UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE UN PORTAL GRID (*GRID PORTAL*) PARA EL USO DE EQUIPOS DE SUPERCÓMPUTO

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

IRVING CARLOS ALVAREZ CASTILLO

DIRECTOR DE TESIS:

M. EN C. JOSÉ LUIS GORDILLO RUIZ

CODIRECTOR DE TESIS:

M. EN I. JORGE VALERIANO ASSEM

MÉXICO, D.F. 2011

Dedicatoria

A toda mi familia, en especial a:

A mis padres por todo su esfuerzo, dedicación, paciencia y apoyo brindados en todos estos años. Y porque sin ellos no sería la persona que soy.

A mi hermana por su apoyo en cada momento.

A mi esposa por su amor, confianza, paciencia, ayuda y empuje que me ha brindado a lo largo de estos años.

A mi hijo por su alegría.

A mis amigos:

Por sus consejos, amistad y compañía.

Agradecimientos

A la Universidad Nacional Autónoma de México, en especial, a la Facultad de Ingeniería, así como a todos sus profesores por su experiencia y calidad recibida en toda mi formación.

Al M. en C. José Luis Gordillo del Departamento de Supercómputo de la UNAM por todo su apoyo, paciencia y guía proporcionados para la realización de esta tesis. También, quiero agradecer a todos los integrantes que han formado o forman parte de este departamento por sus enseñanzas en materia de supercómputo y sobre todo por su amistad.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Los grids computacionales permiten el uso de equipos de cómputo que en su conjunto proporcionan un gran poder de cálculo y de almacenamiento. Estos equipos están distribuidos en diferentes lugares y comúnmente están conectados por redes con un gran ancho de banda que permite mantener ocupados a los equipos del grid computacional.

Debido a las características de los grids computacionales y a su potencial, el cómputo en grid permite estudiar problemas complejos y con mayor precisión, utilizando una infraestructura que se expande a través de límites administrativos y geográficos. Estos problemas serían imposibles de estudiar utilizando un sólo equipo de cómputo por muy robusto que fuera. Este hecho queda de manifiesto en los experimentos llevados a cabo en los detectores del LHC (ver sección 2.2.1) ya que no existe actualmente un equipo con el poder de cálculo suficiente para almacenar y analizar todos los datos generados por dichos experimentos.

Una de la motivaciones de este trabajo es proporcionar una interfaz amigable, accesible y disponible a los usuarios de un grid computacional. Disponible en casi cualquier lugar por ser una interfaz web, amigable en el sentido de que no requiere mucho tiempo de aprendizaje y que es es fácil de utilizar.

El uso de interfaces Grids amigables permite que los usuarios, no expertos en cómputo, dediquen sólo atención a la ejecución de sus aplicaciones y a la administración de su información. Es decir, las interfaces Grid hacen posible que los usuarios se concentren sólo en problemas científicos y no en cómo funciona un grid computacional.

Las interfaces grid benefician y fortalecen a los *science gateway*, grupos de expertos o de investigación en una determinada área del conocimiento que trabajan en conjunto aunque se encuentren geográficamente dispersos. Entre las áreas del conocimiento de los *science gateway* se encuentran: astronomía, química, bioquímica, ciencias de la tierra, sismología, genética, biomedicina, geografía, geofísica, investicación atmosférica, investigación de materiales, neuronciencia, física de altas energías, física teórica, física de astro partículas, visualización y procesamiento de imágenes.

Diversos grupos en México están interesados en grids computacionales. En particular la Delta Metropolitana es un proyecto que conjuntará a las tres supercomputadoras más rápidas del país, ubicadas en la UNAM, UAM y el CINVESTAV. La delta, que puede ser administrada como un grid computacional, puede tener como una interfaz de acceso a los equipos de supercómputo un portal grid.

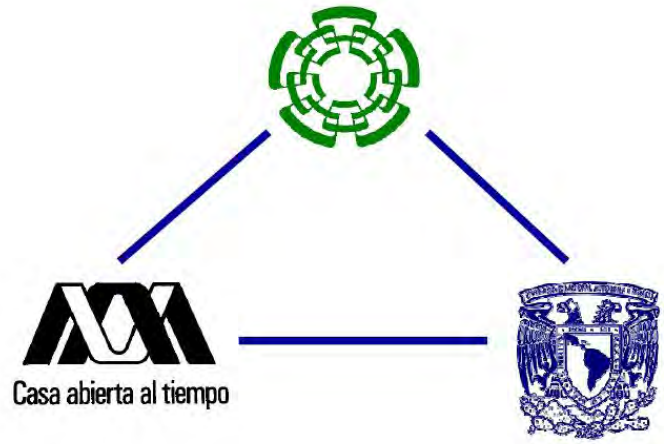


Figura 1.1: Fuente: Redes Ópticas Metropolitanas Académicas[1]

1.2. Descripción del problema

Los servicios proporcionados en un grid computacional son comúnmente mostrados a los usuarios en una interfaz de línea de comandos (CLI) o sesiones remotas de "shell", donde cada comando tiene diferentes opciones a utilizar. Las CLI proporcionan una interfaz con mayor control y más herramientas, en un modo de uso más cercano con los recursos de cómputo, pero debido a su complejidad, al tiempo necesario para aprender a utilizarlas y a que los usuarios (físicos, químicos, ingenierías, etc) de los grids

computacionales no son expertos en cómputo, sino en las aplicaciones que utilizan, se buscaron mecanismos que proveyerán un ambiente más amigable a los usuarios.

De esa necesidad, surgieron los portales Grid: interfaces web para uso de las grids computacionales y portales de acceso a recursos de supercómputo. Este tipo de interfaz se caracteriza por proporcionar un acceso a recursos que resulta más amigable que las CLI, ya que desde un mismo punto, a través de interfaces gráficas y de forma segura, permite consultar información relacionada con equipos de cómputo, envío y monitoreo de trabajos, así como manejo de archivos. Esta interfaz permite que los usuarios de los grids computacionales se dediquen y concentren más en el uso de sus aplicaciones y problemas de su área de conocimiento y no inviertan mucho tiempo conociendo la infraestructura (grid computacional), sobre la que yace su aplicación.

Existe la posiblidad de desarrollar una interfaz o portal grid propia para el uso de grids computacionales. Sin embargo, esto requeriría mucho tiempo de desarrollo e implicaría conocer con detalle todas las diferentes tecnologías involucradas para construir este tipo de interfaz. Además, se volvería a hacer un trabajo que grupos de investigacion (por ejemplo, OGCE y GENIUS [2]) ya han desarrollado en diferentes plataformas; adicionalmente, si por alguna razón existiera un fallo en la seguridad del portal, se comprometería a los equipos de cómputo inmersos en el grid computacional.

Otra alternativa es implementar una interfaz o portal grid ya desarrollado, de preferencia de código abierto. La ventaja de esto es que su tiempo de implementación es menor comparado con desarrollar una interfaz propia, porque no se tienen que conocer todas las tecnologías involucradas a profundidad, además de que varias instituciones y centros de computo (Universidad de Indiana, el Centro de computo de Renaissance, el Instituto de Tecnologia de Rochester, la Universidad de San Diego, el Centro de Supercómputo de San Diego y el Centro de Cómputo Avanzado de Texas) han invertido en su desarrollo, han probado la seguridad y confiabilidad de este tipo de interfaz.

1.3. Resumen del trabajo

En este trabajo se presentarán los conceptos básicos de un grid computacional. Asimismo se mostrarán los componentes principales del software más utilizado para implementar grids computacionales (Globus Toolkit). En este mismo contexto, se describirán algunos de los principales proyectos de grids computacionales en la actualidad.

En el marco de los portales grids, se profundizará en el conocimiento de este tipo de interfaces web, a través del estudio de su funcionamiento y las tecnologías involucra-

das en su implementación. Asimismo, se estudiarán los componentes principales que constituyen a un portal grid para su uso e integración en un grid computacional.

De forma práctica, se implementará un portal grid para equipos de supercómputo, se describirá la infraestructura necesaria, en hardware y software, para la implentación del portal grid; se explicará el proceso de instalación y configuración de los componentes de un grid computacional y de un portal grid; asimismo, se harán las pruebas para mostrar el correcto funcionamiento de un portal grid.

La organización del trabajo es como sigue:

- El capítulo 2 trata sobre los grids computacionales y las tecnologías inmersas en ellas.
- El capítulo 3 versa sobre los portales grids.
- El capítulo 4 presenta la implentación y pruebas de funcionamiento de un Portal Grid.
- El capítulo 5 muestra los resultados de este trabajo.

Finalmente, se concluirá sobre los resultados esperados de este trabajo. Los resultados que se esperán de este trabajo son:

- Descripción de la arquitectura de los portales Grid (portales de acceso a equipos de supercómputo)
- Implementación de un Portal Grid para equipos de supercómputo de la Universidad Nacional Autónoma de México.

Capítulo 2

Tecnologías de Grids Computacionales

El término Grid Computacional es utilizado desde la mitad de los años de 1990 para referirse a las infraestructuras de cómputo distribuido que sirven de herramienta para la ciencia y la ingeniería. El concepto de Grid Computacional fue desarrollado con la idea de hacer posible la compartición de recursos dentro de colaboraciones científicas. La compartición es muy importante porque permite el uso de herramientas e información que ayudan a grupos de investigadores de diferentes áreas del conocimiento a enfrentar problemas más grandes y complejos. Ejemplo de ello es el Gran Colisionador de Hadrones, *Large Hadron Collider* (LHC) (ver sección 2.2.1) de la Organización Europea para la Investigación Nuclear, *Organisation Européenne pour la Recherche Nucléaire* (CERN) que generará más de un PetaByte de información al año. En este caso, esta compartición es imprescindible porque de otra manera no se podría almacenar toda esa cantidad de información en una sola institución u organización. También es importante mencionar la colaboración en el Gran Colisionador de Hadrones porque técnicamente sería imposible analizar tal volumen de información sin los miles de físicos inmersos en este proyecto.

2.1. Conceptos básicos de Grid

2.1.1. Organización Virtual (VO)

Debido a la necesidad de compartir recursos surge el concepto de Organización Virtual, *Virtual Organization* (VO). Una organización virtual es una organización flexible que agrupa a un conjunto de dominios organizacionales (instituciones, organizaciones o individuos) que comparten sus recursos (remotamente) de forma coordinada con el fin de resolver problemas afines y comunes.

La compartición de recursos debe ser ordenada y controlada, a través de reglas de membresía entre las diferentes organizaciones. Esta compartición significa más que un simple intercambio de archivos; significa acceso a recursos (de cómputo, de almacenamiento), software e información en los recursos remotos.

Una característica principal de una VO es que cada miembro define que recursos va a compartir y las condiciones para la compartición de recursos (cómo, dónde, cuándo y quién puede utilizar los recursos).

2.1.2. Definición de Grid Computacional

Un grid computacional es un conjunto de recursos compartidos (de cálculo, de almacenamiento, de información) que se encuentran distribuidos en diferentes dominios organizacionales y pueden ser utilizados por los diferentes miembros de una o varias VOs. Estos recursos son administrados de maneras diferentes, siguiendo políticas de acceso y uso propias de cada uno de los diversos dominios organizacionales. A diferencia de un cluster que se administra de forma centralizada, un grid se administra en forma distribuida. Es decir, en el ambiente de grid computacional se comparten los recursos pero no se pierde el control total de los mismos.

Específicamente, un Grid Computacional es un sistema que coordina recursos computacionales distribuidos utilizando protocolos e interfaces estandarizadas, abiertas y de próposito general para entregar servicios de calidad no triviales [4]. El objetivo de entregar servicios no triviales pretende proporcionar tiempos de respuesta aceptables, capacidad de cálculo, disponibilidad, reservación y seguridad en múltiples recursos; de manera que la combinación de sistemas sea de mayor utilidad que utilizar cada recurso por separado.

Es decir, un Grid Computacional coordina recursos distribuidos porque integra en un

sólo sistema recursos y usuarios que se encuentran en diferentes dominios administrativos.

Los Grids Computacionales están integrados por protocolos e interfaces abiertas que solucionan problemas fundamentales como: autenticación de usuarios; descubrimiento, compartición y acceso a recursos. Al ser abiertos estos protocolos permiten que un Grid Computacional sea interoperable y compatible.

Así un Grid Computacional es un sistema virtual de cómputo que proporciona mecanismos de compartición y coordinación de recursos y en donde estos recursos se encuentran organizacionalmente y geográficamente distribuidos.

2.1.3. Arquitectura del Grid Computacional

Foster et al [4] han propuesto una arquitectura para un Grid Computacional dividida en 4 capas:

1. Estructura (*Fabric*)
2. Protocolos de recursos y conectividad
3. Servicios colectivos
4. Aplicaciones

1. Estructura (*Fabric*)

   Son los recursos (físicos o lógicos finales) que son compartidos en un Grid. Como ejemplo de ello, son: recursos de cómputo, como pueden ser recursos de HPC (ver sección 2.2.3) o de HTC (ver sección 2.2.2); sistemas de almacenamiento (desde un gran sistema de almacenamiento hasta una simple computadora de escritorio), redes de alto rendimiento, sensores, instrumentos, bases de datos, etc.

   En esta capa se ejecutan las operaciones específicas (explícitas) sobre los recursos que se comparten en el Grid. Es decir, capas superiores de la arquitectura de grid ejecutan en esta capa las operaciones (almacenamiento, transferencia de información o cálculos) directamente en los recursos "físicos".

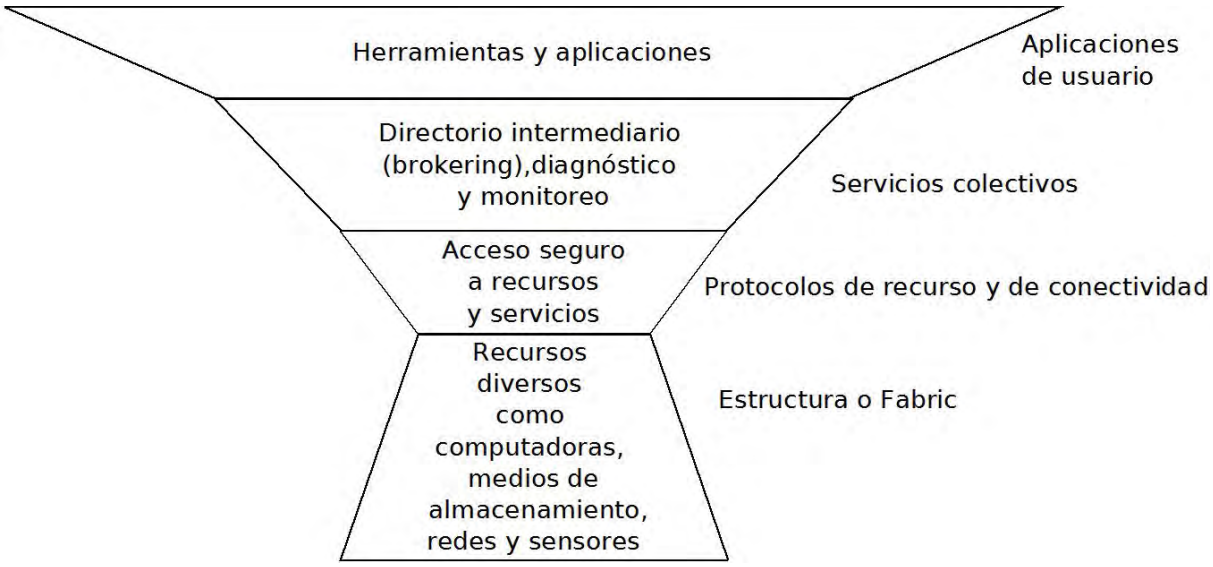


Figura 2.1: Arquitectura del grid computacional

2. Protocolos de recursos y conectividad

   a) Conectividad. En esta subcapa se definen los protocolos de comunicación y de autenticación necesarios para el ambiente de Grid. Los protocolos de comunicación permiten el intercambio de información entre los recursos de la capa de estructura (*Fabric*). Los protocolos de autenticación proporcionan mecanismos de seguridad para verificar la identidad de los usuarios y de los recursos. Esta subcapa está basada en protocolos estandarizados. Algunos de los desafíos que se han presentado en esta subcapa, son:

   - Single Sign On: El usuario de Grid desea autenticarse sólo una vez en el Grid y acceder automáticamente a uno o varios recursos del Grid Computacional, sin necesidad de estar introduciendo una contraseña en cada recurso accedido.
   - Delegación: El usuario de Grid puede delegar a un programa privilegios para que haga ciertas actividades en su nombre.

   b) Recursos

   Esta capa permite al usuario de la Grid Computacional interactuar con los recursos remotos. Es decir, permite operaciones como: inicio, monitoreo y control de procesos, contabilidad del uso de los recursos remotos de forma individual, etc. Esta capa se encuentra arriba de la subcapa de conectividad y sus protocolos pueden llamar a funciones de la capa de Estructura (*Fabric*)

con el fin de acceder y controlar los recursos locales. Es importante recordar que en esta capa se trata con los recursos de Grid en forma individual. Los protocolos pueden ser de los siguientes tipos:

- Protocolos de información: Obtienen información del estado y estructura de un recurso.
- Protocolos de administración: Negocian el acceso a un recurso compartido y a las operaciones a ser desarrolladas en un recurso.

Los protocolos de conectividad y de recursos son limitados y muy estandarizados debido a que tienen que funcionar en todos los tipos de recursos de forma confiable y segura.

3. Colectivas

   Esta capa contiene protocolos y servicios que coordinan a múltiples recursos (colecciones de recursos). Algunos de los desafíos que se encuentran en la capa colectivas son:

   - Reservación, calendarización: Permite la reservación de recursos y calendarización de las tareas en los recursos apropiados.
   - Sistemas de programación habilitados para Grid: hacer posible que modelos de programación sean utilizados en ambiente de Grid. Un ejemplo de esto son algunas implementaciones de MPI para ambiente de Grid.

   Los protocolos de la capa Colectivas son más específicos comparados con la subcapa de recursos.

4. Aplicaciones

   En esta capa se encuentran las aplicaciones de usuario que operan en la Grid.

Según los autores[4], esta arquitectura es extensible, de estructura abierta y satisface las necesidades de las organizaciones virtuales.

2.2. Proyectos relevantes

2.2.1. EGEE

Enabling Grids for E-sciencE , EGEE [1] [11] fue un proyecto de grids computacionales llevado a cabo entre instituciones de diferentes países, principalmente de Europa. La infraestrucutra con que contó EGEE fue la siguiente: 250 sitios en 50 países, más de 68,000 CPUs, más de 20 PetaBytes para almacenamiento, transferencia de información mayor a 1.5 GB/s, servicios de seguridad, soporte a usuarios, etc. Esta infraestrucutura fue proporcionada a los usuarios (investigación e industria) de este grid computacional independientemente de su ubicación física. Según el reporte final de EGEE [10], esta infraestructura apoyó a la investigación de clase mundial en Europa ayudando a hacer más ciencia, a mayor a escala y en menor tiempo. Entre datos reportados resalta que se ejecutaron más de 13 millones de jobs cada mes en esta infraestructura, además de que se reunieron esfuerzos de especialistas de más de 50 países [12].

Los objetivos de EGEE fueron los siguientes:

- Optimizar y expander la infraestructura de Grid Europea a través de utilizar la infraestructura existente, atender a una mayor cantidad de usuarios y agregar más recursos a la infraestructura.
- Preparar la migración de la infraestructura a un modelo de infraestructuras federadas que estén basadas en Iniciativas de Grid Nacionales para su uso multidisciplinario.

El middleware que se utilizó en EGEE para hacer posible la infraestructura de Grid es llamado gLite. El middleware gLite proporciona servicios de seguridad, información, monitoreo, acceso a recursos de cómputo y almacenamiento; asimismo, proporciona servicios de administración de jobs, catálagos de información y replicación de información (*data replication*). GLite [13] está basado en otros middlewares de grid como son: VDT (ver sección 2.2.2) y LCG.

Este proyecto se realizó en tres etapas (EGEE-I, EGEE-II y EGEE-III) durante los años de 2004 a 2010, tan sólo en la última fase (2008-2010) contó con un presupuesto de 32 millones de Euros proporcionados por la Comisión Europea. La cantidad de personas involucradas en este proyecto de grid fue de más de 9,000 por mes.

[1] La continuación de este proyecto es conocida como EGI, que es una organización no temporal

Entre las disciplinas que EGEE ha beneficiado se encuentran:

- Astronomía y Astrofísica
- Arqueología
- Química computacional
- Dinámica de fluidos
- Ciencias computacionales
- Física de la materia condensada
- Ciencias de la Tierra, Geofísica
- Finanzas
- Física de Altas Energías
- Multimedia
- Ciencias de Materiales

Más de 120 artículos fueron enviados a la revista de divulgación de tecnologías grid *International Science Grid This Week* para su publicación en línea.

LHC

En el área de Física de Altas Energías un proyecto que destaca notablemente por su relevancia y tamaño es el Gran Colisionador de Hadrones (*Large Hadron Collider*, LHC), construido por el CERN en la frontera de Francia y Suiza, y el cual es el laboratorio de física de partículas más grande en el mundo. EGEE ayudó a la colección, distribución y análisis de información de los experimentos del LHC, lo que representa un gran logro y demostración del éxito de los grids computacionales[10]. El LHC es un acelerador y colisionador de partículas con una circunferencia de 27 kilómetros y a una profundidad entre 50 y 175 metros; según Sotomayor et al [5] el LHC es la máquina más grande construida por humanos. Esta máquina producirá una gran cantidad de información, aproximadamente generará 10 PetaBytes de información al año. Esta cantidad de información generada por el LHC es el 10 % de la información producida en el planeta en un año. Debido a la gran cantidad de información producida por el LHC es imposible

procesarla y almacenarla en un sólo sitio de cómputo. EGEE proporciona los recursos de cómputo y de almacenamiento, distribuidos en diferentes sitios alrededor del mundo, necesarios para poder analizar la información generada por el LHC del CERN. Según la publicación UNAMirada a la ciencia en su artículo El Gran Colisionador de Hadrones [3] este experimento tratará de reproducir el despuués del Big Bang al acelerar partículas, provocando que choquen entre sí a velocidades muy altas con el fin de generar nuevas partículas. El objetivo del experimento es buscar elementos para describir a toda la materia y las fuerzas del Universo, y explicar su estructura e interrelación.

El LHC está concentrado en varios experimentos:

- El detector *Compact Muon Solenoid*, CMS. El CMS grabará la información de las colisiones de protones contra protones de altas energías. La información de estas colisiones permitirá comprender mejor el origen de la materia en el universo, la partícula de Higgs, la supersimetría y dimensiones espaciales. Cientos de físicos del mundo participan en estudios de simulación, de tipo Monte Carlo, del detector con el fin de comparar las simulaciones con la información generada por el detector. Estas comparaciones ayudarán a calibrar el detector, medidas de procesos físicos y posibles descubrimientos científicos. El CMS tendrá un tiempo de vida de 15 años por lo que es importante contar con los recursos computacionales y de almacenamiento de una forma organizada [4].

- ATLAS

  Es un detector que buscará explicar la física que existe detrás del Modelo Estándar. Este detector proporcionará información de lo que el Modelo Estándar no puede explicar como: el origen de la masa, la generación de partículas, el imbalance de materia y antimateria en el Universo, la materia oscura. La información proporcionada por el detector será analizada por físicos, y estos formularán nuevas teorías acerca de lo que no se ha podido explicar aún.

- ALICE

  ALICE *A Large Ion Collider Experiment* es un detector de iones pesados, *heavy-ion detector* que explota las interacciones de núcleo-núcleo en las energías del LHC. Con el objetivo de estudiar la física de las interacciones de la materia en densidades extremas de energía, donde se espera la formación de una nueva fase de la materia, el plasma *quark-gluon*. Asimismo se estudiarán los hadrones, electrones, muones y fotones producidos en las colisiones de núcleos pesados.

- LHCb

El LHCb (*LHC beauty*, beauty se refire a *botoom quark*) es un experimento que explora que pasó después del Big Bang que permitió a la materia sobrevivir y construir el Universo actual.

2.2.2. Open Science Grid, OSG

OSG [14] está formada por diferentes organizaciones tanto académicas, de investigación y de tecnologías de cómputo ubicados en diferentes sitios de los EUA principalmente, aunque cuenta con colaboradores en Europa; el fin de OSG es formar una Ciber-Infraestructura (CI), con los recursos de sus integrantes, que ayude a la resolución de problemas comunes. Así en OSG, la colaboración mutua (establecida a través de Organizaciones Virtuales) es muy importante tanto para lograr las metas particulares de los proyectos de las VOs como para fortalecer la CI.

OSG está patrocinada con fondos de la *National Science Foundation*, NSF y del *Department of Energy*, DOE de USA; sus fondos son de 50 millones de doláres en el período comprendido en los años de 2006 a 2011. OSG cuenta con alrededor de 54,000 cores de procesamiento y aproximadamente con 24 PB para almacenamiento de información según el reporte anual de OSG en 2010 [15].

HTC

El modelo para la creación de grids computacionales en OSG está orientado en el *High Throughput Computing*, HTC; el HTC divide un problema en pequeños subproblemas independientes, debido al tamaño de estos subproblemas cada uno puede ejecutarse prácticamente en cualquier recurso computacional; así en el HTC no se necesitan recusos dedicados y además el propietario del recurso no pierde el control de su recurso. Para la construcción de la Ciber-Infaestructura, OSG proporciona un software llamado *Virtual Data Toolkit*, VDT. El VDT es una herramienta que empaqueta y facilita el uso de diferentes middlewares de Grid, como son: Globus Toolkit y Condor; además de proporcionar otros softwares como: Apache, MySql, Perl, etc.

Entre los proyectos más importantes que utilizan actualmente OSG se encuentran:

- El Observatorio de Ondas Gravitacionales con Detector Láser, *Laser Interferometer Gravitanional Wave Observatory* (LIGO)[16] es una herramienta que mide cambios en las ondas gravitacionales. Medir estas ondas es complicado porque éstas se debilitan al llegar a la tierra. LIGO detecta las ondas al medir los cambios en los patrones formados al encontrarse dos rayos láser. La sensitividad del

detector, *interferometer* es proporcional a la distancia a la que el rayo láser viaja. Debido a que las señales son muy débiles, el detector tiene que ser muy grande (EnisteinHome [17]). LIGO se encuentra localizado en EUA y tiene dos detectores, el tamaño del detector más grande es de 4 km. Son dos detectores ya que la intensidad de la señal de la onda gravitacional es muy baja y puede ser confundida con otro tipo de señal como puede ser un sismo. De esta manera, si los dos detectores registraron la misma señal se puede concluir que se detectó una onda gravitacional.

La información generada por los detectores es analizada por el OSG y por el *LIGO Data Grid*, este proceso es computacionalmente intensivo puesto que requiere de procesar grandes cantidades de información para detectar una onda gravitacional. La cantidad de información recabada por los detectores de LIGO es de un Terabyte de información por día. La información es dividida en pequeños segmentos y cada segmento es comparado con decenas de miles de patrones para identificar a las probables señales de una onda gravitacional, este proceso requiere de cientos a miles de procesadores, según el investigador de LIGO Ken Blackburn [18].

Figura 2.2: LIGO

- El experimento del LHC: Atlas. Que se describe en la sección 2.2.1
- El experimento del LHC: CMS. Que se describe en la sección 2.2.1

Asimismo, OSG promueve el uso de tecnologías Grid al realizar diferentes actividades como: cursos de entrenamiento, talleres, tutoriales, conferencias.

Los proyectos que se han beneficiado del OSG publicaron más de 100 artículos en el año 2008 y sólo en el período de 2009 a 2010 se publicaron 367 artíiculos según el reporte anual de OSG 2010 [15]. Para los próximos años, OSG espera cumplir con las demandas tecnológicas de la investigación y la ciencia según palabras de Paul Avery, quien es Investigador Principal Asociado(Co-PI) del OSG y además es parte de su Comité(*Co-chair*)[19]. Específicamente, espera satisfacer las necesidades en cómputo, almacenamiento, nuevas arquitecturas multicore, mpi en OSG, administración de la información, etc.

2.2.3. Teragrid

El Teragrid [20] es una infraestructura que integra diferentes equipos de supercómputo[2] (HPC) dedicados, distribuidos en diferentes sitios de EUA (*Indiana University*, IU; *Purdue University*, PU; *Oak Ridge National Laboratory*, ORNL; *National Center for Supercomputing Applications*, NCSA; *Pittsburgh Supercomputing Center*, PSC; *San Diego Supercomputer Center*, SDSC; *Texas Advanced Computer Center*, TACC; *University of Chicago/Argonne National Laboratory*, UC/ANL; *The National Center for Atmospheric Research*, NCAR) y que están conectados por redes dedicadas de alta velocidad (las velocidades son de 30 Gigabits por segundo o de 10 Gigabits por segundo).

Esta infraestructura permite hacer cálculos en los diferentes sitios de supercómputo y la gran velocidad de las redes permite hacer transferencias de información entre ellos. La capacidad de cálculo del TeraGrid es de más de un PetaFlop, cuenta con una capacidad de almacenamiento de 30 PetaBytes y además con 100 Bases de Datos de diferentes disciplinas. Es importante hacer notar que cada equipo de supercómputo tiene una arquitectura diferente a los otros equipos conectados en la infraestructura, por lo que la interacción entre estos equipos es compleja y difícil ya que requiere tanto conocimiento del propio equipo de supercómputo, como del manejo del mismo dentro de la infraestructura de TeraGrid.

El Teragrid permite a los científicos o investigadores comprobar su teorías o hipótesis, en donde el uso de supercomputadoras es indispensable. Por ejemplo, exiten areas de la física que no son concebibles sin el poder de cálculo de los equipos de supercómputo, según Ralph Z. Roskies, Co-Director Científico del PSC. Una de estas áreas es la Cosmología (ciencia que estudia como el cosmos evoluciona) y otra es la física de partículas elemental subatómica (*subatomic elementary particles physics*). En ambas áreas es ne-

[2]Un equipo de supercómputo o de *High Performance Computing*, HPC hace uso de gran poder de cálculo, memoria física robusta y está conectado internamente por redes de alta velocidad; ejemplo de ello, es KanBalam o clusters

cesario el poder de cálculo de las supercomputadoras con el propósito de calcular las implicaciones de lo que se cree que son las ecuaciones apropiadas (de una teoría o modelo). Así los avances en estas dos áreas están estrechamente ligadas con el desarrollo de las supercomputadoras [21].

Diversas disciplinas de diferente índole son beneficiados del TeraGrid. Como son:

- Física
- Química
- Astronomía
- Ciencias Biomoleculares
- Investigación en Materiales

Teragrid utiliza módulos para configurar el ambiente (SoftEnv) del usuario en los recursos de cómputos y de visualización de está infraestructura. Así, puede utilizar PBS para ejecutar trabajos en equipos locales; utilizar Globus Toolkit para trabajos remotos; y Condor para trabajos que no necesitan comunicación entre ellos.

2.3. Globus Toolkit 2

El Globus Toolkit versión 2 (GT2) es un conjunto de bibliotecas, servicios de cómputo y aplicaciones cliente−servidor para la construcción de grids computacionales. GT2 es una arquitectura abierta, creada por una comunidad y es de código libre. Soluciona problemas como: seguridad, administración de recursos, comunicación, detección de fallas, portabilidad, descubrimiento y administración de la información [4]. Debido a su robustez, el Globus Toolkit es uno de los *middleware* de grid más ampliamente utilizado en los grids computacionales.

El objetivo principal de GT2 es implementar protocolos básicos para los grids computacionales. GT2 no implementa los servicios o protocolos de los elementos de (*fabric*) como: calendarización, administración de sistemas de archivos o monitoreo del sistema; su objetivo es conectar los elementos de (*fabric*) con los de la capa de recursos.

2.3.1. Conectividad en GT2

La capa de conectividad en GT2 está definida por los protocolos de la Infraestructura de Seguridad en Grid, *Grid Security Infraestructure* (GSI) que proporcionan el acceso seguro de los usuarios a los grids computacionales. La GSI tiene como base la "Infraestructura de Llave Pública", *Public Key Infraestructure* (PKI).

La PKI es un sistema criptográfico (o de seguridad en cómputo) asimétrico, es decir es un sistema que utiliza diferentes formas de encriptar y desencriptar la información (utilizando una llave pública y una privada) que es enviada a través de una red como puede ser la Internet. La PKI hace uso de firmas digitales, certificados y de Autoridades Certificadoras. La firma digital es una secuencia de bits que permite garantizar la integridad del documento digital. El certificado (de Llave Pública) es un archivo que contiene información de la identidad del usuario, de la llave pública y firma digital de la Autoridad Certificadora. La Autoridad Certificadora, *Certification Authority* (CA) proporciona a entidades (usuarios, recursos, servicios) una identidad digital confiable, con esta identidad las mencionadas entidades pueden acceder a recursos de manera segura [7].

Además de usar PKI, GSI utiliza servicios de autenticación *single sign on*, delegación de certificados y autenticación mutua a través de certificados digitales.

La estructura de GSI es la siguiente:

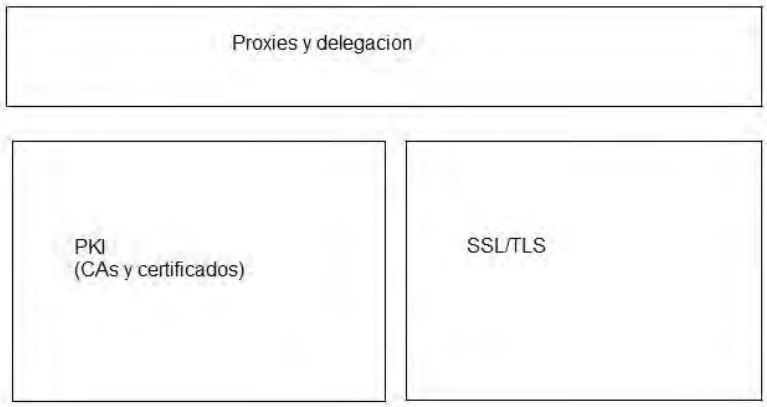


Figura 2.3: Infraestructura de Seguridad en Grid

Las características que proporciona GSI se describen a continuación:

- Autenticación mutua. GSI utiliza certificados tipo X.509 para la autenticación. La autenticación mutua implica que en una conversación (trasmisión de información) se autentica tanto el emisor de la información como el receptor de la misma. Esto garantiza que la información sea recibida por la persona correcta; es decir, la información sólo la recibe la persona que está autorizada al acceso de la misma. El protocolo que utiliza GSI para la autenticación mutua es *Secure Socket Layer*, SSL. Las partes que se autentican confían plenamente en la CA de su contraparte por lo que cada una de ellas tiene un certificado emitido y firmado digitalmente por la CA. El proceso de autenticación mutua se muestra a continuación:

  1. La entidad "A" se conecta con la entidad "B", posteriormente la entidad "A" envía su certificado a la entidad "B". De esta manera, la entidad "B" conoce la presunta identidad de la entidad "A", la llave pública de "A" y la CA que certifica la identidad de "A".
  2. La entidad "B" procede a reconocer el certificado de la entidad "A" como válido al revisar la firma digital de la CA y asegurarse que la CA firmó el certificado de "A".
  3. Se verifica la identidad de "A". La entidad "B" genera y manda un mensaje aleatorio a la entidad "A" y le pide a la entidad "A" encriptarlo.

4. La entidad "A" encripta el mensaje con su llave privada y se lo envía a la entidad "B". La entidad "B" desencripta el mensaje utilizando la llave pública de A. Si el mensaje desencriptado coincide con el mensaje enviado por "B", entonces B reconoce que la identidad de A es correcta y es quien dice ser.

5. El siguiente paso es la autenticación de la entidad "B". Para lo cual, la entidad "B" envía su certicado a la entidad "A". La entidad "A" verifica el certificado de "B" y envía un mensaje aleatorio para que la entidad "B" lo encripte. La entidad "B" encripta el mensaje y se lo envía a la entidad "A". La entidad "A" desencripta el mensaje y lo compara con su mensaje aleaotorio. Si tienen el mismo contenido el mensaje aleatorio y el mensaje desencriptado, la entidad "A" puede estar segura que la entidad "B" es quien dice ser.

Figura 2.4: Autenticación Mutua

- Autenticación *Single Sign On*. La autenticación *single sign on* permite que el usuario se autentique (asegurarse de que el usuario es quien dice ser [7]) ante varios recursos una sola vez. Autenticado el usuario una sola vez, se crea un certificado proxy que puede ser utilizado por uno o varios programa(s) para autenticarse con otros recursos en nombre del usuario, evitando que el usuario se autentique en cada recurso que utilice. Un certificado proxy es un certificado (como una carta poder) firmado digitalmente por el usuario que da derecho al poseedor de realizar operaciones en nombre del firmante (representado), estos derechos son válidos por un determinado período de tiempo. Por ejemplo, los certificados proxy permiten

iniciar un cálculo en múltiples recursos sin necesidad de autenticarse en cada uno de ellos.

- Delegación de certificados. La delegación de certificados es el metódo que reduce el número de veces que un usuario debe proporcionar su contraseña. Así por ejemplo, si un usuario desea utilizar varios recursos, el usuario tiene que ser autenticado en cada uno de ellos, para lo que se crea un proxy al que se delega el proceso de autenticación en cada uno de los recursos.

  A continuación se muestra el proceso de delegación de certificados:

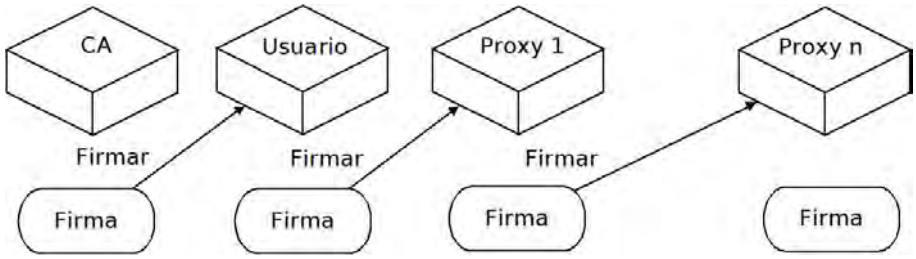


Figura 2.5: Delegación de certificados [8]

En este proceso se crea un certificado proxy firmado por el propietario del certificado. El certificado proxy tiene información del propietario del certificado, tiempo de vida del certificado proxy e información que indica que es un certificado proxy.

Asimismo, en la autenticación mutua, la parte remota recibe además del certificado proxy (firmado por el propietario del certificado) el certificado del propietario. En la autenticación mutua, la llave pública del propietario (obtenida del certificado) es utilizada para validar la firma en el certificado proxy. La llave pública de la CA es utilizada para verificar el certificado del propietario. Se establece así una cadena de confianza desde la CA hasta el certificado proxy [8].

2.3.2. Recursos en Globus 2

Globus Toolkit 2 implementa tres protocolos en la capa de recursos: un protocolo para el inicio de cómputo (o cálculo) en un recurso remoto; otro protocolo para el monitoreo y descubrimiento de recursos; y uno más para el transporte de la información.

El protocolo que inicia el cómputo en un recurso remoto en Globus Toolkit 2 es el Administrador y Reservador de Recursos en Grid, *Grid Resource Allocation and Management*

(GRAM). El protocolo GRAM permite la creación de cómputo remoto de manera segura y confiable, además de proporcionar la administración del cómputo remoto [4]. El protocolo GRAM de GT2 está constituído de tres procesos:

- El proceso *gatekeeper* que permite iniciar el cómputo en un recurso remoto.
- El proceso *job manager* que se hace cargo de la administración del computo remoto.
- El proceso GRAM *reporter* que monitorea y publica información del estado del cómputo en el recurso remoto.

En cuanto al monitoreo y descubrimiento de recursos, GT2 implementa el protocolo Servicio de Descubrimiento y Monitoreo, *Monitoring and Discovery Service* (MDS-2). MDS-2 proporciona un *framework* que permite acceder y descubrir información del recurso remoto. Esta información puede ser el estado o la configuración del recurso remoto. Ejemplo de lo anterior, es la configuración de un recurso remoto que realiza cómputo; estado de la red; o en general, cualquier capacidad que puede brindar un recurso remoto junto con sus respectivas políticas de su uso.

El protocolo MDS-2 de GT2 está formado por los siguientes componentes:

- Un registro local (publicador de información) que concentra la información de un recurso en específico y que además se encarga de publicarla.
- Registro colectivo (nodo index) que es utilizado para solicitar información de lo diversos recursos remotos.

En cuanto al transporte de información, GT2 implementa el protocolo GridFTP que es una versión ampliada del protocolo FTP. GridFTP utiliza protocolos de la capa de conectividad para la transferencia segura de la información entre recursos remotos, proporciona acceso a la información y administra paralelamente las transferencias de alta velocidad[4].

Como se muestra en la figura 2.6 en la subcapa de conectividad, el usuario del Grid computacional se autentica y genera un certificado proxy y un proceso de usuario (1) (el proceso actuará de aquí en adelante en nombre del usuario). Posteriormente en la subcapa de recursos, el proceso de usuario (1) utiliza el protocolo GRAM para crear un nuevo proceso de usuario (2) en el recurso remoto, el proceso de usuario (2) genera un nuevo certificado que podrá utilizar para solicitar otro servicio remoto. Al mismo tiempo, el proceso de usuario (2) es registrado a través del protocolo de MDS-2.

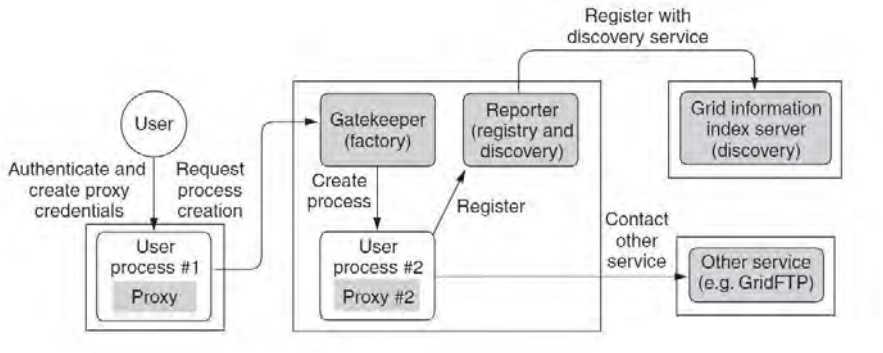


Figura 2.6: Funcionamiento de la capa de recursos en GT2

2.3.3. Colectivas

En la capa colectiva de la Arquitectura del Grid, GT2 implementa una biblioteca para la reservación de múltiples recursos llamada *Dynamically Updated Resource Online Co-allocator* (DUROC).

2.4. Open Grid Services Architecture

La comunidad de Globut Toolkit rediseñó los protocolos de GT2 con el fin estandarizarlos. El rediseño produjó la Arquitectura de Servcios de Grid Abiertos, *Open Grid Services Architecture* (OGSA) con los objetivos de:

- Agrupar tareas comunes en componentes con el objetivo de reutilizar diversas funciones de los protocolos de GT2 y evitar la reimplementación de funciones en las futuras versiones de Globus Toolkit.

- Basar esta nueva arquitectura en los principios de la Arquitectura Orientada a Servicios.

La arquitectura OGSA es utilizada a partir de la versión de GT3.

2.4.1. Arquitectura Orientada a Servicios

En una Arquitectura Orientada a Servicios, *Service Oriented Architecture* (SOA) todos los componentes de software son modelados como servicios. Un servicio es una entidad que proporciona una capacidad a sus clientes mediante un intercambio de mensajes. El servicio se define al identificar secuencias de intercambios de mensajes específicos que causan que el servicio desarrolle una operación. Así, una Arquitectura Orientada a Servicios es aquella en la que todos los elementos que la constituyen son servicios y además cualquier operación visible a la arquitectura es el resultado del intercambio de mensajes. SOA se centra en el diseño de la interfaz. En SOA, se utilizan los servicios con una interfaz bien definida y el servicio puede ser utilizado en multiples contextos. Esto permite que las aplicaciones sean bajamente acopladas (*loosely coupled*) y sean integradas a nivel de la interfaz y no a nivel de la implementación [6].

SOA está constituido de tres elementos:

- Un proveedor de servicio: Este elemento crea la descripción del servicio, publica la descripción en registros y proporciona el servicio (invocación del servicio) a partir de la solicitud de un demandante de servicios.
- Demandante de servicio: Este componente encuentra la descripción de un servicio en un registro y utiliza las descripciones de servicio para invocar al servicio donde está hospedado en el proveedor de servicio.
- Registro de servicio: Publica las descripciones de servicios proporcionadas por los proveedores de servicio y permite la búsqueda de descripciones de servicio a un demandante de servicios.

Ejemplos de servicios:

- Servicio de almacenamiento. Las operaciones que puede realizar este servicio pueden ser almacenamiento y recuperación de la información, reservación de espacio, monitoreo del estado del servicio de almacenamiento, consulta de las políticas para el acceso al servicio de almacenamiento, etc.
- Servicio de transferencia de la información. Puede proporcionar operaciones para solicitar transferencia de información entre servicios de almacenamiento, monitoreo del estado del servicio de transferencia y consulta de las políticas de prioridad de las solicitudes de transferencia.

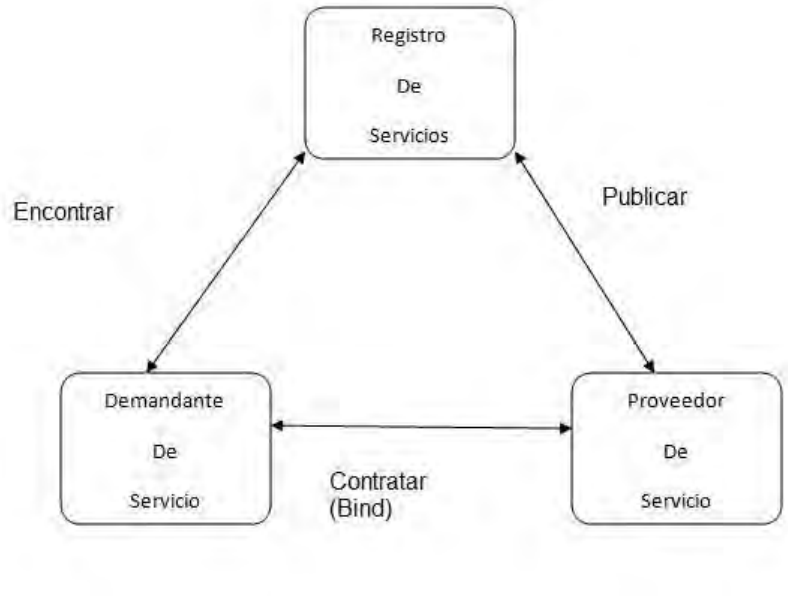


Figura 2.7: Arquitectura Orientada a Servicios

- Servicio de reparación de errores. Este servicio puede monitorear el estado de otros servicios, proporcionar operaciones que permitan la notificación de errores, además de que permite la consulta de políticas sobre quien tiene autorizado recibir tales notificaciones.

2.4.2. Servicios web, *Web Services*

Los *Web Services* son una infraestructura para crear aplicaciones distribuidas. Los *web services* están basados en SOA, donde los clientes fungen como demandantes del servicio y los servidores son los provedores del servicio. Están basados en estándares abiertos como XML y HTTP. Una definición que define ampliamente lo que es un *web service* es la siguiente:

- Un *web service* es una plataforma bajamente acoplada, encapsulada, modular a través de componentes (que se encuentran en el lado del servidor). Estos componentes pueden ser descritos, publicados, descubiertos e invocados por medio de una red. Y una caracterísitca importante del *web service* es que no importa el lenguaje de programación utilizado para la creación del componente.

En la parte de la definición que hace mención de "bajamente desacoplada" debe entenderse que la implementación del *web service* puede modificarse sin afectar al cliente que utiliza el servicio, siempre y cuando la interfaz del servicio no sea modificada. El término "encapsulado" en la definición debe entenderse que la implementación de un servicio web es totalmente transparente al cliente que utiliza el servicio [7].

Web services utiliza los siguientes estandares: SOAP, WSDL y UDDI:

- SOAP es un protocolo de comunicación para intercambio de mensajes (entre clientes y servidores) en un formato XML sobre un protocolo de transporte (generalmente HTTP).
- UDDI es un estándar para el registro, publicación y descubrimiento de servicios.
- WSDL, *Web Service Description Language* es una especificación basada en XML que es utilizada para describrir a un servicio web. Describe lo que el servicio puede hacer, cómo invocarlo, su localización, etc. Los elementos en WSDL son los siguientes:
  - Servicio (*Service*): es un conjunto de puertos relacionados.
  - Puerto (*Port*) : Define un servicio individual que específica una dirección para un contrato (binding).
  - Interfaz (*PortType*): Define de forma abstracta las operaciones que puede proporcionar un servicio.
  - Operación (*Operation*): Define el tipo de mensajes (involucrados en la operación) que puede realizar la interfaz.
  - Mensaje (*Message*): Define los tipos de datos involucrados en el propio mensaje.
  - *Binding*: Identifica el protocolo y formato de la información para las operaciones y mensajes definidas en una interfaz (PortType).

2.4.3. Particularidades de OGSA

OGSA define una arquitectura abierta, estándar para aplicaciones que utilicen los grids computacionales. El próposito de OGSA es estandarizar los servicios que se encuentran en un Grid computacional (servicios de administración de jobs, servicios de administración de recursos, servicios de seguridad, etc) al especificar un conjunto de interfaces

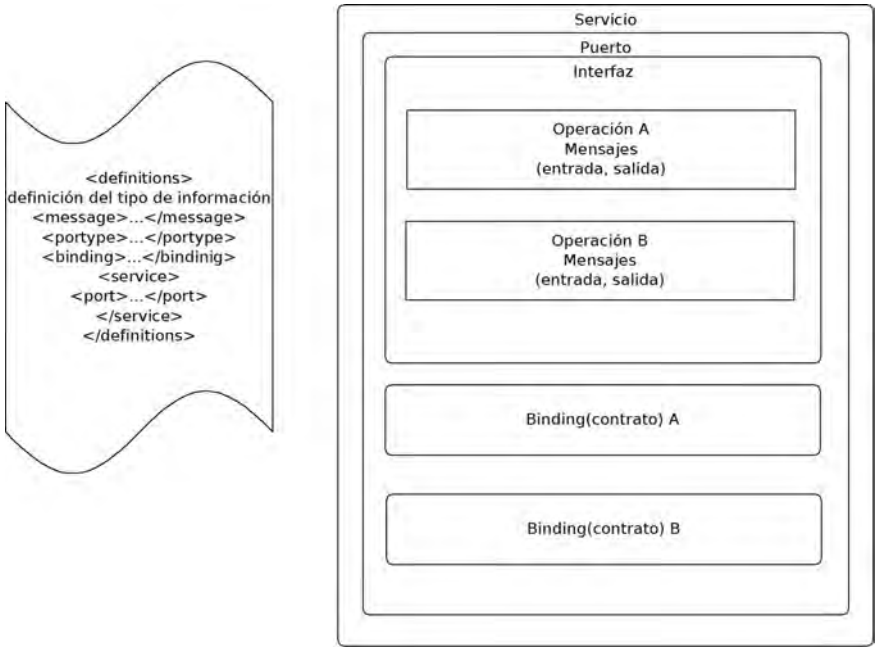


Figura 2.8: Estructura de un documento WSDL [7]

estándar para estos servicios [5]. OGSA está basada en *web services* modificados y ampliados para el ambiente de los Grids computacionales. OGSA amplia el concepto de los web services al introducir interfaces y convenciones siguientes:

- Interfaces para administrar la creación, ciclo de vida y destrucción de servicios de los grids computacionales. Esto debido a la naturaleza dinámica y temporal del ambiente de los grids.
- Soportar el estado que mantienen los servicios de los grids computacionales
- Notificación en el cambio de un servicio de un grid computacional a petición de un cliente.

Los *web services* modificados para el ambiente de Grid y que a su vez cumplen con los requisitos de OGSA son llamados Servicios Grid, *Grid services*. Los servicios en OGSA están compuestos de dos elementos:

- Plataforma de OGSA (*OGSA platform*): Son servicios basados en Grid relacionados con autenticación, envío de jobs, monitoreo de jobs, acceso a información, etc.

- Servicios núcleo de OGSA (*OGSA core services*): Sirven para la creación y destrucción de servicios, administración del ciclo de vida, registro de un servicio, descubrimiento de servicios, notificaciones. OGSA también presenta las interfaces de los servicios Grid como: *GridService*, *Factory*, *Registration*, *HandleResolver* y *Notification*.

Asimismo, OGSA presenta dos conceptos: instancia de servicio, *service instance* e información de servicio, *service data*, que están asociados con cada *Grid service* para soportar las características de temporalidad y de estado de los mismos. OGSA define varios matices que deben seguir los servicios Grid (características de los mismos e interfaces necesarias), pero no especifica cómo las interfaces deben ser implementadas.

2.4.4. OGSI

OGSI, *Open Grid Service Infraestructe* especifica técnicamente cómo implementar los servicios núcleo de OGSA utilizando web services. OGSI especifica qué se necesita implementar para cumplir con OGSA. De esta manera, otra definición de *Grid service* es: un servicio web que cumple con la especificación OGSI.

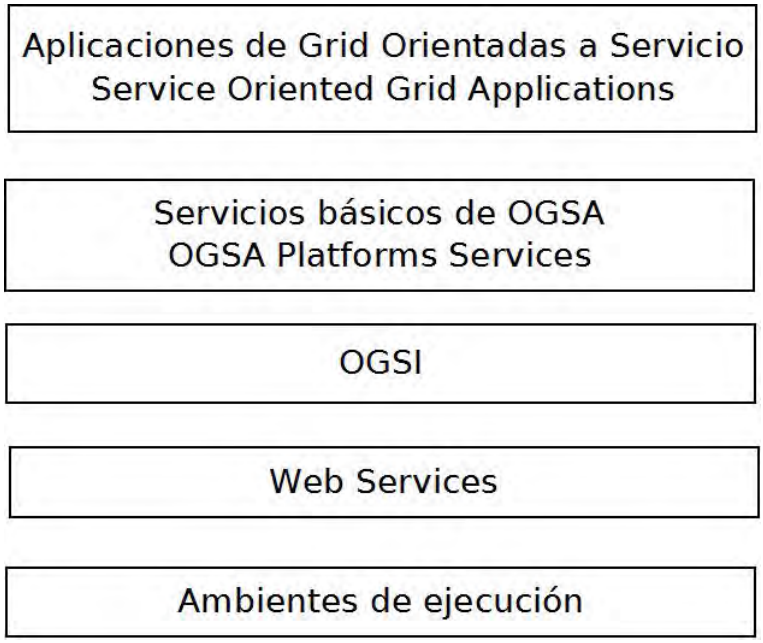


Figura 2.9: Construcción de aplicaciones Grid de acuerdo a OGSA utilizando OGSI

Instancia de servicios, *service instance*

Una característica de un Web service es que es permanente, mientras que un Grid service es temporal. Una instancia de servicio de Grid, *Grid service instance* es la invocación de un servicio Grid que puede ser creado y posteriormente destruido dinámicamente. A un servicio Grid que es capaz de crear una invocación de servicio se le denomina *Service Factory*, que es un servicio permanente. Así un servicio Grid puede involucrar diversas instancias de servicios, las cuales fueron creadas por sus respectivas *factories*. De esta manera, las implementaciones de los servicios son independientes de ubicación, plataformas y lenguajes de programación. Un GSH, *Grid Service Handle* identifica a cada instancia de servicio de Grid. La información del GSH junto con la información que se necesita para interactuar con la instancia del servicio es encapsulada para formar un GSR *Grid Service Reference*. Los GSH son permanentes mientras que los GSR para una instancia de servicio Grid pueden cambiar sobre el tiempo de vida del servicio mismo.

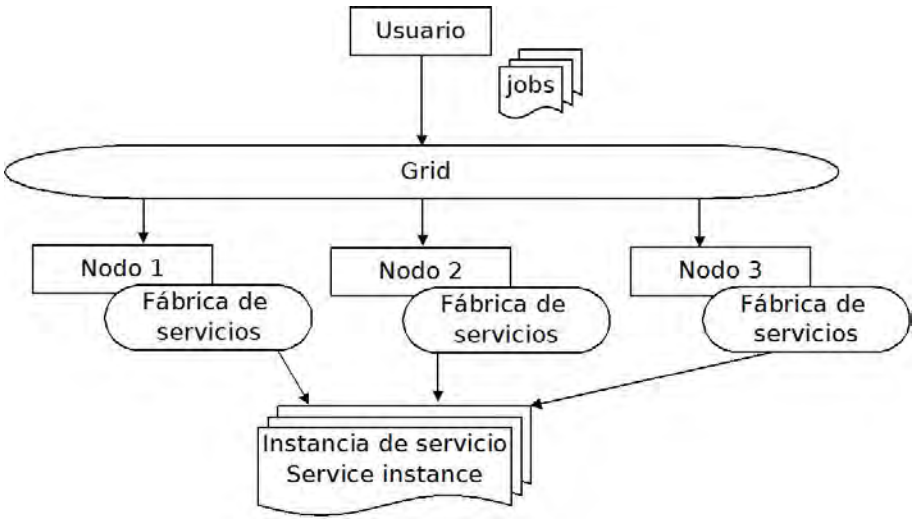


Figura 2.10: Envío de jobs con múltiples instancias de servicios grid

Información de servicio, *service data*

Cada servicio Grid tiene asociada información que específicamente es una colección de elementos XML encapsulados como los Elementos de Información del Servicio, *Sevice Data Elements* (SDE). Esta información del servicio proporciona información de la instancia del servicio y de los estados de su ambiente de ejecución. Así, a diferencia

de los web services que no tienen estado, los Servicios de Grid tienen estado y pueden ser monitoreados por si mismos (*introspected*). Así una Fábrica de Servicios, *Service Factory* puede crear instancias de servicios. Y a su vez, cada una de las instalancias tiene un Conjunto de Información de Servicio, *Service Data Set* (SDS). Ese SDS tiene cero o más SDE donde cada SDE puede ser de diferente tipo.

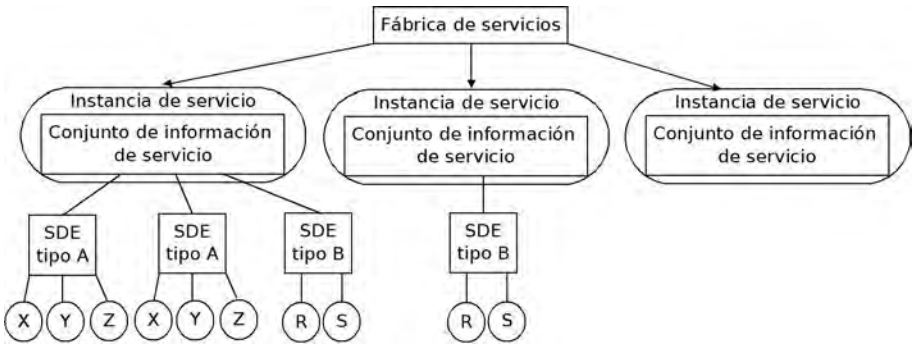


Figura 2.11: Vista de una fábrica de servicio

OGSA portTypes

OGSA proporciona las siguientes interfaces para definir Servicios de Grid (extendidas de los portTypes de WSDL). La interfaz *GridService* es implementada por todos los servicios, las demás interfaces son opcionales.

- Interfaz del Servicio Grid, *GridService*. Es la interfaz base en OGSA. Esta interfaz tiene métodos para el descubrimiento de servicios (*FindServiceData*).
- Interfaz de Fábrica, *Factory*. Esta interfaz es implementada por un servicio de Grid permanente que crea una fábrica, *factory*. Esta interfaz puede crear instancias de servicio Grid por medio del método *createService*.
- Interfaz de Solucionador de Manejador, *HandleResolver*. Esta interfaz permite a un servicio Grid resolver un GSH a un GSR por medio del método FindbyHandle.
- Interfaz de Registro, *Registration*. Al servicio Grid que implementa la interfaz Registration se le conoce como registro. El registro permite el descubrimiento de servicios al mantener grupos de GSH y sus respectivas políticas. La interfaz "*registration*" permite a una instancia de servicio registrar a un GSH con el servicio de registro, y así como un conjunto de información de servicio que contiene información del GSH registrado a los estados de la ejecución de la instancia de servicio.

Para registrar un servicio se utiliza el método *RegisterService* y para realizar la acción contraria se utiliza el metodo *UnRegisterService*.

- Interfaces de Notificaciones, *NotificationSource* y *NotificationSink*. Las notificaciones en OGSA permiten subscribir a las partes interesadas elementos de información de servicio y recibir notificaciones cuando los valores son modificados.

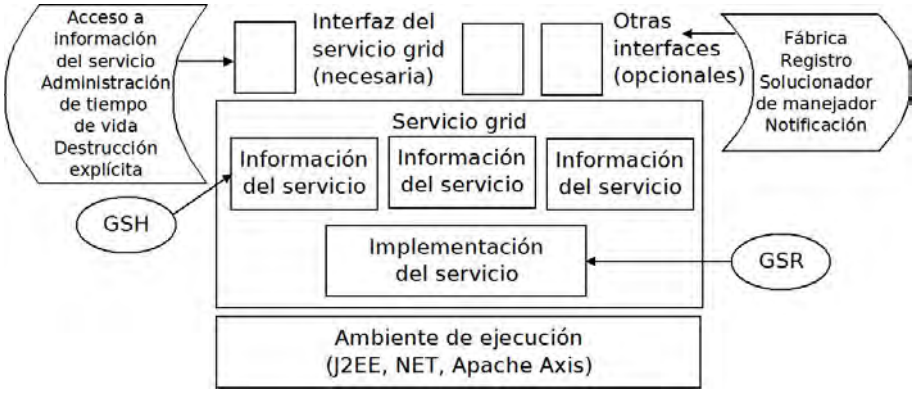


Figura 2.12: Estructura de un servicio Grid en OGSA

2.5. Tecnologías de Portales

2.5.1. Definición de Portal

Un portal se puede definir como un mecanismo o servicio que provee un punto único de entrada a un sistema que integra datos, información, aplicaciones y servicios [9].

2.5.2. Definición de Portal basado en Web

Un portal basado en web es un portal que se apoya en tecnologías web (como son los applets, servlets, portlets, en el API de Java; protocolos HTTP y HTML). La ventaja de estas tecnologías es que son muy conocidas y son ampliamente utilizadas. Los portales basados en web proporcionan las mismas fucionalidades que cualquier portal como son: servicios, aplicaciones e información.

2.5.3. Definición de Portal Grid

Un portal Grid (*Grid Portal*) es un portal basado en Web que proporciona una interfaz simple e intuitiva que permite acceder y utilizar un grid computacional. Esta interfaz brinda información de un grid computacional y además permite el acceso y uso de los recursos del mismo a quienes tengan los permisos correspondientes [9].

Un portal Grid facilita el uso de los grids computacionales al proporcionar una interfaz basada en Web que esconde la complejidad inherente a los grids computacionales. Esta interfaz permite a los usuarios interactuar de una forma sencilla con recursos distribuidos y heterogéneos.

2.5.4. Particularidades de un Portal Grid

Un portal Grid puede proporcionar un medio para monitorear la disponibilidad de los recursos, el estado de los jobs en ejecución o la carga de los recursos de un grid computacional. Para conseguirlo, el portal Grid debe ser capaz de interactuar con el middleware de Grid que se encuentre en los recursos del grid computacional.

Un portal Grid es capaz de proporcionar información de los diferentes recursos de un grid computacional de una forma centralizada, fácil, intuitiva y sencilla en comparación con el middleware de Grid.

Una ventaja de los portales Grid es que no es necesaria la instalación de software adicional, sólo se necesita un navegador web para utilizar los recursos de un grid computacional.

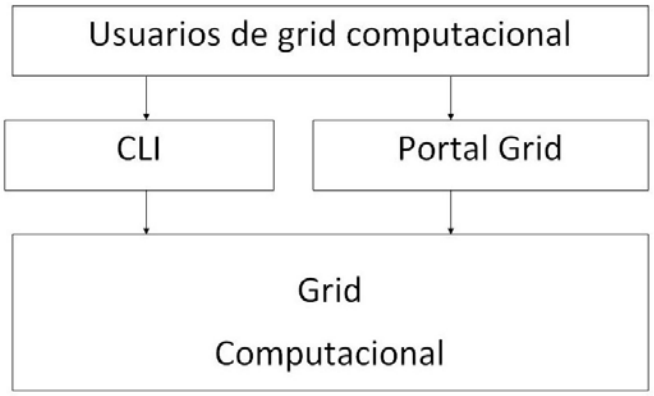


Figura 2.13: Conección a un grid computacional

La figura 2.13 muestra como los usuarios de grid pueden acceder a un grid computacional. Una forma es utilizar una interfaz de línea de comandos y conectarse directamente al middleware de grid, este middleware de grid utilizará los servicios grid que están disponibles en los recursos del grid computacional. Otra forma de acceder al grid computacional, es conectarse a través de un portal Grid, este portal recibirá las peticiones de los usuarios de grid a través de la interfaz web, e interactuará con el middleware de grid con el fin de obtener una respuesta a la petición del usuario.

2.5.5. Tecnologías CoG

Una posible forma de que interactúen los recursos de un grid computacional con tecnologías web es utilizando tecnologías *Commodity Grid Toolkit* (CoG). Las tecnologías CoG permiten la reutilización de código, menos tiempo invertido en la creación de un portal Grid y permiten el uso de diversas APIs como Java o Python. Específicamente, los portales que utilizan este tipo de tecnologías conectan un servicio grid con su correspondiente servicio web. Esto permite realizar servicios más complejos a partir de servicios ya existentes.

Java CoG

Una implementación de tecnologías CoG es Java CoG. Java CoG actúa como puente entre los servicios Grid y los portales Grid utilizando Java. Esta implementacion permite acceder a servicios grid existentes y crear servicios grid basados en Java.

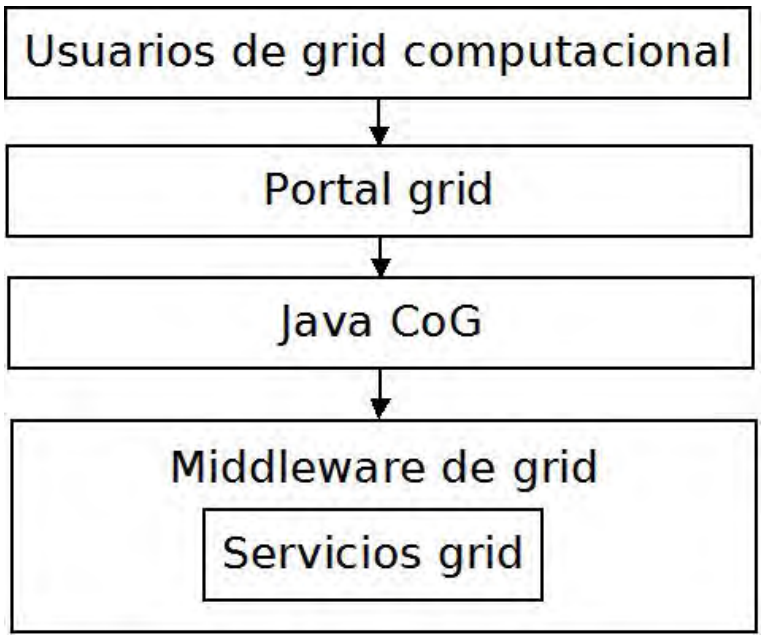


Figura 2.14: Java CoG

Capítulo 3

OGCE, *Open Grid Computing Environment*

3.1. Descripción general

OGCE es un portal Grid que proporciona un punto de acceso sencillo y transparente a los diferentes recursos de un grid computacional que un usuario esté autorizado a utilizar. Según Li, Maozhen et al [7] este portal proporciona a los usuarios de grid una interfaz personalizada de los recursos de software y hardware específicos al dominio particular del usuario de grid.

3.1.1. Arquitectura de los portales Grid

Los portales grid están compuestos de tres capas que se ilustran en la figura 3.1

1. La primera capa consiste de navegadores web.
2. La segunda capa de los servidores web.
3. La tercer capa la conforman los recursos y servicios finales del grid computacional, como pueden ser bases de datos, equipos de alto rendimiento, dispositivos de almacenamiento.

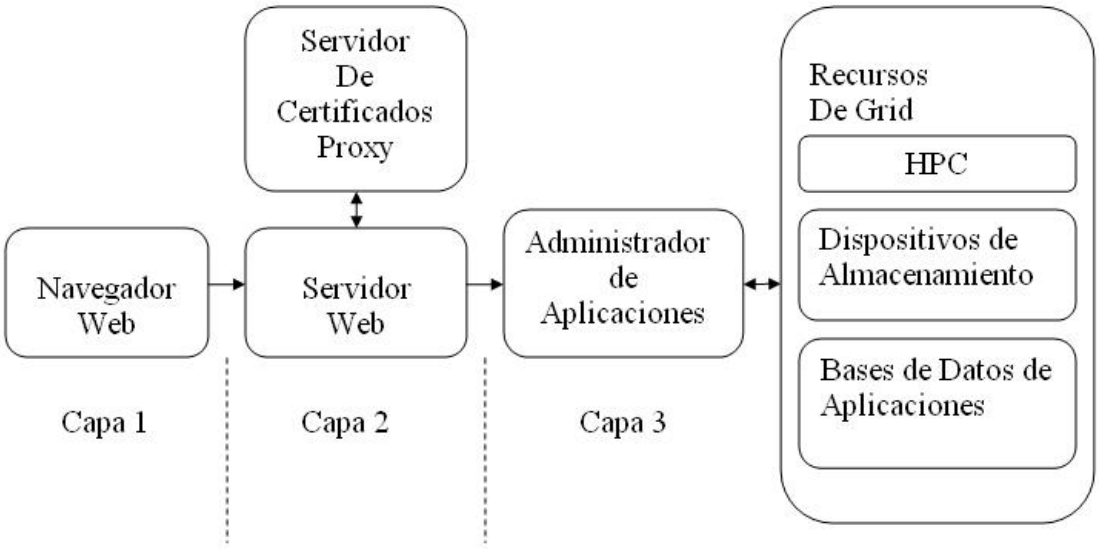


Figura 3.1: Arquitectura de un portal Grid

3.1.2. Funcionamiento de un portal grid

Un usuario de grid utiliza el navegador web para conectarse de forma segura con el servidor web. Posteriormente el servidor web obtiene el certificado proxy del usuario de un servidor de certificados proxy y utiliza este certificado para auntenticar al usuario con los recursos del grid computacional.

Una vez que el portal grid ha obtenido el proxy, el usuario puede interacturar con los recursos de grid. Para utilizar estos recursos, el usuario mediante la interfaz del portal grid puede definir los parámetros necesarios para ejecutar un trabajo, ante esto el servidor web ejecuta el administrador de aplicaciones (*application manager*); el cual controla y monitorea la ejecución de las tareas en un grid computacional. El servidor web delega el certificado proxy del usuario al administrador de aplicaciones para que éste pueda actuar en nombre del usuario ante los recursos del grid.

Algunos de los servicios que proporciona un portal grid son:

- Autenticación:

  Este servicio se encarga de autenticar a los usuarios utilizando su certificado proxy ante un grid computacional mediante un portal grid. Mediante este servicio, el usuario puede pedir al portal grid acceder a los recursos del grid a nombre suyo.

- Adminstración de Trabajos

  Este servicio permite a los usuarios de un grid administrar sus trabajos de una forma cómoda y sencilla. Así los usuarios pueden mandar a ejecución sus trabajos a través de un navegador web (de forma confiable y segura), monitorear el estado de sus trabajos y cancelar trabajos cuando es necesario.

- Transferencia de información

  Este servicio permite a los usuarios del portal grid transferir información en los recursos de grid desde sus máquinas locales. La información que se puede transferir puede ser de diversa índole, como archivos de configuración, archivos de entrada para la ejecución de sus aplicaciones o archivos de salidas generadas por las mismas aplicaciones.

- Servicios de información.

  Este servicio muestra información de los diferentes recursos de un grid. Esta información es recabada de los diferentes recursos de cómputo de un grid y es publicada para el conocimiento de los usuarios. La información recabada puede ser de dos tipos: estática o dinámica. Estática se refiere a información como tipo de Sistema Operativo o CPU; dinámica es la carga que tiene el CPU, memoria libre, espacio en disco.

3.2. Portlets

Los portlets son importantes porque permiten la personalización de los portales grid, hace a los portales poco acoplados con los middleware de grid, etc.

3.2.1. Definición de Portlet

Un portlet gráficamente es una ventana que ofrece un servicio dentro de un portal grid. Es decir, un portlet visualmente tiene su propia ventana, título (de portlet), contenido (de portlet) y acciones (maximizar, minimizar).

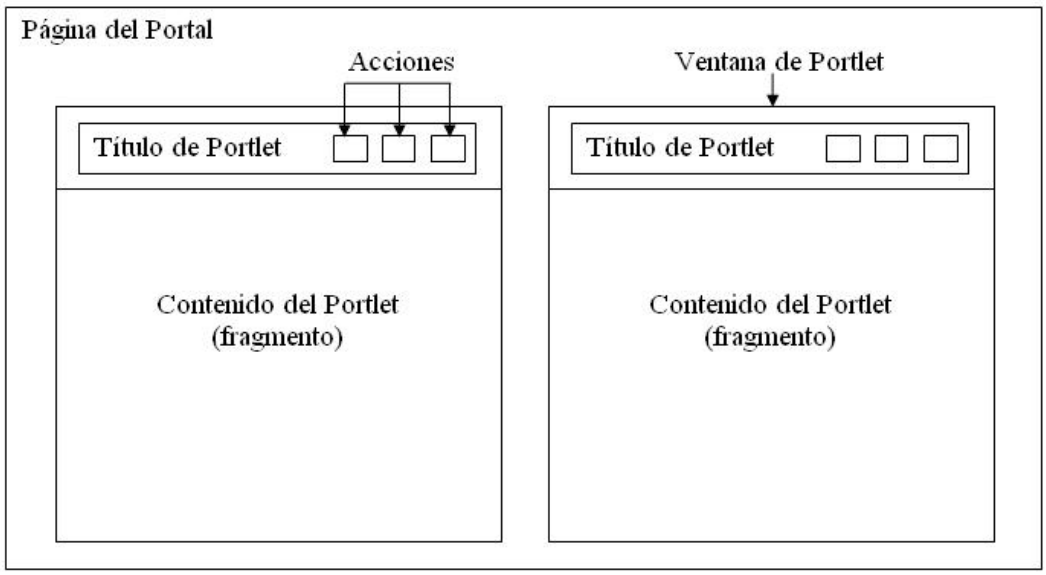


Figura 3.2: Portlet

Específicamente, un portlet es un componente de software administrado por un contenedor de portlet, que maneja las peticiones hechas por un usuario (de un portal grid) y genera contenido dinámico. Así los portlets como componentes de la interfaz de usuario, pasan información a la capa de presentación de un portal.

El contenido que genera un portlet es llamado fragmento. Ese framgmento es un conjunto de código HTML. La totalidad de los fragmentos generados por los portlets forman un documento completo de una página web.

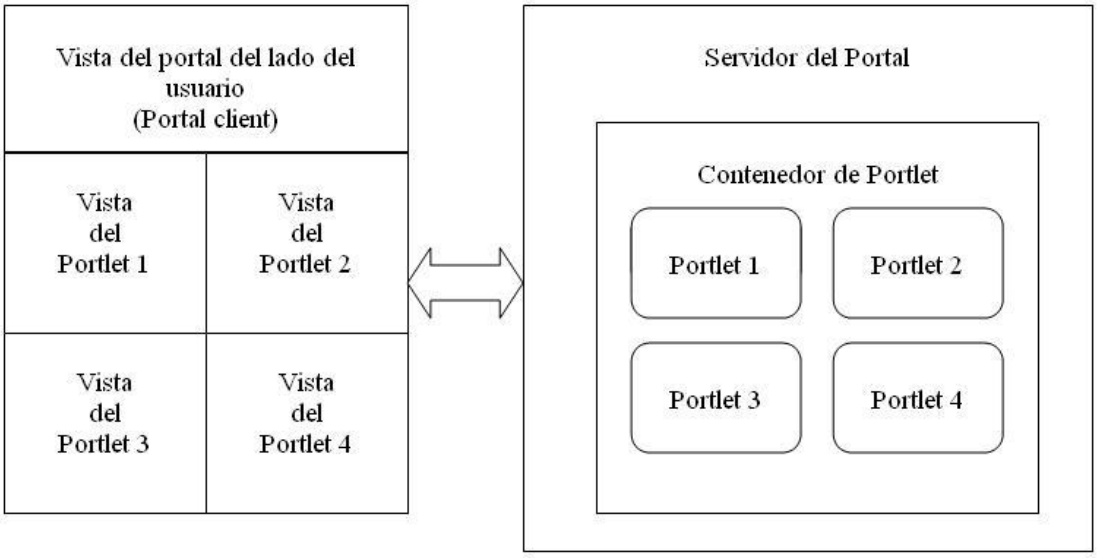


Figura 3.3: Portal con portlets

Contenedor de portlets

Un contenedor de portlets administra el ciclo de vida de los portlets en un portal. Así un contenedor de portlets proporciona un ambiente de ejecución para que los portlets sean invocados, ejecutados y destruidos. Los portlets se basan en la infraestructura del portal para interactuar con otros portlets, buscar certificados, almacenar información, acceder a contenido e información del usuario, etc.

Específicamente el contenedor de portlets está implementado en una capa superior a un contenedor de servlet, lo que le permite reutilizar la funcionalidad que proporciona los contenedores de servlet.

Portlets y Servlets

Los portlets al igual que los servlets procesan peticiones HTTP, realizan acciones y generan salida HTML. En el caso de un portlet su salida es sólo una parte de la página web. El servidor del portal se encarga de reunir toda información generada por los portlets y darles formato para la creación de la página web.

Las características de los portlets hacen que sean más flexibles y dinámicos que los servlets. De esta manera se pueden realizar los siguientes cambios en un portal sin necesidad de reiniciar el servidor del portal:

- Los portlets pueden ser creados y borrados dinámicamente.
- El administrador del portal puede cambiar los parámetros de configuración de un portlet.
- Una aplicación de portlets puede ser instalada o removida de la interfaz del usuario del portal.

Particularidades de un Portlet

La ventana de un portlet está compuesta de los siguientes elementos:

- Barra de título: Se muestra el título del Portlet.
- Decoraciones: Botones para cambiar el estado de la ventana del portlet (maximizar, minimizar); cambio en el modo del portlet, como por ejemplo, editar parámetros de configuración del portlet o solicitar ayuda del funcionamiento del portlet.
- Contenido producido por el portlet.

El ciclo de vida de un portlet es el sguiente:

- Invocación (*initialization*): Se utiliza la clase que inicializa al portlet y lo pone en servicio.
- Manejo de peticiones: Se reciben peticiones de la interacción de un usuario con un portlet en un portal y posteriormente se hace un procesamiento de acciones, se genera y actualiza (*rendering*) el contenido.
    - En el procesamiento de acciones se realizan las acciones solicitadas por un usuario, como pueden ser un click en una liga de un portlet. El procesamiento de acciones debe finalizar antes de que se actualicen(*rendering*) los otros portlets. En esta fase, el portlet puede cambiar el estado del portal.

- Generación y actualización de contenido (*rendering content*): En esta fase cada portlet produce su contenido (de *markup*) que será enviado al usuario del portal. Esta actualzación no cambia el estado del portlet, sólo muestra las actualizacines de la página web sin modificar el estado del portlet. La actualización de portlets puede ser desarrollada de forma paralela.

- Finalización: Se remueve un portlet de la página del portal utilizando la clase correspondiente.

Accediendo web services a través de portlets

Según Li, Maozhen et al [7] un portal web recibe una petición de un usuario, lo cual genera y despacha eventos al portlet utilizando los parámetros de la solicitud y entonces se invoca al portlet a ser desplegado mediante la Interfaz de Invocación de Portlet *Portlet Invocation Interface*, PII. El diseño interno de un portlet es similar al modelo Modelo Vista Controlador (MVC): en la parte del controlador se reciben las peticiones entrantes de PII; mientras que en la parte del modelo (encapsula la información y la lógica de la aplicación que accede a las aplicaciones o contenido web) se realizan las operaciones necesarias sobre los recursos web para obtener la respuesta a la petición del usuario; y finalmente en la parte de la vista se presentan los resultados al usuario.

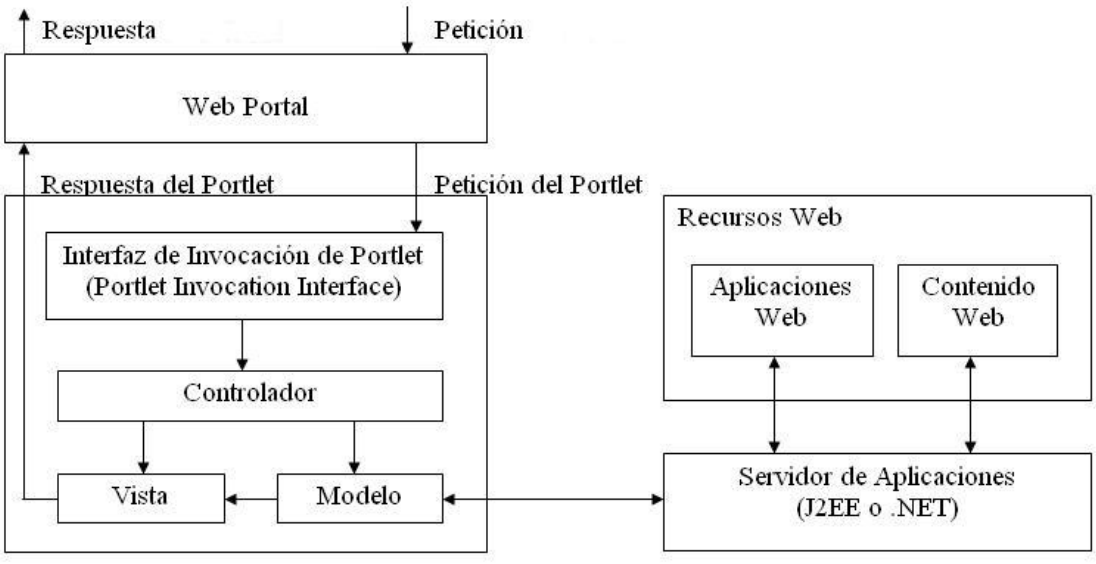


Figura 3.4: Estructura interna de un Portlet

Eventos para acceder a una página web con portlets

La secuencia de eventos para acceder a un a página web a través de portlets es mostrada a continuación:

- Un usuario desde un navegador web y después de haber sido autenticado, realiza una petición HTTP al portal.
- El portal recibe la solicitud
- El portal determina si la petición contiene una acción a ser realizada por alguno de los portlets asociados con la página del portal.
- Si la acción está asociada a un portlet, entonces el portal solicita al contenedor de portlets invocar al portlet para llevar a cabo el procesamiento de la acción.
- El portal invoca a los portlets restantes, a través del contenedor de portlet para obtener sus fragmentos que serán incluidos en la página del portal.
- El portal agrega los fragmentos en la página del portal y se manda esta al usuario.

Especificaciones de Portlets

Con el objetivo de tener interoperabilidad entre las diferentes implementaciones de portlets surgieron las siguientes especificaciones:

- WSRP (*Web Services for Remote Portlets*). Esta especificación proporciona una forma de fácil conexión (*plug and play*) de portlets, actúa como intermediario de aplicaciones de contenido e integra aplicaciones de diferentes orígenes. Así WSRP permite a las aplicaciones utilizar o producir *web services*. Estos web services incluyen elementos e información de presentación que permiten seleccionar y mostrar los portlets de diferentes proveedores sin la necesidad de integración de código. Los objetivos de WSRP son:
  - Permitir a los web services ser conectados dentro de portales que sigan el estándar.
  - Permitir crear y publicar contenidos y aplicaciones como web services.
  - Permitir la publicación de portlets de un portal para que puedan ser utilizados por otros portales sin complicaciones de integración.

- JSR 168. JSR (*Java Specification Request*) 168 es una especificación de portlet que define un conjunto de APIs de Java que permite la interoperabilidad entre portales y portlets. Define a un portlet como un componente web basado en Java, administrado por un contenedor de portlet que procesa peticiones de usuario y genera contenido dinámico. Los portales utilizan a los portlets como componentes de interfaz de usuario conectables que proporcionan una capa de presentación a los sistemas de información. Los objetivos de esta especificación son los siguientes:

  - Define un ambiente de ejecución (contenedor de portlets) para los portlets.
  - Define un API entre el contenedor de portlet y los portlets.
  - Proporciona mecanismos para almacenar información de los portlets.
  - Permite a los portlets incluir servlets y JSPs.
  - Define cómo empaquetar portlets para su fácil uso.
  - Permite que los portlets que cumplan con esta especificación puedan ser ejecutados como portlets remotos utlizando WSRP.

De acuerdo a Li, Maozhen et al [7] estas especificaciones son complementarias. La especificación JSR 168 define un API de Portlet estándar específica a los portales basados en Java, mientras que WSRP define un API general que permite a los portales utilizar portlets de cualquier tipo. De esta manera los portlets que siguen la especificación JSR 168 pueden ser encapsulados (*wrapped*) y presentados como servicios WSRP para su publicación como servicios; mientras que servicios WSRP pueden ser utilizados como portlets con la API de Portlet de Java para ser incluidos en los portales. Es decir, el JSR 168 define un conjunto de APIs de Java que permite a los portlets ejecutarse en portales que sigan esta especificación y WSRP permite a los web services ser mostrados como portlets en una forma de fácil conexión.

GridSphere

El portal Grid OGCE está basado en el *framework* de GridSphere. Este *framework* ayuda a la creación de portales web que utilizan portlets. GridSphere es un proyecto de código libre que proporciona un *framework* de portlet y una infraestructura para reutilizar portlets. De esta manera los desarrolladores pueden crear y empaquetar aplicaciones basadas en portlets de terceras partes y ejecutarlas dentro del contenedor de portlet de GridSphere.

GridSphere tiene un conjunto de portlets y servicios básicos que proporcionan la infraestructura necesaria para crear, desarrollar y administrar un portal web. Las características de GridSphere son las siguientes:

- Soporta portlets de terceras partes, lo que permite que estos sean integrados fácilmente en el contenedor de portlet de GridSphere.
- Portlets base (*core*) que proporcionan una funcionalidad básica para conexión al portal y administración de usuarios.

GridSphere proporciona específicamente un portal, un contenedor de portlets y un conjunto de portlets básicos. Entre estos portlets se encuentran portlets para la administración de usuarios o grupos y un portlet que permite agregar o quitar portlets del ambiente de trabajo del usuario.

Grid Portlet

Un Grid Portlet es un portlet dentro de un portal grid y que además está asociado con un servicio grid final (*backend grid service*). La figura 3.5 muestra como acceder a un servicio grid de un portal grid mediante un portlet:

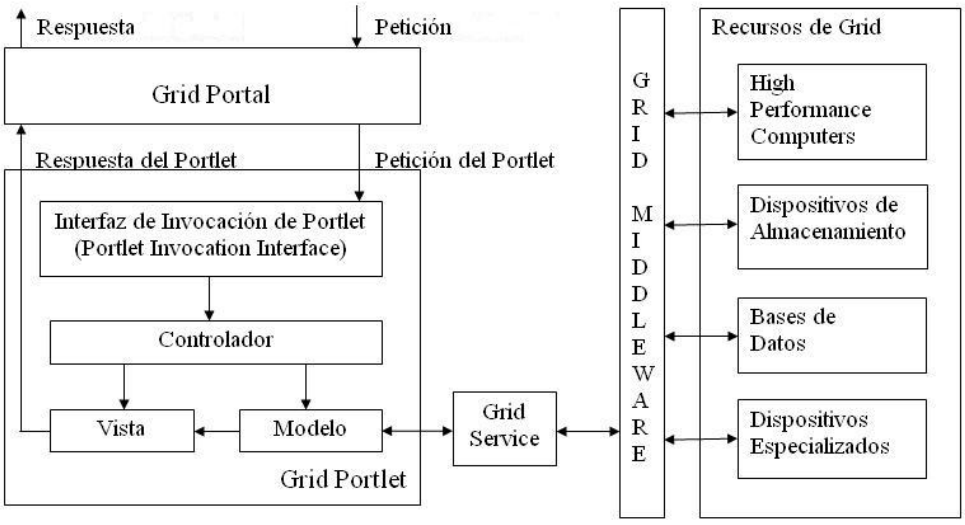


Figura 3.5: Grid Portlet

El Grid Portlet interactúa con un servicio de grid, este servicio es proporcionado por el middleware de grid para accesar a *fabric*, los recursos físicos de grid, (ver sección 2.1.3). Como los servicios grid son proporcionados por diferentes proveedores de servicio que pueden utilizar diferentes middlewares de grid, estos servicios pueden ser mostrados como portlets estándar, de esta forma los portales que utilizan portlets están desacoplados de los middlewares de grid.

3.2.2. Descripción de OGCE

OGCE fue creado con recursos de la *National Science Foundation Middleware Initiative* (NMI) de los EUA en 2003, con el fin de crear colaboraciones para los portales grid. Específicamente estableció un colaboración para portales grid que proporciona un foro en línea para desarrolladores de portales, un depósito de portlets y construyó un conjunto de componentes de portales reutilizables para ser integrados en un contenedor de un portal.

OGCE está vinculado con los siguientes proyectos:

- Java CoG
- CHEF Project
- GPIR y GridPort
- The Alliance Portal Expedition project.

Cabe mencionar que OGCE apoya tecnologicamente al portal de usuarios del Teragrid 2.2.3.

Específicamente, el portal OGCE está compuesto por el contenedor portlet de GridSphere y sus portlets cumplen con la especificación JSR 168.

La arquitectura de OGCE se muestra en la figura 3.6 :

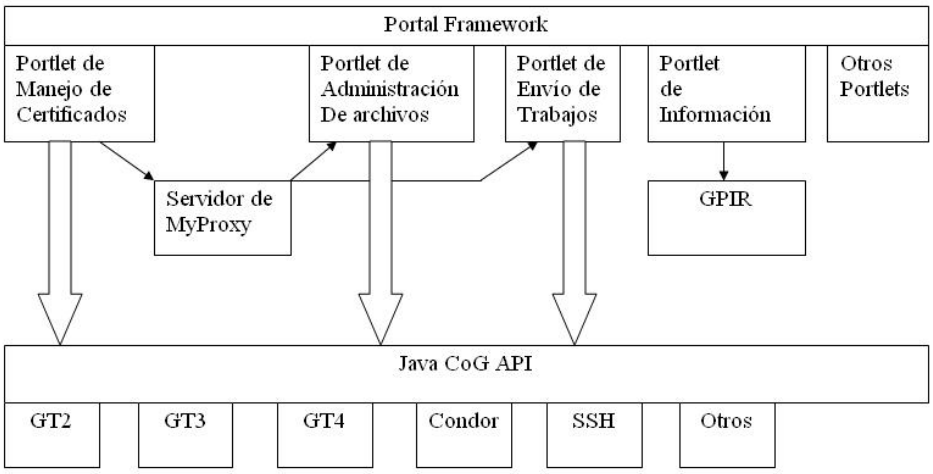


Figura 3.6: Open Grid Computing Environment

3.2.3. Portlet de Manejo de Certificados

Este portlet se encarga del servicio de autenticación de usuarios en un grid computacional. Un usuario dentro de un portal grid debe autenticarse ante un grid computacional mediante su certificado proxy. Terminada la autenticación, el usuario puede pedir al portal grid acceder a los recursos del grid en nombre del mismo usuario.

MyProxy

Un sistema que realiza parte fundamental de la autenticación de los usuarios de un grid computacional (ya sea que utilicen una interfaz web o no) es MyProxy. MyProxy es un sistema que administra certificados para los grids computacionales. MyProxy delega el certificado proxy de un usuario de grid a un Portal Grid, el portal a su vez utiliza este certificado para acceder a los recursos del mismo (grid computacional) en nombre del usuario. Una de las ventajas de MyProxy es que permite recuperar un certificado proxy independientemente de lugar donde se encuentre el usuario y en el momento que lo necesite gracias a su depósito (*repository*) de certificados. Otra característica importante de MyProxy es que al utilizarlo, se puede renovar los certificados proxy de un usuario automáticamente (sin la intervención del usuario) y evitar que trabajos que llevan largo tiempo ejecutándose sean interrumpidos.

Considerando que el certificado del usuario se encuentra almacenado en el servidor de

Myproxy y la creación de dicho certificado se realizó mediante el comando myproxy-admin-adduser, se muestra el proceso de autenticación utilizando Myproxy en la siguiente figura:

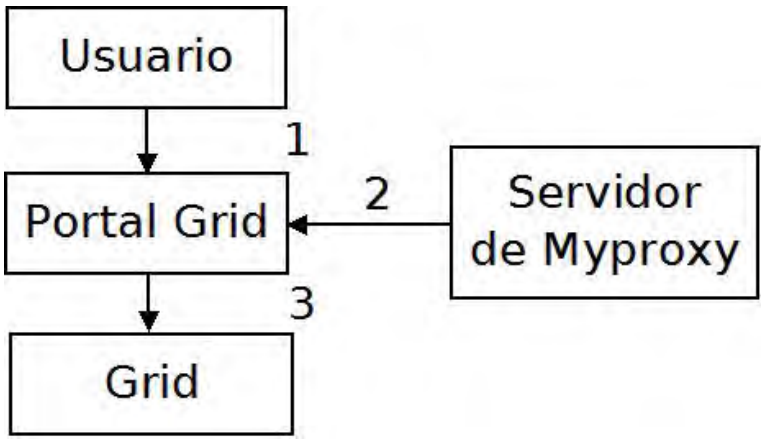


Figura 3.7: Servidor de Certificados

1. Se accede al portal Grid.

2. El portal grid solicitará la información necesaria para obtener el certificado proxy. En el caso del portal Grid OGCE esto se realiza mediante el portlet de MyProxy. En este paso el portal grid ejecutará el comando myproxy_get_delegation para recurperar el certificado proxy del servidor MyProxy utilizando el nombre de usuario y contraseña.

3. Una vez que el portal grid obtiene el certificado proxy del usuario, el portal accederá a los recursos del grid computacional con el certificado proxy a su nombre.

3.2.4. Portlet de Administración de Archivos

El Portlet de administración de archivos permite al usuario de un portal grid:

- Listar archivos en recursos remotos
- Cargar archivos a recursos remotos
- Transferir archivos entre recursos remotos

Este portlet puede mostrar los directorios uno al lado de otro de diferentes recursos de grid, esto permite al usuario listar los archivos de los diferentes recursos para su transferencia.

Específicamente este portlet proporciona funciones básicas del cliente GridFTP en una interfaz amigable. Las versiones de Globus toolkit que soporta este portlet son la 2.4, 3.2.1, 4.0.1 y 5.0 Este portlet realiza la admininstración de archivos directamente utilizando Java CoG.

3.2.5. Portlet de Envío de Jobs

Este portlet permite a un usuario enviar sus trabajos a un grid computacional utilizando el protocolo GRAM de Globus. Asimismo, en este portlet se pueden definir los parámetros para la ejecución de un trabajo, enviar a ejecución a un trabajo y ver el estado en que un trabajo se encuentra. Este portlet soporta las versiones de Globus 2.4, 3.2.1, 4 y 5.0. Para el envío de trabajos este portlet se comunica directamente con el middleware de grid utilizando el API de Java CoG.

Capítulo 4

Implementación y Evaluación

4.1. Descripción de la infraestructura

La figura muestra de forma esquemática los elementos que integran el ambiente de grid bajo el cual se instaló y configuró el portal OGCE versión 2.5:

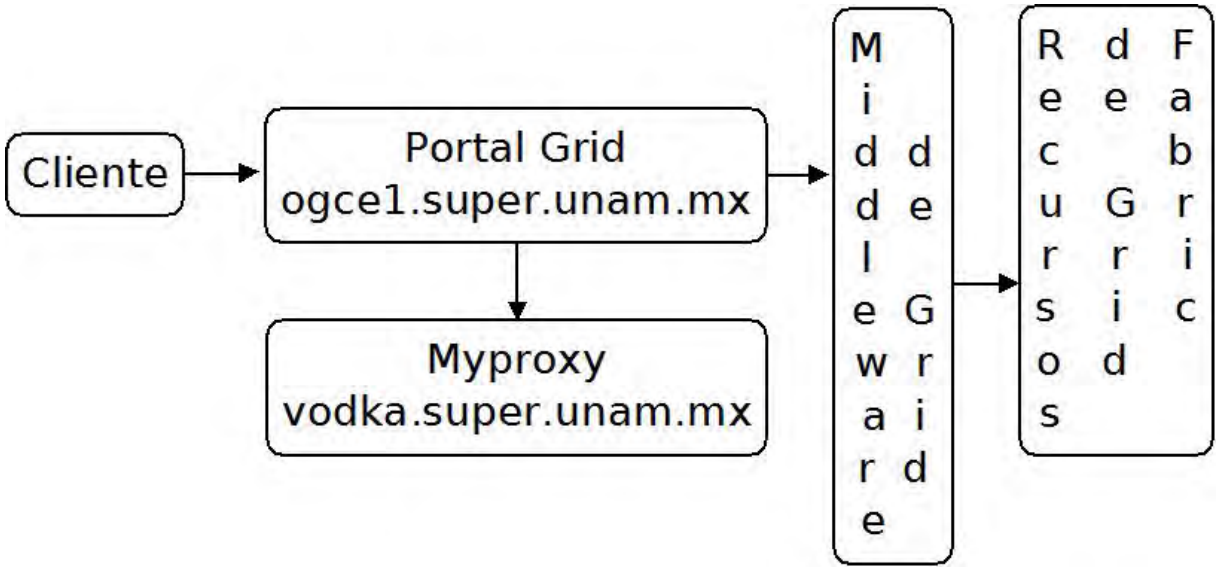


Figura 4.1: Elementos con los que interactúa un portal grid

El portal grid OGCE versión 2.5 se encuentra instalado en una máquina virtual del Departamento de Supercómputo `ogce1.super.unam.mx`, el sistema operativo de la

máquina virtual es Fedora 8, cuenta con un procesador Intel Xeon a 1.8 GHz y tiene de memoria RAM 512 MB.

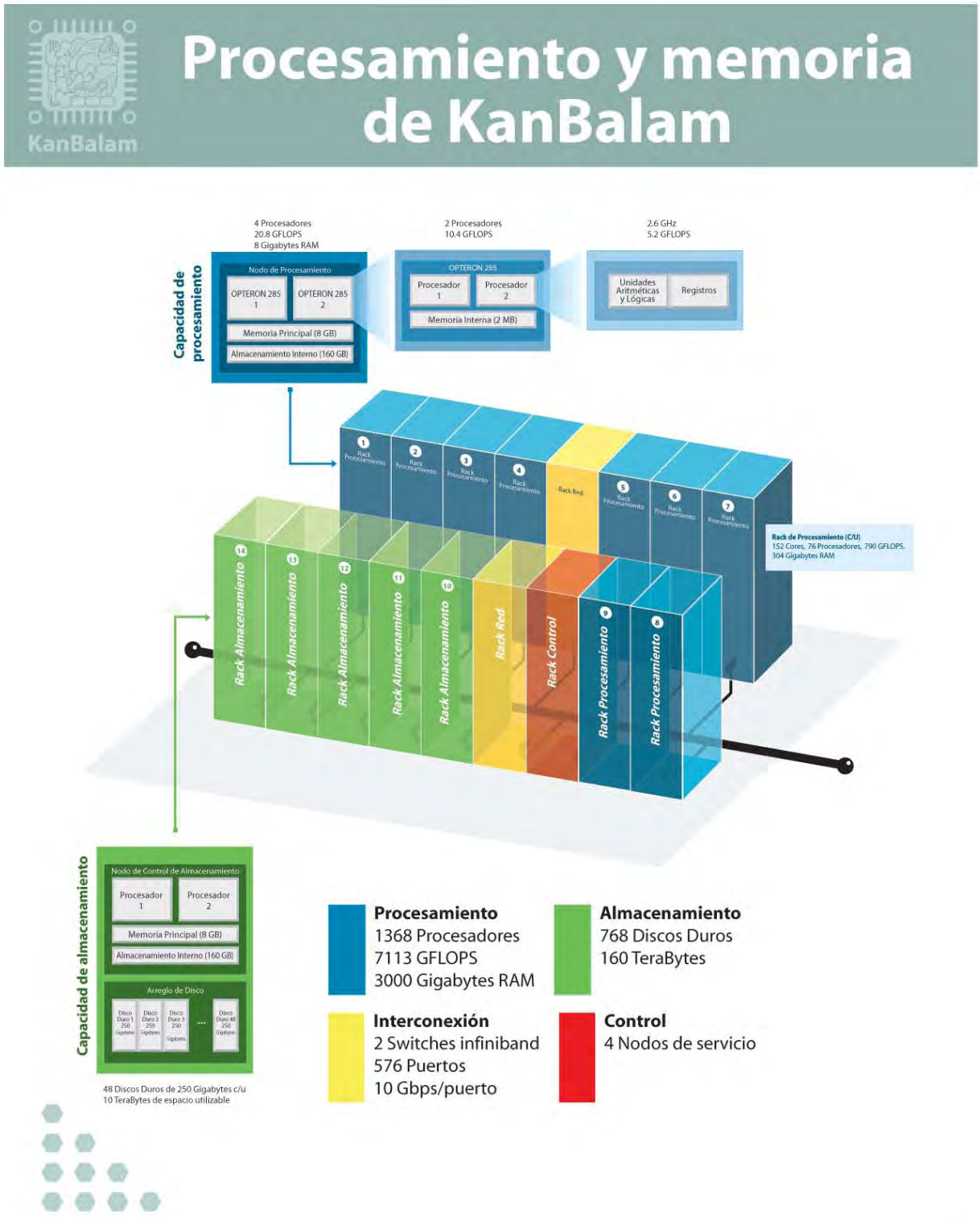
La parte del middleware de grid correspondiente al servidor de manejo de certificados Myproxy fue instalado en la máquina `vodka.super.unam.mx`, cuenta con sistema operativo Ubuntu 8, tiene un procesador Intel Pentium 4 a 2.26 GHz y tiene de memoria RAM 756 MB.

Los restantes partes del middleware de Grid se instalaron en un nodo de login de la Supercomputadora y la versión del middleware es GT5. Este nodo fue dado de alta en el DNS con el nombre de `balam-ln2.super.unam.mx` con el fin de poder acceder a este nodo mediante los servicios del middleware. Los cuales son:

- gridftp
- gram5

Los recursos de *fabric*, son los nodos de cálculo de la Supercomputadora Kanbalam. Los nodos de cálculo son 337 nodos de cálculo regulares, cada nodo con dos procesadores AMD Opteron Dual Core con 8 GB de RAM y 5 nodos de cálculo especiales con dos procesadores AMD Opteron Dual Core por nodo y 64 GB de RAM. El manejo de estos recursos se lleva a cabo mediante el software de administración de recursos LSF.

Figura 4.2: Arquitectura de un cluster LSF


KanBalam
Procesamiento y memoria de KanBalam
4 Procesadores
20.8 GFLOPS
8 Gigabytes RAM
2 Procesadores
10.4 GFLOPS
2.6 GHz
5.2 GFLOPS
Capacidad de procesamiento
Nodo de Procesamiento
OPTERON 285 1
OPTERON 285 2
Memoria Principal (8 GB)
Almacenamiento Interno (160 GB)
OPTERON 285
Procesador 1
Procesador 2
Memoria Interna (2 MB)
Unidades Aritméticas y Lógicas
Registros
Rack Procesamiento
Rack Red
Rack Control
Rack Almacenamiento
Rack de Procesamiento (C/U)
152 Cores, 76 Procesadores, 790 GFLOPS, 304 Gigabytes RAM
Capacidad de almacenamiento
Nodo de Control de Almacenamiento
Procesador 1
Procesador 2
Memoria Principal (8 GB)
Almacenamiento Interno (160 GB)
Arreglo de Disco
48 Discos Duros de 250 Gigabytes c/u
10 TeraBytes de espacio utilizable
Procesamiento
1368 Procesadores
7113 GFLOPS
3000 Gigabytes RAM
Almacenamiento
768 Discos Duros
160 TeraBytes
Interconexión
2 Switches infiniband
576 Puertos
10 Gbps/puerto
Control
4 Nodos de servicio

4.2. Configuración

4.2.1. Configuración del Servidor de MyProxy

A continuación se muestran los pasos para la instalación del servidor de Myproxy en la máquina `vodka.super.unam.mx`:

1. Como root, crear un usuario llamado globus

   % adduser globus

2. Crear un directorio, en el cual se instalará GT5.

   % mkdir /usr/local/globus-5.0/

3. Cambiar de dueño y de grupo al anterior directorio.

   % chown globus:globus /usr/local/globus-5.0/

4. Como globus, agregar la variable de ambiente GLOBUS_LOCATION

   % export GLOBUS_LOCATION=/usr/local/globus-5.0/

5. Configurar y compilar myproxy:

   % ./configure –prefix=$GLOBUS_LOCATION
   % make gsi-myproxy

6. Agregar el contenido de $GLOBUS_LOCATION/share/myproxy/etc.services en /etc/services

7. Copiar el archivo $GLOBUS_LOCATION/share/myproxy/etc.xinetd.myproxy a /etc/xinet.d/myproxy y modificar las variables de ambiente

   server = /usr/local/globus-5.0/sbin/myproxy-server env = GLOBUS_LOCATION=/usr /local/globus-5.0 LD_LIBRARY_PATH=/usr/local/globus-5.0/lib

8. Como root, reiniciar el demonio de xinetd

   % kill -HUP pid_de_xinetd

4.3. Configuración del Portal Grid

4.3.1. Instalación de Java

Para el portal Grid, es necesario contar con una versión de Java superior o igual a la 1.5.

En nuestra implementación se instaló Java 1.6.6 [22], debido a que no se encontró ningún paquete que configurará la última versión del Ambiente de Ejecución de de Java (*Java Runtime Environment*, JRE) de Sun y mantuviera al mismo tiempo los JREs que proporciona el Sistema Operativo.

1. Instalación del *Java Devolopment Kit*, JDK.

   a) Entrar a la página de Sun:
   http://java.sun.com/ o de http://www.oracle.com/technetwork/java/index.html

   b) Dar click en Java SE en la sección de Descargas de Software, *Software Downloads*

   c) Dar click en *Previous Releases*

   d) Dar click en *Archived Releases*

   e) De la sección *Java 2 Platform Standard Edition (J2SE)*, elegir de las opciones de *JDK/JRE - 6* la actualización 6 (*6 Update 6*) y dar click en *Go*.

   f) Dar click en Download JDK

   g) Seleccionar en Plataforma Linux, aceptar la licencia y dar click en continuar.

   h) Dar click en jdk-6u6-linux-i586-rpm.bin y guardarlo en Disco Duro.

   i) Agregar permisos de ejecución al archivo descargado.
   chmod +x jdk-6u6-linux-i586-rpm.bin

   j) Como root, ejecutar el archivo
   ./jdk-6u6-linux-i586-rpm.bin

2. El paso anterior instalará el JDK y el JRE pero sólo configurará adecuadamente al JDK; para configurar adecuadamente el JRE, es necesario descargar el paquete jpackage[23]:

   a) Entrar a la siguiente dirección:
   ftp://jpackage.hmdc.harvard.edu/JPackage/1.7/generic/RPMS.non-free/

b) Descargar el paquete compatible con la versión de Java instalada. En nuestro caso, se descargó la versión 1.6.6:
java-1.6.0-sun-compat-1.6.0.06-1jpp.i586.rpm

c) Instalar el paquete jpackage[24]
rpm -Uvh java-1.6.0-sun-compat-1.6.0.06-1jpp.i586.rpm

3. Configuración del JRE

   Esta configuración funciona en distribuciones de Linux basadas en RPMs (Mandrake, Red Hat, SuSE, etc).

   a) Configurar el JRE con el que deseamos trabajar, con el siguiente comando: alternatives –config java

```
[root@ogce3 ~]# alternatives --config java

There are 3 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
*+ 1           /usr/lib/jvm/jre-1.7.0-icedtea/bin/java
   2           /usr/lib/jvm/jre-1.5.0-gcj/bin/java
   3           /usr/lib/jvm/jre-1.6.0-sun/bin/java
```

   b) Seleccionar el número que corresponda a la versión 1.6. En nuestro caso, tecleamos 3.

4.3.2. Descarga del Portal OGCE

1. Entrar a la página de OGCE

   `http://www.collab-ogce.org/ogce/index.php/Main_Page`

2. Dar click en *OGCE Portal* en la sección de componentes descargables.

3. Ir a la sección *Portal Toolkit 2.5 Download* y descargar el archivo compactado y comprimido:

   ogce-portal-only-release-2.5.tar.gz

4.3.3. Instalación del Portal OGCE

1. Descomprimir y descompactar el archivo que contiene el portal en el directorio $HOME de una cuenta dedicada al servicio del portal exclusivamente, en nuestro caso la cuenta dedicada se llama irving.

   % tar -zxvf ogce-portal-only-2.5-release.tar.gz

2. Hacer una liga al directorio anteriormente descomprimido.

   % ln -s ogce-portal-only-2.5-release ogce-portal-only

3. Cambiarse al directorio ogce-portal-only y descompactar el archivo maven-2.0.7-bin.tar.gz.

   % tar -zxvf maven-2.0.7-bin.tar.gz

4. Agregar las siguientes variables de ambiente. Para hacer que esta configuración sea permanente en una cuenta de usuario dedidicada, agregar las siguientes líneas al archivo $HOME/.bashrc

   a) export JAVA_HOME=/usr/lib/jvm/jre-1.6.0-sun

   b) PATH=$HOME/ogce-portal-only/maven-2.0.7/bin:$PATH

   La primer línea actualiza la variable de ambiente de Java.

   La siguiente línea agrega al PATH los programas que utiliza el administrador de proyectos Apache Maven.

5. Editar las propiedades (*properties*) del archivo pom.xml

   a) Cambiar <portal.server.ip/> por la dirección IP en que se va a encontrar el portal grid.
   <portal.server.ip>132.248.202.135</portal.server.ip>

6. Para instalar el portal, ejecutar el siguiente comando:

   mvn clean install

7. Finalizado el proceso de compilación del portal, iniciar el servidor Apache Tomcat mediante el comando:

   ./startup-tomcat.sh

8. Accesar a la página web del portal:

   http://132.248.202.135:8080/gridsphere

9. Y crear la cuenta de administrador del portal introduciendo la información solicitada por el portal como nombre de usuario, correo electrónico, contraseña, etc.

4.3.4. Configuración del Firewall

En caso de que el portal esté detrás de un firewall, realizar el siguiente procedimiento:

1. Como root, editar el archivo

   /etc/sysconfig/iptables

2. Agregar la siguiente línea, antes de cualquier otra que contenga un *REJECT*

   -A RH-Firewall-1-INPUT -m state –state NEW -p tcp -m tcp –dport 8080 -j ACCEPT

3. Actualizar la nueva regla del Firewall

   /etc/init.d/iptables restart

4.3.5. Configuración del Portlet de Administración de Archivos

1. Estando en el directorio donde se instaló el portal "ogce-portal-only", parar el servidor Tomcat:

   ./shutdown-tomcat.sh

2. Editar el archivo "pom.xml"

   a) Agregar los hosts "balam-ln2.supercomputo.unam.mx,vodka.super.unam.mx" entre las etiquetas
   <gridftp.host.names> y </gridftp.host.names>

3. Reconstruir el portlet de Administración de Archivos, *File Manager*, ejecutando el siguiente comando:

   mvn clean install -f portlets/file-manager/pom.xml

4. Reiniciar el servidor Apache Tomcat:

   ./startup-tomcat.sh

4.3.6. Configuración del Portlet de Envío de Jobs

La configuración del Portlet de Envío de Jobs, gp−job−submission es similar a los pasos 1 y 4 de la sección 4.3.5

1. Editar el archivo "pom.xml"

   a) Agregar el nuevo host "balam-ln2.supercomputo.unam.mx,vodka.super.unam.mx" entre las etiquetas
   <gram.host.names> y </gram.host.names>

2. Reconstruir el portlet, ejecutando el siguiente comando:

   mvn clean install -f portlets/gp-job-submission/pom.xml

Debido a que el portlet de Envío de Jobs no presentaba ciertas características del lenguaje RSL del middleware y que son particulamente útiles en el uso de la Supercomputadora, como los parámetros de Queue y JobType fue necesario modificar parte del código fuente de este portlet:

A continuación se presentan las modificaciones realizadas:

En el archivo submit.jsp del directorio $HOME/ogce-portal-only/portlets/gp-job-submission/src/main/webapp/jsp se agregó lo siguiente:

```
<%-- JOBTYPE --%>
<label for="jobtype">JobType</label>
<input id="jobtype" name="jobtype" type="input"
           class="portlet-form-input-field gp-form-input-field"/>
<br/>

<%-- QUEUE --%>
<label for="queue">Queue</label>
<input id="queue" name="queue" type="input"
           class="portlet-form-input-field gp-form-input-field"/>
<br/>
```

En la clase GRAMJobSubmissionPortlet.java del directorio $HOME/ogce-portal-only/ portlets/gp-job-submission/src/main/java/edu/tacc/gridport/portlets/interactive se añadió lo siguiente:

```
jbean.setJobtype(request.getPreferences().getValue(handle + ".jobtype", null));
jbean.setJobtype(request.getPreferences().getValue(handle + ".queue", null));

req.getPreferences().setValue(handle + ".jobtype", currentJobBean.getJobtype());
req.getPreferences().setValue(handle + ".queue", currentJobBean.getQueue());
```

En la clase JobBean.java del directorio $HOME/ogce-portal-only/portlets/gp-job-submission/ src/main/java/edu/tacc/gridport/portlets/interactive/beans se agregó:

```
     private String jobtype;
     private String queue;
       /**
*          * Get the jobtype associated with the job
*                   *
*                            * @return Returns the jobtype.
*                                     */
       public String getJobtype() {
               return jobtype;
       }
       /**
*          * Set jobtype to be associated with the job
*                   *
*                            * @param jobtype  The jobtype to set.
*                                     */
       public void setJobtype(String jobtype) {
               this.jobtype = jobtype;
       }
       /**
*  *          * Get the queue associated with the job
*   *                   *
*    *                            * @return Returns the queue.
*     *                                     */
       public String getQueue() {
               return queue;
```

```
       }
       /**
*  *          * Set queue to be associated with the job
*   *                   *
*    *                             * @param queue  The queue to set.
*     *                                        */
       public void setQueue(String queue) {
               this.queue = queue;
       }
```

En la clase GRAMJobManager.java del directorio $HOME/ogce-portal-only/portlets/gp-job-submission/src/main/java/edu/tacc/gridport/portlets/interactive/helpers se agregó lo siguiente:

```
String jobtype = req.getParameter("jobtype");
  newJob.setJobtype(jobtype);
String queue = req.getParameter("queue");
  newJob.setQueue(queue);
if (job.getJobtype() != null && !(job.getJobtype()).equals("")) {
        rslSb.append("(jobType=\"" + job.getJobtype() + "\")");
}
if (job.getQueue() != null && !(job.getQueue()).equals("")) {
        rslSb.append("(queue=\"" + job.getQueue() + "\")");
}
  System.out.println("El rsl es: " + rsl);
```

Adicionalmente este portlet generaba un RSL incorrecto porque una vez dados los archivos de error y salida mediante el portal grid, el RSL agregaba otras líneas referentes a los mismos archivos. Este comportamiento no permitía almacenar nada en los archivos de error y de salida. Por lo anterior fue necesario comentar algunas líneas de la clase GRAMJobPreWS.java ubicada en el mismo directorio de la clase (GRAMJobManager.java) anterior:

```
//newRSL.append("(stdout=x-gass-cache://$(GLOBUS_GRAM_JOB_CONTACT)
stdout anExtraTag)");
//newRSL.append("(stderr=x-gass-cache://$(GLOBUS_GRAM_JOB_CONTACT)
stderr anExtraTag)");
```

Finalmente, se recompiló este portlet como se mostró en 4.3.6

4.3.7. Configuración de la CA

Para configurar la CA a utilizar, se deben solicitar a la CA: su certificado y sus políticas de firmado (*CA Signing policy*).

Para la configuración se deben seguir los siguientes pasos:

1. Copiar la información al directorio $HOME/.globus/certificates de la cuenta dedicada a la implementación del portal.

   % cp vodka.ca.tar.gz $HOME/.globus/certificates

2. Descomprimir la información

   % cd $HOME/.globus/certificates

   % tar -zxvf vodka.ca.tar.gz

4.3.8. Creación de cuentas

1. Con cuenta de adminstrador del portal, entrar al portlet de administración (*Administration*).
2. Ingresar a la sección de usuarios (*Users*).
3. Y dar click en Crear nuevo usuario (*Create new user*)
4. Llenar los campos solicitados: nombre de usuario, nombre, correo electrónico, organización, tipo de cuenta, contraseña en el portal.

Figura 4.3: Administración de cuentas

4.3.9. Configuración de cuentas de usuario

Una vez conectado al portal grid, aparece el portlet de bienvenida, en este portlet el usuario del portal podra configurar su información personal, cambiar su contraseña y seleccionar los portlets a utilizar. Para seleccionar los portlets, ir a la sección de configurar membresía de grupo (*Configure group membership*) y marcar (dar un click) los siguientes portlets:

- file−manager
- gp−job−submission
- proxymanager−portlet

Y finalmente dar de alta los cambios, dar click en el botón de *Save*

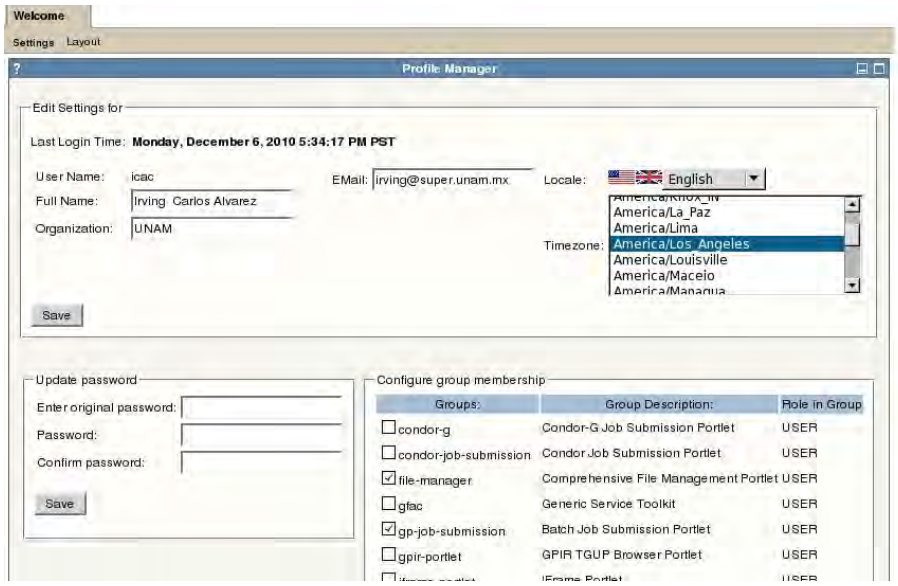


Figura 4.4: Configuración de cuenta de usuario

4.3.10. Configuración del Portlet de Manejo de Certificados Proxy

1. Entrar al portlet de Servidor de Manejo de Certificados Proxy, *ProxyManager-Portlet*
2. Dar click en el botón de *Edit* que se encuentra en la parte superior izquierda del portlet.
3. Cambiar el valor del Hostname por "vodka.super.unam.mx"
4. Poner el nombre de usuario en el servidor de Manejo de Certificados y dar el tiempo de vida del proxy o aceptar el default que es de 2 horas.
5. Por último actualizar estos parámetros en el portal, dar click en *Customize*.

Figura 4.5: Configuración del Servidor de certificados

4.4. Pruebas de funcionamiento

4.4.1. Ingreso al Portal

Para ingresar al Portal Grid OGCE ingrese su nombre de usuario y contraseña.

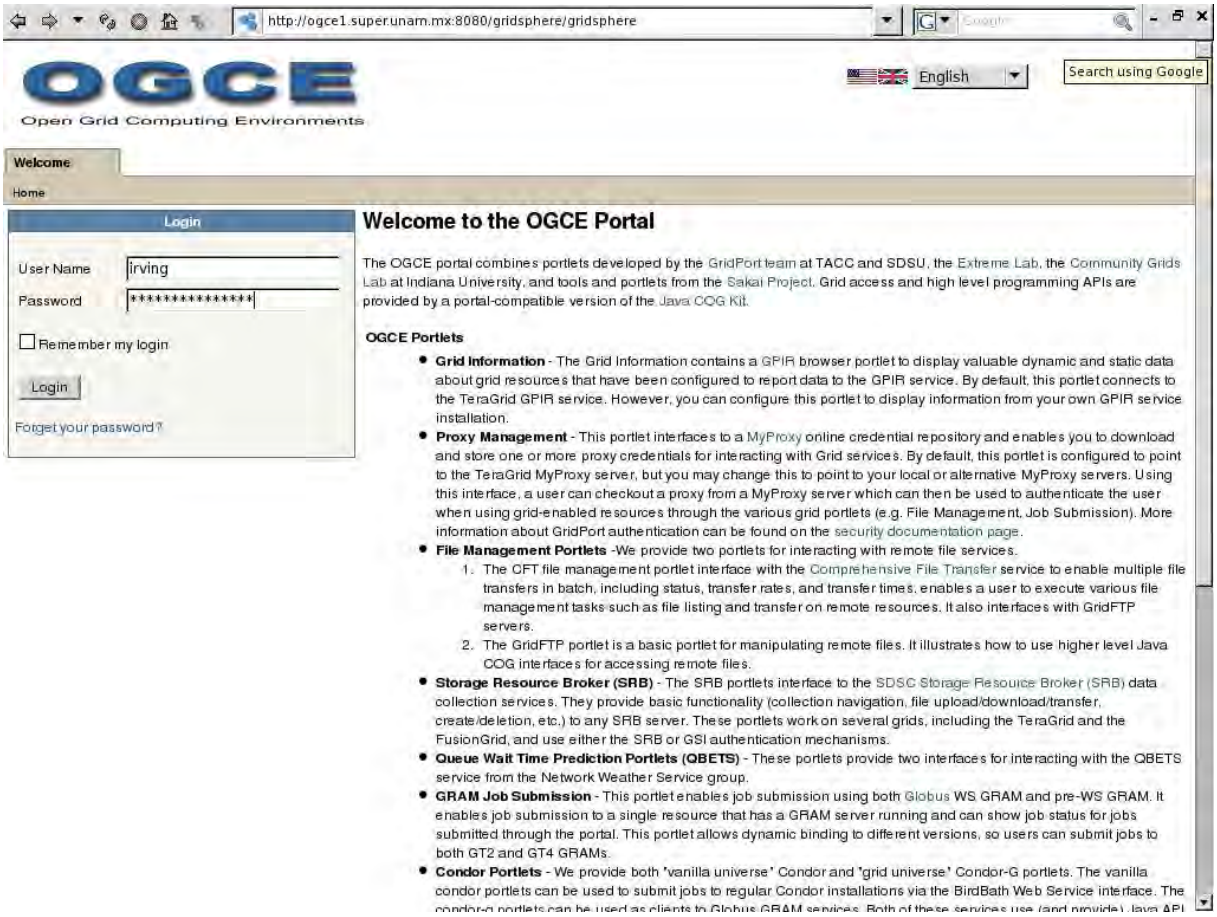


Figura 4.6: Pantalla de inicio del portal grid

4.4.2. Uso del Portlet Proxy−Manager

Seleccionar el portlet ProxyManager y dar click en el botón *Get new Proxy*

Figura 4.7: Portlet del servidor de certificados

Proporcionar la contraseña del certificado proxy y dar un click en el botón de *Get Proxy From MyProxy.*

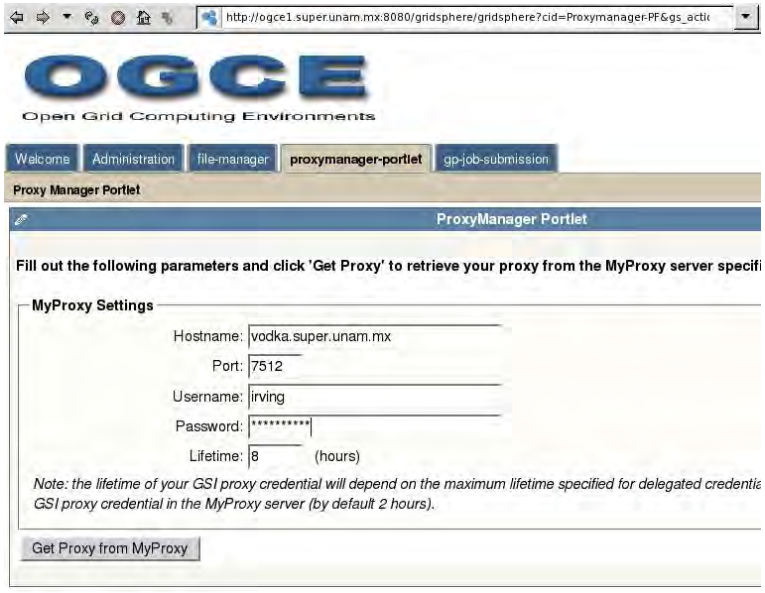


Figura 4.8: Solicitud de un certificado proxy

Lo anterior, mostrará que se obtuvo un certificado proxy del usuario del portal grid.

Figura 4.9: Obtención del certificado proxy vía el portal

Para ver el contenido del certificado proxy, dar click en *view*

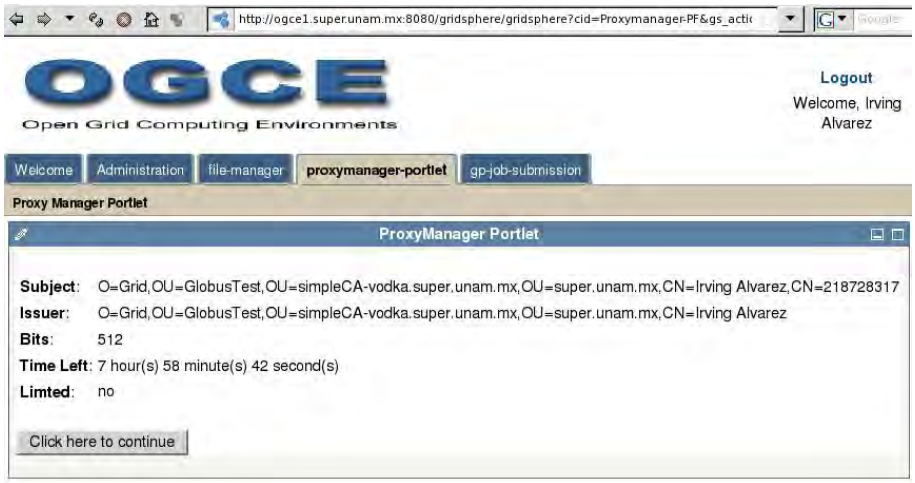


Figura 4.10: Información del certificado proxy

4.4.3. Uso del Portlet File−Manager

Seleccionar el Portlet File−Manager, este portlet tiene dos secciones una de listado de archivos de origen y otra de destino.

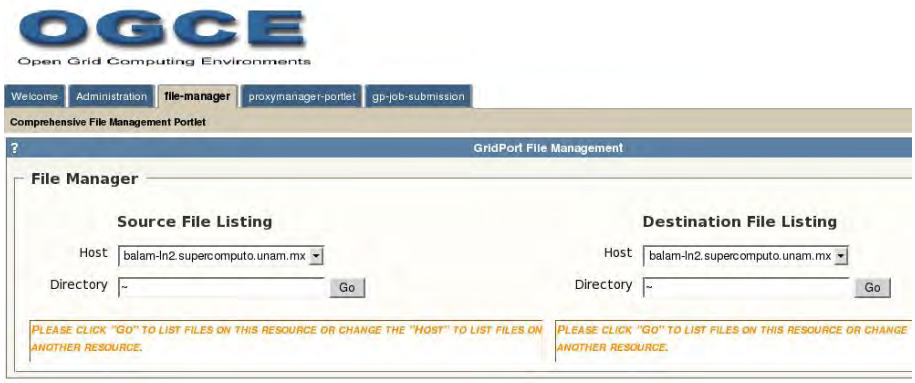


Figura 4.11: Portlet de manejo de archivos

Seleccionar en el listado de archivos de origen el host "balam-ln2.supercomputo.unam.mx" y despues dar click en *Go* para desplegar los archivos.

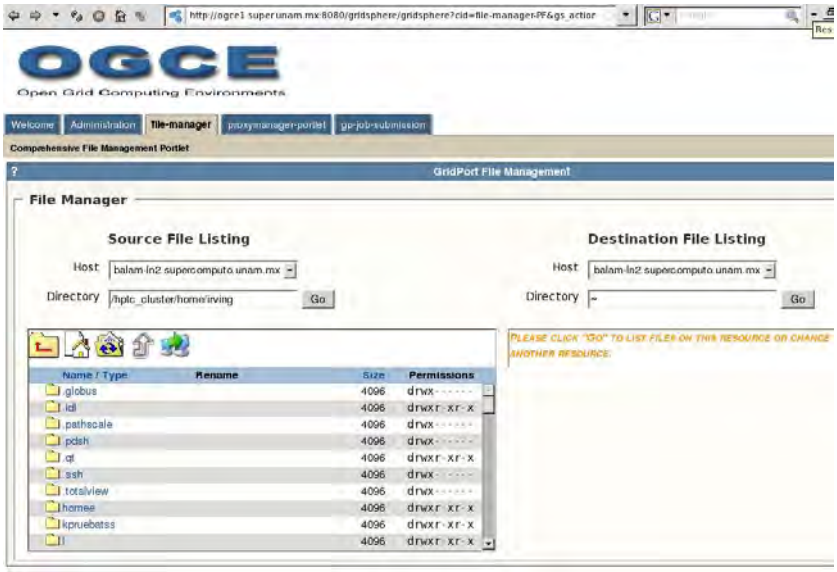


Figura 4.12: Listado de archivos de una máquina origen

Las acciones que se pueden realizar en ambas secciones son las siguientes:

- Ir un directorio atrás
- Ir al directorio $HOME del usuario
- Actualizar el listado de archivos
- Subir archivos a un host origen o destino desde la máquina local del usuario.

Adicionalmente el host origen, puede transferir archivos hacia el host destino.

Dar click en el botón de Subir archivo (*Upload file*), seleccionar un archivo desde la máquina local y subir el archivo, dar click en Upload.

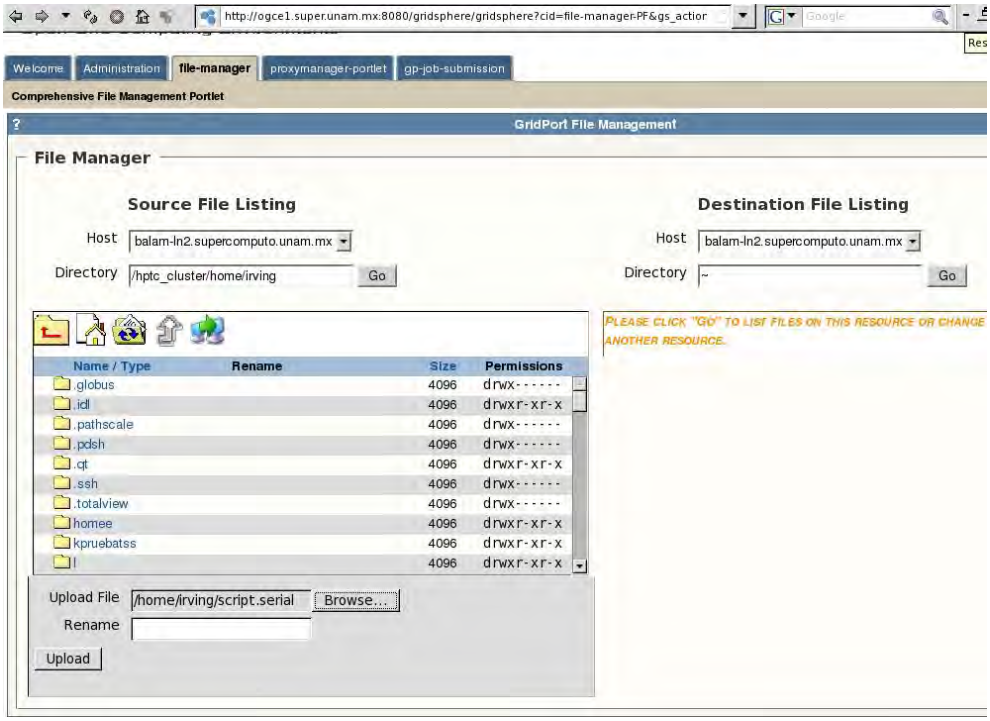


Figura 4.13: Transferencia de archivo desde una máquina local

Figura 4.14: Archivo en la máquina origen

Seleccionar en el listado de archivos destino el host "vodka.super.unam.mx" y después dar click en *Go* para desplegar los archivos.

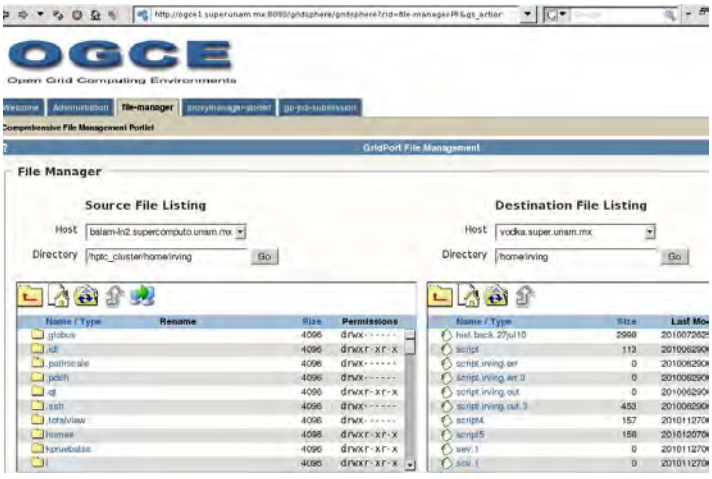


Figura 4.15: Listado de archivos en máquina destino

Seleccionar el archivo "script.serial" en el listado de archivos origen y dar click en el botón Transferir archvios (*transfer files*).

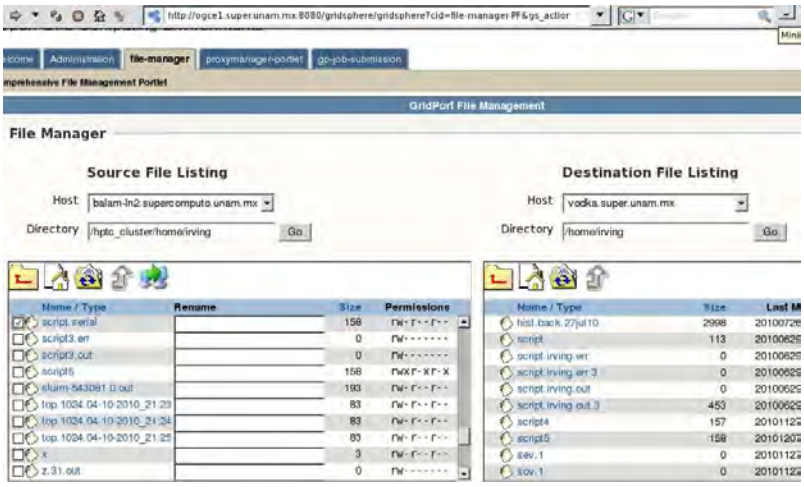


Figura 4.16: Transferencia de archivo desde máqina origen a máquina destino

A continuación se observa el archivo transferido en el host destino enviado de un host origen.

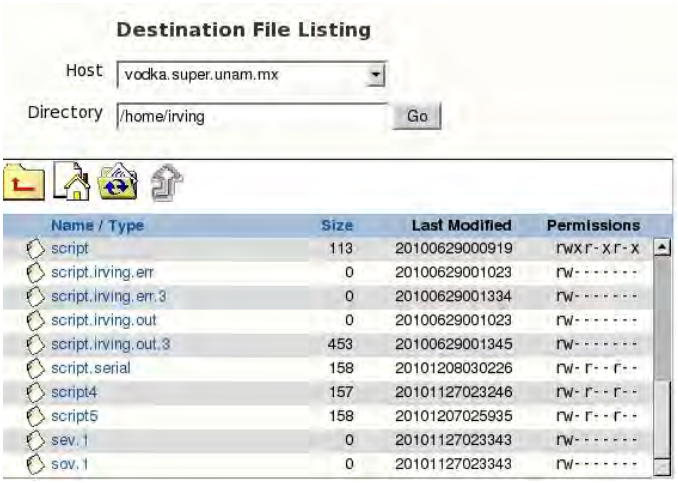


Figura 4.17: Archivo en máquina destino

4.4.4. Uso de Portlet Gp-Job-Submission

Para poder hacer uso de este portlet, dar click en el portlet gp-job-submission. Para mostrar el funcionamiento de este portlet y su interacción con el equipo de supercómputo, ejecutaremos el siguiente programa:

```
#!/bin/sh
/bin/pwd
/usr/bin/whoami
/bin/hostname
/bin/date
if [ $# -eq 2 ]
then
  echo ‘‘/bin/sleep $1’’
  /bin/sleep $1
  echo ‘‘wake up’’
  /bin/date
  for ((i=1; i<=$2; i++))
  do
        echo $i
  done
else
  echo ‘‘Para ejectar este script introduzca dos numeros’’
fi
```

Para ejecutar el programa, en el portlet de gp-job-submission, seleccionar el host `balam-ln2.supercomputo.unam.mx`, completar los parámetros necesarios y solicitar la ejecución del programa dando click en el botón *Submit*, como se muestra a continuación:

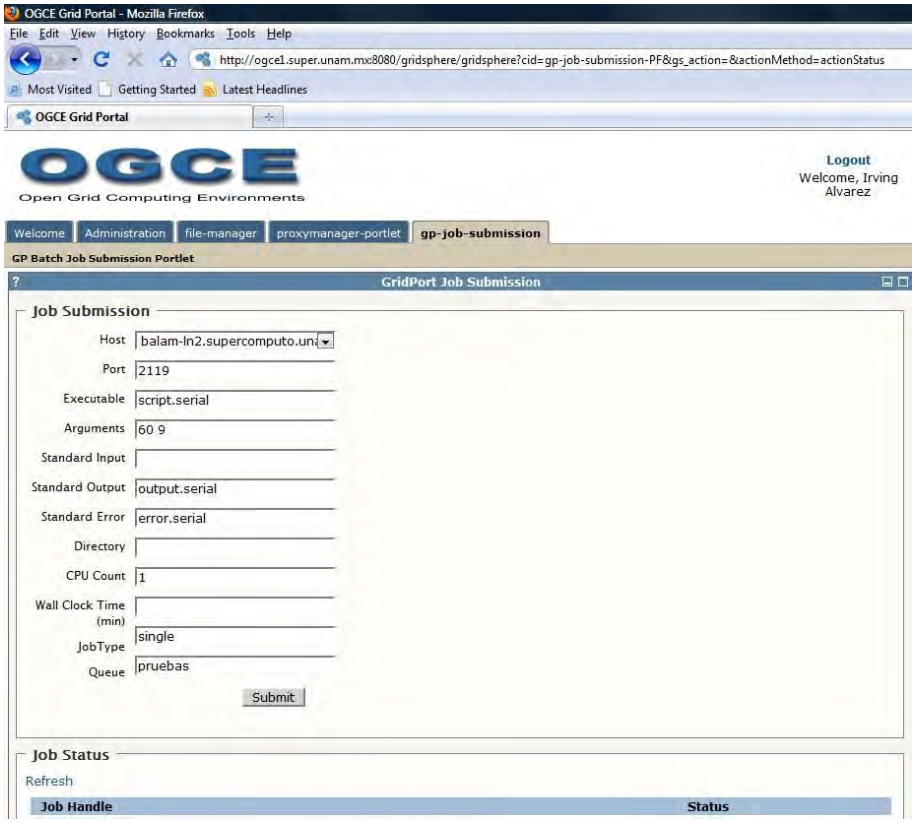


Figura 4.18: Portlet de Envío de Jobs

Una vez solicitada la ejecución del programa, el portal genera un RSL (*Resource Standar Language*), el cual contiene los parámetros de ejecución del programa en lenguaje del middleware de grid para su ejecución. A continuación se muestra el RSL generado por el portal:

```
El rsl es: &(executable=script.serial)(arguments=60 9)(count=1)
(stdout="output.serial")(stderr="error.serial")(jobType="single")
(queue="pruebas")
```

Este RSL es utilizado por el servidor GRAM para solicitar recursos de cálculo a la supercomputadora, esta petición se realiza al Sistema de Calendarización de jobs y Administración de Recursos LSF.

El sistema LSF cuando dispone de los recursos de cómputo, despacha los recursos para la ejecución del programa. Esta asignación de recursos y las propiedades del job se puede verificar vía consola, como se aprecian a continuación:

```
[irving@n340 ~]$ bjobs  -l

Job <113451>, User <irving>, Project <default>, Status <RUN>, Queue <pruebas>,
                     Command <#! /bin/sh;#;# LSF batch job script built by Glob
                     us Job Manager;#;#BSUB -q pruebas;#BSUB -i /dev/null;#BSUB
                     -o /hptc_cluster/home/irving/output.serial;#BSUB -e /hptc_
                     cluster/home/irving/error.serial;#BSUB -n 1;X509_USER_PROX
                     Y=''/hptc>
Thu Jan 13 08:01:28: Submitted from host <n339>, CWD <$HOME>, Input File </dev/
                     null>, Output File </hptc_cluster/home/irving/output.seria
                     l>, Error File </hptc_cluster/home/irving/error.serial>, R
                     equested Resources <type=any>;
Thu Jan 13 08:01:32: Started on <lsfhost.localdomain>, Execution Home </hptc_cl
                     uster/home/irving>, Execution CWD </hptc_cluster/home/irvi
                     ng>;
Thu Jan 13 08:01:32: slurm_id=548513;ncpus=1;slurm_alloc=n314;

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

[irving@n340 ~]$
```

Para monitorear el estado del job mediante el portal grid, este portlet muestra en su parte inferior los jobs enviados a ejecución, su estado y las acciones que se pueden realizar a los mismos. Para ver el estado actual del job, dar click en en *refresh*. A continuación se muestran los estados por los que pasa el job:

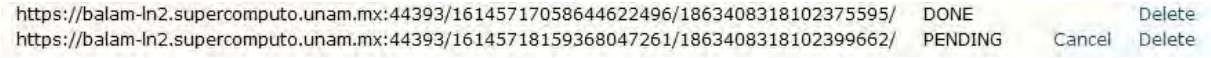


Figura 4.19: Estado pendiente del job

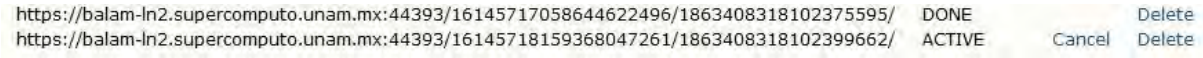


Figura 4.20: Estado activo del job

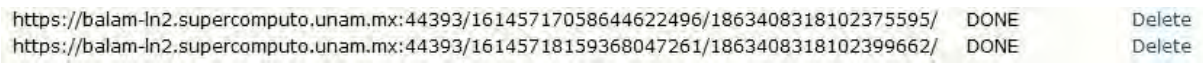


Figura 4.21: Estado finalizado del job

La salida del job es la siguiente:

```
Sender: LSF System <lsfadmin@lsfhost.localdomain>
Subject: Job 113451: <#! /bin/sh;#;# LSF batch job script built by Globus
Job Manager;#;
```

#BSUB -q pruebas;#BSUB -i /dev/null;
#BSUB -o /hptc_cluster/home/irving/output.serial;
#BSUB -e /hptc_cluster/home/irving/error.serial;
#BSUB -n 1;

```
X509_USER_PROXY=''/hptc> Done

Job <#! /bin/sh;#;# LSF batch job script built by Globus Job Manager;#;
#BSUB -q pruebas;#BSUB -i /dev/null;#BSUB -o /hptc_cluster/home/irving/
output.serial;#BSUB -e /hptc_cluster/home/irving/error.serial;#BSUB -n 1;
X509_USER_PROXY=''/hptc> was submitted from host <n339> by user <irving>.
Job was executed on host(s) <lsfhost.localdomain>, in queue <pruebas>,
as user <irving>.</hptc_cluster/home/irving> was used as the home directory.
</hptc_cluster/home/irving> was used as the working directory.
Started at Thu Jan 13 08:01:32 2011
Results reported at Thu Jan 13 08:02:47 2011

Your job looked like:

------------------------------------------------------------
# LSBATCH: User input
#! /bin/sh
```

```
#
# LSF batch job script built by Globus Job Manager
#

#BSUB -q pruebas
#BSUB -i /dev/null
#BSUB -o /hptc_cluster/home/irving/output.serial
#BSUB -e /hptc_cluster/home/irving/error.serial
#BSUB -n 1


X509_USER_PROXY=''/hptc_cluster/home/irving/.globus/job/balam-ln2.supercomputo.
unam.mx/16145718159368047261.1863408318102399662/x509_user_proxy''; export X50
9_USER_PROXY
GLOBUS_LOCATION=''/global/opt/appl/globus-5.0''; export GLOBUS_LOCATION
GLOBUS_GRAM_JOB_CONTACT=''https://balam-ln2.supercomputo.unam.mx:44393/16145718
159368047261/1863408318102399662/''; export GLOBUS_GRAM_JOB_CONTACT
HOME=''/hptc_cluster/home/irving''; export HOME
LOGNAME=''irving''; export LOGNAME
GLOBUS_GASS_CACHE_DEFAULT=''/hptc_cluster/home/irving/.globus/.gass_cache'';
export GLOBUS_GASS_CACHE_DEFAULT
export PATH=/opt/appl/scripts-grid:$PATH

#Change to directory requested by user
cd /hptc_cluster/home/irving
/hptc_cluster/home/irving/script.serial ``60'' ``9''

------------------------------------------------------------

Successfully completed.

Resource usage summary:

    CPU time   :      0.12 sec.
    Max Memory :         5 MB
    Max Swap   :        31 MB


The output (if any) follows:
```

```
/hptc_cluster/home/irving
irving
n314
Thu Jan 13 08:01:32 CST 2011
/bin/sleep 60
wake up
Thu Jan 13 08:02:32 CST 2011
1
2
3
4
5
6
7
8
9


PS:

Read file </hptc_cluster/home/irving/error.serial> for stderr output of this job.



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          ESTADISTICAS DEL JOB

 Duracion:                      1.25 minutos
 Recursos consumidos:           0.02 horas-cpu
 CPU Utilizado:                 0.00 minutos
 Eficiencia de uso de CPU:      0.13 %
```

4.5. Pruebas de aplicaciones

A continuación se mostrará como utilizar varias aplicaciones mediante el uso del portal grid. Para hacer uso de estas aplicaciones fue necesario modificar un archivo de configuración de Globus Toolkit y crear scripts que carguen el ambiente que utilizan las aplicaciones, esto fue debido a que el middleware no proporciona un ambiente para cada aplicación o usuario específico.

4.5.1. Gaussian

Para el uso de la aplicación de gaussian, se hizo el programa o script "script.gaussian", el fin de este script es cargar el ambiente que necesita la aplicación Gaussian para su correcto funcionamiento. El contendido del script es el siguiente:

```
#!/bin/bash

export TMPU="/global/$LOGNAME"
export GAUSS_SCRDIR=$TMPU

export GAUSS_EXEDIR="/opt/appl/g09/bsd:/opt/appl/g09/private:/opt/appl/g09"
export GAUSS_ARCHDIR="/opt/appl/g09/arch"
export GMAIN="/opt/appl/g09/bsd:/opt/appl/g09/private:/opt/appl/g09"
export G03BASIS="/opt/appl/g09/basis"
export F_ERROPT1="271,271,2,1,2,2,2,2"
export PATH="/opt/appl/g09/bsd:/opt/appl/g09/private:/opt/appl/g09:$PATH"

g09 < "$@"
```

Y se modificó el archivo $GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/lsf.pm:

```
if (-f '/opt/appl/scripts-grid/' . $description->executable) {
 $description->add('executable',
 '/opt/appl/scripts-grid/' .  $description->executable);
  }
```

Esta modificación también fue necesaria para Nwchem (ver sección 4.5.2).

Para solicitar la ejecución de la aplicación se colocan los siguientes parámetros en el portal grid:

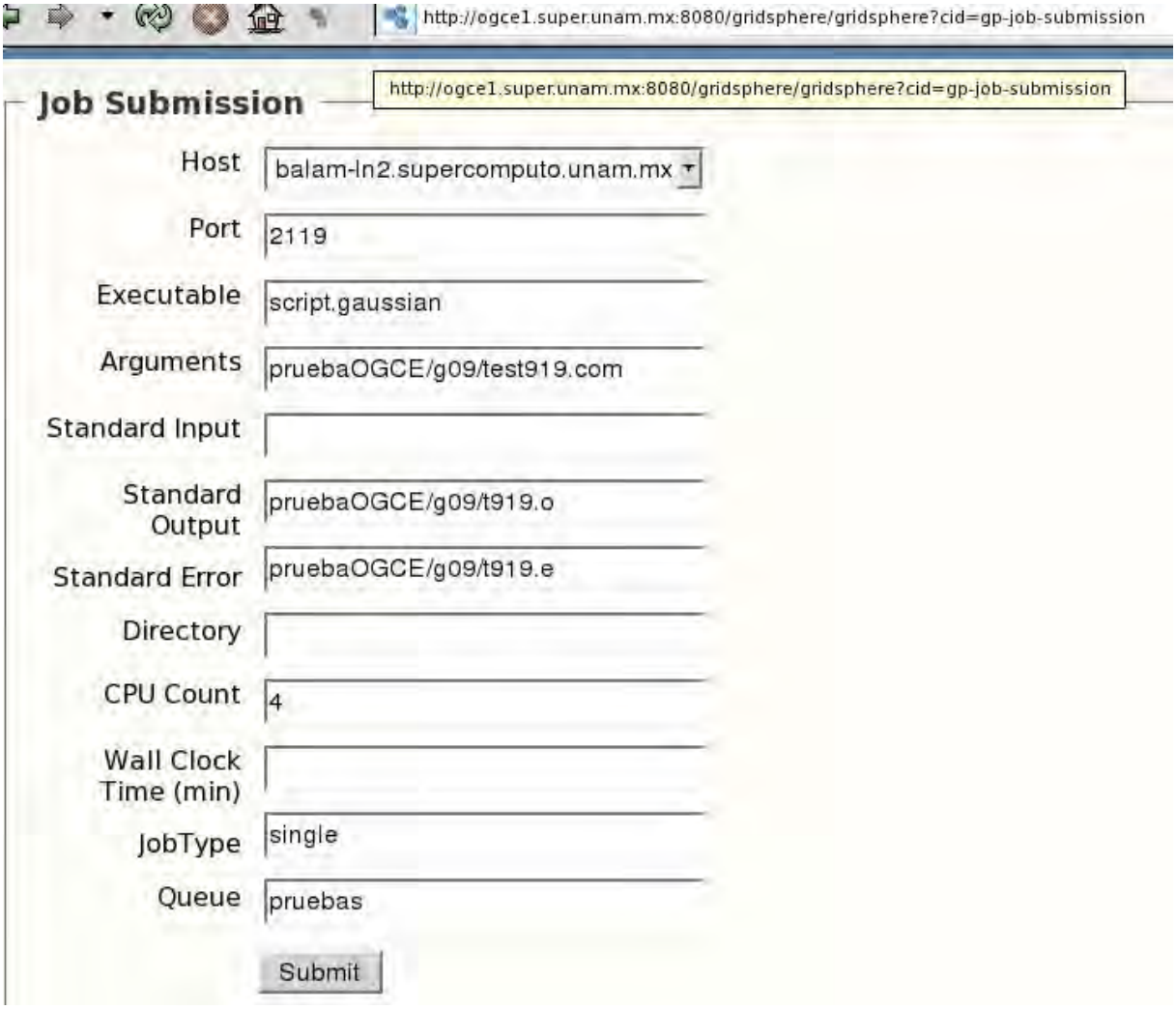


Figura 4.22: Parámetros para la ejecución de un job de Gaussian

- En el campo *Executable* se proporciona el nombre del programa que carga el ambiente, en nuestro caso script.gaussian.
- En el campo *Arguments* se proporciona el archivo de entrada de gaussian.
- El campo *Standard output* y *Standar error* corresponden a los archivos de salida y de error estándar respectivamente incluyendo el directorio donde se colocarán.

- En el campo *CPU count* sólo se pueden proporcionar hasta 4 procesadores debido al tipo de licencia que se cuenta de esta aplicación.
- En el campo *Job type* se proporciona single.
- En el campo *Queue* se proporciona la cola en que se ejecutara la aplicación.

Y finalmente se da click en *Submit* para enviar el job.

4.5.2. Nwchem

El programa que carga el ambiente de nwchem y ejecuta a nwchem es el programa script.nwchem, el contenido de este programa es el siguiente:

```
#!/bin/bash

export PATH=/opt/hptc/bin:/opt/appl/bin:$PATH

/opt/hpmpi/bin/mpirun -srun nwchem "$@"
```

El archivo de entrada "input2.in" de esta aplicación es el siguiente:

```
title "formaldehyde ECP deck"
start ecpchho
geometry units au
  C         0.000000  0.000000 -1.025176
  O         0.000000  0.000000  1.280289
  H         0.000000  1.767475 -2.045628
  H         0.000000 -1.767475 -2.045628
end
basis
  C  SP
   0.1675097360D+02 -0.7812840500D-01  0.3088908800D-01
   0.2888377460D+01 -0.3741108860D+00  0.2645728130D+00
   0.6904575040D+00  0.1229059640D+01  0.8225024920D+00
  C  SP
   0.1813976910D+00  0.1000000000D+01  0.1000000000D+01
  C  D
```

```
   0.8000000000D+00  0.1000000000D+01
  C  F
   0.1000000000D+01  0.1000000000D+01
  O  SP
   0.1842936330D+02 -0.1218775590D+00  0.5975796600D-01
   0.4047420810D+01 -0.1962142380D+00  0.3267825930D+00
   0.1093836980D+01  0.1156987900D+01  0.7484058930D+00
  O  SP
   0.2906290230D+00  0.1000000000D+01  0.1000000000D+01
  O  D
   0.8000000000D+00  0.1000000000D+01
  O  F
   0.1100000000D+01  0.1000000000D+01
  H  S
   0.1873113696D+02  0.3349460434D-01
   0.2825394365D+01  0.2347269535D+00
   0.6401216923D+00  0.8137573262D+00
  H  S
   0.1612777588D+00  0.1000000000D+01
end
ecp
  C nelec 2
  C ul
        1       80.0000000       -1.60000000
        1       30.0000000       -0.40000000
        2        0.5498205       -0.03990210
  C s
        0        0.7374760        0.63810832
        0      135.2354832       11.00916230
        2        8.5605569       20.13797020
  C p
        2       10.6863587       -3.24684280
        2       23.4979897        0.78505765
  O nelec 2
  O ul
        1       80.0000000       -1.60000000
        1       30.0000000       -0.40000000
        2        1.0953760       -0.06623814
  O s
```

```
        0        0.9212952        0.39552179
        0       28.6481971        2.51654843
        2        9.3033500       17.04478500
  O p
        2       52.3427019       27.97790770
        2       30.7220233      -16.49630500
end
scf
  vectors input hcore
  maxiter 20
end

task scf
```

Para ejecutarlo desde el portal grid, se introdujeron los siguientes parámetros:

- En el campo *Executable* se proporciona el programa de script.nwchem.
- En el campo *Arguments* se proporciona el archivo de entrada de nwchem.
- El campo *CPU count* puede ser variable y depende de la cola seleccionada.

Y finalmente se da click en *Submit* para enviar el job.

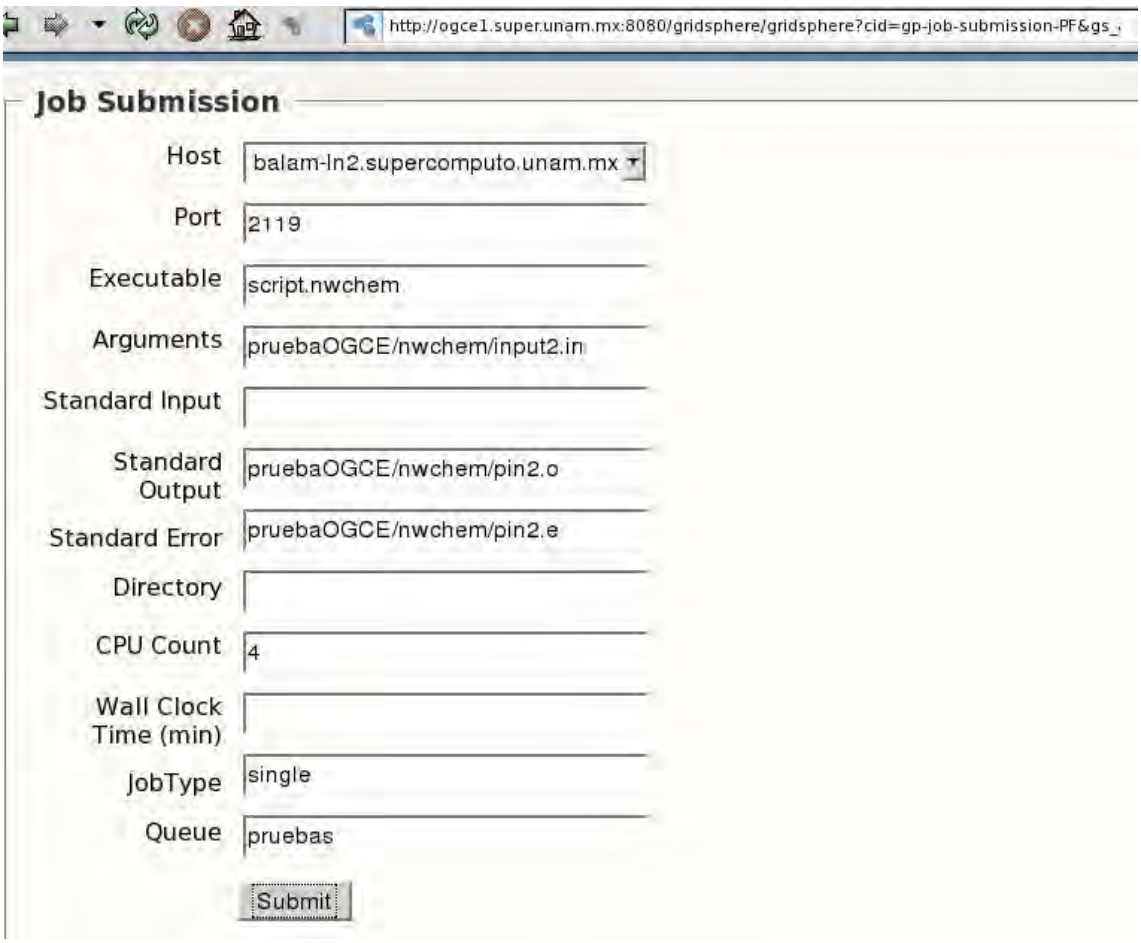


Figura 4.23: Parámetros para la ejecución de un job de Nwchem

4.5.3. MPI

Para ejecutar MPI en Kanbalam, fue necesario hacer modificaciones en el archivo $GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/lsf.pm, las modificaciones fueron:

```
print JOB "export PATH=/opt/appl/scripts-grid:/opt/hpmpi/bin:
/opt/hptc/bin:\$PATH\n";
print JOB "export LD_LIBRARY_PATH=/opt/hpmpi/lib/linux_amd64:
\$LD_LIBRARY_PATH\n";
print JOB "$mpirun -srun",  ' ';
```

El código fuente de un programa en mpi es el siguiente:

```
#include<stdio.h>
#include<mpi.h>

int main(int argc, char *argv[])
{
int rank,nproc,namelen;
char nombre_proc [MPI_MAX_PROCESSOR_NAME];

MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD,&nproc);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Get_processor_name(nombre_proc,&namelen);
printf("Hola proceso %d de %d en %s \n",rank, nproc,nombre_proc);
MPI_Finalize();
return 0;
}
```

Los parámetros para solicitar la ejecutación de este programa de mpi son los siguientes:

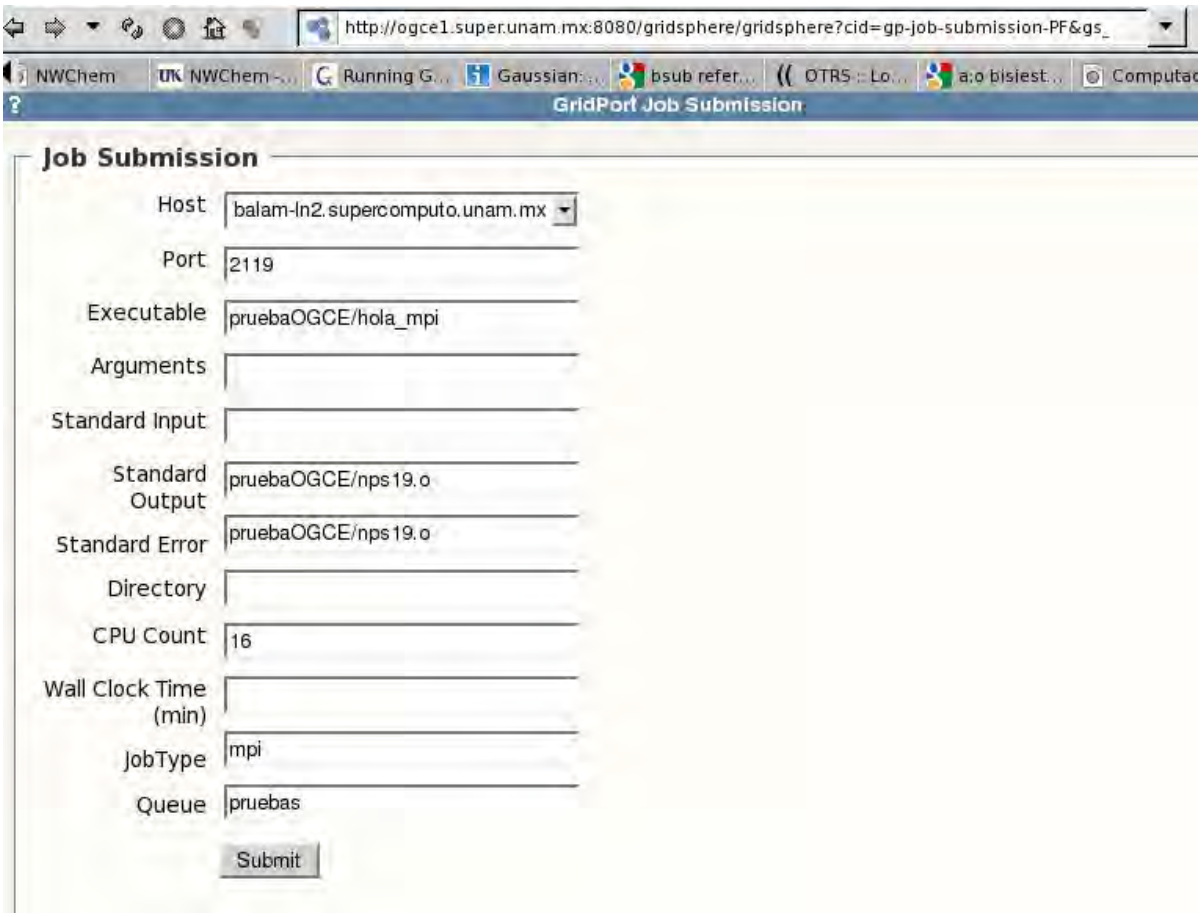


Figura 4.24: Parámetros para la ejecución de un job de MPI

- En el campo *Executable* se proporciona el programa de mpi a ejecutar.
- El campo *CPU count* puede ser variable y depende de la cola seleccionada.
- En el campo *Job Type* se proporciona el valor mpi.

4.5.4. OpenMP

Para ejecutar correctamente OpenMP en Kanbalam, fue necesario hacer modificaciones en el archivo $GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/lsf.pm, las modificaciones fueron las siguientes:

```
if($description->count() lt '4')
  {print JOB "export OMP_NUM_THREADS=",$description->count(),"\n";}
else
  {print JOB "export OMP_NUM_THREADS=4 \n";}
```

El código fuente del programa que utiliza openmp es el siguiente:

```
#include <omp.h>
#define N     1000000

main ()
{
int i,tid,c[N];

#pragma omp parallel shared(c) private(i,tid)
  {

  #pragma omp for schedule(dynamic) nowait
  for (i=0; i <= N; i++)
    {
    c[i] = i*i;
    tid = omp_get_thread_num();
    printf("i = %d, thread = %d, c[i] = %d\n", i, tid, c[i]);
    }
  }  /* end of parallel section */

}
```

Para utilizar openmp mediante el portal grid, se tienen que utilizar los siguientes parámetros:

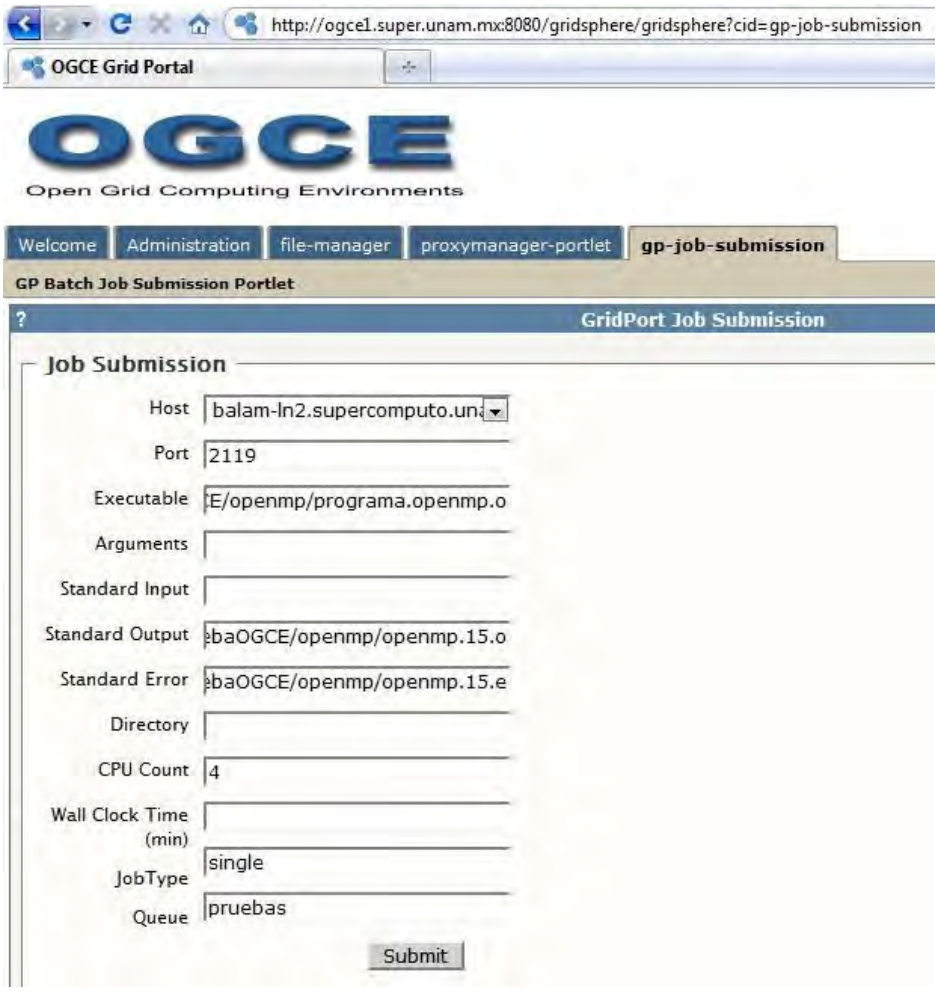


Figura 4.25: Parámetros para la ejecución de un job de OpenMP

- En el campo *Executable* se proporciona el nombre del programa de openmp.
- En el campo *Arguments* se proporciona los argumentos que utilice el programa.
- En el campo *CPU count* sólo se pueden proporcionar hasta 4 procesadores debido a que sólo existen 4 procesadores utilizando memoria compartida por nodo.

Y finalmente se da click en *submit* para enviar el job.

4.6. Estadísticas del Portal Grid

Se utilizó la herramienta JMeter [26] con el objetivo de realizar pruebas de estrés en el portal grid OGCE y verificar su correcto funcionamiento bajo estas condiciones. Mediante esta herramienta se simuló el ingreso de varios usuarios al portal y de esta manera se caracterizó el comportamiento con los diferentes componentes del portal grid.

A continuación en la Fig. 4.26 la duración (eje x) y el porcentaje de carga del procesador (eje y) para distintos números de usuarios, mientras que en la Fig. 4.27 se muestra la carga en memoria:

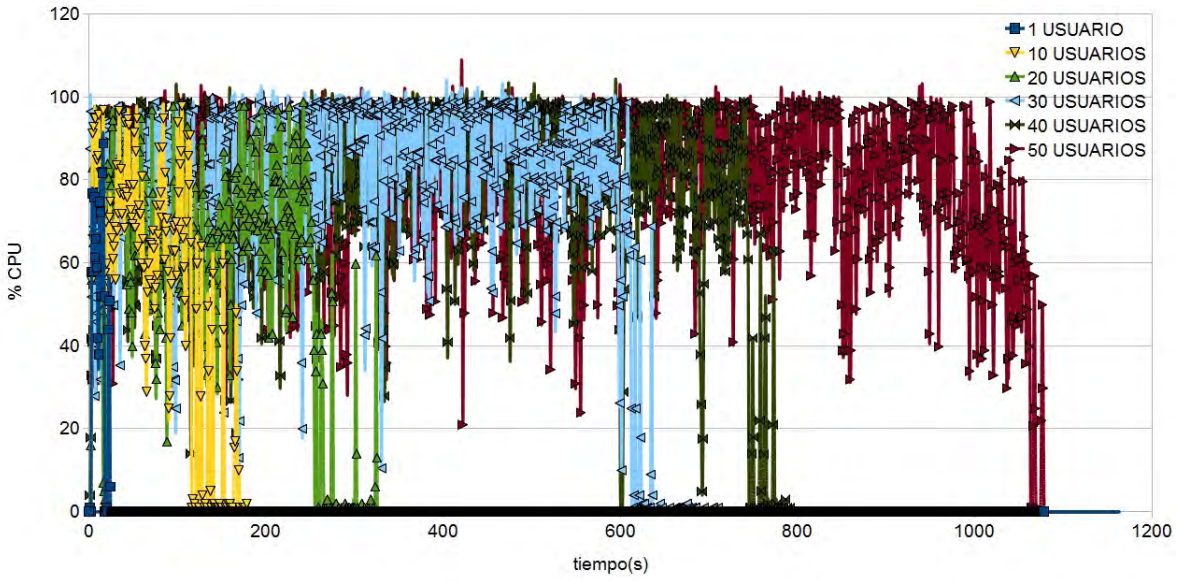


Figura 4.26: Carga del CPU al variar los usuarios conectados en el portal

De la gráfica 4.26 se puede apreciar que conforme se aumenta la cantidad de usuarios en el portal se aumenta el tiempo de ejecución de la prueba. El porcentaje de uso del procesador para todos los casos es aproximadamente 100 %. Para un usuario la prueba dura 20 segundos aproximandamente y cuando el portal atiende simultaneamente a 50 usuarios el tiempo de la prueba se alarga a cerca de los 19 minutos.

De la figura 4.27 no se aprecia un aumento de carga en memoria considerable al aumentar el número de usuarios, el uso de la memoria fue menor de 30 porciento en todas las pruebas realizadas.

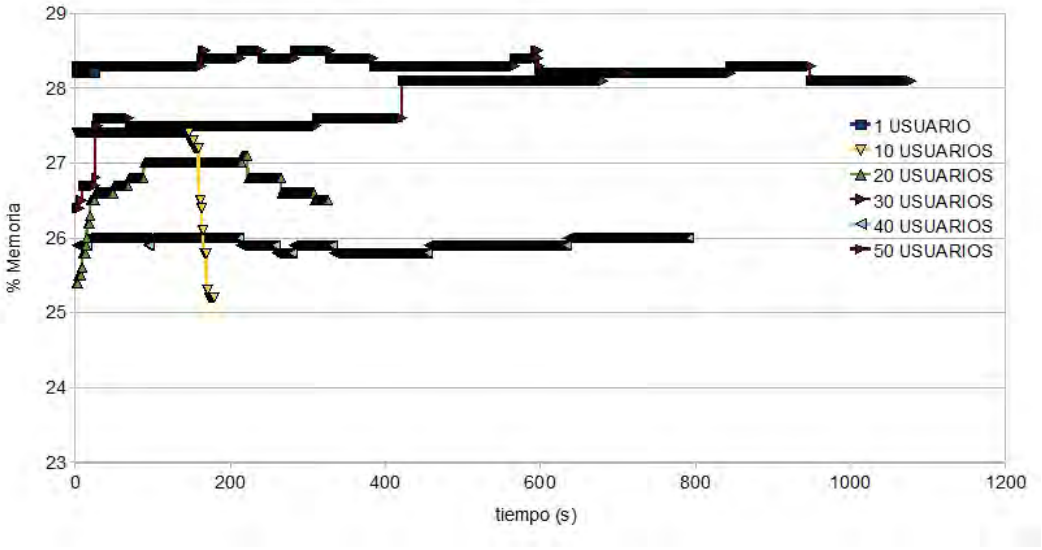


Figura 4.27: Porcentaje de memoria utilizado al incrementar la cantidad de usuarios conectados

A continuación se presentan los tiempos de respuesta obtenidos al variar la cantidad de usuarios que utilizan los difentes componentes del portal:

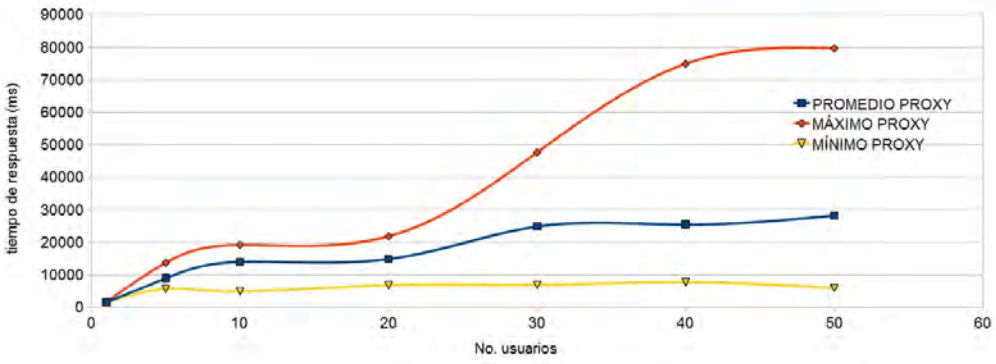


Figura 4.28: Carga al solicitar un certificado proxy

En la figura 4.28 se muestra que para una carga de 50 usuarios el tiempo promedio para obtener un certificado es de 30 segundos y el tiempo máximo que puede llegar a demorarse es de 1 minuto con 20 segundos aproximadamente.

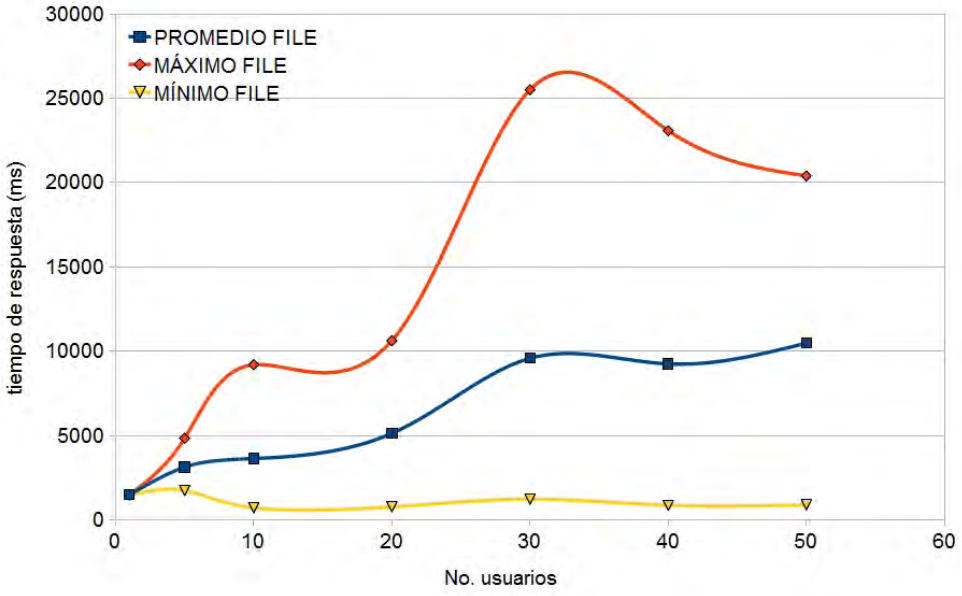


Figura 4.29: Carga al entrar al portlet de manejo de archivos

De la figura 4.29 se puede apreciar que el tiempo promedio de respuesta para entrar al portlet de manejo de archivos es de 11 segundos aproximadamente cuando se encuentran 50 usuarios utilizando el portal y un tiempo máximo de 25 segundos aproximadamente para 30 usuarios.

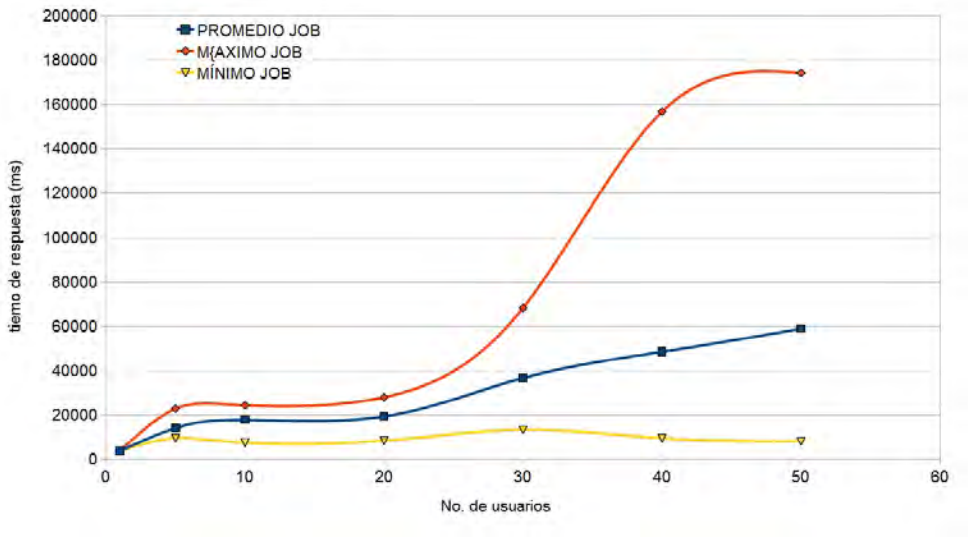


Figura 4.30: Carga al entrar al portlet de envío de jobs

En la figura 4.30 se muestra que el tiempo promedio de respuesta es de 1 minuto al acceder al portlet de envío de jobs cuando la carga es de 50 usuarios; y el tiempo máximo de respuesta de este portlet es de casi 3 minutos para una carga de 50 usuarios.

Los tiempos de respuesta en los principales componentes del portal grid son muy grandes al tener una carga mayor a 30 usuarios. Esto puede explicarse por las características del equipo donde se instaló el portal grid, puesto que se encuentra en una máquina virtual con un procesador y dicha máquina fue adquirida hace más de cinco años. Para tener mejores tiempos de respuesta en los diferentes componentes del portal grid, se recomienda instalar el portal grid en un equipo más potente, con varios procesadores y preferentemente en un servidor dedicado.

4.7. Resumen

En este capítulo se describieron las caracteristícas en hardware y en software de los equipos donde se instaló el portal grid, el servidor de certificados, el middleware de grid y los recursos de grid (*fabric*).

También se describió cómo se llevó a cabo la instalación del portal grid 2.5, la configuración de los diferentes componentes del portal, la modificación del código fuente del portal para poder interactuar adecuadamente con la parte de envío de jobs a una supercomputadora, y así mismo se mostraron los archivos de configuración de aplicaciones específicas en el middleware de grid utilizado.

Se mostró el funcionamiento del portal, al probar sus diferentes componentes: manejo de certificados proxy, manejo de archivos, pruebas de ejecución de diferentes aplicaciones (gaussian, nwchem, mpi y openmp).

Y finalmente se realizaron pruebas de carga al utilizar una herramienta que permite simular a usuarios dentro del portal grid. Esta herramienta y los resultados obtenidos, nos permitirán determinar las características en hardware que debe tener una máquina de producción del portal grid para una determinada cantidad de usuarios a atender.

Capítulo 5

Conclusiones

En esta tesis se describieron algunos de los conceptos fundamentales de un grid computacional, desde su definición hasta la arquitectura propuesta por Foster et al.[4]; se describieron tres proyectos relevantes en los grids computacionales; y se explicó el funcionamiento del middleware Globus Toolkit por capas y las tecnologías que utiliza para la construcción de un grid computacional. Por otra parte, se describieron los conceptos principales de un portal grid; se explicó la arquitectura de un portal grid y su funcionamiento; también se describió el portal grid OGCE, sus principales componentes y las tecnologías que utiliza para su funcionamiento. Y finalmente, se expuso la instalación, configuración y puesta en operación del portal grid OGCE.

El estudio de las tecnologías nos sirvió para entender el funcionamiento de un portal grid y su interrelación con otras tecnologías como el middleware de grid, Java CoG, etc. Lo anterior, nos ayudó a determinar los componentes necesarios para poner en funcionamiento un portal grid, las características de los componentes son:

- Se necesita un sistema operativo linux, unix o mac os para el portal grid. En esta tesis se mostró su instalación en distribuciones linux basadas en RPMs. El hardware necesario está en función de los usuarios que utilicen el portal grid; como sistema de pruebas se puede tomar como referencia la máquina donde se instalo el portal.
- El middleware a utilizar es Globus Toolkit. Se puede usar la versión dos hasta la actual, que es la cinco.
- El hardware para el servidor MyProxy está en función de los usuarios del grid computacional

- El hardware de los recursos de *fabric* puede ser variable, estos recursos pueden interacturar con diversos administradores de recursos y calendarizadores como pueden ser LSF, PBS, SGE, Condor; en esta tesis se ejemplicó como los recursos de fabric interactúan con el middleware de grid GT5 vía LSF.

Uno de los objetivos principales de esta tesis fue la "Implementación de un Portal Grid (*Grid Portal*) que faciliten el uso de equipos de supercómputo", las facilidades obtenidas son:

- Una interfaz (GUI) gráfica de usuario que simplifica el uso de los recursos computacionales.
- Proporcionar un solo punto de acceso para interactuar de forma segura con los diversos equipos de un grid computacional.
- Diversos servicios como administración de archivos, ejecución de programas, así como el monitoreo de los mismos.

Estas facilidades permiten a los usuarios de los equipos de un grid computacional concentarse sólo en la ejecución de sus aplicaciones y en la administración de su información. Es decir, el usuario no se preocupa en cómo funciona el grid computacional, ni en cómo se usa y funciona un equipo de supercómputo, ni en cómo se interrelacionan el grid computacional con el equipo de supercómputo. En nuestro caso el usuario no se preocupa en cómo el portal interactúa con la Supercomputadora Kanbalam. Esta es una caracterísitca importante del portal porque un usuario nuevo podría interactuar con la supercomputadora mediante el portal grid de una forma sencilla, intuitiva, casi inmediata y con poco entrenamiento previo.

Un defecto del portal es que es poco amigable definir un ambiente de ejecución particular, por lo que fue necesario crear scripts en la supercomputadora que se encarguen de crear ambientes específicos para las diferentes aplicaciones que se han probado desde el portal.

Se comprobó que un portal grid ya desarrollado, en específico el portal Grid OGCE versión 2.5, es fácil de implementar y de adaptar, sobre todo en cuestiones de seguridad y de manejo de archivos; sin embargo es necesario modificar su código fuente en la parte de Envío y Monitoreo de Jobs para acoplarse a la configuración de una supercomputadora.

El portal grid OGCE se implementó de forma exitosa en una máquina de servicio. El portal permite interactuar con la supercomputadora Kanbalam con las características

del portal anteriormente mencionadas, permitiendo específicamente enviar jobs de diversas aplicaciones como mpi, openmp, nwchem y gaussian. Esta última es importante para el Departamento de Supercómputo, porque representa el 33 porciento de los jobs en Kanbalam.

Finalmente, este portal, debido a sus características, puede ser utilizado como interfaz en la Delta Metropolitana para su interacción con los diversos equipos de supercómputo, tanto en el manejo de archivos entre los equipos, como en el envío de Jobs a los tres equipos de la Delta Metropolitana.

5.1. Trabajo a futuro

Instalación y configuración de un portlet de información relacionada con los equipos inmersos en un grid computacional, en especial de la supercomputadora Kanbalam, tal información puede ser cantidad de procesadores, memoria RAM, capacidad de almacenamiento, etc.

Realizar pruebas de funcionamiento del portal grid con usuarios de la supercomputadora Kanbalam con el fin de detectar y corregir posibles fallas del portal.

Evaluar la carga en un equipo nuevo para determinar la cantidad de usuarios que es posible atender con tiempos de respuesta aceptables.

Crear scripts de configuración de ambientes de ejecución (como los realizados en la sección 4.5) para diversas aplicaciones instaladas en la supercomputadora Kanbalam.

Virtualizar el servicio de envío de jobs para que las peticiones de realizadas desde el portal grid se distribuyan y balancen equitativamente en los tres nodos de login de la supercomputadora Kanbalam.

Adecuar el funcionamiento del portal grid para su correcto desempeño en la Delta Metropolitana.

Bibliografía

[1] http://www.cudi.edu.mx/primavera_2008/presentaciones/redes_opticas_azaek_fernandez.pdf

[2] https://grid.infn.it/modules/italian/index.php?pagenum=6

[3] http://www.cic-ctic.unam.mx/unamirada/
Número 182 Año IV.

[4] The Grid 2 Blueprint for a New Computing Infraestructure. Ian Foster y Karl Kesselman. 2004

[5] Globus Toolkit 4. Programming Java services. Borja Sotomayor y Lisa Childers. 2006

[6] Building web services with Java. Making sense of XML, SOAP, WSDL, UDDI 2nd. Edition. Steven Graham, et al. 2005

[7] The Grid Core Technologies. Maozhen Li, Mark Baker. Wiley, 2006

[8] http://www.globus.org/security/overview.html

[9] Grid Computing. Making the global infrastructure a reality. Fran Berman, Geoffrey C. Fox y Anthony C. G. Hey Wiley, 2003.

[10] http://www.eu-egee.org/fileadmin/documents/publications/EGEEIII_Publishable_summary.pdf

[11] http://www.eu-egee.org/

[12] http://project.eu-egee.org/index.php?id=117

[13] http://glite.web.cern.ch/glite/common/introduction.asp

[14] http://www.opensciencegrid.org/

[15] http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=965

[16] http://www.ligo.org/

[17] http://einstein.phys.uwm.edu/

[18] http://www.opensciencegrid.org/LIGO_Research_Highlight

[19] http://vimeo.com/5316159

[20] https://www.teragrid.org/

[21] http://www.youtube.com/watch?v=puLlm_edNz8

[22] http://rabbitbrush.frazmtn.com/sun-java-on-F8.html

[23] http://jpackage.org/installation.php

[24] http://wiki.alfresco.com/wiki/Installing_Labs_3_on_Linux_with_command_line_interface

[25] http://www.collab-ogce.org/ogce/index.php/Portal_download

[26] http://www.collab-ogce.org/ogce/index.php/JMeter_Testing