Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

COMPUTACIÓN DISTRIBUIDA: GRID COMPUTING

BENJAMÍN DOMÍNGUEZ HERNÁNDEZ

Asesorado por Ing. Douglas Luna

Guatemala, octubre de 2005

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

COMPUTACIÓN DISTRIBUIDA: GRID COMPUTING

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA

FACULTAD DE INGENIERÍA

POR

BENJAMÍN DOMÍNGUEZ HERNÁNDEZ

ASESORADO POR: ING. DOUGLAS LUNA

AL CONFERIRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2005

UNIVERSIDAD SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	
VOCAL II	Lic. Amahán Sánchez Álvarez
VOCAL III	Ing. Julio David Galicia Celada
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADOR	Inga. Elizabeth Domínguez Alvarado
EXAMINADOR	Ing. Ricardo Morales Prado
EXAMINADOR	Inga. Virginia Tala Ayerdi
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

COMPUTACIÓN DISTRIBUIDA: GRID COMPUTING,

tema que me fuera asignado por la Coordinación de la Carrera de Ciencias y Sistemas en julio de 2004.

Benjamín Domínguez Hernández

AGRADECIMIENTO A:

Dios

Agradezco a Dios por haberme dado la vida y permitirme concluir esta etapa de estudios.

Mis padres y Abuela

Por haberme apoyado en todo momento a lo largo de mi vida estudiantil y por ser mis primeros educadores.

Mis centros de estudios y profesores

Por haber cimentado en mi los conocimientos que pude adquirir en todos estos años. A mis colegios, Liceo Gabriela Mistral y Colegio Salesiano Don Bosco por ser la base de mi carrera en la Universidad de San Carlos y al Ingeniero Douglas Luna por orientarme en el desarrollo de mi trabajo de graduación.

ACTO QUE DEDICO A:

DIOS

Por darme la sabiduría y espíritu para seguir adelante y continuar en los momentos difíciles.

MIS PADRES

JORGE ALFREDO DOMÍNGUEZ ESTRADA Y ANA MARÍA HERNÁNDEZ DE DOMÍNGUEZ, Por darme primero que todo la vida y luego guiarme a través de ella, por todo el apoyo incondicional y comprensión que me brindaron en todo momento a lo largo de mi carrera.

MIS HERMANOS

DANIEL ENRIQUE, ANA ESTER Y GUILLERMO ALFONSO, Por su comprensión y apoyo. Y que esto sea un ejemplo para seguir adelante.

MI ABUELA MARÍA, TÍOS, PRIMOS Y FAMILIA EN GENERAL

Por sus consejos y cariño.

TODOS MIS BUENOS y GRANDES AMIGOS

Dentro y fuera de la universidad, que siempre me apoyaron para lograr metas y con quienes tantas experiencias he vivido.

LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Por la formación académica recibida.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

API	Acrónimo de *Application Program Interface* - Interfaz de programa de aplicación-
ASP	ASP es un Proveedor de Servicios de Aplicaciones -*Application Service Provider*-, provee Servicios y Aplicaciones de software a sus clientes a través del Internet
Atomicidad	Este término se refiere a la unidad de una transacción, de forma tal que si es compuesta de varios pasos estos deben realizarse completamente.
Buffer	Es una fuente de almacenamiento temporal que reside en el propio dispositivo ya sea de entrada, o de salida.
Business-to-Business	Término referido usualmente al comercio electrónico entre empresas.
Cluster	Un cluster es un grupo de múltiples ordenadores unidos, mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador más potente por los usuarios y las aplicaciones. Se espera de un cluster que presente combinaciones de alto rendimiento, alta disponibilidad, balanceo de carga y escalabilidad.
Corba	Es una tecnología con una extremada potencia, que permite el desarrollo de aplicaciones distribuidas.
Criptografía de clave pública	Criptografía de clave o llave Pública es un método que permite intercambiar mensajes en forma segura, en base a la asignación de dos llaves complementarias, una pública, una privada, para las personas que se involucran en una transacción.

Criptografía de clave secreta	En criptografía de llave secreta, existe un único código, o llave, tanto para encriptar como para desencriptar mensajes.
DNS	El DNS -Domain Name Service- es un sistema de nombres que permite traducir de nombre de dominio a dirección IP y vice-versa. Aunque Internet sólo funciona en base a direcciones IP, el DNS permite que los humanos usemos nombres de dominio que son bastante más simples de recordar para acceder diferentes sitios web.
OSPF	Es un protocolo de encaminamiento para redes IP mediante el uso de rutas más cortas y accesibles y la construcción de un mapa de la red.
EJB	Acrónimo de Enterprisse Java Beans. Tecnología para desarrollo de componentes de software que permite agrupar funcionalidades de una aplicación en un ambiente de ejecución.
Firma electrónica	La firma electrónica es una manera de representación y confirmación de la identidad de un sujeto en el medio electrónico.
Global Gris Forum	El Global Grid Forum (GGF) está formado por una comunidad de más de 5000 investigadores y usuarios de tecnologías grid.
Globus Toolkit	Software middleware que permite compartir recursos localizados en diferentes dominios y con diferentes políticas de seguridad y gestión de recursos
GRAM	Acrónimo de Globus Resource Allocation Manager. Utiliza una interfaz única para gestionar el uso de recursos remotos.

Grid	Sistema paralelo y distribuido que permite la compartición, detección y agregación de recursos “autónomos” geográficamente distribuidos.
Gris Aplicación	Aplicaciones distribuidas que se ejecutan y utilizan recursos en toda la grid constituyéndose la razón de ser de toda la estructura computacional.
Gris Fabric	Capa de la Grid que proporciona el acceso y la gestión de los diversos recursos computacionales, mediante mecanismos de reconocimiento del entorno de estos mismos recursos.
***Grid Security Infrastructure* - GSI-**	Protocolo de seguridad para la comunicación y autenticación en un entorno Grid.
GRIP	Acrónimo para *Grid Resource Information Protocol.*
Gris Middleware	Software encargado de realizar gestión de recursos. Análogo a un sistema operativo.
Gris Services	Web Services enriquecidos que permiten gestionar el estado y proporcionan durabilidad a los recursos.
***Hos*t**	Un *host*, literalmente anfitrión, es un ordenador directamente conectado a una red y que efectúa las funciones de un servidor y alberga servicios, como correo electrónico, grupos de discusión o páginas Web accesibles por otros ordenadores de la red.
http	Son las siglas de *Hypertext Transfer Protocol*, el metodo utilizado para transferir ficheros hipertexto por Internet. En el Internet, las páginas escritas en HTML

	utilizan hipertexto para enlazar con otros documentos.
ICMP	Es un protocolo de control robusto encargado de generar mensajes de error en caso de fallas durante el transporte de los datos en una red IP.
Intranet	Es un Internet interno diseñado para ser utilizado en el interior de una empresa, Universidad, u organización. Lo que distingue a un intranet del Internet e libre acceso es el hecho de que el intranet es privado.
Java	Es un lenguaje de programación orientado a objetos creado por Sun Microsystems que permite crear programas que funcionan en cualquier tipo de ordenador y sistema operativo.
Kerberos	Es un protocolo de autenticación que admite la forma ampliada de suplantación denominada delegación, esta permite que el contexto de seguridad de un llamador tenga acceso a los recursos de la red y a los recursos locales del sistema operativo del servidor.
LDAP	LDAP es un protocolo que se utiliza para tener acceso a servicios de directorio como Active Directory.
MDS	Acrónimo de *Metacomputing Directory Service*. Se utiliza como un directorio que permite el descubrimiento y manejo de recursos.
MPI	Acrónimo de Message Passing Interface
Multicast	Significa multidifusión. Modo de difusión de información en vivo que permite que ésta pueda ser recibida por múltiples nodos de la red y por lo tanto por múltiples usuarios.

OGSA	Acrónimo de *Open Grid Services Architecture*
OGSI	Acrónimo de *Open Grid Services Infrastructure*
Organización Virtual -VO-	Conjunto de instituciones y usuarios que comparten recursos y colaboran entre sí para alcanzar una meta común.
Outsourcing	Inglés, Literalmente en inglés quiere decir "ir a buscar la fuente afuera"; acuerdo mediante el cual una compañía brinda a otra un servicio que de otra forma se haría internamente.
P2P	*Peer-to-peer* o Punto-a-Punto es un modelo de comunicaciones en el cual los dispositivos computacionales, computadoras, servidores y otros dispositivos inteligentes se enlazan directamente unos con otros.
Petabytes	Unidad de medida de la capacidad de memoria y de dispositivos de almacenamiento informático. Un PB corresponde a 1.024 billones (2^50) de bytes.
PKI	Acrónimo de Public Key Infrastructure, Infraestuctura de clave pública. La PKI es un sistema para verificar la autenticidad de cada interlocutor implicado en una transacción de Internet.
***Pool* de Ordenadores**	Recursos computacionales que se desea compartir.
Proxy de Usuario	Es una cuenta de servicio configurada específicamente, también conocida como cuenta de proxy, que se utiliza exclusivamente para tener acceso a un recurso indirecto, por ejemplo una base de

	datos, en una aplicación distribuida de varios niveles. Los componentes de nivel medio suelen utilizar un número limitado de cuentas de servicio para conectarse a una base de datos y proporcionar la agrupación de la conexión.
QoS	Sus siglas significan *Quality of Service* o calidad de servicio.
Resource Broker	Entidad que se encarga de buscar y seleccionar los recursos adecuados para que un trabajo sea ejecutado.
S/KEY	Es un mecanismo que implementa el *One Time Password,* es el caso en el que una clave solamente sirve para una sesión, evitando así los potenciales peligros derivados de la captura de un password por parte de un tercero.
SDK	Sus siglas en ingles significan *Software Development Kit* o Herramienta de desarrollo de software.
Secure Shell	SSH es una utilidad que nos permite conectar a un host remoto, ejecutar comandos de forma remota, o realizar transferencia de ficheros entre máquinas, todo mediante comunicación segura y autenticación criptográfica.
Servicios Web Xml	Exponen funcionalidad útil a los usuarios Web mediante un protocolo Web estándar. Los Servicios Web XML son unidades de lógica de aplicación que proporcionan servicios e información a otras aplicaciones.
Single Sign On	Esquema de seguridad en el que los usuarios han de identificarse una única vez para acceder a los recursos de que desean disponer.
SOAP	SOAP es un protocolo ligero basado en

	XML para intercambiar información en un entorno distribuido. Lo utilizan los servicios Web.
SSL	Significa *Secure Socket Layer* o socket de capa segura. Es un protocolo desarrollado por Netscape Communications Corporation para dar seguridad a la transmisión de datos en transacciones comerciales en Internet.
SSP	Acrónimo de *Storage Service Providers***.**
TCP/IP	Protocolo de Control de Transmisión / Protocolo Internet. -*Transmission Control Protocol* / Internet Protocolo-.
Testbeds	Término utilizado en desarrollo de software para denominar los escenarios o casos de prueba.
UDDI	UDDI proporciona un directorio de Servicios Web XML, permitiendo que los negocios que ofrecen Servicios XML sean encontrados. UDDI es actualmente un proceso comunitario con más de 200 compañías participantes.
UDP	Protocolo de Datagrama de Usuario (*User Datagram Protocol*).
Unicast	Significa unidifusión, una dirección que solamente puede ser reconocida por un sistema anfitrión.
VPN	Acrónimo de Virtual Private Network ó Red Privada Virtual. Red en la que al menos alguno de sus componentes utiliza la red Internet pero que funciona como una red privada, empleando para ello técnicas de cifrado.
WSDL	Documento mediante el que se expone la descripción de un Web Service y que es utilizado para construir aplicaciones que se

	comuniquen con el. Un archivo WSDL es un documento XML que describe un conjunto de mensajes y cómo se realiza el intercambio de mensajes.
WSRF	Es el acrónimo para *Web Service Resource Framework.*
XML	El XML -*Extensible Markup Language*, Lenguaje de Marcas Ampliable-, es una forma flexible de crear formatos de información y compartir tanto el formato como los datos en la World Wide Web, intranets y otras redes.
.NET	Microsoft Visual Studio® .NET es un ambiente de desarrollo. Utilizando Visual Studio .NET, los desarrolladores pueden crear aplicaciones y servicios Web.

.

RESUMEN

Durante años se ha luchado por conseguir la integración necesaria entre sistemas de diferentes organizaciones, o, aún entre sistemas dentro de una misma organización, sin embargo, dada la diversidad de tecnologías los desarrolladores se ven en la necesidad de construir sobre diversidad de aplicaciones y para diferentes entornos. La comunidad profesional de la información ha estado fascinada con el potencial de unir diferentes ordenadores obteniendo de esta forma un solo ordenador virtual que permita el procesamiento masivo de datos enfocándose en los recursos.

Algunos de los inconvenientes asociados a las soluciones tradicionales y los sistemas basados en la centralización de los datos por medio de un servidor son la falta de escalabilidad, mantenimiento caro, además, una vez adquirido equipo este podrían estar mucho tiempo desaprovechado.

Sin embargo, en la lucha por la integración de recursos se han desarrollado algunas soluciones como lo son las arquitecturas cluster y arquitecturas de intranet, logrando cierto acercamiento sin llegar a constituirse en la solución optima debido por ejemplo a la dificultad de mantenimiento, escalabilidad limitada, falta de amortización y distribución de la carga entre recursos cuando estos están desaprovechados, entre otros.

La Grid Computing surge como la tecnología que permite la integración de diferentes recursos computacionales para conformar una sola Unidad. Se puede definir como una infraestructura que permite el acceso y procesamiento concurrente de un programa entre varias entidades computacionales independientes.

Es un nuevo paradigma de computación distribuida que está enfocado al acceso remoto a recursos computacionales. Es un paradigma donde cada nodo procesa la parte que le toca. Periódicamente, podemos contactar con cada nodo, para asegurarnos de que todo va bien e incluso realizar correcciones sobre la marcha.

Las organizaciones se interconectan por medio de un Grid y mantienen sus propias políticas de seguridad y gestión de recursos mediante un lenguaje entendible por todos los entornos, Xml. Esto significa que la tecnología usada para construir un Grid es complementaria a otras tecnologías, lo que permite aprovechar los recursos distribuidos en la Intranet de una organización, por ejemplo.

Muchos y variados son los campos de aplicación de la grid Computing entre los que destacan la Supercomputación Distribuida, Sistemas distribuidos en tiempo real y/o procesos de análisis intensivo de datos entre otros.

OBJETIVOS

General

Definir, explicar y presentar los conceptos, características y ventajas de la Computación Distribuida, así como sus aplicaciones y relación con otras tecnologías informáticas.

Específicos

1. Describir los conceptos de la computación distribuida.

2. Dar a conocer las áreas de aplicación de la computación distribuida y su relación con diferentes tecnologías.

3. Dar a conocer la arquitectura y requisitos de la computación distribuida.

4. Describir la forma de implementación y trabajo de una aplicación en sistemas de computación distribuida.

5. Dar a conocer las políticas y requerimientos de seguridad en la computación Grid.

INTRODUCCIÓN

Actualmente, la difusión de la información con el uso del Internet ha crecido increíblemente y es notable la utilidad y los beneficios obtenidos a partir de esto. Por otro lado, la cantidad de implementaciones, a través de la misma Internet, y el almacenamiento de grandes cantidades de datos han hecho de los recursos computacionales una necesidad. Las necesidades de las organizaciones para el procesamiento de datos son cada vez mayores, mientras que el área científica ha puesto sus ojos en el estudio de fenómenos que necesitan a su vez una cantidad de procesamiento enorme y acceso a diferentes recursos que no necesariamente se encuentren dentro de las mismas organizaciones científicas.

Los Grids integran el uso de diferentes tecnologías e infraestructura como redes, comunicación, computación e información que permiten proporcionar en conjunto una plataforma virtual para la computación y gestión de los datos del mismo modo que Internet integra recursos para formar una plataforma virtual de información. De esta forma, se puede acceder a gran cantidad de recursos computacionales dispersos geográficamente y crear una organización compuesta de varias entidades que pueden beneficiarse con el uso de estos recursos.

A diferencia de las tecnologías actuales, Grid presenta una serie de características que permiten la integración y optimización del uso de recursos y no se encuentra orientada a la información como tal, sino que contempla el uso, planificación, gestión y distribución de la información alrededor de los recursos.

El presente trabajo abarca las generalidades de la tecnología Grid, la arquitectura que soporta dicha tecnología y conceptos referentes a seguridad y

planificación de trabajo y recursos en un entorno Grid, así como las aplicaciones y la descripción de una implementación para observar las ventajas potenciales del uso de un ambiente Grid.

1. FUNDAMENTOS DE LA COMPUTACIÓN GRID

1.1 Definición

Grid es una infraestructura que permite el acceso y procesamiento concurrente de un programa entre varias entidades computacionales independientes. Un sistema paralelo y distribuido que permite la compartición, detección y agregación de recursos “autónomos” geográficamente distribuidos, de manera dinámica, en tiempo de ejecución, y depende de su disponibilidad, capacidad, rendimiento, coste y requisitos de QoS de los usuarios.

Uno de los objetivos de un Grid consiste en integrar y optimizar, mediante middleware, el uso de recursos distribuidos de cálculo intensivo y de grandes bases de datos, como si estuvieran en un cluster local.

Las organizaciones que se interconectan por medio de un Grid mantienen sus propias políticas de seguridad y gestión de recursos. Esto significa que la tecnología usada para construir un Grid es complementaria a otras tecnologías, lo que permite aprovechar los recursos distribuidos en la Intranet de una organización, por ejemplo.

Los Grids integran el uso de diferentes tecnologías como redes, comunicación, computación e información que permiten proporcionar una plataforma virtual para la computación y gestión de los datos del mismo modo que Internet integra recursos para formar una plataforma virtual para la información.

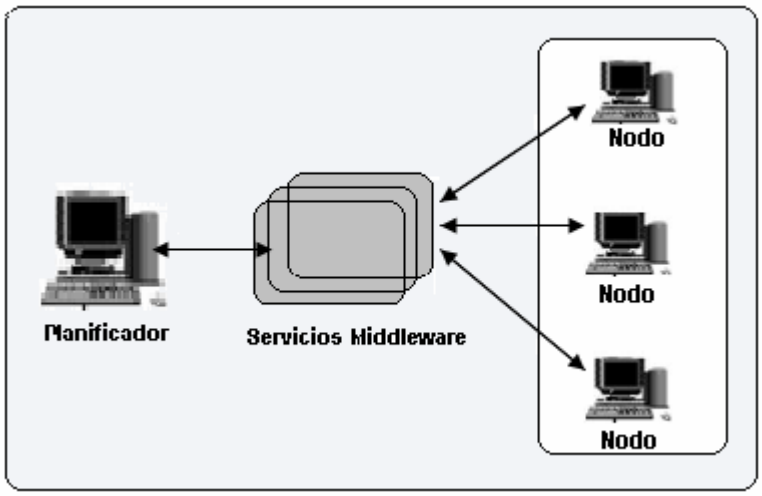
No importa si el usuario accede al Grid para utilizar un único recurso (ordenador, archivo de datos, etc.), o para utilizar varios recursos agregados como un ordenador virtual coordinado, en ambos casos, el Grid permite a los usuarios disponer, de manera uniforme, de una interfaz para los recursos, proporcionando una plataforma poderosa y comprensible para la computación global y la gestión de los datos.

Este nuevo paradigma de computación distribuida fue propuesto por Ian Foster y Carl Kesselman a mediados de los 90. Está enfocado al acceso remoto a recursos computacionales.

La Grid se constituye como un *pool* de recursos computacionales geográficamente dispersos al que tenemos acceso. Una analogía utilizada frecuentemente se refiere a esta tecnología como la red eléctrica de recursos computacionales. La figura 1 muestra el proceso básico del funcionamiento de una Grid.

El procesamiento de información o la ejecución de algún proceso en un nodo perteneciente a la Grid lo decide un “planificador” en función de las características de cada nodo y de cada partición. Cada nodo procesa la parte que le toca. Periódicamente podemos contactar con cada nodo, para asegurarnos de que todo va bien, e incluso realizar correcciones sobre la marcha. Cada nodo devuelve el resultado de la computación realizada.

Figura 1. Funcionamiento básico de una Grid.



1.2 Características Generales y Ventajas

1.2.1 Características

- Un ordenador central distribuye un proceso entre los ordenadores conectados a una red.
- El sistema aprovecha la capacidad de todos los ordenadores conectados. Cuando no están siendo utilizados al 100 por ciento por el usuario reciben tareas del ordenador central.
- Todos los recursos disponibles en la red son aprovechados, independientemente de su arquitectura y sistema operativo.
- Si el usuario está utilizando sólo una parte de la capacidad del ordenador, el resto se aprovecha para el cálculo.
- Si el usuario está utilizando todos los recursos o el ordenador se cae, la tarea se reasigna a otra máquina disponible.

- La aplicación es accesible desde cualquier ordenador de la red. Con un interfaz cómodo y sencillo se pueden añadir al sistema tantas máquinas como se desee.
- La conexión directa entre los PC's (P2P) evita la sobrecarga del ordenador central.
- La tecnología Grid es fácilmente escalable, creciendo según las necesidades de cada empresa.
- Transparente para el usuario que participa en la GRID (como *Worker*).
- A diferencia de las redes convencionales que se centran en la comunicación entre dispositivos, la Grid computing aprovecha los ciclos de procesamiento no utilizado de todos los ordenadores conectados a una red de forma que se resuelven los problemas de las tareas que son demasiado intensivas para que las resuelva una única máquina.

Un cuadro comparativo entre el uso de paradigmas comunes como cliente/servidor "tradicional", el uso de tecnología actual (CORBA, EJB, .Net, …), y el uso de tecnología Grid se presenta a continuación:

Tabla I. Comparación entre tecnologías distribuidas conocidas y Grid.

Tecnologías comunes	Uso de Grid
Orientados a la información	Orientados a los Recursos
Sistemas fuertemente acoplados	Sistemas débilmente acoplados
Seguridad en Segundo Plano	Seguridad en Primer Plano
Orientados a bajas latencias y comunicación síncrona	Orientado al crecimiento y comunicación asíncrona.
No suelen contemplar QoS.	Orientado a QoS.
Pobre Escalabilidad	Gran escalabilidad

1.2.2 Ventajas

Entre los beneficios de utilizar una arquitectura de Grid computing podemos enumerar:

- Integración de sistemas y dispositivos heterogéneos. Grid computing proporciona un conjunto de capacidades de integración horizontal que dirige de forma efectiva los recursos de toda una empresa, e incluso extienden la solución entre múltiples organizaciones. Por ejemplo, un científico que participe en una investigación Grid podría obtener el acceso a un único superordenador conectado a un laboratorio.
- Mejora el coste de los entornos operativos. A través de la visualización, compartición y gestión de recursos mediante funciones heterogéneas de tecnologías de la información, la Grid computing ayuda a simplificar los entornos operativos y su gestión reduciendo su administración y supervisión. Además, como consecuencia de fomentar la utilización eficiente de los recursos, Grid computing puede ayudar a las empresas a construir una infraestructura de tecnologías de la información de costes efectivos que asegure la completa utilización de las inversiones en tecnología existente.
- Incrementa la capacidad de recursos para responder a las fluctuaciones de demanda. Permitiendo a las organizaciones de tecnologías de la información agregar recursos distribuidos y explotar una capacidad no utilizada, los Grids incrementan de forma importante la cantidad de recursos computacionales y de datos disponibles. La Grid computing ayuda a crear infraestructuras de tecnologías de la información que pueden responder rápidamente a oleadas inesperadas en el tráfico y uso de los recursos.

- Permite sacar ventaja de los recursos del Grid como una alternativa ante la recuperación de los desastres tradicionales, los departamentos de tecnologías de la información pueden mejorar la fiabilidad y disponibilidad de sus infraestructuras tecnológicas para aumentar la resistencia a una fracción del coste de los sistemas duplicados.
- Mejora el tiempo de obtención de resultados para nuevos productos y servicios. Aumentando la productividad y la colaboración, las organizaciones mejoran el tiempo en la obtención de resultados. Tanto si estos resultados incluyen llevar un nuevo producto al mercado más rápidamente, resolver un complejo problema de negocio más pronto, o realizar un análisis de datos en profundidad para lanzar un nuevo servicio.
- Facilitan la colaboración y promueven la flexibilidad en las operaciones. La Grid puede agrupar no sólo recursos tecnológicos distintos, sino también a la gente. Facilitando que el personal pueda compartir, acceder y gestionar la información, la tecnología Grid puede mejorar la colaboración y soporte a estrategias de globalización.
- Incrementan la productividad y permiten proporcionar a los usuarios finales un acceso sin restricciones a los recursos informáticos, de datos y de almacenamiento que necesitan, la tecnología Grid puede ayudar a las compañías a mejorar la gestión de recursos humanos.
- La utilización eficaz de los recursos existentes es una de las claves para minimizar costes. Los recursos y procesos organizacionales deben Ayudar a asegurar la óptima utilización de los recursos informáticos, la tecnología Grid puede ayudar a las empresas a evitar las dificultades comunes de sobreprovisionamiento o incurrir en el exceso de costes para infraestructura.

1.3 Requisitos

1.3.1 Grid Middleware

Servicios de gestión (servicios de información y aplicaciones, servicios de red, información de usuario,…), sistemas de ficheros (almacenamiento y caché distribuidos, transparencia de localización) y servicios de meta-información (nombrado de recursos, información de seguridad…). Se le podría comparar a un sistema operativo. Ejemplos: Globus Toolkit, Legion, Boinc, etc.

1.3.2 Aplicación Grid

Aplicaciones distribuidas, alto rendimiento de computación (tareas de procesamiento intensivo de datos…) y aplicaciones de grupo (conferencias como aplicaciones, simulaciones distribuidas…). Las aplicaciones Grid se distinguen de las tradicionales aplicaciones cliente/servidor por la utilización simultánea de un gran número de recursos procedentes de múltiples dominios administrativos, estructuras complejas de comunicación y estrictos requerimientos de rendimiento, entre otras cosas. Las aplicaciones que van a funcionar sobre la GRID. Ejemplos: aplicación meteorológica, aplicación de simulación, aplicaciones con gran uso de cálculos, etc.

1.4 Organización Virtual (VO)

Concepto fundamental en Computación Grid. Es la agrupación de recursos de varios individuos y/o organizaciones distintas que colaboran para alcanzar una meta común. La pertenencia a una VO no es permanente. Puede cambiar según las necesidades. La agrupación de usuarios para formar una VO facilita gestión de recursos y seguridad.

1.4.1 Justificación de las Organizaciones Virtuales

El problema real y específico en que está justificado el concepto de Grid es coordinar la compartición de recursos y la resolución de un problema a través de organizaciones virtuales dinámicas multiinstitucionales. El compartir al que nos referimos no es primordialmente el intercambio de archivos, sino sobre todo el acceso directo a las computadoras, al software, a los datos y a otros recursos, como es requerido por una gama de estrategias de resolución de problemas que requieren colaboración entre industria, ciencia e ingeniería. Este compartir de recursos, debe ser altamente controlado, con los proveedores del recurso y los consumidores definiendo clara y cuidadosamente aquello que se comparte, lo que se permite compartir, y las condiciones bajo las cuales se puede compartir. El conjunto de instituciones y usuarios que comparten estos recursos y se someten a las reglas de compartición forman una organización virtual (*Virtual Organization*, VO).

Los siguientes son ejemplos de VOs: proveedores de servicio a aplicaciones (ASPs, por sus siglas en ingles), proveedores de servicios de almacenaje, proveedores de ciclos de procesamiento, consultores comprometidos con un fabricante de autos para realizar la evaluación de un escenario durante la planeación para una fábrica nueva; miembros de un consorcio industrial que hace una oferta para un avión nuevo; un equipo de gerencia enfocado en paliar crisis, bases de datos y sistemas de simulación que se utilizan para planear la respuesta a una situación de emergencia.

Cada uno de estos ejemplos representa un enfoque para el procesamiento y la resolución de problemas basados en la colaboración e interacción de distintos recursos para cómputo y ambientes que necesitan grandes volúmenes de datos. Como también estos ejemplos demuestran, las

VOs varían enormemente en su propósito, alcance, tamaño, duración, estructura, comunidad, y sociología. Sin embargo, el estudio cuidadoso de los requerimientos subyacentes de la tecnología nos conduce a identificar un conjunto de preocupaciones comunes y sus correspondientes requisitos. En detalle, vemos la necesidad de construir relaciones altamente flexibles para compartir recursos, evolucionando desde los sistemas cliente-servidor a arquitecturas *peer-to-peer* para los sofisticados y exactos niveles de control de cómo son utilizados los recursos compartidos, incluyendo el control de acceso *multi-stakeholder*, delegación, y uso de políticas locales y globales; para compartimiento de recursos variados, ejecución de programas, de archivos, y datos a computar, sensores, y redes; y para los diversos modos de uso, extendiéndose desde un solo usuario a múltiples y del funcionamiento sensible a coste-sensible y aplicaciones que por lo tanto están orientadas a calidad de servicio. Las tecnologías de computo distribuidas actuales no tratan las preocupaciones y los requisitos anteriormente enumerados.

Por ejemplo, las tecnologías actuales del Internet tratan la comunicación y el intercambio de información entre computadoras pero no proporcionan soluciones integradas al uso coordinado de recursos en múltiples sitios para cómputo. Los intercambios *business-to-business* se centran en compartir información (a menudo vía servidores centralizados).

Las tecnologías de cómputo distribuidas tales como CORBA y Java permiten compartir recursos dentro de una sola organización. Los ambientes de cómputo distribuidos soportan el uso seguro de recursos compartidos a través de sitios, pero la mayoría de VOs lo encontraría demasiado pesado e inflexible. Los proveedores de servicios de almacenamiento (SSPs) y los proveedores de servicio a aplicaciones (ASP) permiten a las organizaciones cubrir sus requisitos de almacenaje y cómputo mediante *outsourcing* pero con

ciertas restricciones: por ejemplo, los recursos de SSP son típicamente enlazados hacia un cliente por medio de una red privada virtual (VPN). El uso de compañías externas para realizar procesos de cómputo se apoya solamente sobre el acceso altamente centralizado a los recursos. En resumen, la tecnología actual no se acomoda a los diferentes usos de los recursos ni proporciona la flexibilidad y el control necesario para compartir los mismos y establecer VOs.

Es aquí donde las tecnologías GRID se incorporan al cuadro. Durante los últimos años, los esfuerzos de investigación y desarrollo de la comunidad GRID han producido protocolos, servicios, y herramientas que encaran los desafíos que se presentan cuando intentamos construir VOs escalables. Estas tecnologías incluyen soluciones de seguridad que soportan manejo de credenciales y políticas cuando los cómputos se realizan a través de múltiples instituciones; protocolos de gestión de recursos que proveen acceso seguro remoto a datos y recursos de computo y coasignación a múltiples recursos; protocolos de búsqueda de información y servicios que proveen configuración e información acerca del estado de los recursos, organizaciones, y servicios; y servicios de manejo de datos para localizar y transportar conjuntos de datos entre los sistemas del almacenamiento y las aplicaciones.

Debido a su enfoque dinámico, el uso de tecnologías Grid complementa las tecnologías distribuidas existentes. Por ejemplo, los sistemas de cálculo distribuido de una empresa pueden utilizar tecnologías basadas en Grid para alcanzar un recurso compartido que se encuentra mas allá de los límites institucionales; en el espacio de ASP/SSP, las tecnologías Grid se pueden utilizar para establecer mercados dinámicos para los recursos de almacenaje y computo, superando las limitaciones debidas a las configuraciones estáticas actuales.

Las VOs tienen el potencial de cambiar dramáticamente la manera en que utilizamos las computadoras para solucionar problemas, así como el uso de Internet ha cambiado la manera cómo intercambiamos información. Como se ha ilustrado con los ejemplos presentados la necesidad de realizar procesos en entornos de colaboración, es fundamental para muchas disciplinas y diversas actividades: no se limita a la ciencia, a la ingeniería y a las actividades económicas. Es debido a esta amplia aplicabilidad de los conceptos de VO que la tecnología Grid es importante.

1.4.2 Características de las Organizaciones Virtuales

Consideremos los cuatro escenarios siguientes:

- Una compañía que necesita tomar una decisión sobre la colocación de una nueva fábrica hace uso de un modelo financiero sofisticado de pronóstico de un ASP, proveyéndole acceso a los datos históricos apropiados de una base de datos corporativa en los sistemas del almacenaje que a la vez provee un SSP. Durante la reunión de toma de decisiones, los escenarios "¿que pasaría sí?" se ejecutan interactivamente, aun si los participantes de la simulación están situados en diversas ciudades. El ASP por sí mismo contacta con el proveedor de ciclos de procesamiento durante los escenarios particularmente exigentes, requiriendo por supuesto que el proveedor de dichos ciclos resuelva los requisitos de seguridad y funcionamiento.

- Un consorcio industrial formado para desarrollar un estudio de viabilidad para un avión supersónico "*next-generation*" emprende una simulación multidisciplinaria exacta del avión entero. Esta

simulación integra los componentes de software propietarios desarrollados por diversos participantes, con cada componente operando sobre las computadoras de cada participante y teniendo acceso a la base de datos adecuada y otros datos puestos a disposición para el consorcio por sus miembros.

- Un equipo de manejo de crisis responde a un derramamiento químico usando modelos locales del tiempo y del suelo para estimar la extensión del derramamiento, determinando el impacto basado en la localización de la población así como características geográficas tales como ríos y abastecimientos de agua, creando un plan de mitigación (quizás basado en modelos químicos de la reacción), y personal de respuesta a la emergencia planeando y coordinando la evacuación, notificando hospitales, y así sucesivamente.

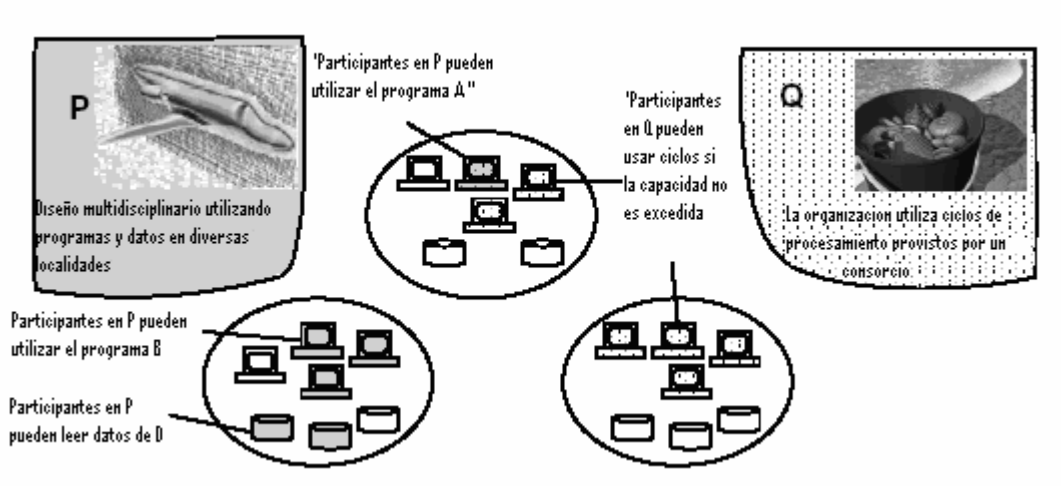
- Los millares de físicos en los centenares de laboratorios y de universidades por todo el mundo vienen juntos a diseñar, crear, funcionar, y analizar los productos de un detector importante en la CERN, el laboratorio europeo de alta física de energía. Durante la fase de análisis, reúnen sus cómputos, almacenaje, y recursos de una red para crear una Grid de datos capaz de analizar petabytes de los datos.

Estos cuatro ejemplos se diferencian en muchos aspectos: el número y el tipo de participantes, los tipos de actividades, la duración y la escala de interacción de los recursos que son compartidos. Pero también tienen mucho en común, en cada caso, un número de participantes mutuamente desconfiados que varían de la relación anterior (quizás ninguna) desean

compartir recursos para realizar una cierta tarea. Además, el compartir conlleva mucho más que simplemente el intercambio de un documento: puede implicar acceso directo a software lejano, a las computadoras, a los datos, a los sensores, y a otros recursos. Por ejemplo, los miembros de un consorcio pueden proporcionar el acceso al software y a los datos especializados y/o reunir sus recursos de cómputo.

Observemos tres organizaciones reales (los óvalos), y dos VOs: P, que liga a participantes en un consorcio aeroespacial del diseño, y Q, que liga a colegas que proporcionaron los ciclos de procesamiento. La organización a la izquierda participa en P, la que esta a la derecha participa en Q, y la tercera a la izquierda es un miembro de P y de Q. (Ver Figura). Las políticas que gobiernan el acceso a los recursos varían según las organizaciones actuales, recursos, y VOs implicadas.

Figura 2. Organizaciones diferentes participando en distintas VO's y compartiendo diferentes recursos.



Fuente: Ian Foster, Varl Kesselman. **(In)The anatomy of the Grid**. Pagina 4.

Al compartir un recurso se tiene una condicional: cada dueño del recurso asegura la disponibilidad, conforme a restricciones de cuando, donde, y qué puede ser hecho. Por ejemplo, un participante en la VO P de la figura puede permitir a los socios de la VO invocar su servicio de simulación solamente para los problemas "simples".

Los consumidores de recursos pueden también poner restricciones en las características de los recursos con que están preparados para trabajar. Por ejemplo, un participante en la VO Q pudo aceptar solamente los recursos de cómputo reunidos certificados como "seguros." La puesta en práctica de tales restricciones requiere de mecanismos para expresar políticas de seguridad, establecer la identidad de un consumidor o de un recurso (autentificación), y para determinar si una operación es consistente con las relaciones de compartición aplicables (autorización). Las relaciones para compartir recursos pueden variar dinámicamente en un cierto plazo, en términos de los recursos implicados, debido a la naturaleza del acceso permitido, y debido a los accesos permitidos a los participantes. Estas relaciones no necesariamente implican un sistema explícitamente nombrado de individuos, sino el uso de un diseño multidisciplinario usando programas y datos provenientes de múltiples locaciones y definidos implícitamente por las políticas que gobiernan el acceso a los recursos.

Por ejemplo, una organización puede permitir el acceso a cualquier persona que pueda demostrar que es un cliente o un estudiante. Dada la naturaleza dinámica de las relaciones de compartición de recursos requerimos de mecanismos para descubrir y caracterizar la naturaleza de las relaciones que existan en un punto particular en el tiempo. Por ejemplo, un nuevo participante que se une a la VO Q debe poder determinar a qué recursos puede

tener acceso, la calidad de estos recursos, y las políticas que gobiernan el acceso.

Compartiendo relaciones que no son simplemente cliente-servidor, sino *peer-to-peer*, los proveedores pueden ser a su vez consumidores, y compartir relaciones que puedan existir entre cualquier subconjunto de participantes. Las relaciones compartidas pueden ser combinadas para coordinar el uso de recursos, perteneciendo cada uno de estos a diversas organizaciones. Por ejemplo, en la VO Q, un proceso que comenzó en un recurso de cómputo puede tener acceso posteriormente a datos o a subcómputos iniciados en otra parte. La capacidad de delegar autoridad de manera controlada llega a ser importante en tales situaciones, al igual que los mecanismos para coordinar operaciones a través de múltiples recursos.

El mismo recurso se puede utilizar de diversas maneras, dependiendo de las restricciones puestas en el uso de los mismos y la meta compartida. Por ejemplo, una computadora se puede utilizar para correr solamente un software específico en un arreglo compartido, mientras que puede proporcionar ciclos genéricos de cálculo en otro. Debido a carencia de conocimiento a priori sobre cómo un recurso puede ser utilizado, limitaciones de funcionamiento, métricas, expectativas, y limitaciones (por ejemplo calidad de servicio) se puede ser parte de las condiciones puestas en el uso del recurso que se comparte.

Estas características y requisitos definen lo que se denomina una organización virtual, un concepto que creemos es fundamental en los sistemas modernos. Las VOs permiten a grupos dispares de organizaciones y/o de individuos compartir recursos de manera controlada, de modo que los miembros puedan colaborar para alcanzar un objetivo compartido.

1.5 Arquitectura de la Grid

La clave en la arquitectura de Grid es la interoperabilidad: el objetivo es establecer la compartición de recursos entre potenciales participantes cualesquiera, lo que se traduce en protocolos comunes. A la hora de especificar la arquitectura, se sigue el modelo de “reloj de arena” (*hourglass*). En esta arquitectura, el cuello de botella, reside en las capas de Recursos y Conectividad (*Resource* y *Conectivity*), que facilitan la compartición de recursos individuales. A continuación se detallan las diferentes capas y funcionalidades.

Figura 3. Arquitectura de la Grid.

Fuente: María Cruz Valiente Blázquez. **(In) Sistemas Distribuidos**. Pagina 3.

1.5.1 Infraestructura: Interfaz a Control Local

Denominada también Grid Fabric. Esta capa proporciona los recursos a los que el acceso compartido es conducido por los protocolos Grid (acceso local a recursos lógicos como CPU, software, ficheros), y a la infraestructura compuesta por los recursos computacionales que queremos compartir: Ordenadores individuales, *Pool* de ordenadores, Clusters, Supercomputadores, Sistemas de almacenamiento.

Es importante que existan mecanismos de interrogación, que permitan descubrimiento de estructura, estado y capacidades, así como mecanismos de gestión que proporcionen control de la QoS entregada. Los principales elementos que ha de proporcionar son los siguientes:

- Recursos computacionales: para comienzo de programas y control de la ejecución de procesos. Aquí las funciones de interrogación obtienen características de hardware y software e información de estado, y los mecanismos de gestión permiten realizar reserva de recursos.
- Recursos de almacenamiento: obtención y entrega de ficheros. Los mecanismos de gestión permiten controlar recursos para transferencia de ficheros (CPU, discos, ancho de banda), y las funciones de interrogación obtienen información acerca de utilización de recursos, como espacio libre y utilización de ancho de banda.
- Recursos de red: Mediante el uso de mecanismos de interrogación se encamina a obtener las características de la red y su carga, y los mecanismos de gestión proporcionan control sobre recursos de la red (priorización, reserva).

1.5.2 Conectividad: Comunicación Fácil y Segura

Esta capa define los protocolos de comunicación y autenticación para las transacciones de red Grid. Por lo que se refiere a los requisitos de comunicación, son cubiertos por la pila de protocolos TCP/IP: Internet (IP e ICMP), transporte (TCP y UDP) y aplicación ((DNS, OSPF). En cuanto a la autenticación, la capa *Conectivity* se define con las siguientes características:

- *Single Sign On*: los usuarios han de identificarse una única vez para acceder a los recursos Grid proporcionados por la capa *Fabric*.
- Delegación: el usuario puede "traspasar" su acceso a los recursos Grid a un programa.
- Integración con soluciones de seguridad locales: debe existir interoperación con estas soluciones locales, generalmente mediante mapeo en el entorno local de la seguridad Grid.
- Seguridad basada en usuario: si el usuario accede a dos recursos diferentes A y B de manera conjunta, la seguridad de A y B no debe interactuar.

Los protocolos empleados por Globus Toolkit, aparte de los indicados para la comunicación, son *Grid Security Infrastructure* (GSI) para autenticación y autorización, empleando certificados X.509.

1.5.3 Recurso: Compartición de Recursos Individuales

Está constituida por protocolos que permitirán la negociación segura, iniciación, monitorización, control, pago y tarificación de las operaciones compartidas sobre recursos individuales, mediante la llamada a funciones de la capa Fabric para el acceso a los recursos individuales. Notar por tanto que están plenamente enfocados a recursos individuales. Estos protocolos se pueden diferenciar en dos clases:

- De información: obtienen información sobre la estructura y estado de un recurso; por ejemplo configuración, carga actual, coste.

- De control o gestión: se encargan de la negociación del acceso al recurso en términos de requisitos (QoS, reserva) y operaciones a ser realizadas (creación de procesos, acceso de datos). Son responsables de que la política requerida por el usuario sea consistente con la ofrecida por el recurso.

Globus Toolkit emplea protocolos basados en estándares en esta capa: GridFTP, *Grid Resource Information Protocol* (GRIP, basado en LDAP, como protocolo de información), *Grid Resource Access and Management* (GRAM, basado en HTTP, para la ubicación de recursos) y el propio LDAP.

1.5.4 Recursos: Coordinación de Recursos Múltiples

Está constituida por los protocolos y servicios no asociados a un recurso específico sino que son globales y capturan interacciones de diferentes recursos. Las principales funciones que lo caracterizan son:

- Servicios de directorio: permiten a las VO participantes descubrir recursos de otras VO.
- Coubicación y programación: ubicación de tareas en recursos para un propósito específico y ejecución de tareas en un momento específico.
- Servicios de monitorización y diagnóstico.
- Servicios de descubrimiento de software: elección del mejor software para un determinado problema.
- Gestión de carga de trabajo.
- Servidores de autorización de comunidad.
- Servicios de colaboración.

1.5.5 Aplicaciones

Mediante la utilización de APIs y SDKs (*Software Development Kit*) las aplicaciones intercambian mensajes de protocolo con el servicio/capa adecuado a fin de ejecutar las acciones adecuadas.

1.6 Campos de Aplicación de la Tecnología Grid

Actualmente, hay 5 aplicaciones generales para la Computación Grid:

1.6.1 Supercomputación distribuida

Aplicaciones cuyas necesidades es imposible satisfacer en un único nodo porque satisfacen necesidades puntuales e intensivas de computación. Ejemplos: Simulaciones, cálculos numéricos, *Data Mining*, análisis de grandes volúmenes de datos.

1.6.2 Sistemas Distribuidos en Tiempo Real

Sistemas que generan un flujo de datos a alta velocidad que debe ser analizado en tiempo real (experimentos de física de alta energía, *e-Medicine*, control remoto de un recurso no-trivial, como sería por ejemplo un equipo médico,…).

1.6.3 Servicios Puntuales

Acceso puntual a un recurso que no nos merece la pena tener en nuestra organización. Similar a las dos aplicaciones anteriores, con las siguientes diferencias: no nos referimos a "'potencia computacional", y no tiene que ser en tiempo real. Ejemplos: acceso a hardware específico para ciertos tipos de análisis (químico, biológico, ...)

1.6.4 Proceso Intensivo de datos

Aplicaciones que trabajan con grandes volúmenes de datos, y que es imposible almacenar en un único nodo. En este caso, los datos se distribuirán a lo largo del Grid.

1.6.5 Entornos Virtuales de Colaboración (Teleinmersión)

Se utiliza la potencia computacional y la naturaleza distribuida del Grid, para crear entornos virtuales 3D distribuidos.

Actualmente, las aplicaciones del Grid Computing están muy orientadas al mundo científico. Las empresas empiezan a interesarse por estas

tecnologías. Se espera en un futuro un Grid Público a nivel mundial, en el que cualquier persona pueda disponer, si lo necesita, del poder computacional de un supercomputador. Además, todos podemos ser también proveedores de ciclos de CPU.

2. GLOBUS TOOLKIT

2.1 Definición

El Globus Toolkit, desarrollado por el Globus Alliance. Es una colección de componentes software que ofrecen la infraestructura básica necesaria para la creación y ejecución de aplicaciones distribuidas (Grid Middleware), así como para la construcción de Grids. Actualmente, Globus se ha convertido en el estándar de facto para la computación distribuida. Globus consta de tres componentes fundamentales: gestión de recursos, servicio de información y gestión de datos; todos ellos usan el protocolo de seguridad GSI para la comunicación y autentificación.

Los componentes anteriores, ya sea de forma independiente o conjunta, facilitan el acceso transparente y seguro a recursos distribuidos geográficamente en diferentes dominios de administración, además de servir como herramientas básicas para implementar las fases de la planificación de trabajos en Grids, a saber: descubrimiento, selección y preparación de recursos; y envío, monitorización, migración y finalización de trabajos.

2.2 Versiones

A continuación se listan las principales versiones existentes del Globus Toolkit:

- GT1: Implementado completamente en C, actualmente está obsoleto.
- GT2: Mayormente C (con algunas interfaces Java), ya no se encuentra en desarrollo y fue ampliamente difundida. Utiliza una herramienta de seguridad: *Globus Security Infrastructure* (GSI), (PKI, SSL v3). Esta versión esta soportada para AIX, HP-UX, Iris, Linux y Solaris. Esta versión goza de una gran aceptación pero tiene algunas deficiencias.

- GT3: Combina C y servicios basados en Java (Implementa OGSI, está basada (casi completamente) en tecnologías estándar como XML, SOAP, WSDL, y demás servicios de la versión GT 2.4, es una versión ampliamente difundida.
- GT4: Web Services (WSRF) y C, actualmente se encuentra en desarrollo y ya se encuentra disponible la versión GT3.9.1.

2.3 Grid Services

El Globus Toolkit a partir de su versión 3 (GT3) introduce el concepto de Grid Service, que es una ampliación considerable a los Web Services. Los Grid Services son la base de la arquitectura GT3.

Al plantear la siguiente generación de tecnologías Grid (OGSA y OGSI), el Global Grid Forum buscó una tecnología de objetos distribuidos que se adaptase a las necesidades de una aplicación Grid. Los Web Services son una buena opción. Sin embargo, a pesar de sus ventajas, también tienen importantes limitaciones.

Principales desventajas de los Web Services:

- *Stateless*
- No-transientes (Persistentes)
- No tienen ”servicios de apoyo” (notificaciones, servicio de persistencia, gestión del ciclo de vida, etc.)

Los Grid Services son Web Services mejorados. Están basados en SOAP y WSDL y, por lo tanto, 100% compatibles con Web Services “tradicionales”.

2.3.1 Introducción a los Grid Services, Web Services XML, ventajas y desventajas

2.3.1.1 XML

XML o *Extensible Markup Language*, es un lenguaje de *tags* o etiquetas que permite definir de un modo muy sencillo la estructura jerárquica a la que pertenece un dato, así como HTML permite definir la forma en que se muestra un dato en nuestro navegador.

Hasta ahora el déficit más importante a la hora de integrar aplicaciones desarrolladas en distintos lenguajes o sobre distintas plataformas, era que cada forma de trasmitir los datos era propietaria de la aplicación que la generaba, y en muchos casos la forma de transmisión no permitía que la comunicación fuera fluida. Este problema sucede incluso al intentar integrar aplicaciones sobre una misma plataforma desarrolladas en un mismo lenguaje. En el mejor de los casos, una vez establecida la comunicación de datos, nos encontrábamos con un verdadero problema a la hora de estructurarlos jerárquicamente de acuerdo al modelo de origen.

XML permite, de un modo sumamente sencillo, estructurar la información de modo de enviarla con total seguridad de que el receptor sabrá que ese dato tiene una relación con otro dato dentro de la misma estructura enviada, también puede saber qué tipo de dato es el que está recibiendo (XML *Schema*), puede establecer cómo mostrarlo (XSL) e incluso cómo tiene que devolverlo (SOAP). Así XML permite la comunicación de una aplicación a otra, o recibir y enviar datos estructurados mediante Internet sin tener que idear mecanismos complejos o excesivamente pesados para rearmar la información como en su origen.

Algunas utilidades de los Xml son:

- Creación de programas que interpretan bien el dato
- Los programas hablan entre ellos sin intervención humana:
    - Computación Distribuida
    - Interoperatividad
    - Monitorización

2.3.1.2 Servicios Web XML

Los servicios Web XML son los bloques de construcción básicos en la transición al proceso distribuido en Internet. Los estándares abiertos y el foco en la comunicación y colaboración entre las personas y aplicaciones han creado un entorno donde los servicios Web XML se están convirtiendo en la plataforma para la integración de aplicaciones. Las aplicaciones se construyen utilizando múltiples servicios Web XML desde diversas fuentes que trabajan conjuntamente con independencia de dónde residen o cómo hayan sido implementadas.

Probablemente existan tantas definiciones de los Servicios Web XML como compañías que los desarrollan, pero casi todas las definiciones tienen estos aspectos comunes:

- Los Servicios Web XML exponen funcionalidad útil a los usuarios Web mediante un protocolo Web estándar. En la mayoría de casos, el protocolo utilizado es *Simple Object Access Protocol* (SOAP).

- Los Servicios Web XML proporcionan un modo de describir sus interfaces con suficiente detalle para permitir a un usuario construir una aplicación cliente para hablar con ellos. Esta

descripción se proporciona generalmente en un documento XML que responde al nombre de documento *Web Services Description Language* (WSDL).

- Los Servicios Web XML se registran de modo que los potenciales usuarios puedan encontrarlos. Esto se realiza mediante *Universal Discovery Description and Integration* (UDDI).

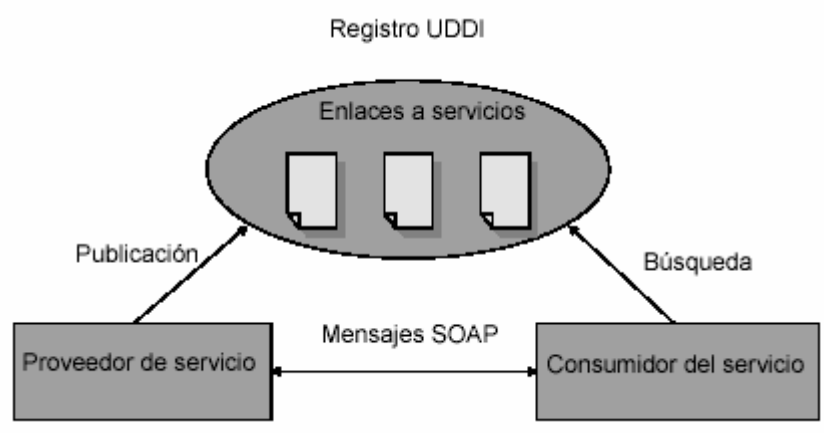
Una de las ventajas principales de la arquitectura de servicios Web XML es que permite a los programas escritos en diferentes lenguajes sobre diferentes plataformas comunicarse entre sí de un modo basado en estándares.

Una ventaja significativa que tienen los servicios Web XML sobre anteriores iniciativas es que trabajan con protocolos Web estándares: XML, HTTP y TCP/IP. Hemos definido un servicio Web XML como un servicio software expuesto en la Web mediante SOAP, descrito con un archivo WSDL y registrado en UDDI.

La mayoría de información está ya disponible en la Web, pero los servicios Web XML han que su acceso programático sea más fácil y fiable. Exponer las aplicaciones existentes como servicios Web XML permite a los usuarios construir nuevas y más potentes aplicaciones que utilicen servicios Web XML como bloques de construcción. Por ejemplo, un usuario podría desarrollar una aplicación de compras para obtener automáticamente la información de precios de varios fabricantes, que permitiera seleccionar un fabricante, enviar el pedido y a continuación realizar seguimiento del envío hasta que sea recibido. La aplicación del fabricante, además de exponer sus servicios en la Web, podría a su vez utilizar servicios Web XML para verificar el crédito del cliente, realizar un cargo en su cuenta y realizar el envío con una

empresa de transporte. En la figura se puede observar la arquitectura básica de un Web Service XML. Este esquema tiene miles de ventajas, entre otras el hecho de que cualquier plataforma es capaz de solicitar una página Web, leer los resultados e interactuar con este servicio, obteniendo así el resultado que nos interesa: Internet Paralelo sólo para máquinas.

Figura 4. Arquitectura Básica de un Web Service XML



Fuente: Mario Muñoz Organero. **Servicios Web**. Pagina 10.

2.3.1.3 Protocolos de Web Services

Existen 2 tendencias principales: el uso de XML-RPC y SOAP. Globus escoge SOAP:

- Soporte completo y minucioso
- Más potente
- Más difícil

2.3.2 WSDL Y ELEMENTOS FUNDAMENTALES

Las siglas WSDL significan *Web Services Description Language*. Para nuestro propósito, podemos decir que un archivo WSDL es un documento XML que describe un conjunto de mensajes y cómo se realiza el intercambio de mensajes. Como WSDL es XML, es legible y editable pero en la mayoría de casos, se genera y se consume por parte de software.

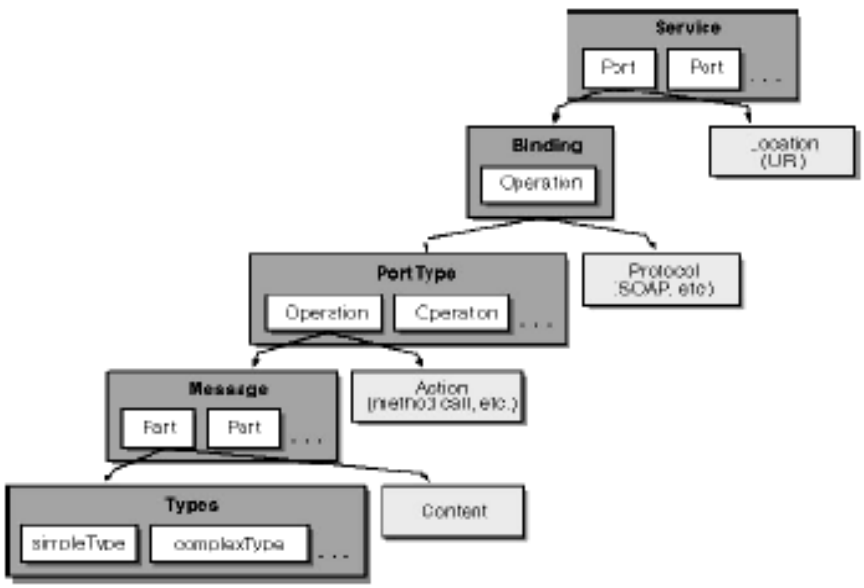
Para ver el valor de WSDL, imaginemos que queremos empezar invocando un método SOAP que proporciona uno de nuestros socios de negocio. Podríamos pedirle algunos mensajes SOAP de ejemplo y escribir nuestra aplicación para generar y consumir mensajes parecidos a los del ejemplo, pero ello puede ser propenso a error. Por ejemplo, podríamos ver un ID de cliente de 2837 y asumir que es un entero cuando de hecho es una cadena. WSDL especifica el contenido de un mensaje de petición y el aspecto del mensaje de respuesta en una notación inequívoca.

La notación que utiliza un archivo WSDL para describir formatos de mensajes está basada en el estándar XML *Schema* lo cual significa que es neutral respecto del lenguaje de programación y que está basado en estándares, lo que lo hace apropiado para describir interfaces de servicios Web XML accesibles desde una amplia variedad de plataformas y lenguajes de programación. Además de describir el contenido de los mensajes, WSDL define dónde está disponible el servicio y qué protocolo de comunicaciones utilizar para hablar con el servicio. Esto significa que el archivo WSDL define todo lo necesario para escribir un programa que interactúe con un servicio Web XML. Existen varias herramientas disponibles para leer un archivo WSDL y generar el código necesario para comunicarse con un servicio Web XML.

WSDL describe las operaciones ofrecidas por los Web Services con los siguientes elementos (Figura 5):

- <portType> Colección de operaciones de los métodos que definen el intercambio ordenado de los mensajes (comparable a librería de funciones). Cada tipo de puerto tiene su nombre.
- <message> Mensajes usados. Descripción de los datos que van a ser transmitidos. Son una colección de *"data values"* de un tipo particular. Son divisibles en partes (parámetros de función), cada mensaje tiene partes y cada parte tiene nombre y tipo.
- <types> Se usa para describir los tipos de datos usados. Definiciones de tipos de datos que son relevantes para el intercambio de mensajes.
- <binding> Protocolos de comunicación usados.

Figura 5. Componentes de un Web Service.



Fuente: Ian Mario Muñoz Organero. **(In) Servicios Web**. Pagina 15.

2.3.2.1 PORTS

Definen un punto de conexión a Web Services. “Port type” + “Binding”. Es una descripción abstracta de una acción soportada por el servicio. Cada operación se corresponde a un mensaje de *input* o de *output*.

Operaciones permitidas:

- *One-Way*: Recibir mensaje sin devolver respuesta.
- *Request-Response*: Recibir mensaje y devolver respuesta (la mas extendida).
- *Solicit-Response*: Enviar petición y esperar respuesta.
- *Notification*: Enviar mensaje sin esperar respuesta.

2.3.2.2 BINDING

Especifica los protocolos que usan cada port y el *encoding*. Sus atributos son:

- Nombre
- Tipos de atributos para port del *binding*

El tipo hace referencia al PortType. Se puede especificar opcionalmente información de *binding* para toda una operación así como para todo el elemento de *binding*.

Atributos soap: *binding*

- Estilo: [rpc | document]
- Transporte (HTTP)

Operation:

- Cada operación ofrecida por Port
- Se define acción SOAP

2.3.2.3 UDDI

Es el acrónimo de *Universal Description, Discovery and Integration*, es un servicio de directorio donde registrar Web Services (Sólo definidos con WSDL) y donde buscar Web Services.

La arquitectura importa. Escoger la estructura adecuada para una aplicación, especialmente una distribuida a través de varios sistemas (como Grid) es de gran importancia. Generalmente, una mala elección de arquitectura no puede arreglarse durante la implementación, con independencia de lo buenos que sean los desarrolladores. Tomar las decisiones incorrectas lleva a un menor rendimiento, menor seguridad y una menor cantidad de opciones cuando una aplicación deba ser actualizada.

Universal Discovery Description and Integration son las páginas amarillas de los servicios Web. Al igual que con las páginas amarillas tradicionales, podemos buscar una empresa que ofrezca los servicios que necesitamos, obtener información sobre el servicio ofrecido y contactar con alguien para más información. Naturalmente, también podemos ofrecer un servicio Web sin registrarlo en UDDI, al igual que podemos abrir un negocio en nuestro sótano y confiar en la publicidad boca a boca, pero si deseamos llegar a un mercado significativo, necesitamos UDDI para que los clientes puedan encontrarnos.

Una entrada de directorio UDDI es un archivo XML que describe un negocio y los servicios que ofrece. Una entrada en el directorio UDDI está formada por tres partes. Las "páginas blancas" describen la compañía que ofrece el servicio: nombre, dirección, contactos, etc. Las "páginas amarillas" incluyen categorías industriales basadas en taxonomías estándares como el *North American Industry Classification System* y el *Standard Industrial Classification*. Las "páginas verdes" describen el interfaz al servicio con suficiente detalle para alguien que desarrolle una aplicación que consuma el servicio Web. El modo en que se definen los servicios es mediante un documento UDDI llamado *Type Model* o tModel.

En muchos casos, tModel contiene un archivo WSDL que describe un interfaz SOAP a un servicio Web XML, pero tModel es suficientemente flexible para describir prácticamente cualquier tipo de servicio.

El directorio UDDI también incluye varias formas de buscar los servicios que necesitamos para construir nuestras aplicaciones. Por ejemplo, podemos buscar los proveedores de un servicio en una ubicación geográfica específica o un negocio de un tipo específico. El directorio UDDI proporcionará información, contactos, enlaces e información técnica para permitirnos evaluar qué servicios satisfacen nuestros requerimientos.

2.3.3 WSRF

Los Web Services permiten a los usuarios acceder y manipular estados y valores persistentes, lo que finalmente resulta en una interacción entre Web Services (Descubrimiento, Inspección y Manipulación). WSRF es el acrónimo para *Web Service Resource Framework* (Estándar de Globus) que propone una forma estándar de implementar el concepto de estado que es necesario mantener en aplicaciones distribuidas con un cierto grado de complejidad.

La base del (WSRF) es el WS-Resource, una entidad compuesta por un servicio Web y un recurso asociado a estados, descrito por un documento de XML (con el esquema sabido). WS-Resource Framework permite definir, crear, acceder a, monitorizar y destruir WS-Resources, empleando mecanismos convencionales de los Web Services, define las funciones que permiten interactuar con los WS-Resources como consultas, manejo del tiempo de vida, etc., WSRF se basa en la especificación de OGSI.

Un recurso asociado a estados se puede definir como un componente que tenga tres características:

- Esta compuesto por los datos de estado definidos en formato de XML.
- Tiene un ciclo vital.
- Puede ser manipulado por unos o más servicios Web.

Algunos ejemplos de estos recursos son archivos, objetos de Java, o filas en una base de datos. Estos recursos pueden ser compuestos (es decir, contienen otros recursos) y sus instancias se pueden crear o destruir mediante fábricas de servicios. Una instancia de un recurso asociado a estados se debe referenciar con un identificador de la identidad o del recurso; por otra parte, los recursos que usan un recurso pueden asignar identidades o alias adicionales.

Las relaciones entre los Web Services y los recursos asociados a estados se definen con el concepto de un patrón de recurso. Un patrón de recurso define los mecanismos que asocian un recurso a los intercambios de mensajes de un Web Service. Esta relación puede ser estática si un recurso se asocia a un servicio cuando está desplegado, o dinámica si el recurso es asociado en el intercambio del mensaje. Se ponen en ejecución los patrones del recurso usando estándares tales como XML, WSDL, y W-Address. Un recurso asociado a estados se puede utilizar en intercambios de mensajes entre Web Services. Los WS-Resources pueden ser creados y destruidos y su estado se puede preguntar o modificar mediante mensajes. Un WS-Resource tiene cuatro características muy importantes:

- Atomicidad: Las actualizaciones del recurso deben hacerse dentro de una unidad transaccional, se hacen en una manera, todo o nada.
- Consistencia: Los recursos deben siempre estar en un estado consistente incluso después de faltas.
- Aislamiento: Las actualizaciones a los recursos se deben aislar dentro de una unidad transaccional dada del trabajo.
- Durabilidad: prevé la permanencia de las actualizaciones hechas al recurso bajo una unidad transaccional del trabajo.

WSRF está dividido en 5 especificaciones que son las que se ven a continuación con detalle.

2.3.3.1 WS - ResourceLifetime

Son los mecanismos para la destrucción de los WS-Resources, incluyendo mecanismos que permiten asignar un momento para dicha destrucción.

2.3.3.1.1 El patrón WS-Resource Factory

Se basa en el patrón de diseño Factory estándar que tiene las siguientes ventajas para los objetos de este tipo:

- Separación de la responsabilidad de la creación compleja en objetos de apoyo (*helper*) cohesivos. Según las características del objeto que se quiere instanciar, se hace una llamada a la clase *helper* asociada.
- Ocultación de la lógica de creación potencialmente compleja.
- Posibilidad de introducir estrategias para mejorar el rendimiento de la gestión de memoria, como objetos caché y reciclaje.

Este patrón de diseño pretende resolver el problema de quién es el responsable de la creación de objetos cuando existen consideraciones especiales, dando como solución la creación de un objeto de Fabricación pura que maneje la creación.

Volviendo al WSRF, se define un WS-Resource Factory como cualquier Web Service capaz de crear uno o más WS-Resources.

2.3.3.1.2 Identidad del WS-Resource

El componente del recurso asociado a estado de un WS-Resource se identifica. Esto es realizado por el componente Web Service, que incluye un

identificador del recurso en las propiedades de la denominada referencia del punto final de WS-Addressing. Así, este punto final pasa a denominarse un “WS-Resource Calificado”, que puede hacerse disponible a otras entidades en un sistema distribuido.

El concepto de “WS-Resource Calificado” es completamente opaco, ya que no está en su mano el examen o la interpretación de los identificadores de un WS-Resource.

2.3.3.1.3 Destrucción del WS-Resource

Cualquiera que pida un WS-Resource estará solamente interesado en su uso durante un periodo limitado. A continuación veremos los dos tipos de destrucción de WS-Resource ofrecidos por el WSRF.

- Inmediata: Cualquier receptor de servicios que quiera destruir un WS-Resource inmediatamente, no tiene más que utilizar el punto final asociado al “WS-Resource Calificado” y mandarle una petición de destrucción. El resultado no se hace esperar, y el WS-Resource termina definitivamente.

- Programada: Un WS-Resource Factory puede dar la habilidad de negociar la terminación programada de un WS-Resource en tiempo de creación. En añadidura, el receptor del servicio puede usar el punto final asociado al “WS-Resource Calificado” para cambiar el tiempo para el cual estaba programada la destrucción, tanto para aumentarlo como disminuirlo.

2.3.3.2 WS - Resource Properties

Se trata de la definición de un WS-Resource, así como mecanismos para obtener, cambiar y borrar propiedades de un WS-Resource en particular.

La especificación de las propiedades de un WS-Resource define el tipo y el valor de aquellos componentes del estado de un WS-Resource que pueden verse y modificarse por los receptores del servicio, a través del interfaz del Web Service. Estas son las ideas principales:

El WS-Resource tiene un “Documento de Propiedades” definido usando el esquema XML. Los receptores del servicio pueden determinar el tipo de un WS-Resource, la definición “portType usando WSDL” de forma estándar (El portType no es mas que la definición de un conjunto de operaciones estándar, o lo que es lo mismo, qué funciones realiza un servicio).

Los receptores del servicio pueden usar intercambio de mensajes de los Web Services para leer, modificar y pedir el documento XML que representa el estado del WS-Resource.

El término “Propiedad de un Recurso” se emplea para designar un componente individual del estado de un WS-Resource, y tiene su propio documento XML para su descripción.

2.3.3.2.1 Documento de Propiedades del WS-Resource

Concretamente, la información se expresa empleando una XML *global element declaration* (GED). Supongamos un estado C que afecta a 3 componentes de un recurso que llamaremos p1, p2 y p3. Entonces, el

documento asociado a las propiedades del recurso, que llamaremos "EjemploPropiedadesRecurso", podría ser:

```
<xs:schema
targetNamespace="http://example.com/EjemploPropiedadesRecurso"
xmlns:tns="http://example.com/EjemploPropiedadesRecurso"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
…
... >
<xs:element name="p1" type= … />
<xs:element name="p2" type= … />
<xs:element name="p3" type= … />
<xs:element name="EjemploPropiedadesRecurso">
<xs:complexType>
<xs:sequence>
<xs:element ref="tns:p1" />
<xs:element ref="tns:p2" />
<xs:element ref="tns:p3" />
</xs:sequence>
</xs:complexType>
</xs:element>
…
</xs:schema>
```

Los receptores del servicio pueden descargar y examinar esta definición XML, que representa el tipo del recurso asociado a estados C.

El receptor sabe que el GED llamado "EjemploPropiedadesRecurso" define el documento de propiedades del recurso del WS-Resource por vía de la declaración del portType en el Web Service.

```
<wsdl:definitions
targetNamespace="http://example.com/EjemploPropiedadesRecurso"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsrp=
"http://www.ibm.com/xmlns/stdwip/web-services/ws-resourceProperties"
xmlns:tns="http://example.com/EjemploPropiedadesRecurso"
...>
...
<wsdl:types>
<xs:schema>
<xs:import
namespace="http://example.com/EjemploPropiedadesRecurso"
schemaLocation="..."/>
</xs:schema>
</wsdl:types>
...
<wsdl:portType name="NombreDeAlgunPortType"
wsrp:ResourceProperties="tns:EjemploPropiedadesRecurso" >
<operation name="...
...
</wsdl:portType>
...
</wsdl:definitions>
```

2.3.3.2.2 Composición de propiedades del WS-Resource

Tener solamente una interfaz de Web Service no resulta productivo. Es por ello que en el estándar WSDL, el diseñador puede añadir uno sencillamente mediante un copiar y pegar de las operaciones definidas en los portTypes ya existentes, de tal manera que se obtendría una composición. Dicha composición también puede afectar a las propiedades del WS-Resource.

En este ejemplo, vemos la modalidad de agregación empleando el atributo xs:ref.

```
<xs:element name="EjemploPropiedadesRecurso">
<xs:complexType>
<xs:sequence>
<xs:element ref="tns:p1" />
<xs:element ref="tns:p2" />
<xs:element ref="tns:p3" />
<xs:element ref="xxxx:AlgunaPropiedadRecursoAdicional"
xmlns:xxxx= ... />
</xs:sequence>
</xs:complexType>
</xs:element>
```

2.3.3.2.3 Accediendo a los valores de las Propiedades del WS-Resource

El estado de un WS-Resource puede leerse y/o modificarse a través del documento de Propiedades, empleando mensajes Web Service. Este

intercambio de mensajes tiene que estar incluido como operaciones WSDL en cualquier portType que use wsrp:ResourceProperties. La función más básica consiste en obtener el valor de una única propiedad de un recurso empleando un intercambio de mensajes sencillo (petición/respuesta). Más complejamente, podemos encontrar una variante que nos permite obtener el valor de múltiples propiedades a la vez.

Estas operaciones están recogidas en la operación "*get*" de WS-ResourceProperties.

<wsrp:GetMultipleResourceProperties>
<wsrp:ResourceProperty> Nombre <wsrp:ResourceProperty>+
</wsrp:GetMultipleResourceProperties>

La respuesta a este mensaje es una secuencia de elementos XML correspondientes a los valores de las propiedades WS-Resources identificadas por los nombres especificados en el mensaje de petición.

Otro ejemplo nos permite ver la petición del valor de la propiedad p1 de un recurso asociado a estados C.

<soap:Envelope>
<soap:Header>
<tns:resourceID> C </tns:resourceID>
</soap:Header>
<soap:Body>
<wsrp:GetMultipleResourceProperty>
<wsrp:ResourceProperty>tns:p1</wsrp:ResourceProperty>
</wsrp:GetMultipleResourceProperty>
</soap:Body>
</soap:Envelope>

La referencia al punto final usada para designar el objetivo de esta petición contiene un identificador en el componente *ReferenceProperties* que identifica el recurso C. En este caso, observamos que la información del identificador está en la cabecera SOAP, cumpliendo la codificación estándar. Si seguimos con el ejemplo, veremos que el WS-Resource respondería con los valores de p1.

<soap:Envelope>
<soap:Body>
<wsrp:GetMultipleResourcePropertyResponse>
<p1>xyz</p1>
</wsrp:GetMultipleResourcePropertyResponse>
</soap:Body>
</soap:Envelope>

Por otro lado, también existe una operación "*set*" que permite insertar, actualizar y borrar valores de las propiedades del WS-Resource. Aquí podemos ver una psuedo-sintaxis para el "set":

<wsrp:SetResourceProperties>
{
<wsrp:Insert >
{cualquier valor a insertar}*
</wsrp:Insert> |
<wsrp:Update >
{ cualquier valor a insertar }*
</wsrp:Update> |
<wsrp:Delete ResourceProperty=îNombreî />
}+
</wsrp:SetResourceProperties>

Hay otra operación que permite al receptor de servicios hacer una búsqueda. Un uso de esto bien sería la búsqueda de recursos, empleando búsquedas masivas, y basándose en los valores del estado de los WS-Resources. Y finalmente, existe una ultima operación, no menos interesante, a través de la cual, un receptor puede recibir notificaciones en caso de que algún valor de las propiedades del WS-Resource cambiara.

2.3.3.3 WS – RenewableReferences

Define una ampliación del WS-Addressing empleando políticas. Entre otras funciones, define mecanismos que se pueden usar para renovar la referencia de un punto final que se ha vuelto inválido, bastante útil en caso de que dicho punto final haga referencia a un WS-Resource. Una referencia a un punto final de WS-Addressing puede contener además información sobre políticas relativas a las interacciones con el servicio. Está claro que la información recibida por el cliente acerca de una política es susceptible de cambio, con lo que sería muy importante que la referencia al punto final se pudiera renovar, y además, debería existir una información que le guiara a la hora de obtener una nueva referencia en caso de invalidez de la actual.

2.3.3.4 WS – ServiceGroup

Se trata de un interfaz para colecciones heterogéneas de Web Services. Los grupos creados por esta especificación se pueden definir como una colección de miembros que igualan el valor de ciertas variables, expresadas en el contexto de propiedades de recurso.

Los interfaces definidos por WS-ServiceGroup deberían poder componerse con otros interfaces de Web Services.

2.3.3.5 WS – BaseFaults

Es una tipificación XML para ser usada cuando aparecen fallos en el intercambio de mensajes asociados a los Web Services.

2.3.3.6 WS – Notification

WS-Notification es una familia de especificaciones separada. Define un sistema Web Service para publicar y suscribirse a las notificaciones relacionadas con las interacciones en el *Framework*. Estas operaciones son consideradas de tipo general y asociadas a un tema, de tal manera que el cliente pueda tener un abanico mayor de funcionalidades. Además, se ofrece una perspectiva de bloques para representar y estructurar notificaciones. Existen 3 subespecificaciones:

- WS-BaseNotification: Describe los roles básicos, conceptos y patrones requeridos para permitir a un suscriptor registrarse.
- WS-Topics: Presenta una descripción XML de los temas y meta datos asociados a las notificaciones.
- WS-BrokeredNotification: Define el interfaz de un "*NotificationBroker*" que implanta un servicio intermediario para administrar suscripciones para otras entidades productoras de mensajes de notificación.

2.3.4 Características de los Grid Services

- *Stateful*: El estado del Grid Service se mantiene de una invocación a otro.
- Puede ser transiente: Podemos crear varias instancias de un mismo Grid Service *"onthe-fly"* y destruirlas cuando ya no son necesarias.

- Servicios de apoyo
- Novedad: Los Grid Services utilizan un enfoque de “Factorías de objetos”.
- En lugar de tener un único servicio sin estado, compartido por todos los usuarios (Web Service), tenemos un servicio-factoría para crear instancias individuales del servicio (Grid Service).
- Las instancias se crean a través de la factoría.
- Cuando queremos invocar una operación del servicio, accedemos a la instancia no a la factoría.
- Podemos tener una instancia por cliente, varias instancias por cliente, varios clientes por instancia, ...
- La destrucción de la instancia puede correr a cargo del cliente o de la factoría.

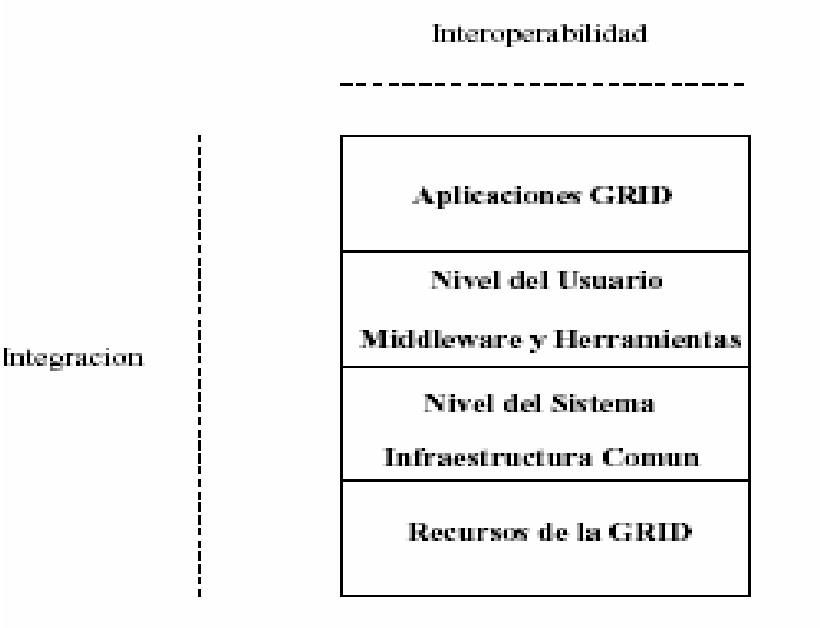
2.4 OGSA y OGSI

La arquitectura OGSA (*Open Grid Services Architecture*) es un conjunto de protocolos y normas estándar abiertos que permiten las comunicaciones entre entornos heterogéneos y geográficamente dispersos. La arquitectura Grid Service está especificada por el Global Grid Forum y se ha convertido en el estándar de facto para las infraestructuras Grid actuales. Permite gestionar servicios de red Grid basados en las normas estándar OGSI (del inglés *Open Grid Services Infrastructure*, infraestructura de servicios de red abierta).

Dicho de otra manera OGSA es una arquitectura de servicios para aplicaciones Grid que se basa en OGSI, que es el estándar que propone un modelo de Web Services mejorados (Grid Services) y cuya implementación se da a partir del GT3.

Para facilitar la utilización y la implementación de esta infraestructura distribuida es necesario establecer una arquitectura global orientada hacia los servicios que simplifique el desarrollo y la puesta en operación de las aplicaciones que harían uso de estos servicios. Se ha definido el modelo OGSA que está constituido de 4 capas, como se muestra en la figura:

Figura 6. Modelo OGSA del Grid Computing.



Fuente: J. Verduzco, N. García y otros. **(In) Grid - ITC**. Pagina 3.

- OGSA *Infrastructure*. Define la infraestructura que conforma el GRID. Incluye computadoras, Cluster, redes de datos e instrumentos científicos.
- OGSA *Services*. Proporciona los servicios necesarios para acceder con seguridad recursos remotos gestionando de manera eficiente la heterogeneidad. Los servicios proporcionados son

seguridad, almacenamiento, información sobre el estado de los recursos y sumisión de trabajos remotos.

- OGSA *Schemes*. Proporciona herramientas y ambientes adaptativos de ejecución en un nivel de usuario que facilitan el acceso a los servicios del GRID.

El objetivo de OGSA es el proveer métodos uniformes para descubrir, acceder e integrar recursos de datos heterogéneos. Los métodos de envío de datos admitidos son SOAP cliente, a fichero, Streams HTTP(S), y GridFTP y FTP a servidor separado. GT utiliza tres herramientas de servicios:

- *Globus Resource Allocation Manager* (GRAM)
- *Metacomputing Directory Service* (MDS)
- *Grid File Transfer Protocol* (GridFTP)

2.4.1 Reliable File Transfer (RFT)

Este servicio tiene un rol similar al planificador *batch* (por lotes) en una máquina. Se le envía un trabajo de transferencia de datos, que podría consistir en 10000 o 100000 ficheros, con la información de los tamaños de buffer. La petición, junto con la información necesaria para reiniciar (en el caso de ser necesario) y las notificaciones, son almacenadas en una base de datos. El servicio entonces ejecuta una transferencia GridFTP a terceros por el usuario.

2.4.2 WS Grid Resource Allocation and Management (WS GRAM)

WS GRAM ofrece un único interfaz para pedir y usar recursos remotos para la ejecución de trabajos. GT4 incorpora hasta 2 implantaciones de GRAM:

- Basada en un protocolo Pre-WS no propietario (Pre-WS GRAM).

- Construida usando interfaces WS (WS GRAM).

2.4.3 WS Monitoring and Discovery System (MDS4)

Se trata de un monitor de nivel de Grid basado en WSRF, usado para descubrimiento y manejo de recursos. Sirve para obtener un gran rango de información relativa a los recursos básicos y colas, y puede servir de interfaz para monitorización de sistemas cluster. Cada servicio basado en WSRF ofrece bastante información de monitorización sobre sí mismo, de tal manera que permite a WS MDS usar sus datos.

Igual que cualquier sistema de monitorización Grid, MDS4 ofrece un servicio de indexado donde los datos son recogidos y almacenados. Además, MDS4 almacena información en la base de datos Xindice, basada en XML, y tiene un interfaz Web para poder ver los datos fácilmente.

3. SEGURIDAD EN LA COMPUTACIÓN GRID

Las necesidades y problemas de seguridad en los sistemas Grid surgen precisamente por las características propias de los entornos de computación Grid:

- La población del usuario es extensa y dinámica. Los participantes de las organizaciones virtuales pueden incluir miembros de diversas instituciones, las cuales, es muy probable que cambien de manera frecuente.
- El servicio de recursos es extenso y dinámico. Debido a que los usuarios y las instituciones individuales son quienes deciden cuando van a contribuir con recursos, la cantidad y la localización de los recursos disponibles puede cambiar de manera muy rápida.
- Una computación (o proceso creado por una computación), durante su ejecución, puede adquirir recursos, comenzar procesos sobre los recursos, y liberar recursos, todo ello de forma dinámica.
- Los procesos que forman una computación pueden realizar comunicaciones utilizando diversos mecanismos, que incluyen *unicast* y *multicast*. Mientras esos procesos forman una entidad única, totalmente conectada de manera lógica, durante la ejecución del programa, se pueden crear y destruir, dinámicamente, conexiones de comunicación a bajo nivel (ej., *sockets* TCP/IP).
- Los recursos pueden requerir diferentes mecanismos y políticas de autenticación y autorización (Kerberos, contraseñas de texto en claro, *Secure Socket Library* (SSL),…).
- Un usuario individual puede tener asociado diferentes espacios locales de nombres, credenciales, o cuentas, en diferentes sitios, para los propósitos de control de acceso y tarificación.
- Los recursos y los usuarios se pueden localizar en diferentes países.

Como se puede apreciar, se hace necesario proporcionar soluciones de seguridad que permitan computaciones en un entorno con las características que se acaban de mencionar. Se deben coordinar las diversas políticas de control de acceso y se debe operar de manera segura en los entornos heterogéneos.

3.1 Requerimientos de Seguridad

Los sistemas Grid pueden requerir de algunas o todas las funciones estándares de seguridad, incluyendo autenticación, control de acceso, integridad, privacidad y no repudio. Una arquitectura de seguridad puede centrarse principalmente en los aspectos de autenticación y control de acceso:

1) Se deben proporcionar soluciones de autenticación que admiten un usuario, los procesos que constituyen una computación del usuario, y los recursos utilizados por esos procesos, para verificar la identidad de cada uno. La autenticación forma la base de una política de seguridad que permite diversas políticas de seguridad locales para que sean integradas dentro de un marco global.

2) Se deben permitir mecanismos de control de acceso local para que sean aplicados sin cambios, siempre que sea posible.

3.1.1 Restricciones

Para poder desarrollar la arquitectura de seguridad que cumpla los requerimientos anteriores, se deben satisfacer, además, las siguientes restricciones que se derivan de las características del entorno y de las aplicaciones Grid:

- Firma única. Un usuario debería se capaz de autenticarse una única vez, por ejemplo, cuando se comienza la computación, y poder iniciar computaciones que adquieren recursos, utilizar recursos, liberar recursos, tener comunicaciones internas; todo ello sin necesidad de autenticación por parte del usuario.
- Protección de credenciales. Se deben proteger las credenciales de los usuarios (contraseñas, claves privadas, etc.).
- Interoperabilidad con soluciones de seguridad local. Mientras que las soluciones de seguridad pueden proporcionan mecanismos de acceso entre dominios, el acceso a recursos locales será determinado, de forma típica, por una política de seguridad local que se hace cumplir por un mecanismo de seguridad local. Resulta poco práctico modificar cada recurso local para que se acomode al acceso entre dominios; en vez de eso, una o más entidades en un dominio (ej., los servidores de seguridad entre dominios) deben actuar como agentes de clientes/usuarios remotos para los recursos locales.
- Infraestructura uniforme de credenciales/certificación. El acceso entre dominios requiere, como mínimo, una manera común de expresar la identidad de un principal de seguridad, como sería un usuario o un recurso real.
- Portabilidad. Se requiere que el código sea a) portable y b) ejecutable en *testbeds* (casos de prueba) multinacionales. En resumen, los aspectos de portabilidad indican que la política de seguridad no puede requerir, directa o indirectamente, la utilización de cifrado en masa.
- Soporte para comunicación segura en grupo. Una computación puede estar constituida por un número de procesos que necesitarán coordinar sus actividades como un grupo. La composición de un grupo de procesos puede y de hecho cambiará durante el periodo de vida de una

computación. Por tanto, se necesita soporte para una comunicación segura (en este contexto, autenticada) para los grupos dinámicos.

- Soporte para múltiples implementaciones. La política de seguridad no debería imponer una tecnología de implementación específica. Todo lo contrario, debería ser posible implementar la política de seguridad dentro de un rango de tecnologías de seguridad, basadas todas ellas tanto en la criptografía de clave pública como en la criptografía de clave secreta.

3.2 Política de Seguridad

Antes de ahondar en los aspectos específicos de una arquitectura de seguridad, es importante identificar los objetivos de seguridad, las entidades participantes y las suposiciones fundamentales. En resumen, se debe definir una política de seguridad, un conjunto de normas que definan los sujetos de la seguridad (ej., usuarios), objetos de seguridad (ej., recursos) y las relaciones entre ellos. Como son posibles diferentes políticas de seguridad, se presenta una política específica que trata los aspectos introducidos anteriormente a la vez que refleja las necesidades y las expectativas de las aplicaciones, los usuarios y los propietarios de los recursos.

3.2.1 Terminología

Se considera la siguiente terminología de seguridad:

- Sujeto. Un sujeto es un participante en una operación de seguridad. En los sistemas Grid, un sujeto suele ser normalmente un usuario, un proceso que opera en nombre de un usuario, un recurso (ordenador, archivo,…), o un proceso que actúa en nombre de un recurso.
- Credencial. Una credencial es una porción de información que se utiliza para probar la identidad de un sujeto. Las contraseñas y los certificados son ejemplos de credenciales.

- Autenticación. Autenticación es el proceso por el cual un sujeto prueba su identidad a un interlocutor, normalmente a través de la utilización de su credencial. La autenticación en la cual ambas partes, es decir, los dos interlocutores, se autentican simultáneamente (cada parte se autentica a la otra parte) se conoce como autenticación mutua.
- Objeto. Un objeto es un recurso que está siendo protegido por la política de seguridad.
- Autorización. Autorización es el proceso por el cual se determina si un sujeto tiene permitido el acceso o puede utilizar un objeto.
- Dominio de confianza. Un dominio de confianza es una estructura lógica, administrativa dentro de la cual se mantiene una política de seguridad local, única y consistente. Dicho de otra manera, un dominio de confianza es una colección de los objetos y de los sujetos gobernados por una administración única y por una política de seguridad única.

3.2.2 Definición

De acuerdo a la terminología anterior, se define la siguiente política de seguridad:

1) El entorno Grid consiste en múltiples dominios de confianza. Este primer elemento establece que la política de seguridad del Grid debe integrar un conjunto heterogéneo de usuarios y recursos administrados de manera local. En general, el entorno Grid tendrá limitación o no tendrá ninguna influencia sobre la política de seguridad local. Por tanto, no se puede pedir que se sustituyan las soluciones locales, y no se pueden sobrescribir las decisiones de la política local. Como consecuencia, la política de seguridad del Grid se debe enfocar en controlar las interacciones entre los dominios y mapear las operaciones entre dominios a la política de seguridad local.

2) Las operaciones que están limitadas a un único dominio de confianza sólo están sujetas a la política de seguridad local. En este caso no se requiere ningún servicio u operación de seguridad adicional. La política de seguridad local se puede implementar con distintos métodos que incluyen firewalls, Kerberos y SSH.

3) Existen sujetos locales y globales. Para cada dominio de confianza, existe un mapeo parcial de los sujetos globales a locales. Cada usuario de un recurso tendrá dos nombres, un nombre global y un nombre local potencialmente diferente en cada recurso. El mapeo de un nombre global a un nombre local es específico del sitio. Por ejemplo, un sitio podría mapear nombre de usuarios globales a: un nombre local predefinido, un nombre local asignado de manera dinámica, o un nombre "único" de grupo. La existencia del sujeto global permite a la política de seguridad proporcionar firma única.

4) Las operaciones entre entidades localizadas en dominios de confianza diferentes requieren una autenticación mutua.

5) Se asume que un sujeto global autenticado que se mapea a un sujeto local serán equivalentes. De este modo, podrá ser autenticado localmente como ese sujeto local. Dicho de otra manera, dentro de un dominio de confianza, la combinación de la política de autenticación del Grid y el mapeo local cumplen el objetivo de seguridad del dominio del *host*.

6) Todas las decisiones de control de acceso se realizan de manera local en base al sujeto local. Esta política requiere que las decisiones de control de acceso se mantengan bajo la supervisión de los administradores del sistema local.

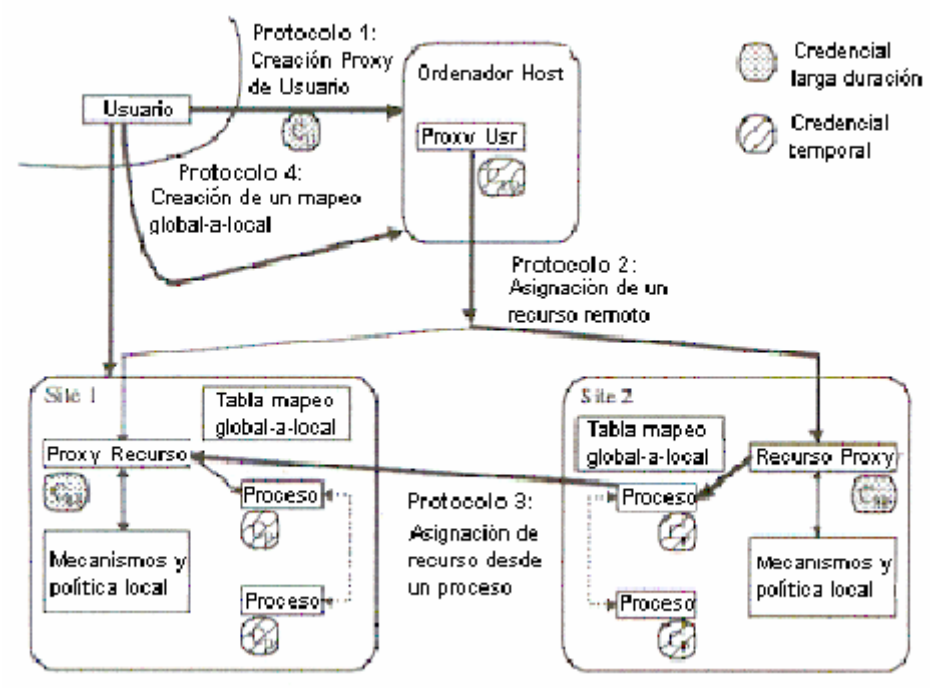
7) Se permite que un programa o proceso actúe en nombre de un usuario y que se le delegue un subconjunto de los derechos del usuario. Este elemento es imprescindible ya que se necesita soportar la ejecución de programas de larga duración que pueden adquirir recursos de manera dinámica sin la interacción adicional del usuario. Además, también se necesita este elemento para poder soportar la creación de procesos por otros procesos.

8) Los procesos que se ejecutan en nombre del mismo sujeto dentro del mismo dominio de confianza pueden compartir un conjunto único de credenciales. Las computaciones Grid pueden implicar a cientos de procesos en un único recurso. Este elemento de la política de seguridad permite escalabilidad de la arquitectura de seguridad para aplicaciones paralelas a gran escala, evitando la necesidad de crear una credencial única para cada proceso.

3.3 ARQUITECTURA DE SEGURIDAD

Con la política de seguridad que se acaba de describir, se proporciona un contexto dentro del cual resulta posible construir una arquitectura específica de seguridad. Se va a especificar un conjunto de sujetos y objetos que se encontrarán bajo la jurisdicción de la política local y se van a definir los protocolos que controlarán las interacciones entre estos sujetos y objetos. La figura muestra una visión de esta arquitectura. En dicha figura se representan los siguientes componentes: entidades, credenciales y protocolos. Las líneas gruesas representan los protocolos, la línea curva que separa al usuario del resto de la figura indica que el usuario puede desconectar una vez que se ha creado el *proxy* del usuario y las líneas punteadas representan la comunicación autenticada entre procesos. Un *host* es un ordenador que proporciona estaciones cliente con acceso a ficheros e impresoras como recursos compartidos de una red de ordenadores.

Figura 7. Arquitectura de Seguridad de una Grid Computacional.



Fuente: María Cruz Valiente Blázquez. **(In) Sistemas Distribuidos – Seguridad en entornos Grid**. Pagina 13.

Como el interés para este trabajo se centra en entornos computacionales, los sujetos y los objetos de la arquitectura deben incluir esas entidades a partir de las cuales se forma la computación. Una computación va a estar compuesta por varios procesos, donde cada proceso actúa en nombre del usuario. De este modo, los sujetos serán usuarios y procesos. Los objetos en la arquitectura deben incluir un rango amplio de recursos que están disponibles en el entorno Grid: ordenadores, repositorios de datos, redes, dispositivos de visualización, etc.

Las computaciones Grid pueden crecer y disminuir de manera dinámica, adquiriendo los recursos cuando necesiten resolver un problema y liberándolos cuando ya no se necesitan. Cada vez que una computación obtiene un recurso, lo hace en nombre de un usuario en particular. Si embargo, de cara a la autenticación, normalmente resulta poco práctico para el usuario interactuar directamente con cada recurso: el número de recursos implicados puede ser muy grande, o, como algunas aplicaciones se pueden ejecutar durante un periodo de tiempo excesivamente largo (días o semanas), el usuario puede preferir que una computación opere de manera individual sin su intervención.

Por tanto, se introduce el concepto de *proxy* de usuario que puede actuar en nombre del usuario sin requerir su intervención. El *proxy* de usuario actúa como sustituto del usuario. Tiene sus propias credenciales, elimina la necesidad de tener al usuario *on-line* durante una computación y elimina la necesidad de tener disponibles las credenciales del usuario para cada operación de seguridad. Además, debido a que el tiempo de vida del *proxy* está bajo el control del usuario y puede estar limitado a la duración de una computación, las consecuencias de comprometer sus credenciales no son tan extremas que si se exponen las credenciales de los usuarios. Un *proxy* de usuario es un proceso manager que tiene permiso para actuar en nombre de un usuario durante un periodo limitado de tiempo.

Dentro de la arquitectura, se define además una entidad que representa un recurso, el *proxy* de recurso, que sirve como interfaz entre la arquitectura de seguridad del Grid y la arquitectura de seguridad local.

Cuando se tiene un conjunto de objetos y sujetos, la arquitectura se determina especificando los protocolos que se utilizan cuando interactúan los objetos y los sujetos. En la definición de los protocolos se va a utilizar la siguiente notación:

- U, R, P: Se referirán, respectivamente, a usuario, recurso y proceso.
- UP, RP: Se referirán, respectivamente, a *proxy* de usuario y *proxy* de recurso.
- $C_{X:}$ Se referirá a la credencial del sujeto X. Muchos de los protocolos cuentan con la capacidad de afirmar que una porción de datos proviene, sin modificación, de una fuente conocida X. Esta condición será verdadera siempre y cuando dicha porción de datos esté firmada por X (firma electrónica reconocida).
- Sig_X{texto}. Indicará que el "texto" está firmado por el sujeto X.

El rango de interacciones que se puede dar entre entidades en un Grid computacional se define por la funcionalidad del sistema Grid subyacente. Sin embargo, basándose en la experiencia y en los sistemas Grid actuales que han sido desarrollados hasta la fecha, resulta razonable suponer que el sistema Grid va a incluir las siguientes operaciones:

- Asignación de un recurso a un usuario, es decir, creación del proceso.
- Asignación de un recurso a un proceso.
- Comunicaciones entre procesos localizados en diferentes dominios de confianza.

A continuación, se deben definir protocolos que controlen las interacciones UP-RP, P-RP y P-P. Además, la introducción del *proxy* de usuario lleva a que se debe establecer como interactúa el usuario y el *proxy* de usuario (U-UP). A la vista de la arquitectura, se cumple este requerimiento permitiendo al usuario hacer "*log on*" en el sistema Grid, creando un *proxy* de usuario utilizando el protocolo 1. El *proxy* de usuario puede entonces asignar recursos (y por tanto, crear procesos) utilizando el protocolo 2. Utilizando el protocolo 3, un proceso creado puede asignar, directamente, recursos adicionales. Por último, el protocolo 4 se puede utilizar para definir un mapeo de un sujeto global a uno local.

3.3.1 Protocolo 1: Creación del Proxy de Usuario

Un *proxy* de usuario es una entidad que actúa en nombre de un usuario. En la práctica, el *proxy* de usuario es un proceso especial iniciado por el usuario que se ejecuta en algún *host* local a ese usuario. El tema principal en el protocolo de creación del *proxy* de usuario es la naturaleza de las credenciales dadas al *proxy* y cómo el *proxy* puede obtener estas credenciales.

Un usuario podría permitir a un *proxy* actuar en su nombre dando a dicho *proxy* las credenciales apropiadas, por ejemplo, una clave privada o una contraseña. De esta manera, el *proxy* podría utilizar esas credenciales directamente. Sin embargo, esta aproximación tiene dos desventajas muy significativas: introduce un riesgo creciente al comprometer las credenciales y no permite restringir el tiempo durante el cual el *proxy* puede actuar en nombre del usuario. En lugar de eso, lo que se puede hacer es generar una credencial temporal, C_{UP}, para el proxy de usuario; el usuario le indica su permiso firmando esta credencial con un secreto, por ejemplo, clave privada. C_{UP} incluye el intervalo de validez, así como otras restricciones impuestas por el usuario, por ejemplo, nombres de *host* (desde donde se permite operar al *proxy*) y sitios objetivo (donde se permite al *proxy* iniciar procesos y/o utilizar recursos). En la creación del *proxy* de usuario se deben seguir los siguientes pasos:

1. El usuario consigue acceso al ordenador donde se ha creado el *proxy* de usuario, utilizando cualquier forma de autenticación local que se haya dispuesto en ese equipo.

2. El usuario crea la credencial del *proxy* de usuario, C_{UP}, utilizando su credencial, C_U, para firmar una tupla que contiene la identificación del usuario, el nombre del *host* local, el intervalo de validez para C_{UP}, y cualquier otra información que sea requerida para el protocolo de

autenticación utilizado para implementar la arquitectura (por ejemplo, una clave pública si se utiliza una autenticación basada en certificados):
C_{UP} = Sig_U { id-usr, *host*, hora-inicio, hora-fin, info-autenticación,…}.

3. Se crea un proceso de *proxy* de usuario que se proporciona con C_{UP}. Es responsabilidad de la política de seguridad local proteger la integridad de C_{UP} en el ordenador en el cual se localiza el *proxy* de usuario.

3.3.2 Protocolo 2: Asignación de Recursos

En este caso hay que considerar dos clases: asignación de recursos por un *proxy* de usuario y asignación de recursos por un proceso. En este protocolo se considerará la primera.

Todas las operaciones asociadas a los recursos están controladas por una entidad, llamada *proxy* de recurso, la cual es responsable de planificar el acceso a un recurso y de mapear una computación en ese recurso. Su funcionamiento es el siguiente: un *proxy* de usuario que requiere acceso a un recurso, en primer lugar determina la identidad del *proxy* de recurso para ese recurso y entonces le envía una petición. Si la petición tiene éxito, se le asigna el recurso y se crea un proceso en ese recurso. La petición puede fallar si el recurso solicitado no está disponible (fallo de asignación), si no se reconoce al usuario (fallo de autenticación), o si el usuario no está autorizado a utilizar el recurso en el modo que ha solicitado (fallo de autorización). Es responsabilidad del *proxy* de usuario hacer cumplir todos los requerimientos locales de autorización. Dependiendo de la naturaleza del recurso y de la política local, la autorización se puede comprobar en el momento de la asignación del recurso o en el momento de la creación del proceso, o bien, puede estar implícito en la autenticación, en cuyo caso no se realiza ninguna comprobación.

Se define, por tanto, como Protocolo 2 el mecanismo utilizado para realizar una petición a un *proxy* de recurso desde un *proxy* de usuario. En la creación del *proxy* de recursos (y creación del proceso) se deben seguir los siguientes pasos:

- El *proxy* de usuario y el *proxy* de recurso se autentican mutuamente utilizando, respectivamente, C_{UP} y C_{RP}. Como parte de este proceso, el *proxy* de recurso se asegura de que no han caducado las credenciales del *proxy* de usuario.

- El *proxy* de usuario envía una petición firmada al *proxy* de recurso de la forma Sig_{UP} (especificación-asignación).

- El *proxy* de recurso comprueba si el usuario que ha firmado las credenciales del *proxy* está autorizado por la política local para hacer la petición de asignación.

- Si la petición puede llevarse a cabo, el *proxy* de recurso crea una tupla CREDENCIALES-RECURSO, que contiene el nombre del usuario a quien se le ha asignado el recurso, el nombre del recurso, etc.

- El *proxy* de recurso pasa de manera segura la tupla CREDENCIALES-RECURSO al *proxy* de usuario (esto es posible gracias a la operación realizada en el paso 1.).

- El *proxy* de usuario examina la petición CREDENCIALES-RECURSO y, si lo aprueba, firma la tupla para generar C_P, una credencial para el recurso solicitado.

- El *proxy* de usuario pasa C_P de manera segura al *proxy* de recurso.

- El *proxy* de recurso asigna el recurso y pasa el nuevo proceso(s) C_P (la última transferencia confía en el hecho de que el *proxy* de recurso y los procesos están en el mismo dominio de confianza).

La verificación llevada a cabo en el paso 3, puede requerir un mapeo de las credenciales del usuario a un identificador de usuario local o a un nombre de cuenta si la política del *proxy* de recurso es comprobar la autorización en el momento de la asignación del recurso.

El protocolo crea una credencial temporal para los nuevos procesos creados. Esta credencial, C_P, le da al proceso la capacidad de autenticarse por sí solo y la capacidad de identificar al usuario en cuyo nombre fue creado el proceso. La asignación de un único recurso puede dar como resultado la creación de múltiples procesos en el recurso remoto. A todos los procesos se les asigna la misma credencial (paso 8).

3.3.3 Protocolo 3: Asignación de un recurso desde un proceso

Mientras que con la asignación de un recurso desde un *proxy* de usuario se hace necesario iniciar una computación, el caso más común es que la asignación del recurso sea iniciada de manera dinámica desde un proceso creado mediante una petición de asignación de recurso anterior. El Protocolo 3 define el proceso por el cual esto puede llevarse a cabo de la siguiente forma:

- El proceso y su *proxy* de usuario se autentican mutuamente utilizando, respectivamente, C_P y C_{UP}.
- El proceso envía una petición firmada a su *proxy* de usuario, de la forma:

Sig_P { “asignar”, parámetros de petición de asignación }.

- Si el *proxy* de usuario decide llevar a cabo la petición, inicia una petición de asignación de recurso al *proxy* de recurso especificado utilizando el Protocolo 2.
- El manejador de proceso resultante es firmado por el *proxy* de usuario y devuelto al proceso que realizó la petición.

La creación de credenciales específicas de proceso en el Protocolo 3 resulta en una delegación de un conjunto de derechos del usuario al proceso. La autenticación siempre se va a llevar a cabo entre el *proxy* de usuario y el *proxy* de recurso. Como consecuencia, este protocolo de firma única favorece la relación existente de confianza entre un usuario y un recurso que se estableció cuando al usuario se le concedió el acceso al recurso.

3.3.4 Protocolo 4: Registro del Mapeo

Un componente central de la política de seguridad y la arquitectura resultante es la existencia de un mapeo “correcto” entre el sujeto global y el correspondiente sujeto local. Para conseguir la conversión de un nombre global (por ejemplo, un ticket o un certificado) a un nombre local (por ejemplo, *login* o *id* del usuario) se utiliza la tabla de mapeo que mantiene el *proxy* de recurso. Aunque un administrador del sistema local podría crear la tabla de mapeo, esta arquitectura exige una cierta carga administrativa e introduce la posibilidad de error. (Algunos sitios prefieren gestionar la tabla de mapeo de manera explícita como parte de su proceso de creación de cuentas. En tal caso, estos sitios consideran el Protocolo como una característica opcional). Por tanto, se ha añadido una técnica que permite al usuario añadir un mapeo a la tabla.

La idea básica que hay detrás de esta técnica, presentada como Protocolo 4, es la de probar que el usuario mantiene las credenciales globales y

locales del sujeto. Esto se lleva a cabo mediante una autenticación global y una autenticación directa al recurso utilizando un método de autenticación local. El usuario declara entonces un mapeo entre las credenciales globales y locales. Esta declaración del usuario se coordina a través del *proxy* de recurso, ya que se encuentra en una posición desde la que puede aceptar credenciales globales y locales.

El mapeo de un identificador global a uno local se realiza de la siguiente forma:

- El *proxy* de usuario se autentica con el *proxy* de recurso.

- El *proxy* de usuario envía una petición firmada del tipo MAPASUJETO-UP al *proxy* de recurso, proporcionando como argumentos el nombre del sujeto global y el nombre del sujeto recurso.

- El usuario accede al recurso utilizando el método de autenticación del recurso e inicia un proceso de registro del mapa.

- El proceso de registro del mapa envía una petición del tipo MAPASUJETO-P al *proxy* de recurso, proporcionando como argumentos el nombre del sujeto global y el nombre del sujeto recurso.

- El *proxy* de recurso espera peticiones del tipo MAPA-SUJETO-UP y MAPA-SUJETO-P con argumentos coincidentes.

- El *proxy* de recurso asegura que el proceso de registro del mapa pertenece al sujeto recurso especificado en la petición del mapa.

- Si se encuentra un coincidente, el *proxy* de recurso configura un mapeo y envía confirmaciones al proceso de registro del mapa y al *proxy* de usuario.

- Si no se encuentra ningún coincidente dentro del tiempo establecido por el temporizador MAPA-*TIMEOUT*, el *proxy* de recurso purga la petición pendiente y envía una confirmación a la entidad que está esperando.

- Si no se recibe confirmación dentro del tiempo marcado por el temporizador MAPA-*TIMEOUT*, se considera que la petición ha fallado.

En los cuatro primeros pasos se muestran las actividades que realiza el usuario para la autenticación global (1 y 2) y para la autenticación del recurso (3 y 4).

La correspondencia entre las peticiones de tipo MAPA-SUJETO-P y MAPA-SUJETO-UP se debe gestionar tanto desde el *proxy* de usuario como desde el proceso de mapeo. Esto asegura que el mismo usuario está en posesión de las credenciales globales y locales. Si los resultados del protocolo de mapeo se almacenan en una base de datos accesible al *proxy* de recurso, entonces el usuario sólo necesita ejecutar el protocolo de mapeo una única vez por recurso. La política de administración del sistema local determina el tiempo durante el cual se mantiene como válido un mapeo. Sin embargo cabe esperar que un mapeo se mantenga válido durante toda la duración de las credenciales globales o durante toda la duración de la cuenta local del usuario.

Parte del protocolo de mapeo requiere que el usuario acceda al recurso para el cual se ha creado el mapeo. Esto requiere que un usuario se autentique al sistema local. En consecuencia, el protocolo de mapeo será tan seguro como el método de autenticación local. Con lo cual, los recursos con una fuerte

autenticación (por ejemplo, basada en Kerberos, S/KEY o *Secure Shell*) darán como resultado un mapeo más seguro.

3.4 Implementación de la Arquitectura de Seguridad GRID

Globus Security Infraestructure (GSI) es una implementación real de la arquitectura de seguridad descrita. GSI fue desarrollado como parte del proyecto Globus10 y se enfoca en:

- Entender la infraestructura básica requerida para soportar la ejecución de un amplio rango de aplicaciones Grid computacionales.
- Desarrollar implementaciones de prototipos para esta infraestructura.
- Evaluar las aplicaciones sobre *testbeds* a gran escala. Como parte del proyecto Globus se ha desarrollado GUSTO, un caso de prueba (*testbed*) que abarca unos 20 *sites* en 4 paises. GUSTO incluye centros de supercomputadores, laboratorios, centros de recurso, laboratorios de la NASA, universidades y compañías. Este caso de prueba ha sido utilizado para un amplio rango de experimentos de aplicaciones intensivas de comunicación.

4. ASIGNACIÓN Y PLANIFICACIÓN DE RECURSOS GRID

4.1 Servicios para asignación y planificación de recursos Grid

Para hacer más fácil la utilización de las infraestructuras distribuidas que forman el Grid, surge la necesidad de establecer una arquitectura global que sea plasmada en la práctica en una serie de servicios básicos en forma de middleware, y que simplificarán el modo de desarrollar aplicaciones que puedan hacer uso de estas infraestructuras.

En particular un proyecto Grid está orientado principalmente a aplicaciones paralelas e interactivas, por lo que se requiere que el control sobre los recursos y la planificación de los distintos trabajos de estas aplicaciones en concreto sea eficiente, automático y de una manera consistente.

Para ejemplificar la forma en que los recursos Grid se deben gestionar se toma como base la implementación que han hecho algunos proyectos como el Crossgrid (proyecto de Supercomputación y análisis de grandes volúmenes de datos) en cuanto ha planificación y asignación de recursos.

Las tareas relacionadas con la planificación y la selección eficiente de recursos están basadas en un middleware, cuya entidad principal es el *Resource Broker* (RB), que proporcionará los servicios necesarios para la ejecución automática de trabajos secuenciales.

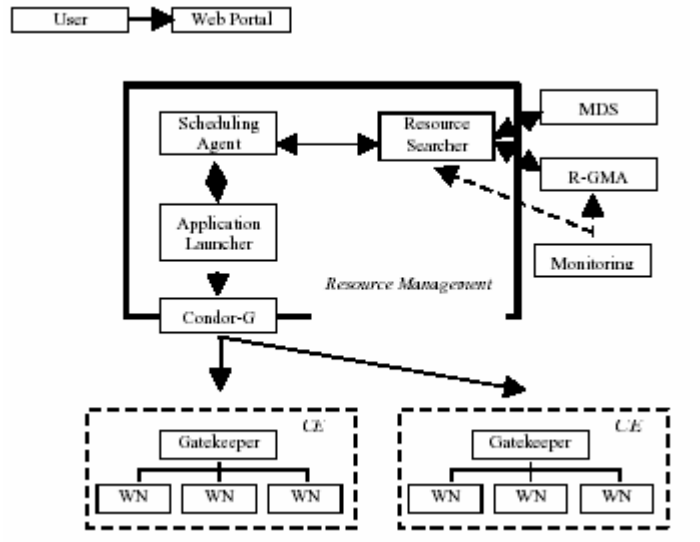
Mediante estos servicios un usuario es capaz de enviar un trabajo junto con una descripción de los requerimientos del mismo, y el RB se encargará de buscar y seleccionar los recursos necesarios. Cuando se hayan localizado los

recursos se pasará a la siguiente etapa que consiste en mandar el trabajo para su ejecución utilizando aquellos seleccionados, tomando las precauciones necesarias y reenviando el trabajo si algo fallara. Continuamente se monitorizará el estado del trabajo para ver que el trabajo se está ejecutando correctamente, reenviando el trabajo a estos u otros recursos si algo fallara. Se obtendrá finalmente la salida del mismo para reenviarla al usuario, haciendo la ejecución totalmente transparente para éste.

Aunque un proyecto Grid se beneficia de estos servicios, extensiones a éstos y nuevos componentes específicos son necesarios para el control de aplicaciones paralelas interactivas escritas en MPI (*Message Passing Interface*), y también para solventar algunas deficiencias del sistema original, dando lugar a los *Scheduling Agents* (SA), que constituyen una extensión de la funcionalidad proporcionada por el RB.

En los siguientes puntos se hace referencia a los servicios relacionados más específicamente con la gestión y planificación de recursos, así como los cambios más importantes en estos componentes para acomodar las necesidades de las aplicaciones interactivas. La figura muestra los componentes que gestionan los recursos y su funcionamiento en una arquitectura de Grid.

Figura 8. Planificación de recursos Grid.



Fuente: A. Fernández, E. Heymann, J. Salt y M. A. Senar **(In)Allocation and Scheduling Services of Grid Resources**. Pagina 3.

4.1.1 Selección de Recursos

Una parte importante dentro del sistema de planificación de recursos es poder averiguar cuáles de estos recursos están disponibles en un momento determinado y hacer una selección de los mismos para ejecutar la aplicación. Cuando el usuario envía un trabajo junto con una descripción del mismo, éste será transformado por el SA a un *classad* que contiene los requerimientos y preferencias del trabajo. Los recursos también tienen asociados *classads* definiendo lo que cada recurso provee y que están publicados en un sistema de información, generalmente MDS o R-GMA.

En el caso de la arquitectura propuesta en Crossgrid se adoptan los nodos de computación (*Computing Elements* o CE) que son la representación

de una granja local compuesta por diferentes nodos trabajadores (*Working nodes* o WN), y que son el punto de acceso a la misma. Adicionalmente existen nodos de almacenamiento (*Storage Elements* o SE) desde donde se accederán los datos.

En la primera fase de la selección el *Resource Selector* consulta qué recursos cumplen los requerimientos, parciales o totales, que el trabajo está imponiendo. En el caso de se mande un trabajo secuencial éste se deberá ejecutar en un solo CE, ya que únicamente requerirá el uso de una cpu que se corresponderá finalmente con un WN de ese CE. En este caso se construirá una lista de posibles CEs que cumplen todos los requerimientos, ordenados según alguna función de preferencia (número de cpus del CE libres, potencia de cálculo, memoria libre, etcétera) que el usuario puede establecer.

En el caso de un trabajo MPI se necesitarán varias cpus para correr el trabajo, que serán seleccionadas en un único CE o en un grupo de ellos, con el siguiente criterio:

- Primero se intentan seleccionar los CEs que tienen todas las cpus requeridas libres, con lo que se intenta correr primero en un solo cluster para evitar las latencias de los mensajes entre diferentes clusters.
- Después se forman grupos de CEs que cumplan los requisitos necesarios para cada uno de ellos, y que conjuntamente reúnan el número de cpus libres requeridas.
- La ordenación entre los diferentes grupos se hace de menor a mayor número de CEs diferentes, y dentro de los grupos con el mismo número de elementos, se utilizará la función de preferencia ponderada con el número de componentes.

4.1.2 Planificación y Reservas temporales

Parte de la planificación se ha hecho con el servicio anterior de selección de recursos, ordenando los recursos a utilizar con una serie de criterios que se acaban de nombrar. En la siguiente fase se enviará el trabajo al primer CE o grupo de CEs de la lista en el caso de trabajos MPI. Debido a la naturaleza en la que se obtienen los datos sobre los CEs y sus cpus libres por ejemplo, que no es completamente actualizada en tiempo real, puede darse el caso de que se produzcan casos en los que el RB/SA considere alguna cpu como libre cuando realmente no lo está porque se acaba de mandar un trabajo. Este es sólo un ejemplo de los casos en los que puede fallar el intento de ejecución de un trabajo, por lo cual es necesario un método para poder reenviarlo a la siguiente selección de la lista.

Además se pueden dar posibles casos de interbloqueos entre diferentes trabajos mandados que requieran varias cpus, cuando un trabajo se está esperando un WN que está ocupado por un segundo trabajo, y éste está esperando un WN que está ocupado por el primero. Esto se soluciona con un método de reservas locales temporal que afecta al *Resource Selector*. Cuando se decide a qué grupo de CEs enviar el trabajo, se establece una reserva por un espacio de tiempo, de manera que estos recursos no serán tenidos en cuenta por el siguiente trabajo que busque recursos.

Estas reservas son locales al SA y no es un sistema general de reservas, sino que simplemente modifica los servicios locales de selección de recursos. Cuando se ha superado cierto intervalo, o el trabajo ha acabado o fallado en esos recursos, la reserva es retirada y esos recursos pueden volver a ser utilizados.

4.1.3 Prioridades

Las aplicaciones interactivas requieren un tiempo de respuesta corto para que el usuario pueda estar satisfecho con la ejecución de la misma. Para permitir que este tipo de aplicaciones puedan ejecutarse anteriormente a otras, por ejemplo aplicaciones en *batch*, se introduce acá el concepto de prioridades. Con las prioridades es posible determinar qué trabajos son necesarios ejecutar anteriormente, de manera que si tenemos varios trabajos pendientes de ejecución, podremos determinar el orden en que se van a mandar.

Además en el caso de que todos los recursos estén ocupados por la ejecución de trabajos, y nos llegue uno más prioritario, será posible determinar qué trabajos pueden ser parados momentáneamente para permitir la ejecución del nuevo trabajo más prioritario.

4.1.4 Envió del Trabajo

Cuando se tienen todas las decisiones hechas sobre los recursos a los que mandar el trabajo, y las distintas prioridades sobre otros, se deben realizar los pasos adecuados para permitir la ejecución del mismo. Dependiendo del tipo de trabajo, si es en *batch* o interactivo, si es secuencial o paralelo escrito en las diferentes variantes de MPI soportadas, se realizaran diferentes aproximaciones para conseguir el objetivo. El *Application Launcher* envía los ejecutables de forma confiable a los distintos recursos seleccionados implicados, y empezar la ejecución. Se pueden Utilizar herramientas como Condor-G y ampliaciones al mismo para llevar a cabo estas tareas.

4.2 Caso Práctico: Cálculo del número PI por el método Montecarlo en un ambiente Grid

4.2.1 Descripción

Se describe la implementación de una Grid para el cálculo del número PI utilizando el método de Montecarlo, para ejemplificar los procesos que deben existir en dicho cálculo y mostrar el manejo de recursos en la implementación de un ambiente de computación distribuida. Dicha aplicación es un proyecto piloto desarrollado por el Instituto Nacional de Estadística e Informática de Perú y consta de los siguientes subsistemas:

- Iniciación
    - Gridfinal: Para iniciar el interfaz gráfico del usuario final con el Grid.
    - INIT: Para desplegar el *Master* y *Worker* sobre la GRID.

- *Master*
    - *Master*: Proceso encargado de coordinar todos los procesos *Worker*, gestión del trabajo.

- *Worker*
    - *Worker*: Todos aquellos nodos que realizarán el proceso del calculo de PI.

El cálculo concretamente de la constante PI que se hace dentro la aplicación es basado sobre el método de Montecarlo que se describe de la siguiente forma:

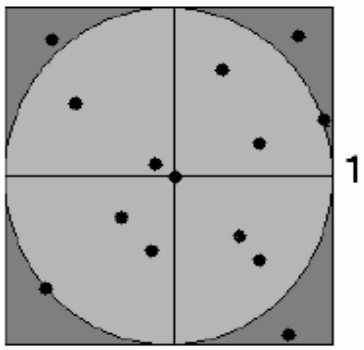
Figura 9. Cálculo del número PI por el método de Montecarlo.

$$f(x,y) = 1$$

$$g(x,y) = \begin{cases} 1 & \text{when } x^2 + y^2 \leq 1 \\ 0 & \text{when } x^2 + y^2 > 1 \end{cases}$$

$$\frac{\pi}{4} = \int_0^1 \int_0^1 g(x,y) f(x,y) dx dy$$

$$\bar{G}_N = \frac{1}{N} \sum_i g(X_i, Y_i) = \frac{\pi}{4}$$



Fuente: INEI Perú **(In) Proyecto Grid Computing**. Pagina 13.

Desglose del Cálculo de PI: N = k * n

4.2.2 Implementación en el planificador

Implementación en el *Master*:

$$\pi = \frac{[\pi_1 + \pi_2 + \ldots + \pi_k]}{k}$$

- k = número de trabajos (*JOB*)

4.2.3 Implementación en el/los trabajador/es

Implementación en el *Worker*:

$$\pi_n = \frac{4 * \left[\sum_i^n g(X_i, Y_i) \right]}{n}$$

- n = tamaño del trabajo (SIZE)
- n1 = n2 = n3 = … = nk

Ejemplo: 500 = 50 (JOB) * 10 (SIZE)

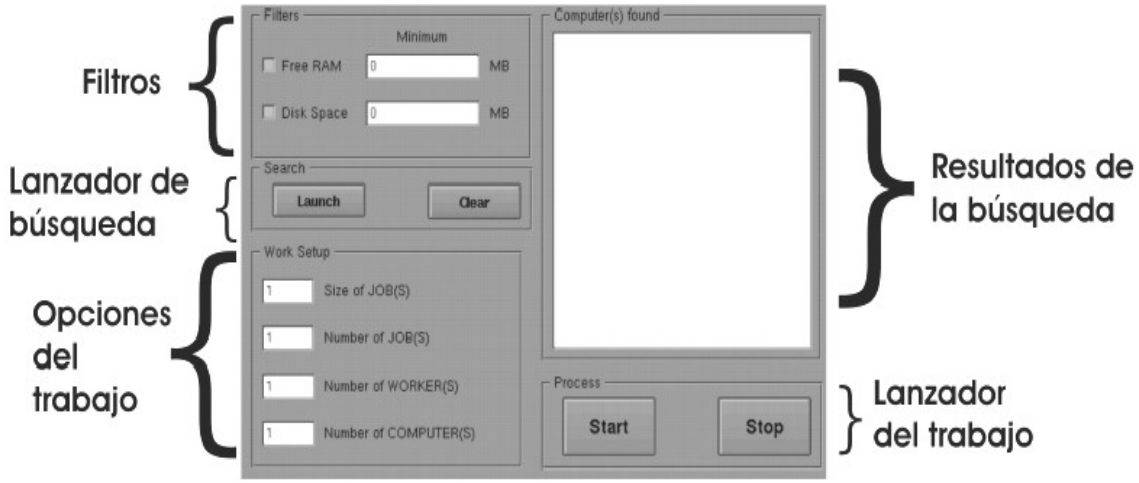
4.2.4 Proceso Global

La aplicación se divide en estas etapas cronológicamente:

- Iniciación (usuario).
- *Master* y *Worker*(s) desplegados sobre la GRID.
- *Master* y *Worker* hacen su trabajo respectivo.
- Fin de la aplicación.

- Iniciación: Se realiza solamente sobre una máquina. Su función es llenar los datos del proceso de cálculo de PI y buscar los maquinas disponible sobre la GRID.

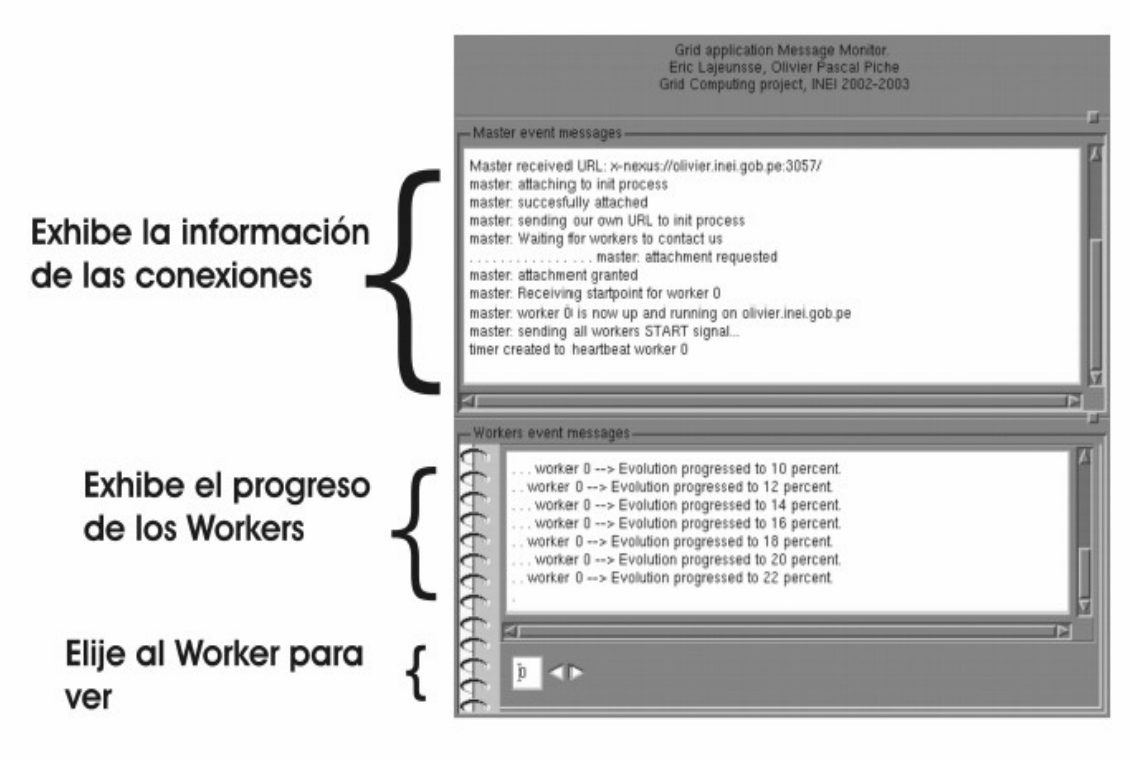
Figura 10. Pantalla del proceso Inicial en el cálculo de PI.



Fuente: INEI Perú **(In) Proyecto Grid Computing**. Pagina 16.

- *Master*: Un proceso master es desplegado sobre una maquina de la GRID y cuya función será coordinar la evolución de cada *Worker* y escribir los resultados y las estadísticas del proceso global.

Figura 11. Pantalla del proceso *Master* en el cálculo de PI.



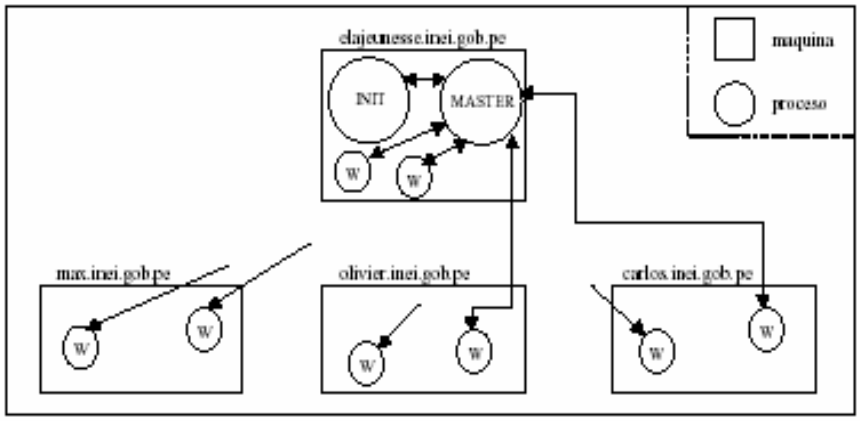
Fuente: INEI Perú **(In) Proyecto Grid Computing**. Pagina 18.

- *Worker*: El proceso *Worker* es transparente al usuario. Dichos procesos son desplegados sobre máquinas de la GRID y su función es calcular una parte del cálculo total de PI (trabajos). Cada proceso *Worker* manda su resultado y hace una petición para procesar otra parte del cálculo hasta que todas las partes sean culminadas.

  Ejemplo de la ejecución:

  - 4 maquinas
  - 8 *worker*

Figura 12. Funcionamiento del proceso global del cálculo de PI.

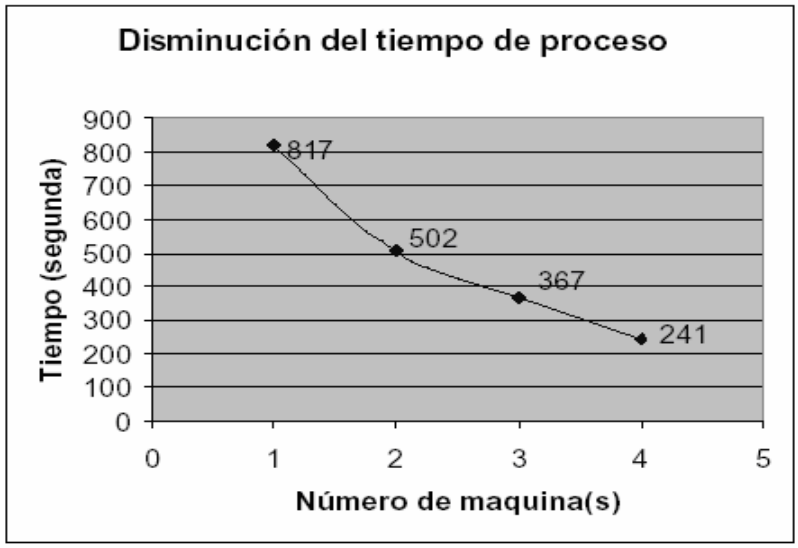


Fuente: INEI Perú **(In) Proyecto Grid Computing**. Pagina 22.

4.2.5 Análisis y Resultados

La distribución de la aplicación sobre la GRID disminuye el tiempo de proceso total de la aplicación.

Figura 13. Análisis del resultado de una arquitectura Grid en el cálculo de la constante PI.



Fuente: INEI Perú **(In) Proyecto Grid Computing**. Pagina 23.

CONCLUSIONES

1. La difusión de entornos de aplicación Grid permitirá el uso de recursos computacionales que se encontraban fuera del alcance de las organizaciones, lo que tendrá un alto impacto de desarrollo propicio para implementaciones científicas a gran escala e integración de servicios entre sectores empresariales.

2. La implementación de tecnologías Grid permite resolver una gran cantidad de problemas a través de la integración de múltiples organizaciones, lo que significa no solo un ahorro de costos y tiempo sino que garantiza la calidad, gestionando la información de manera que se pueda involucrar aquellas organizaciones que cumplan los requisitos apropiados.

3. La implementación de una Grid computacional pasa por 2 componentes: el *Middleware* que debe soportar los servicios de gestión de la información y aplicaciones, los servicios de red y de almacenamiento, y, que, por lo tanto, funciona como un sistema operativo que gestiona dichos procesos y la parte de aplicación que se compone de aquellas tareas que se procesaran en el sistema Grid.

4. El uso de servicios Web XML ha posibilitado la comunicación entre aplicaciones de manera tal que se describe la información, y la misma, puede ser estructurada por un receptor que sabrá la relación que guardan los datos enviados por un emisor, son los servicios Web XML los que dan cabida al desarrollo de ambientes de integración de aplicaciones como Grid, y son los bloques básicos de todo el proceso distribuido actual bajo Internet.

5. Mediante el uso de servicios Web se pueden manipular los estados de un recurso, lo que se traduce en la interacción entre los servicios Web; un recurso asociado a estados es un componente con estados definidos en XML, un ciclo de vida y puede ser manipulado por varios servicios Web, esto permite el manejo y la gestión de recursos como pueden ser archivos, objetos java, filas de una tabla, etc. en un ambiente Grid.

6. El *middleware* Grid se encarga de proporcionar los servicios que permiten seleccionar los recursos a utilizar y planificar su uso, manteniendo un control sobre el avance del trabajo que se realiza sobre los recursos disponibles y tomando las medidas necesarias si algo fallase. Las herramientas *middleware* existentes permiten la planificación eficiente de recursos, gestionando y teniendo en cuenta aspectos como la selección eficaz de recursos de acuerdo a los requisitos que el trabajo impone, orden del trabajo y prioridades entre otros criterios.

RECOMENDACIONES

1. Es muy importante, cuando se va a implementar un ambiente Grid contar con la infraestructura necesaria, sobre todo si esta infraestructura se convertirá en los recursos a los que el acceso compartido será conducido por los protocolos Grid, en todo caso se deben tomar en cuenta los recursos computacionales, de almacenamiento y de red.

2. Un punto medular de cualquier aplicación son los mecanismos que garanticen su integridad, a diferencia de una Grid de información –Internet- en una Grid de recursos los usuarios anónimos no deben acceder sin las credenciales necesarias, por lo tanto, se deben establecer los mecanismos de seguridad que permitan la entrada a usuarios previamente registrados y autorizados.

3. La seguridad en entornos Grid se debe orientar a desarrollar mecanismos de control de acceso que tengan gran flexibilidad y estén basados en políticas; implementar políticas de control de acceso entre dominios para que se permita la comunicación segura entre grupos y, mecanismos de delegación para soportar escalabilidad a un gran número de usuarios y recursos.

BIBLIOGRAFÍA ELECTRÓNICA

1. Humphrey, Marty. Wasson, Glenn. Morgan, Mark. Beekwilder, Norm. **An Early Evaluation of WSRF and WS-Notification via WSRF.NET** **http://www.cs.virginia.edu/~gsw2c/WSRFdotNet/wsrf_Grid2004.pdf** 1/2004

2. Lajeunesse, Erick. Piché, Olivierde. **Proyecto Grid Computing**. **http://www.pcm.gob.pe/portal_ongei/publica/proyectos/4819.pdf** 09/08/2004

3. Foster, Ian. Keselman, Carl. Tuecke Steven. **The Anatomy of the Grid. Enabling Scalable Virtual Organizations.** **www.globus.org/alliance/publications/papers/anatomy.pdf** 2000

4. Foster, Ian. Argonne National Laboratory, Chicago. **What is the Grid?** **http://www.fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf.** 20/07/2002

5. Borja Sotomayor. **Grid Computing. Un nuevo paradigma de computación distribuida.** **http://www.eside.deusto.es/eventos/semana/eventos/pdf/grid_computing.pdf** 09/04/2003

6. Borja Sotomayor. **Introducción a la Computación Grid.** **http://people.cs.uchicago.edu/~borja/lectures/IntroduccionGrid.pdf** 30/04/2004

7. Foster, Ian. **Internet Computing and the Emerging Grid. Nature Web Matters.** **http://www.nature.com/nature/webmatters/grid/grid.html.** 2000

8. Bote Lorenzo, Miguel L. Asencio Pérez, Juan. Gómez Sánchez Eduardo. **Computación Grid e Ingeniería del Software Basada en Componentes en CSCL.** **http://ulises.tel.uva.es/uploaded_files/asignadorGridcorto.pdf** 09/2003

9. Bote Lorenzo, Miguel L. Asencio Pérez. **Grid Computing and Component-Based Software Engineering in Computer Supported Collaborative Learning.**
http://ulises.tel.uva.es/uploaded_files/BoteICCS04.pdf 06/2004

10. Valiente Blázquez, María Cruz. **Seguridad en Entornos Grid.**
http://inf.uc3me.es/pfc/pfc.pdf 2003

11. Fernández, Alvaro. Heymann, Elisa. Salt, José. Senar, Miguel A. **Servicios de asignación y planificación de Recursos Grid. Barcelona, España.**
http://www.rediris.es/rediris/boletin/66-67/ponencia2.pdf 12/2003-01/2005

12. Vásquez Poletti, José Luis. **Análisis de la Arquitectura de Globus Toolkit**
http://asds.dacya.ucm.es/jlvazquez/files/081004GT4slides.pdf 08/10/2004

13. European CrossGrid Project.
http://www.crossgrid.org 1/2005.

14. The Globus Project, Argone National Laboratory. **Grid Architecture.**
http://www.lip.pt/computing/projects/crossgrid/doc/DataGridOverview.pdf 16/07/2001

15. F. Giacomini, F. Prelz. **Definition of architecture, technical plan and evaluation criteria for scheduling, resource managements, security and job description.**
http://server11.infn.it/workload-grid/docs/DataGrid-01-D1.2-0112-0-3.pdf

16. Brooks Davis, Craig Lee The Aerospace Corporation El Segundo, CA **Grid Computing With FreeBSD.**
http://people.freebsd.org/~brooks/pubs/usebsd2004/fbsdgrids.pdf 29/06/2004

17. Algunas Respuestas clave sobre el Grid Computing
http://www.vnunet.es/Actualidad/An%C3%A1lisis/Infraestructuras/Soluciones/20030317009 17/03/2003

18. Rodríguez González, David. **Uso de Bases de Datos en Grid.**
http://grid.ifca.unican.es/cursos/presentaciones/DB_GRID.ppt 06/2004

19. Escudero Garzás, José Joaquín. **Grid e IPv6.** **http://www.it.uc3m.es/~fvalera/int_red/trabajos/Grid%20IPv6.pdf** Madrid. 10/2004

20. Verduzco, J. Garcia, N. Pecero, J. Dominguez, P. Pillon, M. Capit, N y otros. **Grid-ITC Una Intra-GRID de Bajo Costo para las Necesidades de Calculo Intensivo del Instituto Tecnológico de Colima.** **http://www-id.imag.fr/~mpillon/arquivos/griditcjournal.pdf** 08/2004

21. Alvaro Fernández Casani. Grupos Trabajo Middleware IrisGrid. **Integración de VOs y middleware para EGEE.** **http://asds.dacya.ucm.es/GridMiddleware/slides/VO_CSIC.pdf** 26/11/2004

22. Borja Sotomayor. **The Globus Toolkit 3 Programmer's Tutorial** **http://gdp.globus.org/gt3-tutorial/multiplehtml/** 2003, 2004**.**

23. Wolter, Roger. Microsoft Corporation. **Fundamentos de los servicios Web XML** **http://www.microsoft.com/spanish/msdn/articulos/archivo/151102/voices/fundamentos_xml.asp** 12/2001

24. Proyecto Seti@home **http://setiathome.ssl.berkeley.edu/** 2004

25. IBM. **Globus Toolkit 4 Early Access: WSRF** **http://whitepapers.zdnet.co.uk/0,39025945,60107995p-39000590q,00.htm** 26/10/2004

26. The Globus Alliance **http://www.globus.org/** 1/2005.