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".

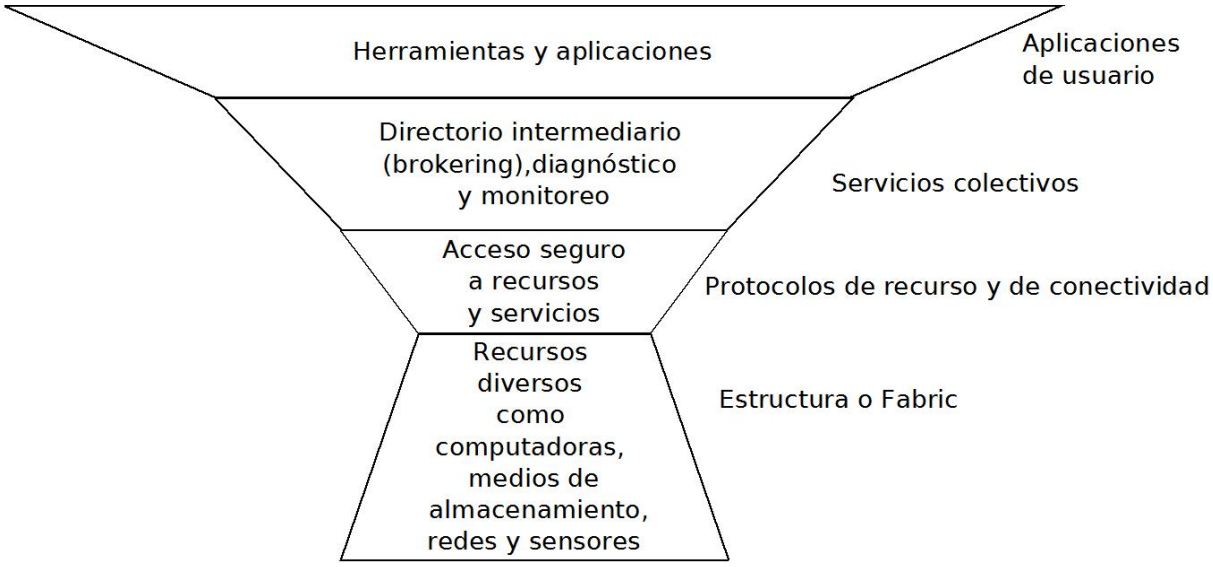

Figura 2.1: Arquitectura del grid computacional

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:

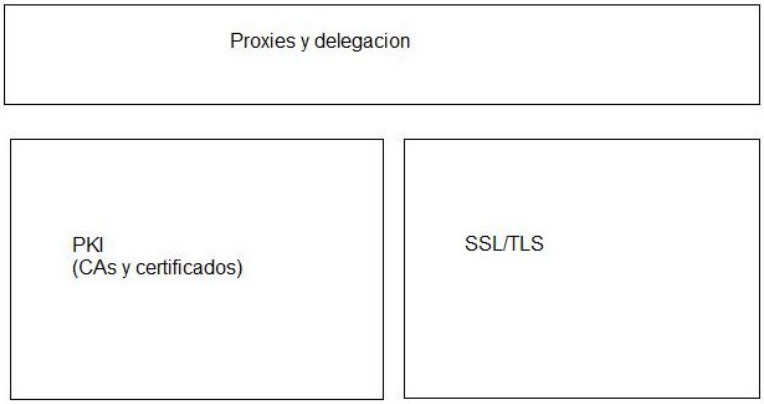

Figura 2.3: Infraestructura de Seguridad en Grid

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.

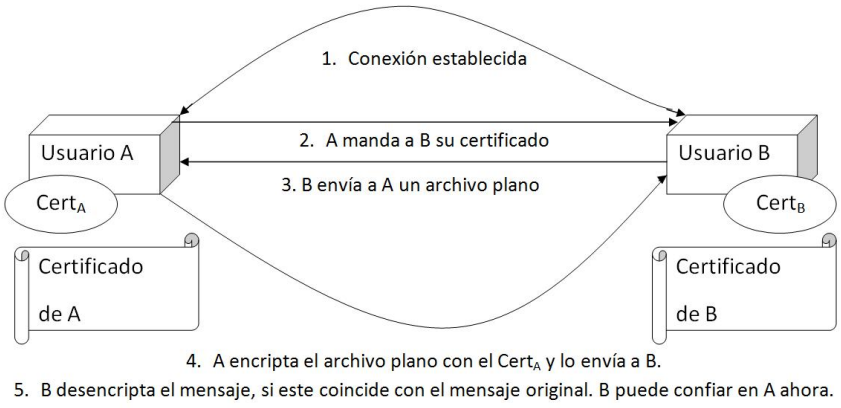

Figura 2.4: Autenticación Mutua

- 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:

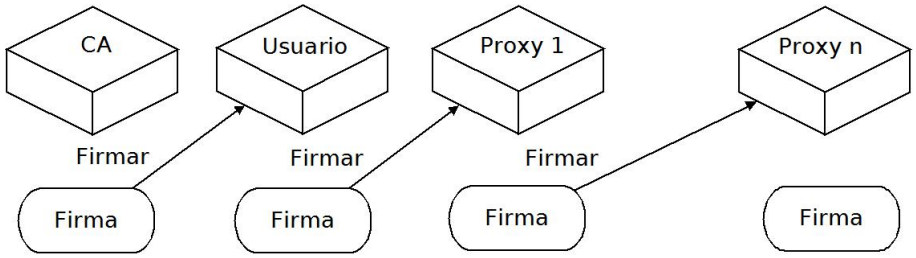

Figura 2.5: Delegación de certificados [8]

  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.

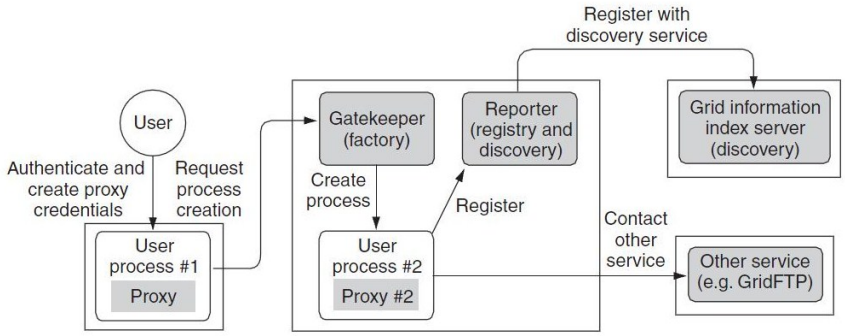

Figura 2.6: Funcionamiento de la capa de recursos en GT2

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.

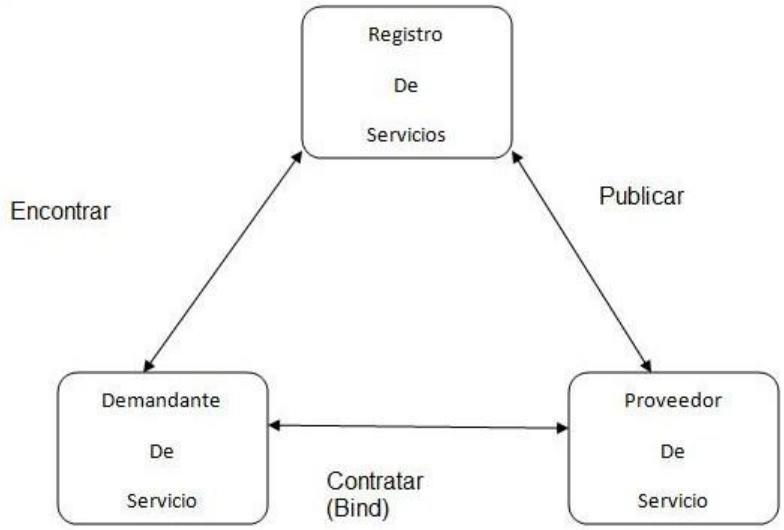

Figura 2.7: Arquitectura Orientada a Servicios

- 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

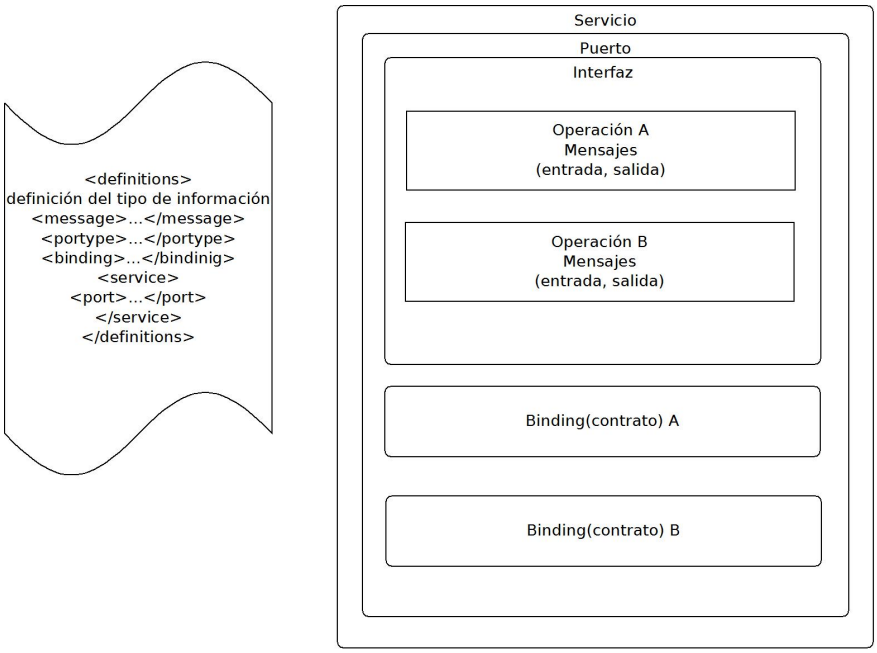

Figura 2.8: Estructura de un documento WSDL [7]

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.

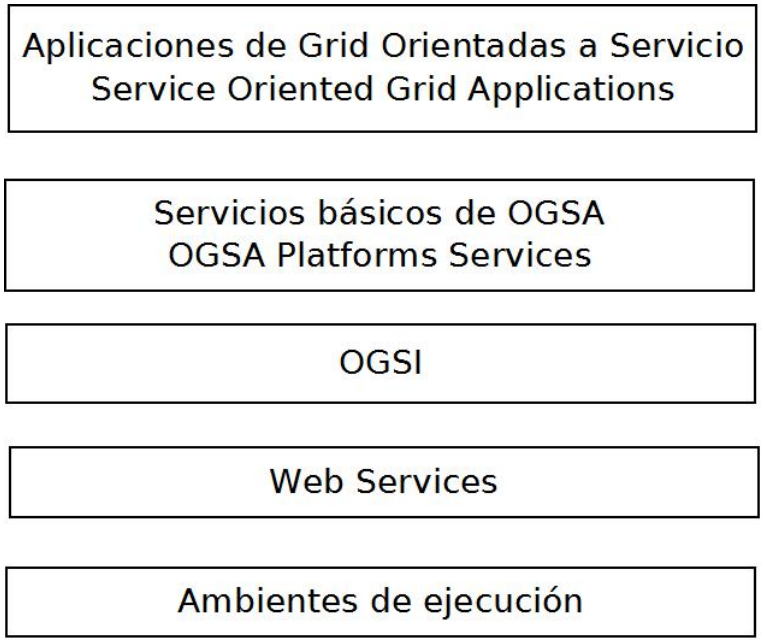

Figura 2.9: Construcción de aplicaciones Grid de acuerdo a OGSA utilizando OGSI

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.

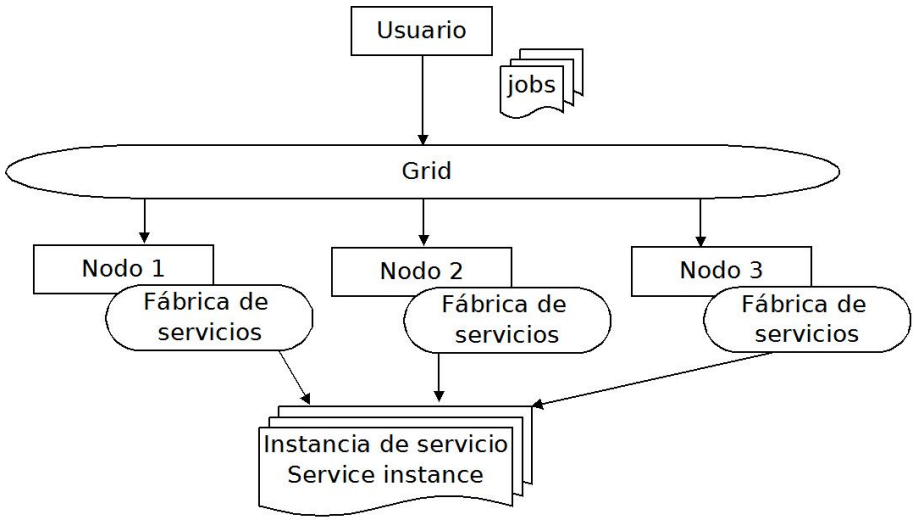

Figura 2.10: Envío de jobs con múltiples instancias de servicios grid

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.

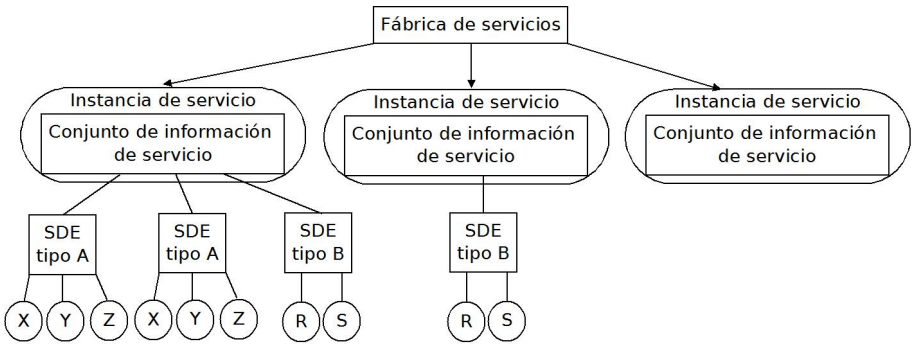

Figura 2.11: Vista de una fábrica de servicio

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.

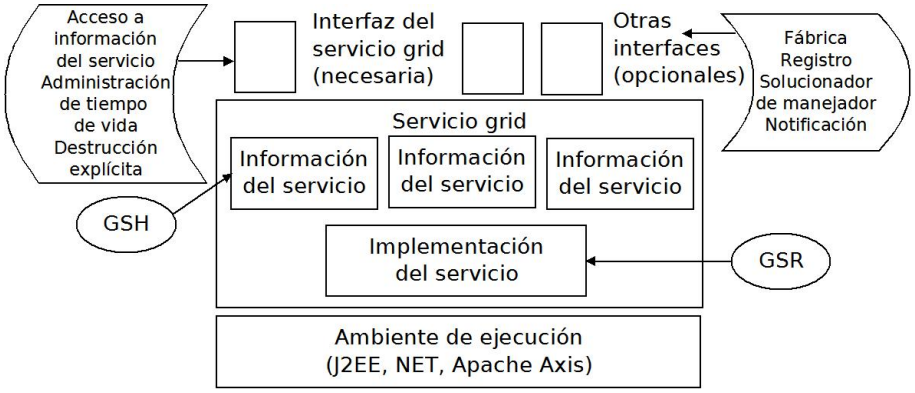

Figura 2.12: Estructura de un servicio Grid en OGSA

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.

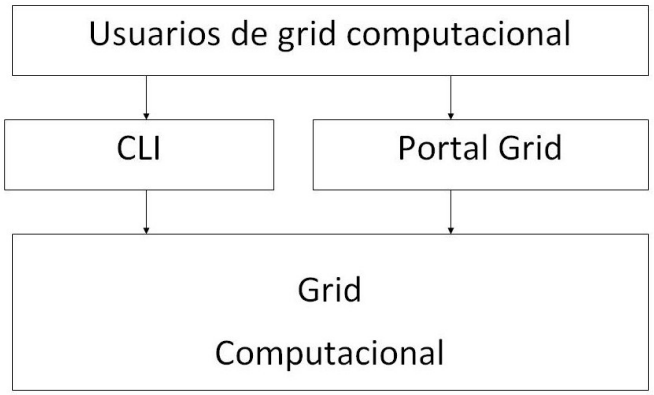

Figura 2.13: Conección a un grid computacional

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.

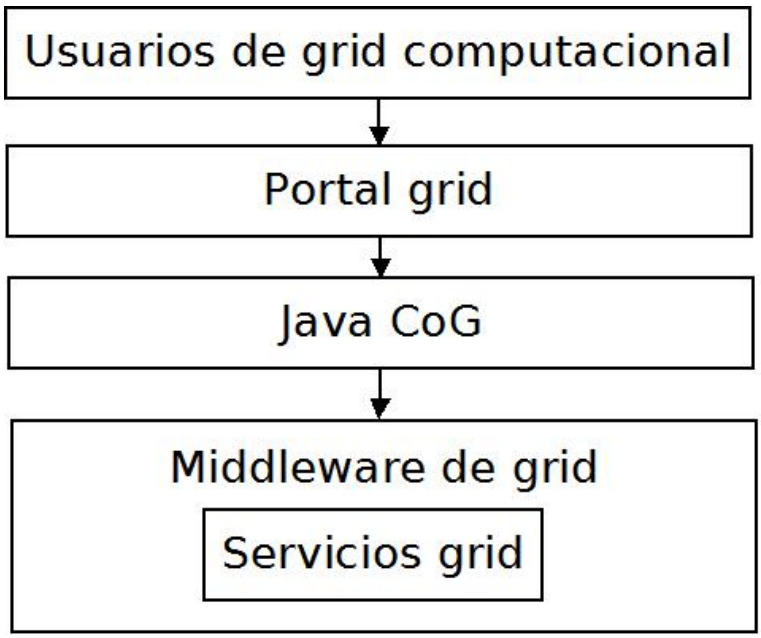

Figura 2.14: Java CoG