecución de una aplicación.

1.2.3.5 Capa de Aplicación (Application)

En el nivel superior de la arquitectura residen los protocolos que permiten a una aplicación construida en término de servicio; que se comunique con cualquiera de las capas anteriormente definidas, incluso la capa de infraestructura. Esto se debe a que cada capa tiene protocolos bien definidos que proveen el acceso al uso de servicios, tales como: el manejo de recursos, acceso a datos, entre otros.

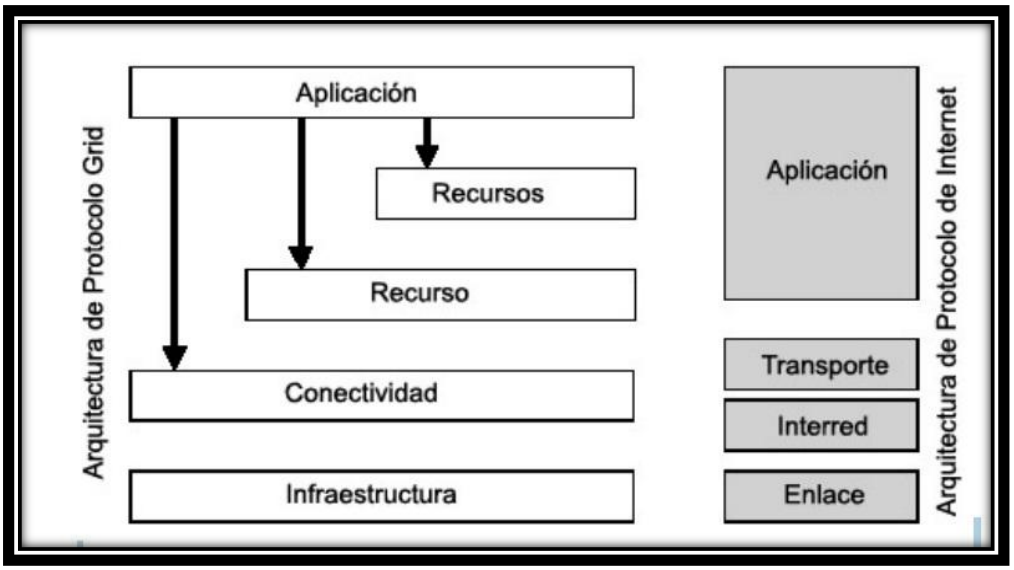


Figura 1-3 Arquitectura *Grid*.

1.2.4 Componentes del Software Grid

Los componentes de software en la arquitectura *Grid* son muy importantes a la hora de diseñar y poner la arquitectura en funcionamiento. En la siguiente sección se tratarán los principales componentes de software (Beristes, *et al.*, 2003).

1.2.4.1 Componentes de administración

Todo sistema *Grid* debe contar con componentes de administración, que le sirvan para realizar varias tareas. Entre ellos, no debe faltar el componente que mantiene los recursos disponibles del *Grid* en un registro, para decidir la asignación de trabajos. Otros componentes de suma importancia son los de medición, cuya función es determinar la capacidad disponible y la utilización de los nodos conectados al sistema; esta información se almacena con el propósito de planificar, determinar el estado, el congestionamiento y la recolección de estadísticas acerca del uso de recursos. La computación autónoma se debe al uso de software avanzado de administración *Grid*, que maneja automáticamente muchos de sus aspectos, su objetivo es la recuperación de varias fallas y pausas en el servicio, buscando la manera de lograr el procesamiento de la carga de trabajo actual.

1.2.4.2 Administración Grid distribuida

Un sistema *Grid* puede estar organizado en una jerarquía, formada de clústeres de clústeres, que corresponde con la topología de conectividad. Para lograr el aumento de la escalabilidad, el trabajo de administrar este tipo de *Grid* debe estar distribuido, al igual que los datos y la planificación.

1.2.4.3 Software donado (servidor)

Para que un nodo pueda formar parte del *Grid* es necesario que se asocie como miembro del mismo, e instale un software que permita el manejo de sus recursos, incorporándolos al sistema. Los mecanismos de autentificación e identificación para que una máquina pueda unirse al *Grid*, se llevan a cabo a partir de un procedimiento como el uso de certificados.

Los mecanismos de autentificación (*login*), pueden ser propios del *Grid* o los proporcionados por el sistema operativo. En un sistema *Grid*, la información de los nuevos recursos incorporados es distribuida a todos sus miembros. Las máquinas donantes cuentan con un

software que determina o mide cuán ocupadas se encuentran. Esta información se pondría a disposición del software de administración del *Grid,* con fines de planificación. El software instalado en una determinada máquina puede aceptar trabajos ejecutables del sistema de administración y ejecutarlos.

1.2.4.4 Software de sumisión

Cualquier máquina miembro del *Grid* puede ser usada para enviar trabajos a este. Sin embargo, en algunos sistemas *Grid,* esta función es implementada como un componente separado e instalado en los nodos de sumisión (*submission nodes*). Cuando el Grid es construido usando recursos dedicados en lugar de recolectados, usualmente se instala el software por separado en el escritorio de usuario (*Workstation submission*).

1.2.4.5 Planificadores

Uno de las herramientas más importante que incorporan los sistemas *Grid* son las de planificación. Este software, denominado "programación de trabajo", es el responsable de localizar el nodo donde se ejecutará el trabajo solicitado por el cliente. Los planificadores varían desde el más simple, que asigna ciegamente la próxima máquina que posee los recursos necesarios para el trabajo, hasta lo complejos, al implementar colas de prioridad, y el trabajo es ejecutado en un nodo según su prioridad, de manera similar se pueden poner restricciones o políticas a los planificadores.

Los planificadores más avanzados monitorean el progreso de los trabajos remitidos, manejando el flujo total de trabajo. Si un trabajo se ve afectado por una falla en un nodo o por otra situación, el planificador lo reacomoda en otro nodo del *Grid* de ser posible; siempre y cuando el trabajo remitido no sea el causante de la falla. Típicamente, los trabajos tendrían asociados distintos códigos de finalización, algunos de los cuales serán susceptibles de re-planificación y otros no.

1.2.4.6 Comunicaciones

Un sistema *Grid* debe ser capaz de incluir software que ayude a los trabajos a comunicarse entre sí. Se puede dividir un trabajo en un grupo de sub-trabajos, ellos deben ser capaces de localizar otro sub-trabajo y establecer comunicación. El estándar abierto *MPI* se incluye con frecuencia como parte del sistema *Grid* para dar este tipo de soporte de comunicaciones.

1.2.4.7 Observaciones y medidas

Con frecuencia el software donado incluye alguna herramienta que mide la carga corriente y la actividad en una máquina dada, este software es conocido como sensor de carga. Esta información no solo es útil para planificar, sino que también lo es para descubrir patrones de uso global en el *Grid*. Estos patrones permiten predecir comportamientos, lo cual deriva en un mejor aprovechamiento del *Grid*.

1.3 Conceptos avanzados en el *Grid*.

1.3.1 Las capacidades de la computación Grid

La computación *Grid* ofrece la resolución de problemas grandes y costosos, a nivel computacional y de gestión de la información, que una simple máquina o clúster es incapaz de resolver. Las potencias y capacidades que tiene este sistema están dadas por varios factores que se describirán a continuación (Beristes, 2002).

1.3.1.1 Explotar recursos subutilizados

En la mayoría de las organizaciones o centros de investigaciones, existe una gran cantidad de recursos computacionales subutilizados. Se estima que los computadores de escritorio realmente están ocupados alrededor del 5% del tiempo en un día laboral, incluso algunos servidores pueden estar relativamente ociosos, lo que significa una gran pérdida de recursos que pudieran aprovecharse. La computación *Grid* provee un marco de trabajo para explotar estos recursos de manera eficiente.

Los recursos que pueden ser aprovechados son: los ciclos de la *CPU* y el almacenamiento que pueden ser agregado a un almacenamiento virtual. En la actualidad, existen varias áreas de aplicación que requieren acceso a fuentes de datos distribuidas: bibliotecas digitales, que proveen servicios para descubrir, manipular y presentar objetos digitales; sistemas *Grid* para el procesamiento de datos distribuidos y archivos persistentes para mantener colecciones de datos, durante un potencial cambio en la tecnología subyacente.

1.3.1.2 Capacidad de procesadores (CPUs) paralelos

El potencial de esta característica es uno de los rasgos más importantes de un *Grid*, donde las aplicaciones pueden ser escritas para usar algoritmos que pueden ser divididos en partes

ejecutables independientes. Una aplicación *Grid* intensiva en uso de *CPU* puede ser pensada como varios sub-trabajos más pequeños, cada uno ejecutándose en una máquina diferente.
Otras fuentes de contención de inter-trabajo en una aplicación de *Grid* paralela, incluyen latencias de comunicaciones de mensajes entre los trabajos, la red, las capacidades de comunicación, los protocolos de sincronización, el ancho de banda de entrada-salida, los dispositivos de almacenamiento, y latencias que interfieren con los requerimientos de tiempo real.

1.3.1.3 Recursos virtuales y organizaciones virtuales para la colaboración

La computación *Grid* tiene la capacidad para proveer un entorno de colaboración entre una audiencia amplia. Dicha capacidad, se incrementa entre las organizaciones utilizando un sistema *Grid* en vez de un sistema distribuido típico. El incremento se produce mediante la virtualización de objetos o de organizaciones.

1.3.1.4 Acceso a recursos adicionales

Los recursos adicionales pueden proporcionarse en número y capacidad variable. La capacidad de comunicación no solo es aprovechada para el acceso a Internet, sino también para proporcionar caminos redundantes necesarios para soportar potenciales fallas de la red y tráfico de datos excesivo.

1.3.1.5 Balance de recursos

El *Grid* provee un efecto de balance de recursos a través de la planificación de trabajos. Mediante un planificador avanzado se puede minimizar el tráfico en las comunicaciones, cuando los trabajos se comunican vía Internet, o con recursos de almacenamientos, esto puede conducir a un balanceo de recursos.

1.3.1.6 Confiabilidad

En algunos sistemas se suele utilizar hardware costoso para lograr la confiablidad, a través del uso de chips con circuitos redundantes. El *Grid* garantiza la confiabilidad de manera más barata y eficiente, mediante el uso de un conjunto de componentes independientes de un mismo tipo. Dichos componentes son relativamente baratos y geográficamente dispersos, por

lo cual, ante una falla específica en una determinada ubicación, las otras partes no deberían ser afectadas.

1.3.1.7 Administración

La meta de manejar sistemas heterogéneos uniformemente y virtualizar los recursos de un *Grid*, generará nuevas oportunidades para gestionar una infraestructura de tecnología de información más grande y más dispersa. El *Grid* permite manejar las prioridades entre diferentes proyectos. Cuando el sistema requiere mantenimiento, los trabajos pueden redirigirse a otras máquinas sin frenar los proyectos involucrados.

1.3.2 Áreas de aplicaciones de la computación Grid

La computación *Grid* suministra a los usuarios acceso seguro, consistente, penetrante y barato. Su infraestructura debe proporcionar a los usuarios un servicio seguro en todos los niveles: capacidad de cómputo, integridad de datos, seguridad de acceso, entre otros.

El servicio debe ser consistente, basado en estándares, de esta manera el acceso y las operaciones sobre el *Grid* estarán definidas por dichos estándares, evitando la heterogeneidad.

La idea de penetración, no es tanto la posibilidad de acceder a cualquier recurso, sino la posibilidad que tiene cualquier usuario de acceder al sistema. De esta manera, se asegura que una vez conectado desde cualquier punto, pueda extraer del *Grid* toda la potencia requerida.

Por último, el acceso y uso del *Grid* debe tener un coste económico que le haga atractivo para que su utilización se universalice.

Existe una multitud de aplicaciones reales que hacen uso de mini-grid, casi todas ellas están centradas en el campo de la investigación, en el terreno de las ciencias físicas, médicas y del tratamiento de la información.

La computación *Grid* se aplica en muchas áreas de trabajo, de ellas existen cinco grandes áreas de trabajo determinadas por las necesidades de cálculo, espacio para el almacenamiento de los datos y tiempo de respuesta. Las áreas son: supercomputación distribuida, sistemas distribuidos en tiempo real, proceso intensivo de datos, servicios puntuales y entornos virtuales de colaboración

1.3.3 Topologías de Grid

Para la definición de una topología *Grid* se toman en cuenta dos aspectos importantes: las organizaciones que utilizará el *Grid* y los clústeres involucrados. Según sea la topología se definen los mecanismos de seguridad, el acceso a los datos y la integración de recursos, aplicaciones y servicios. La complejidad del diseño del *Grid* está dada por la cantidad de organizaciones que actuarán sobre él. Además, mientras más organizaciones requieran el acceso a los recursos, se incrementarán los requisitos de seguridad y de esta forma se vuelven más complejas la disponibilidad y la capacidad, para ello, existen tres tipos de topologías (Beristes, *et al.*, 2003):

1.3.3.1 Intragrid

Es la más simple entre las topologías, se implementa en una sola organización, proporcionando un conjunto de servicios básicos de *Grid*. Dicha organización puede tener un conjunto de máquinas que tiene un dominio común de seguridad y comparte datos en una red privada. Esta topología se caracteriza por tener: un ancho de banda disponible y alto, un solo proveedor de seguridad y un único ambiente dentro de la red. Además, es fácil de diseñar, proporciona de forma estática los recursos computacionales y facilita la compartición de datos entre los sistemas *Grid*.

1.3.3.2 Extragrid

Es la unión de dos o más *intragrid*, involucrando a más de un proveedor de seguridad, además, se incrementa el nivel de la complejidad en la gestión. Las características más notables de una *extragrid* son: múltiples organizaciones implicadas, la necesidad de conexión *WAN* remota y la seguridad dispersa.

Los recursos se proporcionan de forma dinámica, el diseño será más complicado y los servicios de información se hacen pertinentes, para asegurar que los recursos tengan acceso a la dirección del trabajo en el tiempo de ejecución.

1.3.3.3 Intergrid

Este tipo de topología requiere la integración dinámica de aplicaciones, recursos, clientes y cualquier organización autorizada para obtener el acceso al *Grid*. Se usa principalmente en empresas de diseño, industrias de ciencia, fabricantes, y por los negocios en la industria

financiera. Las características que definen esta topología son similares al *extragrid*, pero los datos y las aplicaciones deben ser globales y públicas, en caso que no lo sean, se deben modificar.

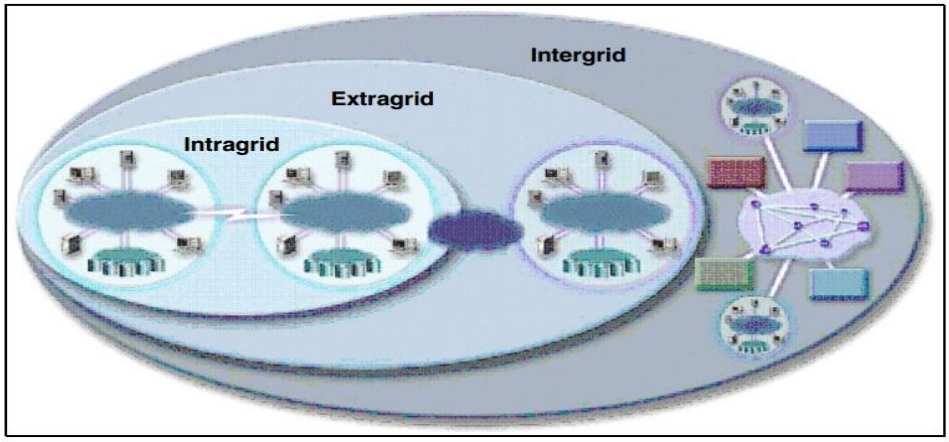


Figura 1-4 Topologías de *Grid.*

1.3.4 Modelos de administración de recursos:

Es sumamente elemental escoger el modelo correcto para la administración de recursos en un sistema *Grid*, pues de ello depende el éxito de una organización *Grid*. Según Silva (Silva, 2006) existen tres modelos para la administración, estos son:

1.3.4.1 Jerárquico

Este modelo introduce el concepto de componentes activos y pasivos, donde los pasivos pueden ser cualquiera de los siguientes:

- Recursos que pueden ser compartidos por los dueños, tales como: disco duro, *RAM* (*Random Access Memory*), ancho de banda, etc.
- Recursos consumidos (tareas), que pueden ser computacionales o no-computacionales.
- Trabajos, que son agrupados lógicamente por una o más tareas.
- Planificadores, que son componentes que trazan los trabajos con el tiempo a los recursos.

Los componentes activos pueden ser:

- Los servicios de información que describan: recursos, trabajos, planificadores, y agentes dentro del sistema de la administración de recursos.
- Usuarios que someten los trabajos para la ejecución.

- Agentes tales como: el mando del dominio que compromete los recursos para el uso dentro del dominio local; el mando de admisión para acomodar o rechazar los trabajos dependiendo de la carga del sistema; y el mando del trabajo para coordinar entre diferentes componentes dentro del sistema de administración de recursos.

Estos componentes pueden actuar mutuamente entre sí de muchas maneras. Por ejemplo, cuando un usuario somete un trabajo, se intercepta por el mando del trabajo y los agentes de admisión, quiénes determinan si es seguro agregar el trabajo a la cola de trabajo del sistema. El trabajo es enviado entonces al planificador, que realiza el descubrimiento del recurso y pide a los agentes del mando la determinación de la disponibilidad de este, a su vez, el planificador envía el trabajo al agente del despliegue que negocia para los recursos requeridos, y obtiene las reservaciones para los recursos.

1.3.4.2 Modelo del Dueño abstracto (AO)

Este modelo se basa en la noción de dueños abstractos de recursos, también conocidos, como corredores. A los usuarios de este modelo no les interesa quien posee un recurso específico, sino, el acceso, costo, y medios de pago para esos recursos.

A la fundación de este modelo, todos los recursos de los procesadores individuales e instrumentos del propio *Grid*, asignan dueños abstractos firmemente relacionados a los planificadores. Los clientes negocian con dueños abstractos enviando los objetos del recurso a través de las llamadas remotas del procedimiento. Algunas de las informaciones enviadas por el cliente incluyen los atributos del objeto de recurso, el estilo de la negociación, el acercamiento de la recogida y la autorización. El estilo de la negociación brinda información específica sobre si el trabajo será ejecutado inmediatamente, o devuelve el estado del trabajo: pendiente, cancelado, y así sucesivamente.

1.3.4.3 Modelo Económico

Este modelo incluye componentes de los dos modelos mencionados anteriormente. Es preferible su uso cuando se cuenta con recursos geográficamente distribuidos poseídos por organizaciones alrededor del mundo; con diferentes tipos de modelos de la administración de recursos. Además, brinda mejores incentivos a los dueños del recurso para compartirlos, ofreciendo los servicios financieros para la ganancia. Lo cual se traduce en el rendimiento de

la inversión para los usuarios y los dueños, a través de la mejora y la extensión de servicios computacionales que se crean eficazmente para el mercado de suministro y demanda. Para lograr estas metas, el modelo económico proporciona una administración con las herramientas y servicios, que permite a los usuarios y los dueños expresar sus requisitos. Sus componentes básicos son: aplicaciones del usuario, el planificador local, de recurso y el middleware del *Grid*.

1.3.5 Tipos de Grid

La computación *Grid* proporciona la integración de recursos computacionales de cualquier tipo: unidades de procesamiento, unidades de almacenamiento y unidades de comunicación; por lo que pueden clasificarse cinco tipos diferentes de *Grid,* según sea el recurso computacional:

1.3.5.1 Grid de cómputo (computational Grid)

La capacidad de procesamiento es el principal recurso para ser compartido entre los nodos, donde el *Grid* de cómputo agrega el poder de procesamiento desde los diferentes ordenadores distribuidos en un sistema *Grid* (Bart, *et al.*, 2003; Beristes, 2002; Magoules, *et al.*, 2009).

El *Grid* de cómputo se reconoce por las siguientes características:

- Está compuesto de clústeres que a su vez pueden ser de clústeres.
- Habilita la capacidad de barrido para mejorar el uso de los recursos.
- Proporciona el poder computacional para procesar trabajos a gran escala.
- Satisface el requisito comercial a la demanda para el acceso instantáneo a los recursos.

1.3.5.2 Grid de datos (Data Grid)

En este tipo de *Grid*, el principal recurso a ser compartido es la capacidad de almacenamiento. El desconocimiento de la ubicación de los datos por parte del usuario lo convierte en un proceso transparente.

El *Grid* de datos se enfoca en la responsabilidad de almacenar y proveer el acceso a los datos, a través de múltiples organizaciones, a los usuarios. Además, el mismo puede agregar nuevos conceptos como una base de datos federada. El *Grid* de datos también prepara los datos, dispositivos de almacenamiento y recursos de la red localizados en los distintos dominios administrativos; respeta las políticas locales y globales de cómo los datos pueden ser usados,

proporciona alta velocidad y un acceso fiable a los datos(Bart, *et al.*, 2003; Magoules, *et al.*, 2009).

1.3.5.3 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 descargar 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 (Bart, *et al.*, 2003; Magoules, *et al.*, 2009).

1.3.5.4 Barrido (Scavenging)

Es comúnmente utilizado a través de un 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 facilitan el control de estas máquinas cuando se encuentran libres. Las computadoras pueden integrarse a diferentes organizaciones virtuales, aunque pertenezcan a una misma organización real, de igual manera pueden integrarse a una organización virtual aunque pertenezcan a diferentes organizaciones reales.

El barrido normalmente es llevado a cabo de manera que no obstruya al usuario. Si la máquina permanece en estado de ocupada con un trabajo local no-*Grid*, el trabajo del *Grid* normalmente se suspende o se demora. Esta situación crea de alguna manera tiempos de

terminación imprevisibles para los trabajos del *Grid*, aunque no molesta a las máquinas donantes de recursos (Bart, *et al.*, 2003; Gangotena, *et al.*, 2009; Magoules, *et al.*, 2009).

1.3.5.5 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 una *meta-Grid*, con la habilidad para dirigir las peticiones hacia el *Grid* que soporta el modelo adecuado (Bart, *et al.*, 2003; Gangotena, *et al.*, 2009).

1.4 Herramientas de software.

Existen diversas plataformas de software, que proveen los servicios necesarios para la construcción de sistemas basados en *Grid*. De ellos se destacan los siguientes: *Avaki*, *Data Synapse, Entropia, United Devices, Globus y Platform Computing (Bart, et al., 2003; Beristes, 2002; Beristes, et al., 2003).*

Dichas plataformas se emplean según sea la necesidad de la empresa. Si el interés de la empresa consiste en implementar el *Grid* sobre los ciclos de *CPU*, o sea, el barrido (*Scavening*), los productos tales como: *Data Synapse, Entropia y United Devices* proporcionan el barrido de manera eficiente. En cambio, si el interés de la empresa radica en utilizar los recursos dedicados para implementar *Grid* computacional, es recomendable el uso de *Globus o Avaki*; debido a que ambos proveen los servicios y los componentes necesarios para llevar a cabo la implementación del *Grid* computacional.

De todos los productos anteriores se utilizará en el presente trabajo el *Globus Toolkit*, por varias razones, tales como(Martín Llorente, 2007):

- La mayoría de los otros productos son privados, en cambio, el *Globus* es un paquete de código abierto.
- El *Globus* dispone de las herramientas necesarias para compartir datos y el manejo de la seguridad.
- Es el estándar de facto más universal y utilizado por la mayoría de los proyectos *Grid* a nivel mundial.

1.4.1 Globus Toolkit

Es una implementación particular de una arquitectura *Grid*, desarrollada por el proyecto *Globus Alliance*, la cual proporciona un *middleware* de código abierto. Además, provee las bibliotecas y los componentes que permiten el desarrollo de aplicaciones *Grid* orientadas a servicios. Los componentes del núcleo de *Globus* abarcan las características básicas relacionadas con la seguridad, el acceso y la administración de recursos, el movimiento y la administración de datos, el descubrimiento de recursos, entre otros.

La estructura subyacente en este *middleware* es la de suma de servicios. Lo cual significa que el middleware se compone de múltiples servicios, que implementan los distintos componentes necesarios para integrar y administrar los recursos que conforman el *Grid* (Ian Foster, 2006).

1.4.2 Pirámides de Globus Toolkit

El *Globus* tiene dos pirámides de apoyo construidas sobre una infraestructura de seguridad. Estas son (Beristes, *et al.*, 2003):

- Administración de recursos (*Resource management*), que proporciona soporte para la asignación de recursos y la administración de trabajos. *Globus* no dispone de un planificador de trabajo para encontrar los recursos disponibles, sino que provee un conjunto de herramientas e interfaces para implementar un planificador.
- Administración de datos (*Data management*), esta pirámide proporciona el apoyo para transferir los archivos entre las máquinas en el *Grid* y para la dirección de estos traslados.

1.4.3 Componentes de las herramientas de Globus

Globus proporciona componentes para cada pirámide previamente presentada. Estos componentes son (Beristes, *et al.*, 2003):

- *GRAM/GASS*: Los componentes primarios de la pirámide de la administración de recursos son: el Gerente *Grid* para Asignación de Recursos, *GRAM* (*Grid Resource Allocation Manager*) y el Acceso Global al Almacenamiento Secundario, *GASS* (*Global Access to Secondary Storage*). El *GRAM* es el módulo que proporciona la ejecución remota y la dirección de estado de la ejecución. Mientras que el *GASS*

suministra el mecanismo para transferir el archivo de salida de los servidores a los clientes.

- *GridFTP*: es un componente importante para el traslado de datos de forma segura y eficiente. La réplica de catálogos y de direcciones se usa para registrar y manejar las dos copias de datos, completas o parciales.
- *GSI*: Todos los componentes anteriores se construyen encima de la Infraestructura de Seguridad de *Grid* subyacente, *GSI* (*Grid Security Infrastructure*). Esto proporciona las funciones de seguridad, incluso, la comunicación confidencial y autorización

Conclusiones Parciales

La computación distribuida evolucionó desde la época de los cuarenta y hasta finales del siglo pasado. Uno de sus modelos más importantes es la computación *Grid*, que surge gracias a las investigaciones de los padres del *Grid*, y para dar respuesta a los proyectos ambiciosos que requieren de muchas capacidades tanto en procesamiento, almacenamiento como en comunicación. Este modelo de computación necesita de una arquitectura especial que sea abierta y organizada en capas. Además, cuenta con numerosos componentes de software. Se diferencia de otros modelos por su gran capacidad y potencia. Esto se debe al aprovechamiento de recursos subutilizados, la capacidad paralela y la planificación del trabajo, entre otros.

Globus Toolkit es una herramienta de software para la implementación del *Grid*, que posee todos los protocolos y estándares para llevar a cabo su funcionamiento. Además, brinda una herramienta de seguridad a través de un certificado.

Capítulo 2

Diseño e implementación de la infraestructura del mini-grid.

2 DISEÑO E IMPLEMENTACIÓN DE LA INFRAESTRUCTURA DEL MINI-GRID.

El presente capítulo describe el diseño de la infraestructura del mini-grid y su implementación. Las primeras secciones están dedicadas al diseño y la instalación del software necesario para la conformación del diseño. Luego se procede a la implementación, mediante la instalación y configuración de la herramienta *Globus Toolkit* en todas las máquinas integrantes del mini-grid.

2.1 Infraestructura del mini-grid

Las investigaciones que desarrolla el CEI en áreas de Inteligencia Artificial, Bioinformática, Bases de Datos, Computación Gráfica, entre otras, enfrentan problemas teóricos y prácticos que en múltiples oportunidades involucran un gran volumen de datos y alto procesamiento (Del Toro Melgarejo, *et al.*, 2009).

El CEI tiene un cluster dedicado que junto al resto de los computadores pueden ofrecer una capacidad de procesamiento que satisface las necesidades de las investigaciones desarrolladas en el centro.

En este trabajo se ha decidido simular el clúster *heaven* en máquinas virtuales, para experimentar y explorar la computación *Grid*. Para ello, se utilizan dos máquinas reales, en una de ellas está simulado el servidor y un cliente en términos de máquinas virtuales, y en la otra máquina se emplea una máquina virtual para la simulación de un segundo cliente. Para el diseño de la infraestructura del mini-grid, es necesario plantearla a nivel físico, de red y direccionamiento.

2.1.1 Planteamiento de red

El modelo de red implementado en un ambiente de laboratorio utiliza un *switch*, al cual, están conectadas todas las máquinas involucradas en el mini-grid. En este, todas las máquinas están en la misma subred (10.12.16.0), y cada una de ellas tiene una dirección *IP* fija. En la siguiente figura se muestra el diagrama de interconexión para el entorno *Grid* que se desea

probar. Para ello se emplea un grupo de direcciones *IP*, comprendidas en el rango desde 10.12.16.86 hasta 10.12.16.88.

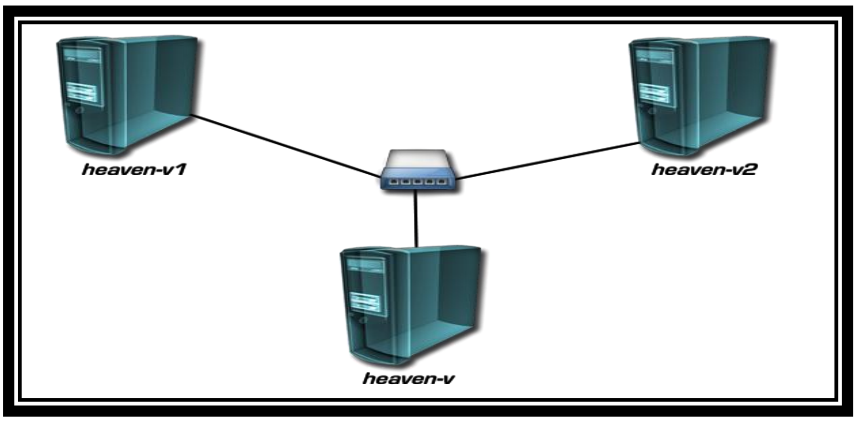


Figura 2-1 Diseño de red.

2.1.2 Planteamiento físico

El planteamiento físico consiste en describir los componentes de hardware y software de los recursos computacionales y de redes involucrados en la construcción del mini-grid. A continuación se detallan estos recursos:

Hostname	Hardware	Software
heaven-v.grid.cei.uclv.edu.cu	Memoria:512 MB Disco Duro:12 GB	Planificadores: *Torque* y *Maui*. Servicio: *NFS* y *DNS*.
heaven-v1.grid.cei.uclv.edu.cu	Memoria:324 MB Disco Duro:8 GB	SO: Linux/Debian6. Compiladores: *gcc*, g++. *JDK* versión 6. *MPICH2*.
heaven-v2.grid.cei.uclv.edu.cu	Memoria:324 MB Disco Duro:8 GB	

Tabla 2-1 Recursos de las máquinas virtuales del clúster heaven.

2.1.3 Esquema de direccionamiento

Se utilizan solamente direcciones *IP* estáticas para identificar los recursos, debido a que todos deben estar siempre identificados. En la tabla 2-2 se muestra el esquema de direccionamiento *IP* de la infraestructura de red del mini-grid. Esta tarea se logra a través de la edición del fichero (/etc/network/interfaces) en todas las máquinas, como se indica a continuación:

```
allow-hotplug eth0
iface eth0 inet static
        address 10.12.16.88
        netmask 255.255.255.0
        network 10.12.16.0
```

Hostname	Dirección *IP*	Máscara	Getway
heaven-v.grid.cei.uclv.edu.cu	10.12.16.88	255.255.255.0	10.12.16.254
heaven1.grid.cei.uclv.edu.cu	10.12.16.87	255.255.255.0	10.12.16.254
heaven2.grid.cei.uclv.edu.cu	10.12.16.86	255.255.255.0	10.12.16.254

Tabla 2-2 Esquema de direccionamiento del mini-grid.

2.2 Software y servicios

La configuración de una infraestructura *Grid*, a nivel de servicios, se resume en las siguientes tareas: instalación del sistema operativo, configuración del *Globus Toolkit* 5.2.2 y la configuración de un conjunto de herramientasadicionales, que se describen a continuación (Cruz Chávez, *et al.*, 2009).

2.2.1 Configuración del servidor DNS

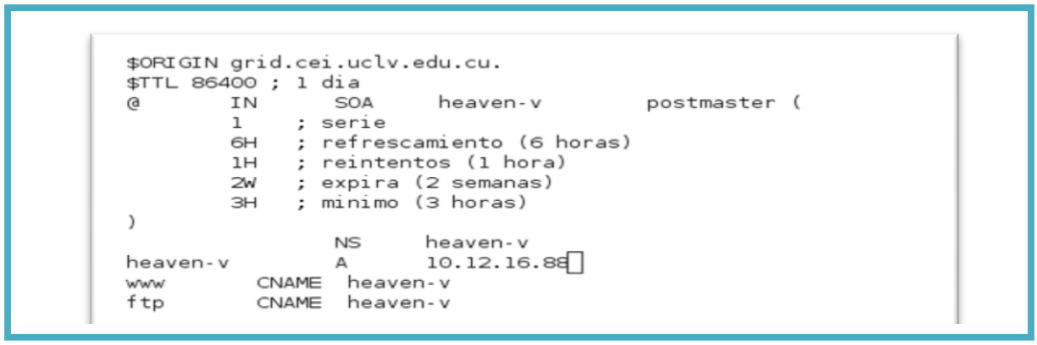
Para emplear un servidor *DNS* capaz de resolver el nombre del dominio, se instala el *DNS* bind9, luego se edita el contenido del fichero (/etc/bind/named.conf.local)[4] y se añade el siguiente contenido:

```
zone “grid.cei.uclv.edu.cu”{
type master;
file “db.grid.cei.uclv.edu.cu”};
zone “16.12.10-in-addr.arpa”{
type master;
```

Para comprobar la sintaxis de los archivos de configuración se ejecuta el comando: **named-checkconf**[5].

[4] Los ficheros, archivos o directorios se encierran entre paréntesis, y no se escriben en cursiva ni negrita.

El siguiente paso es crear el archivo (/var/cache/bind/db.grid.cei.uclv.edu.cu), e incluir el siguiente contenido:



```
$ORIGIN grid.cei.uclv.edu.cu.
$TTL 86400 ; 1 dia
@       IN      SOA     heaven-v        postmaster (
        1       ; serie
        6H      ; refrescamiento (6 horas)
        1H      ; reintentos (1 hora)
        2W      ; expira (2 semanas)
        3H      ; minimo (3 horas)
)
                NS      heaven-v
heaven-v        A       10.12.16.88
www         CNAME   heaven-v
ftp         CNAME   heaven-v
```

Como último paso se reinicia el servicio: **service bind9 restart.**

2.2.2 Configuración del sistema de archivos de red usando Network File System (NFS)

El servicio de archivos *NFS*, es un middleware que se encarga de compartir el directorio base con todo un grupo de máquinas, su tarea principal es cuidar la integridad de los datos y mantener actualizado el sistema de archivos en todas las máquinas conectadas al servidor *NFS* (Cruz Chávez, *et al.*, 2009; Del Toro Melgarejo, *et al.*, 2009).

Para ello se Instala un servidor *NFS* (por ejemplo, *nfs-kernel-server*), luego se editan los archivos (hosts.allow y exports) y se les agrega el siguiente contenido:

```
portmap: 10.12.16.0/24
nfs:10.12.16.0/24
```

Una vez concluida la tarea anterior, es preciso reiniciar los servicios *NFS* y *PORTMAP* con el comando: **service portmap restart && service nfs-kernel-server restart**.

2.2.3 Configuración de Ganglia para el monitoreo en tiempo real del mini-grid.

Ganglia es un sistema de monitoreo distribuido para Sistemas de Cómputo de Alto Rendimiento y *Grid*. Está diseñado para tener un consumo muy bajo de recursos por nodo y una alta concurrencia; actualmente se emplea en muchos sistemas alrededor del mundo y soporta un manejo de hasta dos mil nodos (Cruz Chávez, *et al.*, 2009; Del Toro Melgarejo, *et al.*, 2009).

[5] Los comandos del sistema operativo se escriben en negrita.

Ganglia se ejecuta con dos demonios: el *Ganglia Monitoring Daemon* (*gmond*) y el *Ganglia Meta Daemon* (*gmetad*), además del *Ganglia Web Frontend*. El *Ganglia Monitoring Daemon* (*gmond*), es un servicio multi-hilo, el cual corre en cada uno de los nodos del clúster que se desea monitorear. No es necesario tener un sistema de archivo *NFS* en común, ni una base de datos específica. *Gmond* tiene su propia base de datos distribuida y su propia redundancia.

2.2.4 Configuración de Torque para la administración de recursos locales.

Antes de la configuración del *Torque* es preciso dar una visión de algunos conceptos (Arguibel, 2006):

- Sistema *batch* (lotes): es un conjunto de recursos (computadoras, sistemas de almacenamiento, etc.) que se combinan para presentar una visión abstracta de un sistema capaz de ejecutar tareas (en general *batch*, es decir, no interactivas). El agrupamiento de recursos facilita la administración. Los trabajos pueden correr en las máquinas que satisfagan en número y características los requerimientos de dicho trabajo.
- Administrador de recursos: es un componente de software que provee la funcionalidad de bajo nivel para lanzar, retener, cancelar y monitorear trabajos. Los trabajos tienen características respecto a sus requerimientos y son colocados en una o varias colas, teniendo en cuenta dichas características.
- *Torque*: es un administrador de recursos para sistemas *batch*, que consiste en un sistema de encolado de trabajos *batch* flexibles, que permite especificar requerimientos de los trabajos, administrar distintas colas con diferentes características, además, ejecutar, cancelar y monitorear los diferentes trabajos.

Para instalar *Torque*, primeramente se descarga el archivo (<versión>.tar.gz)[6] correspondiente. Luego, se procede a descomprimir dicho archivo en un directorio de la máquina que será el servidor de *PBS* (nodo máster) y ejecutar (*configure*, *make* y *make install*) como se indica a continuación:

[6] Disponible en: http://www.clusterresources.com/downloads/torque/.

```
tar -xzvf torque-2.5.9.tar.gz
cd torque-2.5.9
./configure , make, make install
```

La configuración del *Torque* consiste en las siguientes tareas (Arguibel, 2006; Del Toro Melgarejo, *et al.*, 2009):

- Configurar el servidor: En primer lugar, es necesario inicializar la base de datos que guardará las configuraciones de *Torque*. También es necesario configurar una primera cola básica. Como usuario *root* se ejecuta el siguiente comando: **./torque.setup torque**. En el servidor se deben instalar todos los servicios necesarios: *pbs_server*, *pbs_mom* y *pbs_sched*. El servidor de *PBS* debe conocer cuáles son los nodos de cómputo. Para ello, el directorio (/var/spool/torque) contiene los nombres de los nodos en el fichero (*nodes*), uno por línea.
- Generar los paquetes: A fin de que los nodos de cómputo y los nodos clientes puedan funcionar como tales, es necesario generar paquetes de distribución en el servidor y ejecutar como usuario con permisos sobre el directorio en que se descomprimió el paquete de instalación (<versión>.tar.gz): **make packages**. Los siguientes comandos, generan los paquetes necesarios para nodos clientes y de cómputo: **torque-package-server-linux-i686.sh**[7], **torque-package-mom-linux-i686.sh**, **torque-package-clients-linux-i686.sh, torque-package-devel-linux.i686.sh, torque-package-doc-linux-i686.sh.**
- Configurar los nodos de cómputos y clientes: En los nodos de cómputo, existe un archivo (server_name) dentro del directorio (/var/spool/torque). Este archivo contiene el nombre del nodo servidor, por ejemplo: (nodo1). También existe otro archivo (config) dentro del subdirectorio (mom_priv), que contiene líneas en forma de variables. Para indicar el nombre del servidor, se hace de la forma ($pbsserver <nombre del servidor>). En cada nodo de cómputo, es necesario instalar el paquete correspondiente, **torque-package-mom-linux-i686.sh**. Previamente se copia el paquete, **torque-package-mom-linux-i686.sh –install**, a la máquina en la que se

[7] Todos los nombre de paquetes generados por *Torque* se escriben normal pero en negrita.

instalará. Para que un nodo pueda enviar trabajos, debe ser configurado como cliente. Esto se hace mediante la instalación del paquete correspondiente, **torque-package-clients-linux-i686.sh**. Como usuario *root* ejecutar: **torque-package-clients-linux-i686.sh –install**.

Para la comodidad del uso de *Torque* se pueden iniciar todos los servicios de manera automática, esto se logra copiando los servicios a la carpeta (/etc/init.d). Una vez concluido el proceso de configuración, ya es posible experimentar las funcionalidades del *Torque*: encender y apagar los servicios, enviar trabajos y verificar el estado de las colas.

2.2.5 Configuración de MPICH.

MPI es el estándar de la librería de paso de mensajes que se publicó en 1994. El estándar de *MPI* es el resultado del acuerdo general de los participantes en los Foros de *MPI* (Forum, 2013), organizado por más de cuarenta organizaciones. Los participantes incluyen a vendedores, investigadores, académicos, diseñadores de biblioteca de software y usuarios. *MPI* ofrece portabilidad, estandarización, actuación y funcionalidad (Sterling, *et al.*, 2002).

La ventaja que ofrece utilizar el *MPI* es la estandarización en muchos niveles. Por ejemplo, desde la estandarización de las sintaxis de *MPI*, cualquier usuario puede confiar en su código de *MPI* para ejecutarlo bajo cualquier aplicación de *MPI* que corre en su arquitectura.

Desde que se estandarizó la conducta funcional de llamadas de *MPI*, dichas llamadas deben comportarse de la misma manera sin tener en cuenta la aplicación. Esto garantiza la portabilidad de sus programas paralelos. Sin embargo, este comportamiento puede variar entre las diferentes aplicaciones.

La instalación y configuración del *MPICH*, debe realizarse en todas las máquinas que conforman el *Grid*, y no solamente en el nodo servidor, debido a que el *Grid* pretende ejecutar los trabajos en paralelo, en cualquiera de los nodos disponibles. El proceso de instalación y configuración del *MPICH* es el siguiente:

```
tar -xzvf mpich2-1.4.1pl.tar.gz
cd mpich2-1.4.1pl
./configure , make, make install
```

MPICH-G2 (unix.msc, 2013) es una aplicación para habilitar implementación de *MPI* para sistemas *Grid*. Es decir, mientras se utilizan los servicios del *Globus Toolkit*® (por ejemplo, inicio del trabajo, seguridad), *MPICH-G2* le permite acoplar las máquinas que utilizan diferentes arquitecturas, para ejecutar las aplicaciones de *MPI*.

2.2.6 Configuración del planificador Maui

Se define el planificador como un componente de software que decide dónde, cuándo, y cómo correr trabajos de acuerdo con una serie de políticas, prioridades y límites. El planificador de un sistema *batch* consulta y controla el administrador de recursos, en lo que respecta a trabajos encolados para correr y requerimientos de los mismos (Arguibel, 2006).

Maui (clusterresource, 2013), constituye una herramienta optimizada y configurable capaz de soportar un arreglo de políticas de planificación, prioridades dinámicas y reservaciones extensivas entre otras. Este planificador debe descargarse[8] e instalarse, en el nodo servidor, de la siguiente manera:

```
tar -xzvf maui-3.1.1.tar.gz
cd maui-3.1.1
./configure , make, make install
```

Al configurar todos estos componentes mencionados anteriormente, resta la configuración del *Globus Toolkit* para tener una infraestructura de servicios adecuada para el mini-grid. Esta infraestructura se refleja en la siguiente figura:

[8] Disponible en http://www.clusterresources.com/downloads/maui/temp/.

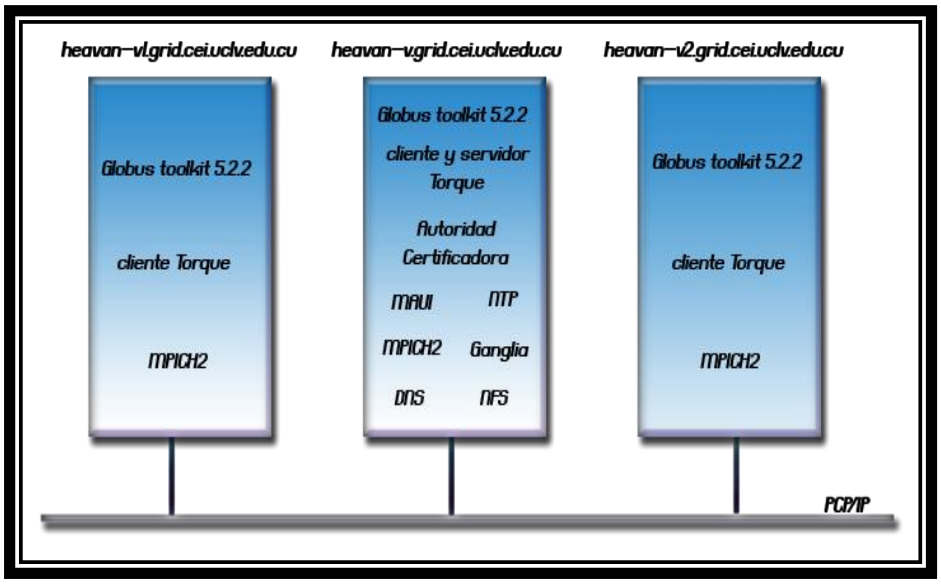


Figura 2-2 Infraestructura de servicios para el mini-grid.

2.3 Instalación del *Globus Toolkit* 5.2.2

Antes de realizar la instalación del software hay que cumplir con una serie de prerrequisitos:

- Sistema Operativo: cualquier distribución de *LINUX*, ya que el *Globus* corre bajo la plataforma *LINUX*. En el mini-grid del *CEI* se ha escogido la distribución *Debian 6*.
- Actores involucrados: hay que disponer de dos cuentas de usuarios para llevar a cabo la instalación y la configuración del *Globus*. La primera cuenta es del superusuario (*root)* y la segunda es del usuario (*globus)*[9] que se encarga de la mayor parte de la instalación.
- Software de Base: antes de la instalación debe asegurarse la instalacion de un conjunto de software y compiladores tales como: *openssl*, compilador C (*gcc*), *GNU tar, GNU make, GNU sed, perl*, y librerías *libltdl, zlib*.
- Software adicional: los software mencionados anteriormente, en su mayoría, forman parte del sistema operativo o del *Globus Toolkit 5.2.2*, pero, se necesita de dos software adicionales y son: *Java JDK* y *Apache Ant*. Estos últimos deben descargarse[10] e instalarse.

[9] Para diferenciar el software *Globus* y la cuenta del usuario *globus*, se decidió escribir este último con minúscula.

[10] Disponibles en: http://repos.uclv.edu.cu/debian

- Variables de ambientes: se deben configurar las variables de ambientes necesarias, tales como: *JAVA_HOME, ANT_HOME y GLOBUS_LOCATION*.

Una vez concluido lo anterior, se procede a la instalación, que consiste en descargar el paquete fuente de *Globus Toolkit5.2.2* (Alliance, 2013f) y extraer sus ficheros en una determinada carpeta, luego se crea un directorio para la instalación. El siguiente paso es ingresar en la carpeta donde se alojan los ficheros de instalación, compilarlos y proceder a la instalación con los comandos típicos de *Debian*.

Los pasos anteriores cubren el núcleo del *Globus* y los procesos de configuración se describen en los siguientes epígrafes.

```
root:~$ mkdir /directorio de instalación
root:~$ chown globus:globus /directorio de instalación
globus:~$ cd /directorio de Globus Toolkit5.2.2
globus:~$./configure –prefix=/directorio de instalación
globus:~$make (puede demorar algún tiempo)
globus:~$make install
```

2.4 Seguridad en el *Grid*

La tecnología *Grid* pretende integrar recursos separados, incluso pertenecientes a diferentes organizaciones o dominios administrativos. Dado este planteamiento, es evidente la necesidad de implementar una infraestructura de seguridad que permita a los administradores del *Grid*, controlar el uso de sus recursos y, además, capaz de proporcionar al usuario confianza sobre la autenticidad de estos recursos (Lechner, 2006; Vanessa Barrios, 2006).

2.4.1 Comunicación segura y encriptación.

La comunicación segura se caracteriza por (Gangotena, *et al.*, 2009):

- Privacidad: únicamente el transmisor y el receptor entienden la conversación.
- Integridad: el receptor debe estar seguro de que el mensaje recibido no fue alterado.
- Autenticación: las partes involucradas en la comunicación son realmente quienes dicen ser.

- Autorización: es el mecanismo que decide cuando un usuario está autorizado a realizar tareas.

Entre los métodos de encriptación se encuentran: la encriptación simétrica (Beristes, *et al.*, 2003; Gangotena, *et al.*, 2009; Losilla Anadón, 2005)), la cual se basa en el uso compartido de una llave secreta, y la encriptación asimétrica, que es un método en el que los receptores y emisores generan dos llaves, una pública, que se conoce en ambas partes, y otra privada.

2.4.2 Certificados y autoridad certificadora

Un certificado digital es un documento que asocia una llave pública a un usuario particular. Dicho documento es firmado digitalmente por un tercero, que es la autoridad certificadora.

En el *Globus Toolkit5.2.2* se utilizan los certificados digitales basados en el estándar X509, que definen la sintaxis del certificado y no están relacionados con ningún algoritmo en particular.

Ellos incluyen información, en forma binaria, sobre la versión y el nombre del certificado, del sujeto (usuario o *host*), información del nombre, llave pública y un identificador único; además, el nombre y la firma de la autoridad certificadora (Alliance, 2013c, 2013e; Lucena López, 2002).

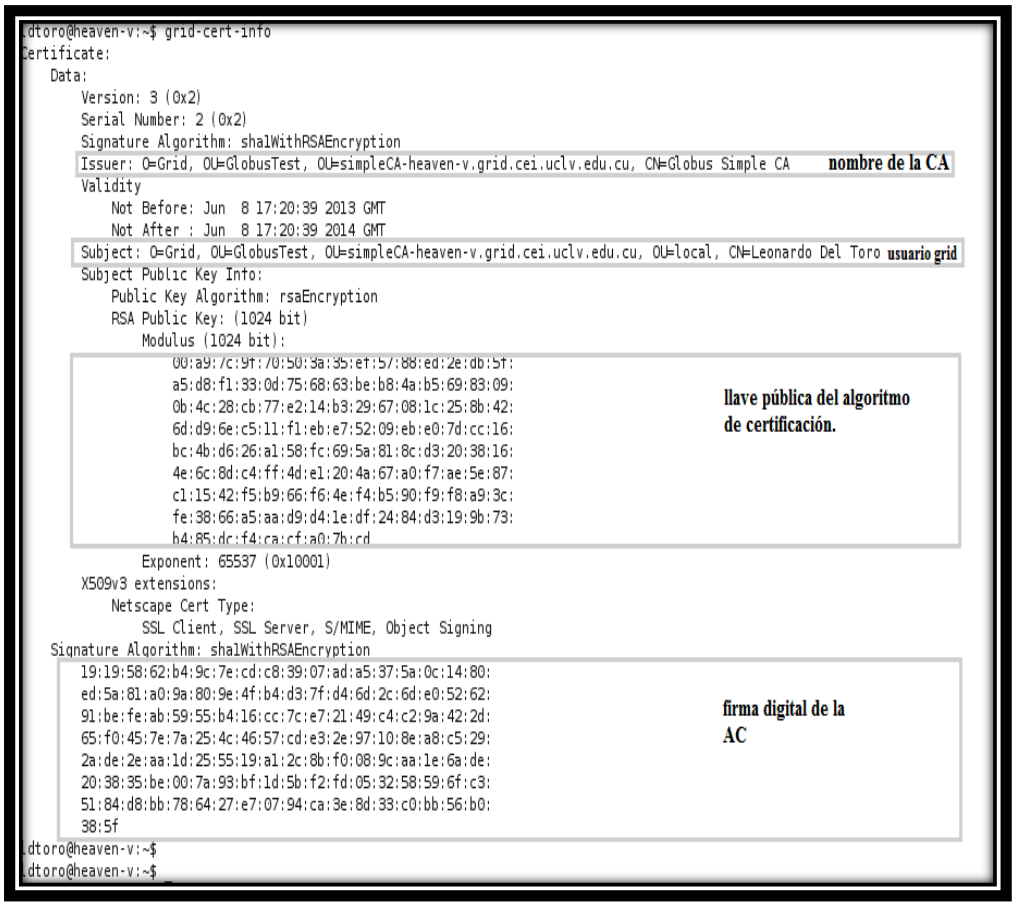


Figura 2-3 Certificado de usuario.

La autoridad certificadora (CA)[11] es responsable de identificar afirmativamente las entidades que solicitan certificados, emitir, remover y archivar certificados; proteger el servidor de la CA y mantener un espacio de nombres único para los propietarios de certificados. Los nombres en los certificados X.509 no están codificados, aparecen simplemente como "nombres comunes" y se codifican como (*distinguished names)*, los cuales se representan como una lista de pares de nombres separados por comas. Los más comunes son los siguientes (Alliance, 2013c, 2013d, 2013e):

O: Organización.

OU: Unidad de la Organización.

CN: Nombre Común (generalmente el nombre de usuario).

El contenido de un certificado se muestra en la Figura 2-3.

[11] Se va a referir a la autoridad certificadora en el resto del documento con la forma abreviada CA.

2.4.3 Configuración de seguridad en el mini-grid

La seguridad en el mini-grid se logra mediante la implementación del paquete (*simple CA)*, que representa el papel de la autoridad certificadora. Esta configuración se realiza en la primera máquina del mini-grid, que va a ser su máquina certificadora. Antes que la CA pueda firmar certificados, tiene que crear su propio certificado, generando un par de llaves: pública y privada, con esta última protege y firma su propio certificado (Alliance, 2013d). Esto se lleva a cabo ejecutando, como usuario *globus*, en la máquina que va a cumplir el rol de la autoridad certificadora, el script ***grid-ca-create***[12]**.** Este script necesita información del sujeto, además, se necesita configurar el tiempo de validez del certificado, un correo electrónico y una contraseña. Una vez concluido lo anterior, se genera un paquete que después se instala en esta máquina y en las demás. A continuación se muestra una figura para la configuración de la autoridad certificadora:

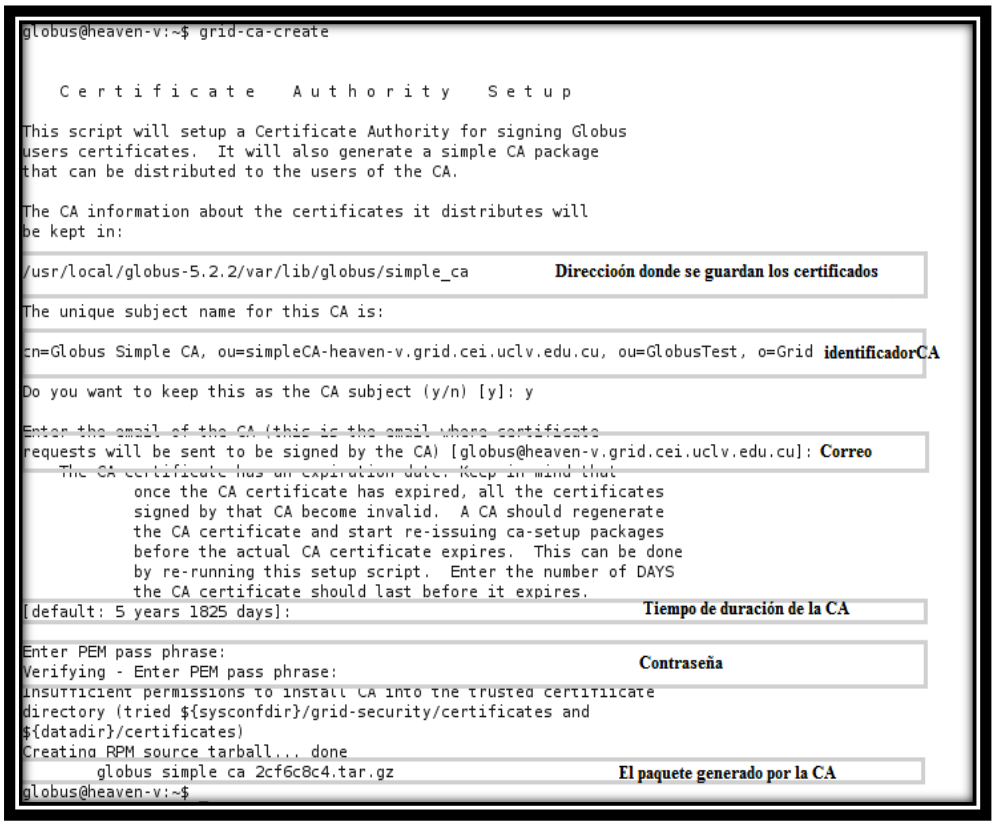


Figura 2-4: Configurar la autoridad certificadora.

[12] Todos los scripts y comandos del *Globus toolkit*5.2.2 están escritos en negritas y cursivas para resaltarlos.

En el resto de las máquinas se debe instalar el paquete generado por la simple CA (los archivos <hash_certificado>.0 y <hash_certificado>.signing_policy).

Para obtener el certificado de usuario o *host*, se deben realizar dos procesos. El primero, es el encargado de emitir la solicitud del certificado, lo cual, tanto usuario o *host* genera su par de llaves y firma la pública para demostrar que realmente posee la llave privada. Esta acción en *Globus Toolkit* se logra ejecutando el script ***grid-cert-request***. En este proceso se solicita una contraseña para proteger la clave privada del solicitante. Este script genera tres archivos: uno de ellos es la solicitud del certificado (usercert_request.pem, en el caso de usuarios, y hostcert_request.pem, en caso de *host*); la llave privada del certificado (userkey.pem, para usuarios, y hostkey.pem, para *host*), y un archivo vacío inicialmente, que es el certificado (usercert.pem, para usuarios, y hostcert.pem, para *host*).

El segundo proceso para obtener el certificado es la verificación que hace la CA sobre la solicitud, la CA debe verificar la identidad del solicitante y que efectivamente posea la llave privada, dicho proceso se logra en *Globus Toolkit* a través del script ***grid-ca-sign***. Una vez ejecutado el script, la CA crea y firma un certificado y lo envía al solicitante mediante algún mecanismo de transferencia. Los certificados de usuarios (hostcert.pem) se guardan en la carpeta (/home/<nombre_usuario>/.globus) y los certificados de host se almacenan en la carpeta (/etc/grid-security). En la siguiente figura se demuestra cómo se solicita y se emite un certificado:

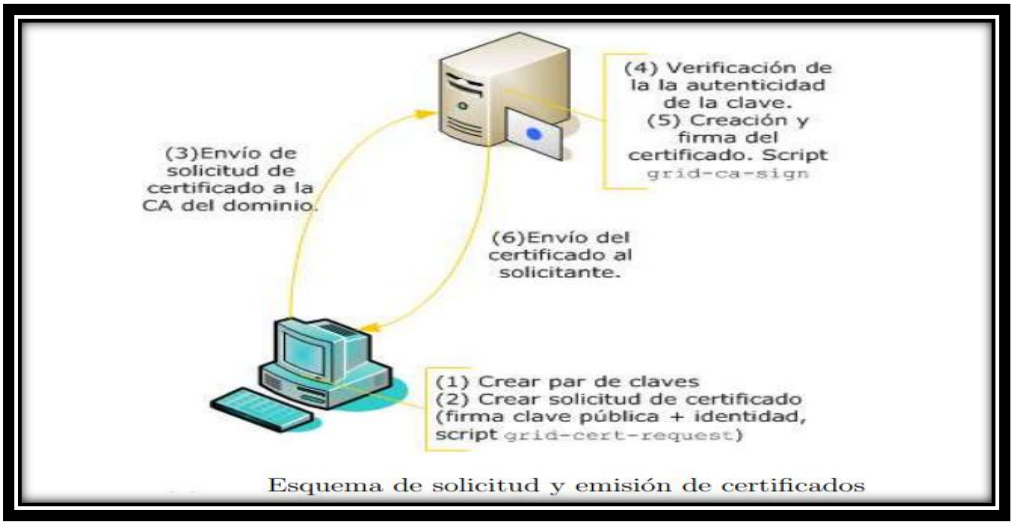


Figura 2-5 Esquema de solicitud y emisión de certificados.

La delegación local se usa para conseguir un certificado proxy de sesión. El certificado proxy se utiliza para autenticar al usuario y a programas de usuarios ante recursos del *Grid*. Este certificado se crea usando el script ***grid-proxy-init***. El certificado proxy es de sesión con un tiempo de vida corto, el motivo detrás de este modelo es proveer un mecanismo por el cual un usuario pueda autentificarse en el *Grid* y realizar varias operaciones sin utilizar su certificado de largo término, y por ende, sin ingresar su contraseña.

La autenticación entre usuarios se realiza para reconocer mutuamente sus certificados, para autenticarse el usuario A frente al usuario B se realiza el siguiente proceso (Lechner, 2006):

- A envía su certificado a B.
- B confirma el certificado enviando un mensaje de prueba al usuario A.
- A recibe el mensaje, lo encripta con su llave privada y lo envía a B.
- B desencripta el mensaje recibido con la llave pública de A, almacenada en el certificado enviado por A.
- B verifica que el mensaje desencriptado es igual al mensaje enviado y comprueba la identidad de A.
- A está autenticado frente a B.

Para simplificar el proceso de autenticación entre los usuarios, se obtiene un certificado proxy con un tiempo de validez y una contraseña, lo que permite que cada vez que el usuario requiera autenticarse frente a otro, se cumpla el proceso de autenticación mutua sin necesidad de reingresar la frase de contraseña *PEM*.

La autorización se realiza desde el lado de la CA en base a un ***grid-mapfile***, que es un archivo que guarda la lista de usuarios autorizados, y solo estos usuarios son los que pueden invocar los servicios del mini-grid. El archivo ***grid-mapfile*** es construido como usuario *root* en el directorio (/etc/grid-security), cada entrada del archivo tiene dos partes: la primera de ellas es el nombre del usuario según su certificado (*distinguished name*), el cual se obtiene ejecutando como usuario el script ***grid-cert-info –subject***. La segunda parte de la entrada es el nombre local del usuario. Luego, como usuario *root* se ejecuta el script ***grid-mapfile-add-entry*** –dn "*grid-cert-info -subject*" –ln "*whoami*".

En la siguiente figura se muestra cómo crear un certificado proxy y la información que este contiene:

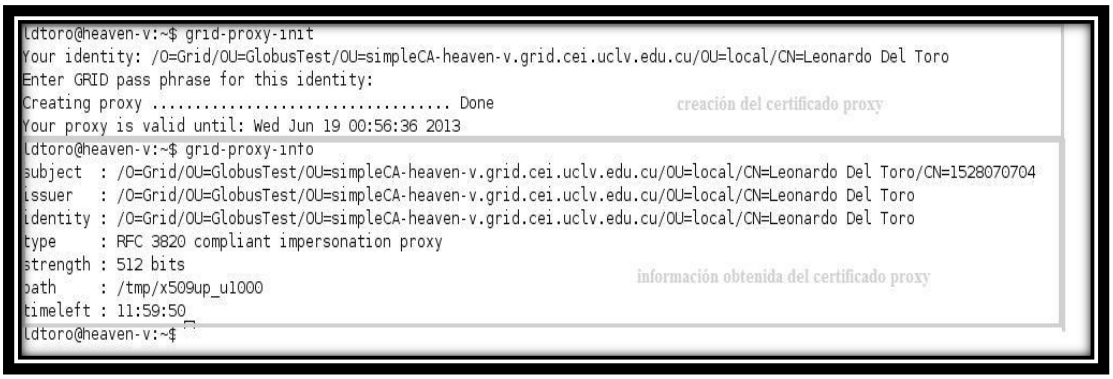


Figura 2-6 Creación de proxy.

2.5 Gestión de datos en el entorno *Grid*

Globus Toolkit provee las herramientas necesarias para la gestión de datos, las cuales, pueden ser para transferencia o réplica de datos. De las herramientas soportadas por *Globus* se encuentran: *GASS, Gridftp, RLS,* entre otras.

2.5.1 Globus Access to Secondary Strong (GASS)

Es el protocolo o herramienta utilizada por los clientes y servidores de administración de recursos, para transferir datos de sumisión de trabajo de un lado a otro entre el cliente y el servidor. Además, proporciona la lectura y escritura; provee soporte para la sintaxis de *http* y *https* (Silva, 2006).

2.5.2 Grid File Transfer Protocol (GridFTP)

El *GridFTP* es un protocolo de transferencia de datos que proporciona, seguridad, alto rendimiento, robustez y fiabilidad. Basado en Highly *Popular Internet File Transfer Protocol* (Alliance, 2013b), y por las recomendaciones de *Globus Grid Fourm,* se resume el protocolo en lo siguiente:

- Basado en *GSI*.
- Existen múltiples canales para transferencias paralelas.
- Transferencias parciales de archivos y transferencias entre servidores.
- Autenticación de los canales de datos.

Globus Toolkit5.2.2 provee:

- Una implementación de servidor ***globus-gridftp-server***.
- Una implementación de cliente ***globus-url-copy***.

El cliente ***globus-url-copy*** es capaz de acceder a los datos mediante múltiples protocolos, tales como: *ftp, file, gsiftp, gsissh, http y https*. Las transferencias pueden ser de dos tipos: la primera involucra un solo servidor (*two party transfer*) y la otra, denominada (*third party transfer*) involucra a dos servidores. En la siguiente figura se muestran los dos tipos de transferencias:

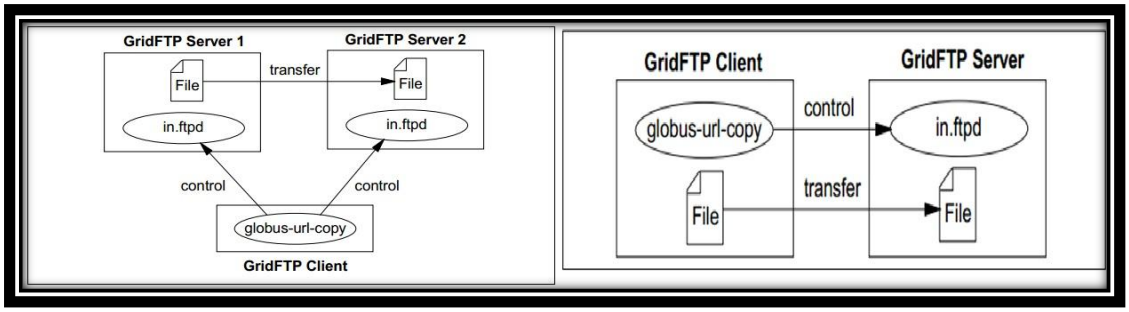


Figura 2-7 Tipos de transferencias.

El formato general de un cliente *Gridftp* en *Globus* es el siguiente:

globus-url-copy [<opción>] <url_source> <url_dest>.

Donde los <url_source> y <url_dest> pueden ser archivos almacenados localmente o en una máquina remota.

- *file*:/camino/del/fichero
- *ftp*://<hostname>:<port>/camino/del/fichero
- *gsiftp*://<hostname>:<port>/camino/del/fichero
- *sshftp*://<hostname>/camino/del/fichero

2.5.3 Los modos de seguridad para la transferencia de archivos.

Para el correcto funcionamiento del *GridFTP* se debe configurar la variable de entorno '*GLOBUS_THREAD_MODEL*=pthread'. Es necesario instalar el *GridFTP* en todos los nodos donde se pretende transferir datos, esta instalación ya se hizo por defecto cuando se instaló el *Globus Toolkit* 5.2.2.

Existen cuatro opciones de seguridad para ejecutar el cliente/servidor de la herramienta *GridFTP* (Alliance, 2013b):

- El modo anónimo: Es el modo donde no hay seguridad ninguna, en el cual, cualquier usuario que tiene un cliente *ftp*, es capaz de gestionar datos. Para ejecutar el servidor *GridFTP*, basta con correrlo con la opción (-aa). Al ejecutarlo, el servidor devuelve el

puerto donde se está escuchando, por lo tanto, el cliente se conecta a esta dirección y hace una transferencia de archivos. El modo anónimo no debe ser ejecutado durante mucho tiempo y hay que asegurarse de apagar el servicio una vez terminada la transferencia con la opción <Ctrl> + <c>.

- archivo de contraseña: si el usuario confía en su red y desea el mínimo de seguridad para sus transferencias, se puede ejecutar el servidor *GridFTP* con un texto en claro. Este proceso, consiste en crear un archivo de contraseña y luego correr el servidor con la opción (*-password-file*). A su vez, el cliente se conecta al servidor con el nombre de usuario y la contraseña introducida.
- *SSHFTP*: el tipo de seguridad en este caso introduce el protocolo control de canal *ssh*, que es una manera simple de obtener seguridad alta en el canal. Donde quiera que se desee ejecutar el servidor *GridFTP*, debe aceptar conexión *sshftp*, y para ello, basta con correr el servidor con la opción (*-enable-sshftp*). Esto crea un fichero *sshftp*, en caso de ejecutar el servidor como usuario *root*, dicho fichero se guarda en (/etc/grid-security); en el caso de ser ejecutado por otro usuario, el fichero *sshftp* se almacena en el directorio (/home/<nombre_usuario>/.globus). Una vez configurado el servidor ya no es necesario ejecutarlo, sino que, directamente el cliente puede hacer una transferencia al servidor *sshftp*.
- *GSIFTP*: este es el modelo de seguridad más potente para la ejecución de gestión de datos. Su potencia es lograda por el uso de la *GSI*. El usuario cliente que va a transferir datos debe ser un usuario *Grid*, tener un certificado digital y crear un proxy para la delegación. Y si la transferencia involucra un servidor en una máquina remota, el usuario debe ser autorizado a usar este servicio, o sea, el archivo ***grid-mapfile*** en la máquina remota, debe tener una entrada de este usuario. Una vez verificados todos los aspectos de la seguridad *GSI*, el cliente puede utilizar el programa cliente del *GridFTP* y hacer una transferencia de cualquier tipo.

A continuación, se muestran dos transferencias, la primera es entre un cliente y un servidor, y la segunda entre dos servidores:

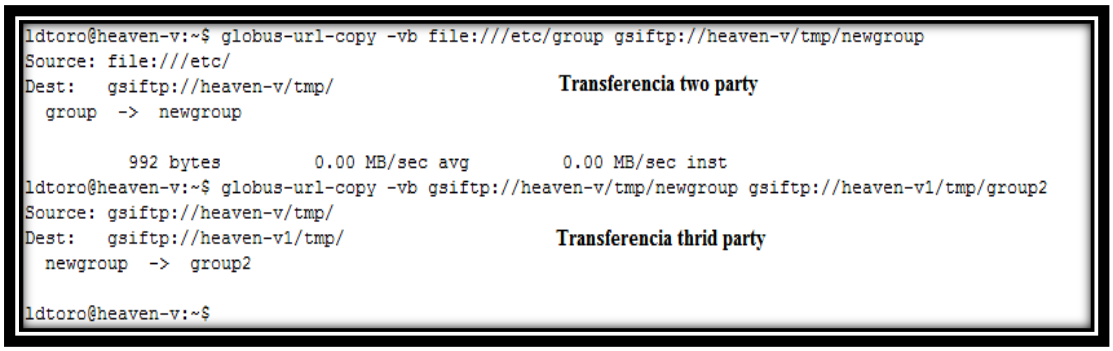


Figura 2-8 Transferencias vía *GSIFTP*.

2.6 ADMINISTRACIÓN DE LA EJECUCIÓN

2.6.1 Grid Resource Allocation and Management GRAM

La herramienta *GRAM*, incluida en el software *Globus Toolkit* 5.2.2, proporciona servicio de acceso, monitoreo, inicio, planificación y administración de trabajos en recursos *Grid*. *GRAM* es implementado en conjunto con el servicio de delegación, para soportar delegación de credenciales, monitoreo y administración del cómputo de manera integral (Czajkowski, *et al.*, 1988).

GRAM no es un planificador de recursos, sino un conjunto de servicios, que permite la comunicación con planificadores de recursos locales. Su arquitectura no es compacta, sino de múltiples componentes; utiliza la *GSI* y delegación de credenciales como medio de seguridad para ejecutar tareas. Estas, son enviadas al servidor a través del cliente *GRAM*, que puede chequear y reenviar el trabajo en orden para abordar errores en la comunicación. Además, puede cancelar una tarea en cualquier punto de su ciclo de vida. *GRAM* proporciona una transferencia de archivos de alto rendimiento, tanto antes, como durante la asignación. El acceso a la información se puede tener a través de *HTTP, GASS y GridFTP* (Alliance, 2013a).

Existen varios componentes que son necesarios para el funcionamiento del *GRAM*:

- Componentes de *Globus*: el *GRAM* utiliza el *GridFTP* para acceder a elementos de almacenamientos remotos y para monitorear el contenido de los archivos escritos por el trabajo durante la ejecución. Otro componente de *Globus* utilizado por el *GRAM* es la delegación, que permite a los clientes delegar credenciales dentro del entorno de *hosting*.
- Planificadores de recursos locales.

- Generadores de eventos.

La herramienta *GRAM5* incluye dos servidores: ***globus–gatekeeper*** y ***globus-job-manager***, este último es ejecutado automáticamente después de iniciar el ***globus-gatekeeper***.

GRAM5 incluye tres programas clientes para enviar trabajos al servidor:

- ***globus-job-run***: proporciona una interfaz de línea de comando. Despacha los trabajos al recurso *GRAM* y espera hasta que concluya su ejecución.
- ***globus-job-submit***: este cliente envía trabajos al servidor *GRAM* y no espera su terminación.
- ***globusrun***: este cliente provee más flexibilidad al enviar, monitorear y cancelar los trabajos. La diferencia con los otros clientes, consiste en que este cliente utiliza el lenguaje *RSL* para proveer la descripción de trabajos.

La herramienta *GRAM* logra la uniformidad de su interfaz mediante la implementación de un lenguaje de especificación de dominio, el cual se denomina *Resource Specification Language* (*RSL*), que proporciona una manera sencilla para expresar los requerimientos del trabajo, independientemente del administrador de recursos donde se pretende ejecutar este trabajo (Alliance, 2013a).

2.6.2 Conectar GRAM con Torque

GRAM no es un administrador local de recursos, sino que depende de otros administradores de recursos locales (*LRM*). Para la ejecución de trabajos con la herramienta *Globus*, es necesario instalar y configurar algún tipo de administrador de recursos locales. El *Globus Toolkit* 5.2.2 tiene la facilidad de integrarse con una amplia gama de administradores locales, tales como: *Condor*, *PBS*, *Grid engine*, entre otros. En el mini-grid se seleccionó el administrador de recursos *PBS*, implementados con *Torque* y el planificador *Maui*.

Para poder enviar trabajos a un clúster *PBS* vía *Globus*, es necesario contar con el adaptador *jobmanager-pbs*. El procedimiento de configuración se describe a continuación:

```
export PBS_HOME=/var/spool/torque
:~$ make gram5-pbs
:~$ make install
```

Para la conexión entre *GRAM* y *Torque* es preciso configurar el archivo *jobmanager-pbs.conf*. Esta configuración se realiza en base al software que esté instalado en el servidor. En el caso del mini-grid abordado en este trabajo, este archivo se configura de la siguiente manera:

```
# For the mpi jobtype, the pbs LRM implementation supports both the
# MPI 2-specified mpiexec command and the non-standard mpirun command common
# in older mpi systems. If either of these is path to an executable, it will
# be used to start the job processes (with mpiexec preferred over mpirun). Set
# to "no" to not use mpiexec or mpirun
mpiexec=/usr/local/bin/mpiexec
mpirun=/usr/local/bin/mpirun                    direcciones de mpirun y mpiexe

# The qsub command is used to submit jobs to the pbs server. It is required
# for the PBS LRM to function
qsub="/usr/local/bin/qsub"
# The qstat command is used to determine when PBS jobs complete. It is
# required for the PBS LRM to function unless the SEG module is used.
qstat="/usr/local/bin/qstat"
# The qdel command is used to cancel PBS jobs. It is required for the LRM
# to function.
qdel="/usr/local/bin/qdel"

# The PBS LRM supports using the PBS_NODEFILE environment variable to
# point to a file containing a list of hosts on which to execute the job.
# If cluster is set to yes, then the LRM interface will submit a script
# which attempts to use the remote_shell program to start the job on those
# nodes. It will divide the job count by cpu_per_node to determine how many
# processes to start on each node.
cluster="1"
remote_shell="/usr/bin/ssh"
cpu_per_node="2"                    cantidad de procesadores por nodos
```

Figura 2-9 Configuración del archivo *jobmanager-pbs.conf*.

2.6.3 Clientes y servidores de GRAM

Una vez concluido el proceso de instalación y configuración de ambas herramientas, *GRAM* y *Torque*, se puede proceder a probar el funcionamiento de *GRAM*, mediante la ejecución de trabajos en el servidor. Para poder ejecutar los trabajos en el mini-grid, es preciso asegurarse de: tener las credenciales del *host* (certificado *host*), el certificado del usuario que va a ejecutar el trabajo, un certificado proxy para dicho usuario y la autorización a través del ***grid-mapfile,*** que mapea el usuario *Grid* con el usuario real donde se está ejecutando la tarea (Alliance, 2013a).

Para probar el comportamiento de los clientes que provee la herramienta *GRAM*, se utiliza una cuenta de usuario, que despacha un trabajo al servidor. Previamente el usuario *root* despliega el servidor ***globus-gatekeeper*** ubicado en el directorio (/etc/init.d) donde está la instalación de *Globus*, así, este proceso ejecuta de manera automática el servidor ***globus-job-manager***, luego

el usuario que desea la ejecución del trabajo, ejecuta el script ***globus-gatekeeper***. De la ejecución se obtiene como resultado, una cadena con información sobre la conexión, la cual se muestra en la Tabla 2-4.

Componente	Ejemplo	Significado
Nombre del host	heaven-v.grid.cei.uclv.edu.cu	El host donde se ejecuta el trabajo.
Puerto	2811	El puerto donde escucha el *GRAM*.
Nombre del servicio	*Jobmanager-pbs*	El nombre del servicio *GRAM*.
Credencial	<id del certificado host *heaven*-v>	La credencial que utiliza el *GRAM*.

Tabla 2-3 Detalles de la conexión.

El cliente *GRAM* despacha los trabajos mediante la conexión con el servidor y el nombre del ejecutable, a continuación se muestran algunos escenarios de trabajos. En el primer ejemplo se utiliza el cliente ***globus-job-run***:

Figura 2-10 Información de la conexión y experimento con el cliente *globus-job-run*.

Este ejemplo muestra la ejecución del cliente ***globus-job-submit***, el cual se integra con otros comandos, tales como: *globus-job-status*, *globus-job-get-output* y *globus-job-clean*, para conocer informaciones acerca del estado del trabajo y devolver el resultado entre otras.

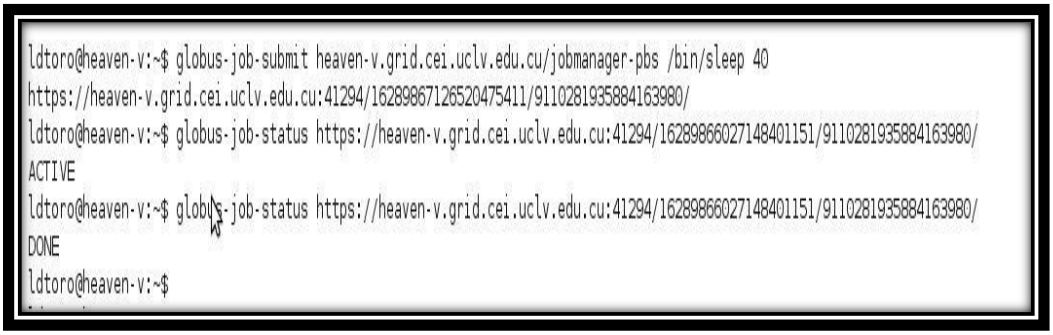


Figura 2-11 Experimentando el cliente *globus-job-submit*.

El cliente ***globusrun*** tiene varios argumentos que se utilizan para indicar una tarea específica:

Argumento	Uso
-p	Chequear las sintaxis del lenguaje *RSL*.
-a	Chequear la autenticación *GRAM*.
-j	Chequear la versión del *Globus* y del *jobmanager*.
-s	Retornar la salida de la ejecución al cliente.
-b	Un trabajo lote (*batch*).
-k	Terminar el trabajo.
-f	Leer la especificación del trabajo desde el fichero *RSL*

Tabla 2-4 argumentos del cliente *globusrun*.

A continuación se muestra un escenario donde el cliente ***globusrun*** despacha un trabajo:

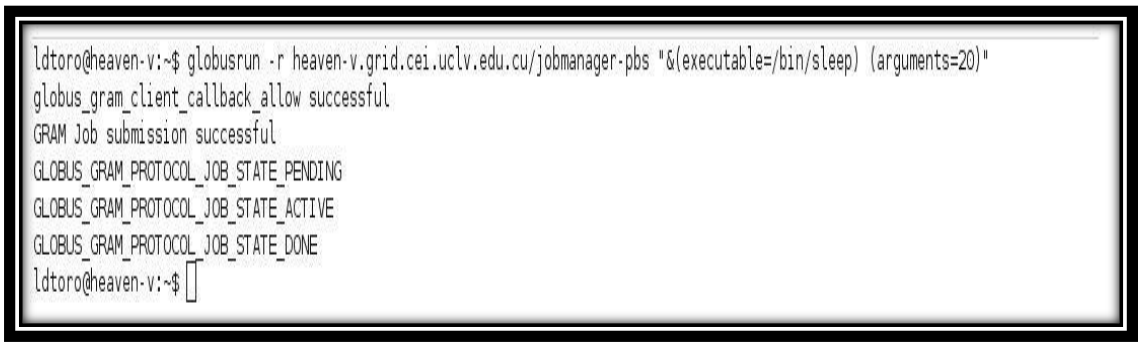


Figura 2-12 Experimentar el cliente *globusrun*.

Las pruebas de ejecución de trabajos en paralelo que se llevan a cabo en el mini-grid, se discuten en el tercer capítulo.

Conclusiones Parciales

Para desarrollar un sistema *Grid*, es necesario diseñar en primer lugar una infraestructura para conectar a sus integrantes. Además, se debe configurar un conjunto de software adicional que se integran con la herramienta *Globus Toolkit*. El proceso de instalación y configuración del *Globus Toolkit*5.2.2 fue logrado en este capítulo utilizando una cuenta con privilegios para la configuración, y empleando la infraestructura de seguridad que provee el *Globus*.

En un sistema *Grid* es preciso tener una máquina que cumpla con el rol de máquina certificadora, para que pueda firmar las solicitudes de certificados emitidos por los *hosts* y los usuarios del sistema.

Todos los usuarios que desean utilizar los servicios *Grid* deben tener un certificado digital firmado por la autoridad certificadora, pues así, podrán transferir archivos y ejecutar trabajos. Este capítulo cumple con estos objetivos, ya que se diseñó e implementó la infraestructura del mini-grid.

Capítulo 3

Resultados y propuestas para un *Grid* universitario.

3 RESULTADOS Y PROPUESTAS PARA UN *GRID* UNIVERSITARIO.

En el presente capítulo se introducen las principales formas de ejecutar un programa, en particular, uno en paralelo. Luego se describen las distintas pruebas realizadas (utilizando la tecnología *Globus*, *Torque* y *MPICH*) y se hace un breve análisis de los resultados obtenidos. Además, se facilitan propuestas para la extensión de la tecnología *Grid* a nivel universitario.

3.1 Resultados de las pruebas de funcionamiento en el mini-grid

El *Globus Toolkit* provee tres clientes para la ejecución de trabajos, además, tiene la facilidad de integrarse con diversos administradores de recursos locales y tiene la capacidad de ejecutar trabajos en paralelos, utilizando los planificadores adecuados para seleccionar los nodos disponibles y que cumplan con los requisitos del programa.

Para las pruebas realizadas en el mini-grid se ha decidido trabajar con programas en paralelos sencillos, debido a las siguientes razones:

- La escasez de los recursos que conforman este mini-grid, ya que cuenta con tres nodos y son máquinas virtuales.
- El bajo rendimiento de dichos recursos, incluso, dos máquinas virtuales comparten el mismo procesador.
- El objetivo de este trabajo no radica en verificar el rendimiento ni tampoco se pretende demostrar la capacidad del *Grid* en reducir el tiempo de ejecución, sino, en probar su funcionamiento, verificar su configuración y explorar esta tecnología.
- La limitación del tiempo para realizar este trabajo, ya que es el primero en su categoría en la UCLV.

Se utilizó un programa que se ejecuta en paralelo para validar el comportamiento de la tecnología *Grid*, y se compararon los resultados obtenidos, con los resultados de la ejecución de este mismo trabajo con la herramienta *Torque* (qsub) y *MPICH* (mpirun).

El programa en paralelo es una aplicación en lenguaje C que calcula la sumar los primeros (10000) números naturales, mediante la división del conjunto de datos en sub-trabajos, y donde cada uno se ejecuta por un proceso independiente, utilizando un nodo determinado.

En el mini-grid se decidió asignar dos procesadores lógicos a cada nodo, de tal manera que el total de los procesadores donde se puede ejecutar un trabajo es a lo sumo seis. Partiendo de todo lo anterior, se procedió a ejecutar el trabajo por los tres clientes de *Globus*, y además, se utilizaron el *Torque* y *MPICH*. Los resultados de la ejecución se muestran en las siguientes cinco figuras:

```
ldtoro@heaven-v:~$ globus-job-run heaven-v.grid.cei.uclv.edu.cu/jobmanager-pbs -np 6 -x '&(jobtype=mpi)'
Examples/SUMA1
Procesador 5. Terminos: 1670. Suma parcial: 15306385
Procesador 2. Terminos: 1666. Suma parcial: 6939723
Procesador 1. Terminos: 1666. Suma parcial: 4164167
Procesador 3. Terminos: 1666. Suma parcial: 9715279
Procesador 4. Terminos: 1666. Suma parcial: 12490835
Procesador 0. Terminos: 1666. Suma parcial: 1388611
La sumatoria calculada es: 50005000.
ldtoro@heaven-v:~$
```

Figura 3-1 Ejecución de un programa en paralelo, utilizando *globus-job-run*.

```
ldtoro@heaven-v:~$ globus-job-submit heaven-v.grid.cei.uclv.edu.cu/jobmanager-pbs -np 6 -x '&(jobtyp
e=mpi)' Examples/SUMA1
https://heaven-v.grid.cei.uclv.edu.cu:41779/1629006614087489404l/705051003699386508/
ldtoro@heaven-v:~$ globus-job-get-output -r heaven-v.grid.cei.uclv.edu.cu/jobmanager-pbs https://hea
ven-v.grid.cei.uclv.edu.cu:41779/1629006614182328364 6/705051003699386508/
Procesador 2. Terminos: 1666. Suma parcial: 6939723
Procesador 5. Terminos: 1670. Suma parcial: 15306385
Procesador 1. Terminos: 1666. Suma parcial: 4164167
Procesador 0. Terminos: 1666. Suma parcial: 1388611
Procesador 3. Terminos: 1666. Suma parcial: 9715279
Procesador 4. Terminos: 1666. Suma parcial: 12490835
La sumatoria calculada es: 50005000.
ldtoro@heaven-v:~$
```

Figura 3-2 Ejecución de un programa en paralelo, utilizando *globus-job-submit*.

```
ldtoro@heaven-v:~$ globusrun -s -r heaven-v.grid.cei.uclv.edu.cu/jobmanager-pbs  "&(jobtype=
mpi) (executable=Examples/SUMA1) (count=6)"
Procesador 2. Terminos: 1666. Suma parcial: 6939723
Procesador 5. Terminos: 1670. Suma parcial: 15306385
Procesador 1. Terminos: 1666. Suma parcial: 4164167
Procesador 3. Terminos: 1666. Suma parcial: 9715279
Procesador 0. Terminos: 1666. Suma parcial: 1388611
Procesador 4. Terminos: 1666. Suma parcial: 12490835
La sumatoria calculada es: 50005000.
ldtoro@heaven-v:~$
```

Figura 3-3 Ejecución de un programa en paralelo, utilizando *globusrun*.

```
ldtoro@heaven-v:~/Examples$ mpirun -f ../.nodes -np 6 ./SUMA1
Procesador 0. Terminos: 1666. Suma parcial: 1388611
Procesador 1. Terminos: 1666. Suma parcial: 4164167
Procesador 3. Terminos: 1666. Suma parcial: 9715279
Procesador 4. Terminos: 1666. Suma parcial: 12490835
Procesador 2. Terminos: 1666. Suma parcial: 6939723
Procesador 5. Terminos: 1670. Suma parcial: 15306385
La sumatoria calculada es: 50005000.
ldtoro@heaven-v:~/Examples$
```

Figura 3-4 Ejecución de un programa en paralelo, utilizando *mpirun.*

```
ldtoro@heaven-v:~/Examples$ qsub ./suma1q
258.heaven-v.grid.cei.uclv.edu.cu
ldtoro@heaven-v:~/Examples$ cat SUMA1.o258
Procesador 3. Terminos: 1666. Suma parcial: 9715279
Procesador 5. Terminos: 1670. Suma parcial: 15306385
Procesador 2. Terminos: 1666. Suma parcial: 6939723
Procesador 0. Terminos: 1666. Suma parcial: 1388611
Procesador 4. Terminos: 1666. Suma parcial: 12490835
Procesador 1. Terminos: 1666. Suma parcial: 4164167
La sumatoria calculada es: 50005000.
ldtoro@heaven-v:~/Examples$
```

Figura 3-5 Ejecución de un programa en paralelo, utilizando *Torque* (*qsub*).

Otras de las pruebas que se realizaron en el mini-grid, son aquellas que verifican el estado del administrador de recursos locales (información cerca de las colas de trabajos) y el estado del planificador (los procesadores y nodos activos, los trabajos en ejecución, entre otras informaciones). En las dos figuras siguientes se muestra el contenido de la ejecución de los comandos de verificación de estado del *Torque* y *Maui*:

```
ldtoro@heaven-v:~/Examples$ showq
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME


     0 Active Jobs       0 of    6 Processors Active (0.00%)
                         0 of    3 Nodes Active      (0.00%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME


0 Idle Jobs

BLOCKED JOBS----------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME


Total Jobs: 0   Active Jobs: 0   Idle Jobs: 0   Blocked Jobs: 0
ldtoro@heaven-v:~/Examples$
```

Figura 3-6 Estado del planificador *MAUI.*

```
ldtoro@heaven-v:~/Examples$ qstat -q

server: heaven-v.grid.cei.uclv.edu.cu

Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
eternity           --   720:00:0    --       --    0   0 32   E R
batch              --      --       --       --    0   0 --   E R
                                                ----- -----
                                                  0     0
ldtoro@heaven-v:~/Examples$
```

Figura 3-7 Estado del administrador de recursos *Torque*.

Las últimas pruebas se realizaron a través de una página web, conectando con el nodo donde se encuentra instalado el *Ganglia*, que es una herramienta poderosa para el monitoreo de recursos en tiempo real. Con esta herramienta el administrador del *Grid* puede verificar el estado de cualquier nodo de su sistema, por lo tanto, le brinda la capacidad de elegir adecuadamente los nodos donde se pretende la ejecución del siguiente trabajo. En la siguiente figura se muestra una instancia de tiempo real de los nodos del mini-grid:

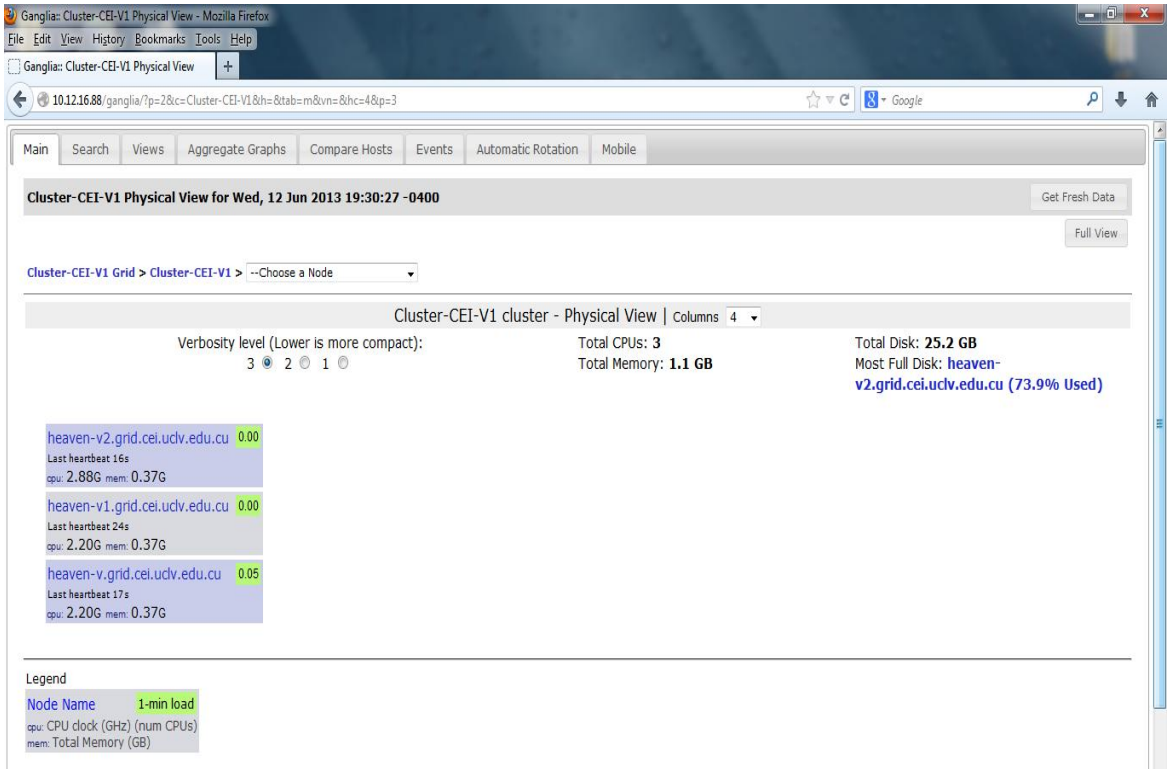


Figura 3-8 Monitoreo en tiempo real utilizando *Ganglia*.

3.1.1 Comparación de los resultados

Los resultados obtenidos de la ejecución del programa anteriormente mencionado, se comportaron de manera similar en todas las herramientas utilizadas, validando los mismos. Cabe señalar que el tiempo de ejecución de los trabajos al utilizar *Globus*, fue superior a la ejecución del mismo trabajo por las otras herramientas. Esto se debe a varias verificaciones, tales como: ¿tiene el usuario, que ejecuta el trabajo, un certificado proxy válido? ¿Posee un certificado digital firmado por la autoridad certificadora donde se encuentra corriendo el ***globus-gatekeeper***? ¿Está el usuario autorizado a utilizar los recursos? ¿Existe una entrada en el archivo ***grid-mapfile*** para este usuario? Además de estas cuestiones, los recursos empleados en el mini-grid no son óptimos.

3.2 **Propuesta para expandir la computación *Grid* en la UCLV**

La computación *Grid* es una tecnología que pretende agrupar la mayor cantidad de recursos computacionales y administrarlos con el fin de aprovecharlos eficazmente. Además, procura mostrar estos recursos como una única máquina con potenciales que satisfacen las necesidades de cómputos requeridos en todo tipo de investigación.

La Universidad Central "Marta Abreu" de Las Villas (UCLV) cuenta con diversos grupos de investigación, los cuales, cada vez necesitan más recursos computacionales para el desarrollo de sus investigaciones. De manera similar, existen, además de los laboratorios, un grupo de clústeres dedicados, que en conjunto pueden ofrecer una potencia de cálculo que podría ser suficiente para las investigaciones universitarias.

Este último epígrafe facilita una propuesta para expandir la computación *Grid* en todas las áreas universitarias. Existen dos maneras para diseñar e implementar el *Grid* en la universidad: una es a través de la certificación cruzada, y la otra, mediante el control central. La base de dichas propuestas se fundamenta en las organizaciones virtuales (ver capítulo 1, página 10-11).

3.2.1 Diseño de Grid universitario con la certificación cruzada

En esta propuesta, una organización virtual como se muestra en la figura 3-9, puede ser una facultad determinada. Esta organización debe contar con una máquina que juegue el rol de la

máquina certificadora y que tenga la autoridad suficiente para certificar cada uno de los nodos y los usuarios de dichos nodos.

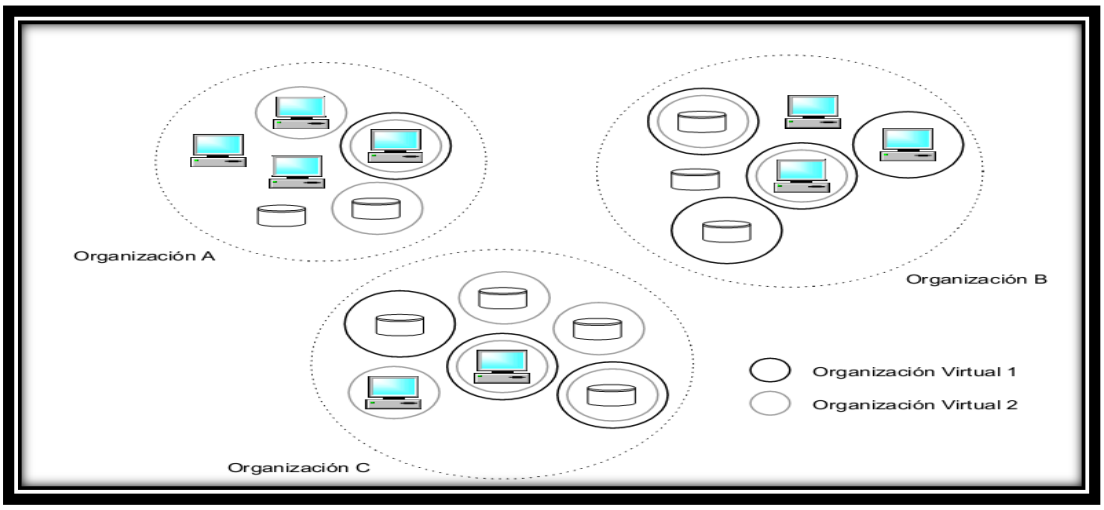


Figura 3-9 Organizaciones virtuales.

Para el funcionamiento de esta organización es necesario cumplir con una serie de requisitos, tales como: diseñar la infraestructura de red, instalar y configurar el *Globus Toolkit* en todos los nodos, instalar los software necesarios para la administración de recursos (*Torque, Condor*, entre otros) y el software de trabajo en paralelo. (Ver anexo A con información detallada del proceso de instalación y configuración del *Globus Toolkit* 5.2.2).

El siguiente paso es la configuración de enrutadores (*routers*) para la conexión de todas las organizaciones virtuales. El último paso es la certificación cruzada, que es un proceso de autentificación y autorización entre todas las máquinas (Beristes, *et al.*, 2003). Este proceso se lleva a cabo mediante la copia del paquete (*globus_simple_ca_*<número-hash>.tar.gz) de cada máquina certificadora, y su instalación en todas las máquinas que están en otras organizaciones, pero manteniendo siempre la autoridad certificadora local como la autoridad por defecto (ver Figura 3-3). O sea, el usuario que esté en la organización A debe solicitar primero un certificado a su administrador local, y de manera similar se solicita otro certificado para cada máquina certificadora en las otras organizaciones, mediante la ejecución del script ***grid-cert-request –ca*** <número-hash-de-la-CA-remota>.

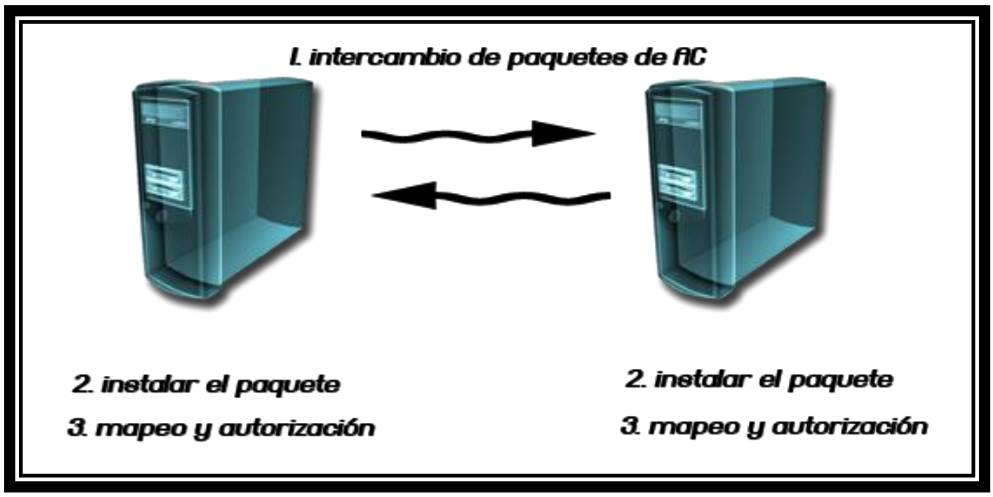


Figura 3-10 Certificación cruzada entre dos máquinas.

Cuando todas las máquinas tienen los paquetes instalados, pueden solicitar certificados de host y usuarios a las autoridades remotas, para que estén autorizados a emplear recursos en las organizaciones remotas. Cuando se solicita el certificado a la autoridad remota, esta última, debe agregar al archivo de autorización ***grid-mapfile*** una nueva entrada por cada usuario remoto que solicita ser certificado.

Esta propuesta agrega más seguridad al sistema *Grid*, permite a los administradores locales de las organizaciones virtuales tener más control sobre los recursos y decidir qué recursos serán compartidos, quién los utilizará y cuáles son sus permisos sobre estos recursos.

El modelo de certificación cruzado es difícil de implementar e involucra a varios administradores. A continuación, se muestra un diseño de este modelo:

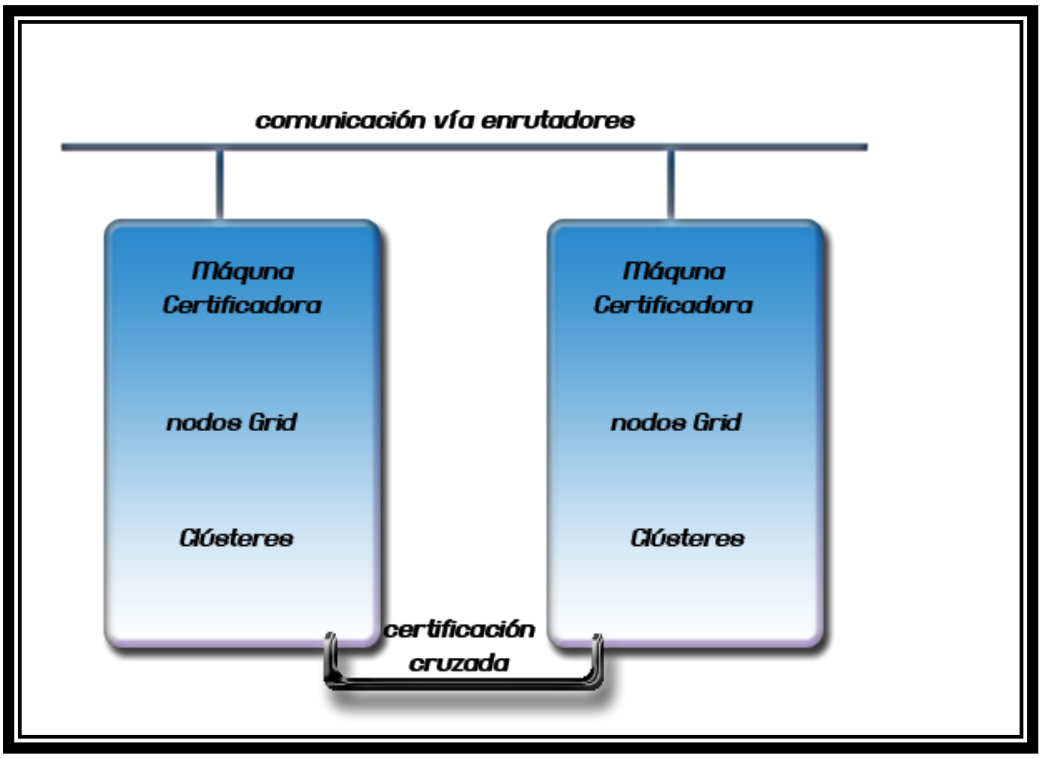


Figura 3-11 Infraestructura de un grid universitario con la certificación cruzada.

3.2.2 Propuesta del modelo central

Esta propuesta propone la existencia de una única máquina certificadora, puede estar dentro de una organización virtual determinada o independiente (forma su propia organización). Esta máquina es la autoridad certificadora de todo el *Grid*, por lo cual, el paquete de autoridad generado por ella debe instalarse en todas las máquinas.

El modelo de control central involucra un único administrador, que es el responsable de firmar las solicitudes de certificación, tanto de *host* como de los usuarios. Esto representa una ventaja sobre el modelo anterior, ya que es más fácil de diseñar e implementar; aunque tiene desventajas referentes a la seguridad, ya que el administrador es el único que autoriza a los usuarios para utilizar los recursos. Así, las organizaciones virtuales no tienen el control total de sus recursos, todo está a disposición del administrador de la única máquina certificadora. Además, una falla en la máquina certificadora significa graves fallos en el *Grid*, ya que no se puede certificar a nuevos usuarios ni autorizarlos. A continuación se muestra el diseño:

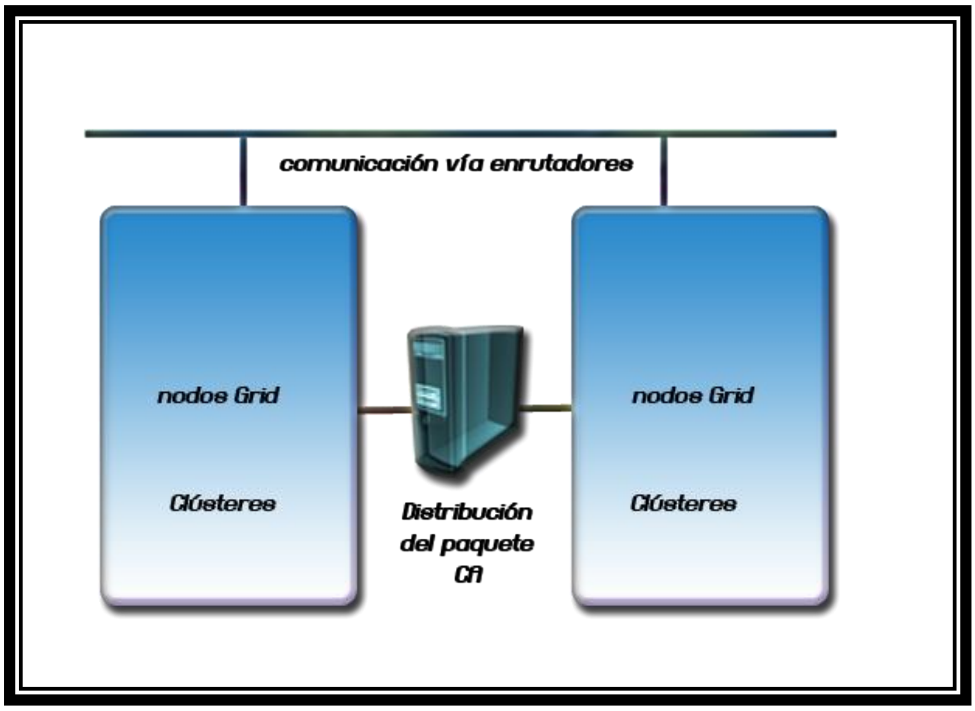


Figura 3-12 Infraestructura del *Grid* universitario con una sola máquina certificadora.

Conclusiones parciales

Para validar el diseño logrado en el segundo capítulo, se procedió a ejecutar dos trabajos sencillos, utilizando los tres clientes *GRAM* (*globus-job-run*, *globus-job-submit* y *globusrun*), estas ejecuciones mostraron que dichos clientes se comportan de igual modo y ejecutan correctamente el trabajo. Para la comparación de la efectividad de los resultados obtenidos, fue esencial ejecutar los mismos trabajos utilizando otras herramientas, tales como: *PBS* y *MPICH*. Los resultados de la ejecución mediante estas herramientas proporcionaron los mismos resultados que los obtenidos por la herramienta *Globus*.

La herramienta *Ganglia* monitorea en tiempo real la disponibilidad de los recursos, así, el administrador del *Grid* podrá verificar la disponibilidad de los nodos y decidir cuáles son los posibles candidatos para el próximo trabajo.

En este capítulo se facilitan dos propuestas para diseñar un sistema *Grid*, que podría aplicarse en un futuro cercano.

Conclusiones generales

i. La caracterización de la computación *Grid* permitió concebir un diseño que se ajusta a las características del CEI, lo que resulta en que:
   - La dimensión del problema a tratar sugirió la selección de la topología *intragrid*.
   - Se selecciona a *Globus Toolkit* como herramienta que brinda todos los componentes necesarios para conformar un sistema *Grid*.

ii. El modelo virtual *Grid* implementado mostró un correcto desempeño, permitiendo ejecutar aplicaciones seriales y paralelas, con resultados similares a los obtenidos al ejecutar las mismas, sobre un clúster o un computador independiente.

iii. La implementación del mini-grid muestra la flexibilidad que brinda *Globus Toolkit* para adaptarse a otros entornos y extender esta tecnología al entorno universitario.

iv. La ejecución de una aplicación sobre una infraestructura *Grid* incorpora un costo adicional relacionado con: la verificación de los certificados del cliente/computador, la autorización para el uso de los recursos del *Grid* y la comunicación con el gestor de recursos.

v. Para minimizar el efecto del costo añadido, se sugiere el empleo de un *Grid* en trabajos de alta complejidad computacional y de consumo de recursos.

Recomendaciones.

i. Implementar el *Grid* sobre la configuración existente en el clúster *heaven* del (*CEI*), a partir de las experiencias acumuladas en este proyecto.
ii. Implementar un servicio de correo en el computador donde radique la autoridad certificadora, para facilitar la gestión de los certificados.
iii. Se propone limitar la cantidad de usuarios en el sistema, debido a que el proceso de gestión de certificados se realiza manualmente,
iv. Realizar un proyecto que incluya las propuestas que se proponen en el presente proyecto, para extender la tecnología *Grid* al entorno universitario.

Referencias Bibliográficas.

1. Alliance, G., (2013a) GT 5.2.2 GRAM5: System Administrator's Guide. In.
2. Alliance, G., (2013b) GT 5.2.2 GridFTP : System Administrator's Guide. In.
3. Alliance, G., (2013c) GT 5.2.2 GSI-OpenSSH: System Administrator's Guide. In.
4. Alliance, G., (2013d) GT 5.2.2 SimpleCA: Admin Guide. In.
5. Alliance, G., (2013e) GT 5.2.2: GSI C Admin Guide. In.
6. Alliance, G., (2013f) "Instaling GT", disponible en: www.globus.org/toolkit/docs/5.2/5.2.2/admin/install [Accesado: 2013].
7. Arguibel, A., (2006) Guia instalación Torque. In.
8. Bart, J., Ferreira, L., Biebrstein, N., Gilzean, C., and y Strachowaki, R., (2003) Enabling Applications for Grid Computing with Globus. In (Vol. Primera Edición.): IBM.
9. Beristes, V., (2002) Fundamentals of Grid Computing. In (Primera Edición. ed.): IBM, ReedBook.
10. Beristes, V., Ferreira, L., Armstrong, J., Kendzierski, M., Neukotter, A., Takagi, M., Bing-Wo, R., Amir, A., Murakawa, R., Hernandez, O., Magowan, J., and y Bieberstein, N., (2003) *Introduction to Grid Computing with Globus.*
11. Buyya, R., (2004) Grid Computing Information Centre (GRID Infoware) FAQ. In.
12. Café, G., (2012) "Antepasados de la computación grid", disponible en: http://www.gridcafe.org/antepasados-grid_ES.html [Accesado: 2013].
13. clusterresource, (2013) "index.php", disponible en: http://docs.adaptivecomputing.com/maui/index.php [Accesado: Abril 2013].
14. Cruz Chávez, M. A., Cruz Rosales, M. H., and y Zavala Díaz, J. C., (2009) Estudio de Modelos Teóricos de tipo NP-completos en el Laboratorio Nacional de GRIDS de Súper Cómputo, Utilizando Algoritmos Evolutivos de Optimización con Técnicas de Procesamiento Distribuido. In A. Rodríguez León, And y R. Rivera López (Eds.). : Universidad Autónoma del Estado de Morelos.

15. Czajkowski, K., Foster, I., Kesselman, C., Smith, W., Karonis, C., and y Tuecke, S., (1988) *A Resource Managament Architecture for Metacomputing Systems.*
16. Del Toro Melgarejo, L., and y Gálvez Lio, D., (2009) Implementación de un clúster de computadoras de alto desempeñobasado en herramientas de código abierto. In.
17. Fernández, J. A. V., (2012) Computación Grid II. In.
18. Foster, I., (2006) Globus Toolkit version (GT4) Tutorial. In.
19. Foster, I., and y Karonis, N., (1998) A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. In.
20. Foster, I., and y Kesselman, C., (1998) *The Grid: Blueprint for a New Computing Infrastructure.*
21. Foster, I., Kesselman, C., and y Tuecke, S., (2001) The Anatomy of Grid. In.
22. Fourm, M., (2013) "MPI Forum", disponible en: http://www.mpi-forum.org [Accesado: Marzo 2013].
23. Gangotena, A. G., Bonilla, D. M., and y Bernal, M. I., (2009) *Implementación de un Mini-Grid de Cómputo.* Escuela Politécnica Nacional., Ecuador, Quito.
24. Lechner, M., (2006) *Grid,GlobusToolkit y potencia computacional sin límites.* Universidad Nacional del Sur., Argentina.
25. Liu, M. L., (2004) Computación distribuida: conceptos y aplicaciones. In.
26. Losilla Anadón, G., (2005) Computación y seguridad en GRID. In: Instituto de biocomputación y física de sistemas complejos.
27. Lucena López, M., (2002) Criptografía y Seguridad en Computadores. In.
28. Magoules, F., Pan, J., Tan, K., And y Kumart, A., (2009) *Introduction to Grid Computing.* Primera Edición. ed., Estados Unidos: Chapman & Hall CRC.
29. Martín Llorente, I., (2007) Diseño de Infraestructuras Grid. In. Madrid, España.: Universidad Complutense de Madrid.
30. Mejía Mesa, A., (2004) *Guía Práctica para Manejar y Reparar la Computadora.* Edición Actualizada ed., Medellín, Colombia: Imprelibros Cargraphics S.A.
31. Silva, V., (2006) *Grid Computing for Developers.* Primera Edición. ed., EEUU: Charles River Media, Inc.

32. Sterling, T., Bell, G., and y Kowalik, S., (2002) Beowulf Cluster Computing with Linux, MIT Press, Paperback. Inc.
33. unix.msc, (2013) "MPICH", disponible en: http://www-unix.mcs.anl.gov/mpi/mpich/ [Accesado: Marzo 2013].
34. Vanessa Barrios, V., (2006) *Grid Computing*. Universidad Nacional del Nordeste., Corrientes, Argentina.

Anexos.

Proceso de instalación del Globus Toolkit 5.2.2

Paso1) preparar el destino de la instalación, como *root* ejecutar:

1.1) mkdir /destino/de/instalación ---en este caso /usr/local/globus-5.2.2

1.2) chown –R globus:globus /destino/de/instalación

Paso2) proceso de instalación básica, como usuario *globus* ejecutar:

2.1) cd /carpeta/de/los/ficheros/de/instalación ---en este caso gt5.2.2-all-source-install

2.2) ./configure –prefix=/destino/de/ instalación

2.3) make ----puede demorar bastante tiempo

Paso3) proceso de instalación de Simple CA, este proceso se lleva a cabo únicamente en la máquina que juega el rol de la autoridad certificadora; como *globus:*

3.1) grid-ca-create //al ejecutar este script, y contestar correctamente las //preguntas que genera el script, se genera un paquete .tar.gz que contiene los ficheros de //la autoridad certificadora. Este paquete se instala en esta máquina y en el resto de las //máquinas que conforman el *Grid*. Una manera de instalar este paquete es generarlo
//para *DEBIAN* y luego instalarlo en todos los nodos. Con los siguientes comandos se //lograrán estas tareas.

3.2) grid-ca-package –d –cadir <carpeta donde se guarda la información sobre la CA>

Ahora como *root* ejecutar:

3.3) dbkg –i <nombre del paquete *DEBIAN* generado en el paso 3.2>

Paso4) certificar los *hosts*, en todas las máquinas que conforman el *Grid* se debe ejecutar como *root* el comando de solicitud de certificado y como *globus* ejecutar el comando que firma la solicitud.

4.1) grid-cert-request –host <nombre del *host*>
4.1.1) scp –r hostcert_request.pem root@<máquina certificadora>:/tmp/<nombre>
---este paso se usa únicamente en las máquinas remotas.
4.2) grid-ca-sign –in /<dirección del fichero hostcert_request.pem> -out /<dirección del fichero hostcert.pem> --sobrescribe el fichero hostcert.pem inicialmente vacío.
4.2.2) grid-ca-sign –in /<dirección del fichero hostcert_request.pem> -out /tmp/<nombre>

Paso 5) certificar usuarios, ejecutar como usuario a certificar el paso 5.1 (usuario local), 5.1.1 (usuario remoto), y como *globus*, el resto de los pasos.

5.1) grid-cert-request
5.1.1) scp –r usercert_request.pem root@< máquina certificadora>:/tmp/<nombre>
---este paso se usa únicamente en las máquinas remotas.
4.2) Grid-ca-sign –in /<dirección del fichero usercert_request.pem> -out /<dirección del fichero usercert.pem> --sobrescribe el fichero usercert.pem inicialmente vacío.
4.2.2) Grid-ca-sign –in /<dirección del fichero usercert_request.pem> -out /tmp/<nombre>
4.2.3)) scp –r /tmp/<nombre>.pem root@<host>:/home/<usuario>/.Globus/usercert.pem ---este paso se usa únicamente en las máquinas remotas.

Paso 6) el último paso de configuración de seguridad es la autorización de los usuarios para utilizar los servicios *Grid*, esto se logra adicionando al fichero las informaciones necesarias por cada usuario.

```
6.1)  globus-gridmap-add-entry –dn “Grid-cert-info -subject” –ln “whoami”
```

Paso 7) configuración *FTP*, en todas las máquinas se debe ejecutar el servicio *GridFTP* con el puerto 2811. Este proceso puede ser manual, mediante la ejecución del paso 7.1 o la configuración del servicio correctamente para que se ejecute como un *daemon* del sistema operativo, y de esta manera, el servicio es arrancado automáticamente al iniciar la sección. Una vez que el servicio está en marcha se pueden hacer las transferencias.

```
7.1)  /dirección del servidor Gridftp  start
//la dirección en este caso es /usr/local/Globus-5.2./etc/init.d/Globus-Gridftp-server start
```

Paso 8) configurar el *GRAM* para que arranque en el puerto 2119, esto se logra manual o automáticamente, de forma similar al paso anterior. Cuando el servicio está en marcha, los clientes *Grid* pueden ejecutar trabajos.

```
8.1)  /dirección del servicio globus-gatekeeper        start
//la dirección en este caso es /usr/local/Globus-5.2./etc/init.d/Globus-gatekeeper  start
```

Glosario de términos técnicos.

- **AFS**

Sistema de archivo *Andrew,* es un sistema de archivos montable en redes.

- **ARQUITECTURA ABIERTA:**

Cualquier diseño de computadora o de periférico que tiene especificaciones públicas.

- **AUTENTICACIÓN**

Proceso mediante el cual el sistema valida la información de inicio de sesión de un usuario.

- **AUTORIZACIÓN**

Proceso mediante el cual un usuario autenticado es autorizado a acceder a determinados recursos.

- **CLÚSTER**

Es un conjunto de computadoras independientes e interconectadas por una red, usadas como un recurso unificado de cómputo como si fuese una sola unidad. Es utilizado para dar solución a problemas complejos y algoritmos en paralelos.

- **CÓDIGO ABIERTO**

Es un programa donde su código fuente está disponible públicamente, aunque los términos de licencias varían respecto a lo que se puede hacer con estos códigos.

- **COLA DE TRABAJO**

Lista de trabajos o tareas que esperan para ejecutarse.

- **COMANDO**

Instrucción que el usuario le da al sistema, el cual generalmente está contenido en un archivo ejecutable.

- **CPU**

Es la unidad central de procesamiento, donde se ejecutan las instrucciones de los programas y se controla el funcionamiento de los distintos componentes de la computadora.

- **CREDENCIAL**

Acredita a un usuario a utilizar determinados recursos a través de un certificado de autoridad.

- **DELEGACIÓN**

Proceso que permite que una credencial de delegación pueda ser usada a través de múltiples solicitudes de servicios.

- **DFS**

Sistema de archivos distribuido, es otro tipo de sistema montable en redes.

- **DNS**

Sistema de nombre de dominio, por lo cual, los host tienen nombres de dominio y dirección IP.

- **ENCRIPTACIÓN**

Es el proceso de codificar datos para prevenir un acceso no autorizado.

- **ESTÁNDAR**

Guía técnica definida por una organización, que se utiliza para uniformizar una determinada área de desarrollo de hardware o software.

- **ETHERNET**

Red de transmisión basada en bus o topología estrella con control de operaciones descentralizadas, con velocidad para transmitir datos a 10/100/1000 *Mbps*.

- **FTP**

Protocolo de transferencias de archivos, utilizado para copiar archivos desde y hacia computadoras remotas.

- **GUI**

Interfaz gráfica del usuario.

HOST

Máquina conectada a una red, dedicada a ejecutar programas de usuarios.

- **HTTP**

Protocolo de transferencia de hipertexto, es el estándar utilizado para acceder a páginas web.

- **HTTPS**

Es la versión de seguridad del protocolo HTTP, utiliza mecanismos y cifrados a través de un puerto seguro.

- **INTERNET**

Conjunto de redes y puertos de enlace a nivel mundial que usan colecciones de protocolos TCP/IP para comunicarse entre ellas.

- **IP**

Protocolo de Internet, se encarga de la transmisión de datos dividíos en paquetes, el envío y la recepción. Además, la unión de paquetes para reconstruir el mensaje original.

- **MIDDLEWARE**

Es una herramienta que se encuentra entre el sistema operativo y los programas de aplicaciones, su función es asegurar que la aplicación puede correr óptimamente en varias computadoras.

- **MPI**

Interfaz de paso de mensajes, es una aplicación que permite a los mensajes formateados entrar y salir de una aplicación.

- **NFS**

Sistema de archivos de red. Otro tipo de sistemas montables en redes.

- **NTP**

Protocolo de tiempo de red, es un servidor para mantener ajustado el tiempo de reloj en una red.

- **PROTOCOLO**

Un conjunto de reglas o estándares diseñados para que las máquinas puedan comunicarse entre ellas.

- **PROXY**

Es un programa que se encarga de proteger a las entidades de una red, al tiempo que proporciona acceso a un determinado recurso.

- **RECURSO**

Cualquier pieza de un sistema computacional que pueda ser asignada a un programa o a un proceso durante la ejecución.

- **SERVIDOR**

Una máquina que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes.

- **SETI@HOME**

Es un proyecto de la universidad de Berkeley y apoyado por la NASA, utiliza computadoras conectadas a Internet para la búsqueda de inteligencia extraterrestre.

- **TCP/IP**

Protocolo de control de transmisiones y protocolos en internet. Es el estándar de facto para las comunicaciones y transmisiones de datos sobre redes.

- **X.509**

Es un estándar de certificados para firmar documentos y garantizar la seguridad y fiabilidad en las comunicaciones.