

# Interfaz Cloud para Servicios de Gestión de Recursos

Mauricio Morales Franco

Facultad de Informática  
Master en Investigación Informática 08-09

Universidad Complutense de Madrid  
Madrid, Madrid, España

[maomorales@gmail.com](mailto:maomorales@gmail.com)

**Resúmen.** En este artículo se analiza como funciona GRAM para la gestión de recursos y cómo puede ser enviado a través de WSGRAM un trabajo a un Grid. Se mencionan además algunos trucos para simplificar la instalación de Globus Toolkit con GRAM y WSGRAM, luego se implementará un prototipo de un Web Service tipo RESTful que servirá para recibir POSTs con descripciones de trabajo que luego se ejecutarán en el Grid.

## Contenido

- 1) Introducción
- 2) GRAM
  - 2.1) Funcionamiento de GRAM
  - 2.2) Funcionamiento de WS GRAM
- 3) Instalación de Gobus Toolkit
  - 3.1) VDT
  - 3.2) Instant-Grid
  - 3.3) Grid Roll with Naregi Integration
  - 3.4) Manual de Globus Alliance
- 4) Puesta en marcha del Grid
- 5) Implementación de Web Services en GRAM
- 6) Desarrollo del Prototipo RESTful en PHP
- 7) Conclusiones, ventajas y desventajas. Enlace al código fuente.

## 1. Introducción

Conociendo la importancia de los GRID y las oportunidades que este tipo de infraestructuras están proporcionando a la ciencia y a la industria, he decidido realizar este artículo para estudiar con un cierto nivel de detalle el funcionamiento de la ejecución de trabajos en Globus Toolkit 4 enfocado al servicio GRAM (Grid Resource Allocation and Management) y WSGRAM que está hecho para la creación de Web services, además se presentará un prototipo funcional de un Web service tipo RESTful creado en PHP para recibir un RSL de un trabajo en un POST y ejecutarlo vía globusrun-ws.

## 2. GRAM

Globus Toolkit cuenta con un paquete para la administración – envío, monitoreo y anulación de tareas – de recursos de Grid computacionales, y también un paquete de Webservices para hacer interfaz con esta administración. Estos son conocidos como GRAM – *Grid Resource Allocation and Management* – y como WS GRAM a la implementación de los Webservices.

Es importante aclarar que GRAM no es un planificador de trabajos, pero se puede comunicar con diferentes motores de planificación de trabajos usando un formato estándar de mensajes. GRAM tampoco se puede ver como un reemplazo de RPC, pues para aplicaciones que no requieran todas las características que éste ofrece puede resultar más costoso su uso en términos computacionales.

WS GRAM está basado en WSRF<sup>1</sup> de Globus Toolkit, que es un aprovechamiento muy potente de Webservices para aplicaciones distribuidas que requieren compartir información continuamente.

GRAM es una solución que promete bastante rendimiento pues se ha realizado aprovechando los servicios existentes de Globus, WSRF, GSI<sup>2</sup>. No hay implementaciones redundantes.

### 2.1. Funcionamiento de GRAM

GRAM es responsable de leer y procesar el RSL<sup>3</sup> que contiene la descripción del trabajo y activar el monitoreo remoto para administrar las tareas que ya han sido creadas.

Para ejecutar una tarea remotamente, el servidor de GRAM (gatekeeper) debe estar ejecutando en un ordenador remoto y la aplicación requiere ser compilada en esa máquina remota. La ejecución inicia cuando una aplicación en la máquina local hace una petición de trabajo al ordenador remoto, la especificación del trabajo debe llevar el nombre y puerto del ordenador remoto así como el ejecutable, *stdin*, y *stdout*.

Después de esto GRAM crea un administrador para el trabajo, que se encarga de la ejecución del trabajo y la comunicación con el usuario.

---

<sup>1</sup> WS Resource Framework: <http://www.globus.org/wsrf/>

<sup>2</sup> Grid Security Infrastructure: <http://www.globus.org/security/overview.html>

<sup>3</sup> Resource Specification Language: <http://tinyurl.com/cxhagd>

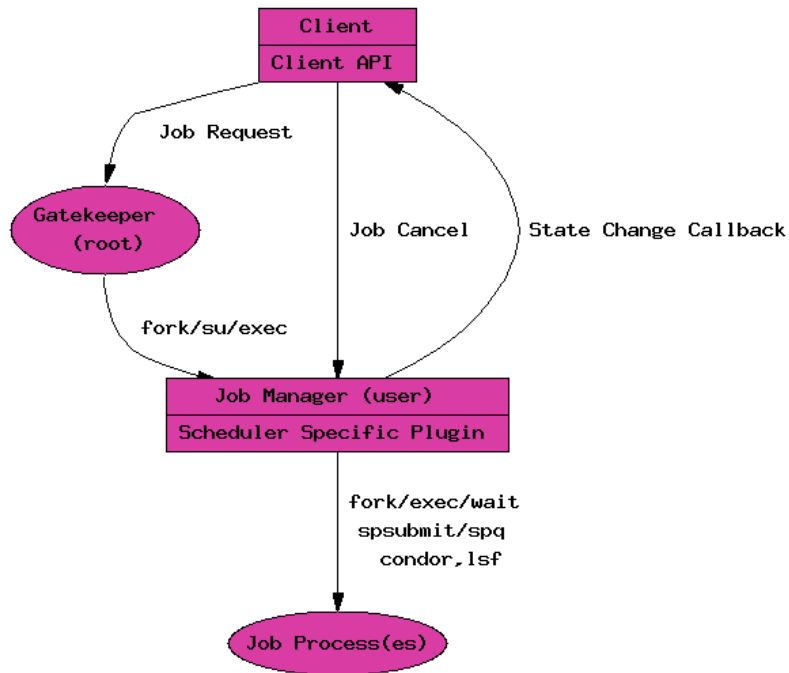


Fig 1. Arquitectura de GRAM

## 2.2. Funcionamiento de WS GRAM

Como lo he mencionado en la página anterior, WSGRAM provee una serie de interfaces con Web services consistentes con el modelo WSRF. La arquitectura está dividida en diferentes componentes que se conectan entre sí para aprovechar por ejemplo las funcionalidades del core de Globus.

El núcleo de WSGRAM se basa en el conjunto de Web services que dependen de WSRF, los Web services de WSGRAM son:

### 2.2.1. ManagedJob

Cada trabajo enviado tiene su propio administrador, el servicio provee una interface para monitorear el estado del trabajo o terminar el trabajo.

### 2.2.2. ManagedJobFactory

Este servicio provee una interfaz para crear recursos *ManagedJob* del tipo apropiado para realizar un trabajo en el planificador local.

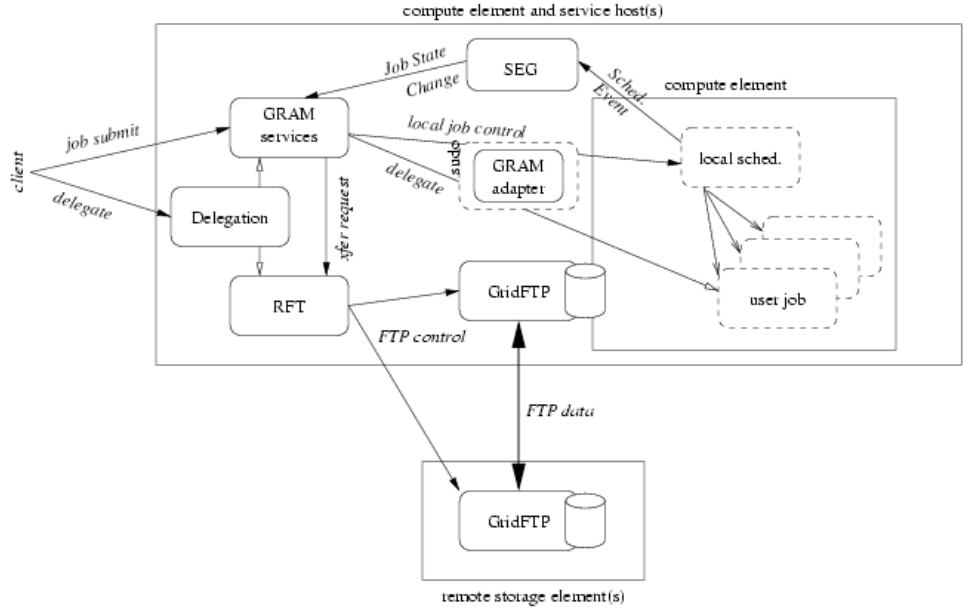


Fig 2. Componentes de WSGRAM.

Los componentes en WSGRAM están diseñados para soportar diferentes tipos de escenarios de configuración. Un concepto global de la forma en que funciona puede ser visto en la figura 3. El principal componente es "ManagedJob" quien es el responsable de la creación del trabajo, después de que un cliente ha obtenido el recurso, se inicia un ciclo de vida que puede finalizar en éxito o en una destrucción del trabajo.

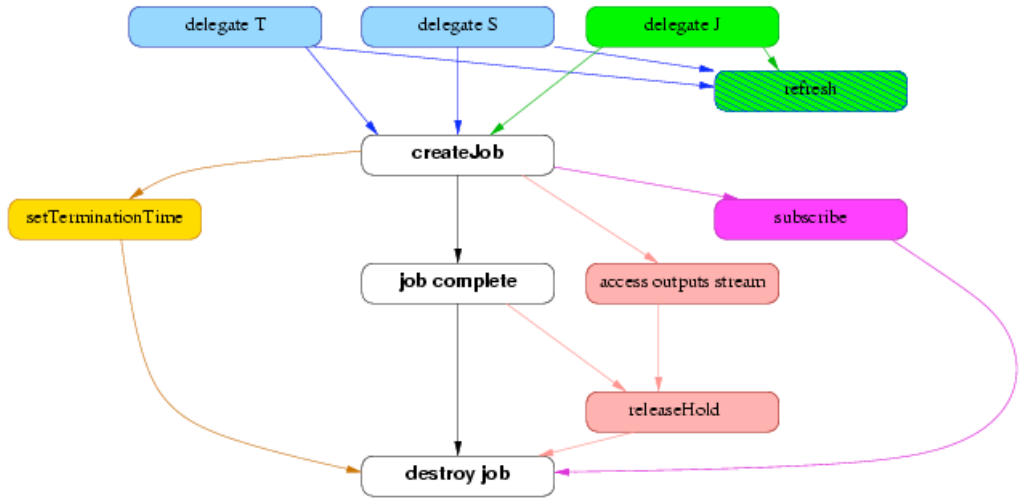


Fig 3. Actividades principales en un trabajo WS GRAM

El cliente puede solicitar opcionalmente que algunas actividades de montaje (staging activities) ocurran antes o después de la ejecución del trabajo, si son solicitadas en la creación del trabajo, las

credenciales EPR<sup>4</sup> deben ser entregadas (delegate T, delegate S en la figura anterior), estas operaciones de delegación son realizadas antes de invocarse *createManagedJob*. Se pueden especificar credenciales tanto para el montaje como para transferencia, o puede ser una misma. Las credenciales de montaje dan los privilegios a WS GRAM de interactuar con RFT, mientras que las credenciales de transferencia dan privilegios a RFT de interactuar con servidores GridFTP, en el EPR se puede especificar un atributo para que las credenciales se refresquen.

El cliente puede acceder a los ficheros de salida que ha escrito el proceso antes de que el trabajo termine su ejecución o que la actividad de montaje envíe la información a otro host.

Antes de continuar y analizar los métodos que proveen los Web services, vamos a introducir un poco sobre los componentes que usa WS GRAM en su funcionamiento.

### 2.2.3. Componentes usados por WS GRAM

Como se ha visto en la figura 2, este servicio usa diferentes componentes que pueden ser catalogados como: los de Globus Toolkit, los internos y los externos

#### 2.2.3.1. Componentes De Globus Toolkit

- **RFT**  
Reliable File Transfer, es un servicio de Globus<sup>4</sup> para transferencia de datos que es utilizado por WSGRAM en el montaje de datos (preparación de ficheros) antes de iniciar la ejecución del trabajo y también al finalizar. RFT es muy confiable y con mejor soporte para operaciones como reintentar y reiniciar.
- **GridFTP**  
Los servidores GridFTP son requeridos para acceder a elementos de almacenamiento remoto, también como sistemas de archivos accesibles por el trabajo. El RFT actúa como un objeto intermedio para transmitir la información entre el almacenamiento remoto y el sistema en cuestión.
- **Delegation**  
Este componente de Globus<sup>4</sup> es usado por los clientes para delegar credenciales para el uso de WSGRAM y RFT.

#### 2.2.3.2. Componentes Internos

- **Scheduler Event Generator**  
Este componente le permite a WSGRAM la capacidad de monitorear trabajos.
- **Fork Starter**  
Este programa inicia la aplicación del usuario y espera por su finalización, guarda el tiempo de inicio, tiempo de finalización y estados de “salida” para cada proceso que él inicia. Este plugin es usado por el *Scheduler Event Generator* para actualizar la información de los cambios del trabajo.

---

<sup>4</sup> EPR. Básicamente, un archivo XML donde se especifican credenciales para delegar acceso.

### 2.2.3.3. Componentes Externos

- **Local Job Scheduler**

WSGRAM opcionalmente puede usar un planificador local para administrar algunos recursos, esto se hace con el comando estándar de Unix *fork()*, sin embargo los elementos que hay que controlar en una escala más grande están bajo el control de planificadores como PBS, LSF, etc.

- **Sudo**

La intención en la implementación es que WSGRAM no tenga que usar un superusuario con todos los privilegios sobre el sistema. WSGRAM utiliza la utilidad de Unix *sudo* para ganar acceso a las cuentas de los usuarios, es decir que los trabajos se ejecutan en el contexto del usuario que los solicita. Esto permite una administración flexible de que usuarios pueden utilizar el sistema.

### 2.2.4. Modelo de Seguridad

WSGRAM utiliza un muy buen nivel de seguridad para los llamados a los Web services, igual que los provee el core WSRF de Globus. Esta seguridad involucra autenticación de los clientes, prevención de alteración de mensajes y hasta privacidad opcional para el contenido de los mensajes.

Los trabajos son ejecutados dentro de la cuenta del usuario Unix, la autenticación de WS GRAM administra cuáles clientes del Grid pueden enviar trabajos a cuáles usuarios, luego se utiliza el comando *sudo* una vez se haya verificado que el cliente tiene permisos para acceder a la cuenta del usuario.

También ya hemos hablado sobre la delegación que hace parte de la seguridad de WS GRAM, los clientes opcionalmente pueden delegar algunos privilegios a WS GRAM, normalmente para facilitar el montaje de datos “file staging”, o para el proceso del trabajo en sí mismo. Cuando no se realiza delegación, WS GRAM deshabilita el montaje “staging” y el trabajo no tendrá posibilidad de solicitar operaciones privilegiadas en el Grid.

Además del control, se pueden activar las opciones de auditoría, que permiten guardar un registro de las operaciones registradas en el sistema.

### 2.2.5. Protocolo

El protocolo de WS GRAM está centrado en el uso de un recurso que lo contiene todo, por verlo de una forma más simple, a través de un *ManagedJob* creado a través de la operación *createManagedJob()*. Sin embargo es importante conocer los métodos existentes:

Método	Descripción
DelegationFactory::requestSecurityToken	Delegación de credenciales
Delegation::refresh	Actualización de credenciales
ManagedJobFactory::getResourceProperty y getMultipleResourceProperties	Recuperación de información del planificador y trabajo. Antes de iniciar el trabajo. Como estado de etapa de montaje.
ManagedJobFactory::createManagedJob	Paso obligatorio para crear el recurso ManagedJob que se describa en la entrada
ManagedJob::release	Permite al ManagedJob continuar en un estado donde estaba esperando o programado para

	esperar
ManagedJob::setTerminationTime	Permite reprogramar el tiempo de finalización
ManagedJob::destroy	Permite abortar el trabajo
ManagedJob::subscribe	Permite suscribirse a notificaciones de cambio, esto también se puede hacer en la creación del trabajo.
ManagedJob::getResourceProperty y getMultipleResourceProperties	Permite recuperar información sobre el estado del recurso en cualquier momento (dentro de su ciclo de vida)

### 3. Instalación de Globus Toolkit

La intención no es explicar aquí como se instala Globus Toolkit paso a paso, pero sí ofrecer algunas alternativas para realizar la instalación.

Varias universidades y grupos de investigación alrededor del mundo han creado mecanismos que facilitan la instalación de Globus Toolkit, es importante saber que esta instalación no es un proceso tan sencillo como instalar un paquete cualquiera, pues Globus Toolkit exige que se cumplan algunos requerimientos del sistema y luego se deben realizar algunas operaciones para configurar sus diferentes servicios, en especial todo lo relacionado con seguridad.

En la investigación que he realizado he encontrado diferentes alternativas estables que son usadas por diferentes compañías y universidades para realizar una puesta a punto de un Grid.

#### 3.1. VDT

Virtual Data Toolkit (VDT) es un software que integra diferentes herramientas para configurar un Grid de una forma más sencilla que instalando todo por separado. Instalar un Grid desde cero es una actividad que puede consumir mucho tiempo y para novatos puede requerir de varios intentos de instalación.

VDT es un producto de Open Science Grid (OSG), quien lo usa como su distribución middleware de Grid. Ha sido fundado por la Fundación Nacional de Ciencia y el Departamento de Energía de los Estados Unidos.

A la fecha la última versión estable de VDT es 1.10.1 y el software se ofrece para diversas plataformas y sistemas operativos, inclusive hay distribuciones en RPMs.

<http://vdt.cs.wisc.edu>

#### 3.2. Instant-Grid

Instant-Grid es una distribución tipo Live-CD que se basa sobre una distribución Knoppix con un entorno preconfigurado para Grid con Globus Toolkit.

Diferentes centros de Investigación de Alemania han participado en este proyecto. Este proyecto es muy interesante por ser amigable y flexible para una demostración de desarrollo y muy orientado a principiantes.

<http://instant-grid.de>

### 3.3. Grid Roll with Naregi Integration

Este proyecto ha sido desarrollado en Chile por la red universitaria Reuna, diferentes universidades le han apoyado. Todo se basa en una imagen que trae un sistema operativo GNU/Linux con Globus Toolkit e incluso integración con Sun Grid Engine (SGE).

El proyecto es muy interesante pero en principio viene para pre-configurarse en el Grid de Naregi<sup>5</sup>, aunque se podrían adecuar los certificados y servicios para una instalación personalizada.

<http://wiki-clgrid.reuna.cl>

### 3.4. Manual de Globus Alliance

El manual de Globus que aunque propone una instalación con más pasos, se encuentra en muy buen estado de documentación y muy claro, ideal para instalaciones de producción y para conocer mejor los componentes que forman a Globus Toolkit.

El documento con enlaces para descargas se encuentra en el sitio web oficial de Globus Toolkit.

<http://www.globus.org/toolkit/docs/4.0/admin/docbook>

## 4. Puesta en Marcha del Grid

Para el montaje del prototipo me basaré en Instant-Grid que es la distribución más sencilla de tener funcionando en pocos minutos, antes de pensar en esta opción dediqué una buena cantidad de horas configurando Globus Toolkit 4 por separado en una distribución GNU/Linux Debian 4. Tuve varios tropiezos y teniendo en cuenta que los fuentes de este prototipo están disponibles en la red (ver al final del documento) es mucho mejor una solución que se pueda probar rápidamente.

Para tener funcionando Instant-Grid en el ordenador basta con descargar un software como VirtualBox[6] y montar sobre un Virtual Host la imagen del disco de Instant-Grid que se puede descargar desde su sitio web.

Instant-Grid dejará una instalación de desarrollo completamente funcional para nuestras pruebas, inclusive trae configurados los certificados CA que requieren los diferentes servicios de Globus.

Si el arranque ha sido exitoso, los puertos comunes abiertos en el Host deben ser los que se ven en la figura 4.

Para que el prototipo funcione es importante instalar un paquete que no trae incluido Instant-Grid, PHP-CLI (Command Line Interface).

La forma más sencilla de instalarlo es a través de "apt":

```
$ apt-get update
```

---

<sup>5</sup> Naregi: National Research Grid Initiative es un proyecto japonés que busca proveer computación de alto desempeño en ambientes ampliamente distribuidos.

<sup>6</sup> VirtualBox: Iniciativa libre de Sun Microsystems para virtualizar máquinas en la mayoría de plataformas. <http://www.virtualbox.org>

```
$ apt-get install php5-cli
```

```
server:0 21:45:56 www # nmap localhost
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2009-02-11 21:46 CET
Interesting ports on localhost (127.0.0.1):
Not shown: 1669 closed ports
PORT      STATE SERVICE
68/tcp    open  dhcp
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp  open  nfs
5001/tcp  open  complex-link
5432/tcp  open  postgres
6000/tcp  open  X11
8009/tcp  open  ajp13
8080/tcp  open  http-proxy
8443/tcp  open  https-alt
8888/tcp  open  sun-answerbook

Nmap finished: 1 IP address (1 host up) scanned in 0.273 seconds
server:0 21:46:03 www #
```

Figura 4. Puertos abiertos después de la ejecución de Instant-Grid.

Con esto ya tenemos lo básico.

## 5. Implementación de Web Services en GRAM

Ya he mencionado que GRAM provee de una serie de facilidades para la implementación de Web Services, de hecho, los llamados que se hacen a través del método `globusrun-ws` en un ordenador local, establecen conexión con el servidor remoto vía SOAP.

El servicio de hecho es un Web Service, pero para usarlo se debe tener instalado en el ordenador local los binarios necesarios para ejecutar `globusrun-ws`.

GRAM provee de un API en Java [7] y en C que puede ser usado para implementar otro tipo de Web Services de acuerdo a las necesidades, no hay un API oficial para otros lenguajes programación.

A continuación veremos el prototipo.

---

<sup>7</sup> GRAM API: <http://www.globus.org/cog/distribution/1.1/api/org/globus/gram/package-summary.html>

## 6. Desarrollo del Prototipo RESTful en PHP

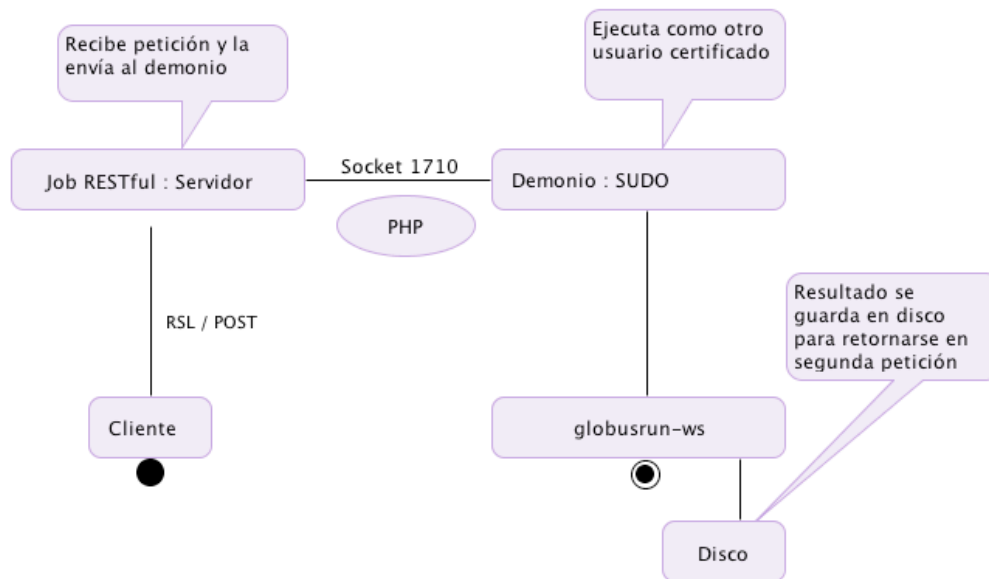
PHP es un lenguaje orientado a objetos basado en Scripts que en los últimos años se ha convertido en una herramienta muy potente y flexible, bien usado provee de excelentes características de seguridad.

Cuando hablé del funcionamiento de GRAM mencioné como la utilidad “*sudo*” de Unix es utilizada para lanzar los trabajos desde el entorno local del usuario que solicita el trabajo.

Para el desarrollo de este prototipo se van a tomar todas las solicitudes de trabajo y se ejecutarán sobre el entorno de un mismo usuario, que en este caso será el usuario certificado de Instant-Grid llamado *knoppix*.

El servidor Web Apache será quien recibe la petición, y esta luego será procesada por el pequeño programa en PHP, pero nótese que los procesos que se lancen desde este entorno se ejecutarán a nombre del usuario “*www-data*” que es por defecto el usuario Apache.

En este caso y para resolver el problema, un demonio (proceso en background) se ejecutará a nombre del usuario *knoppix*, este proceso tendrá un socket abierto en el puerto 1710 que está encargado de recibir los datos del comando completo para ejecutar a su nombre.



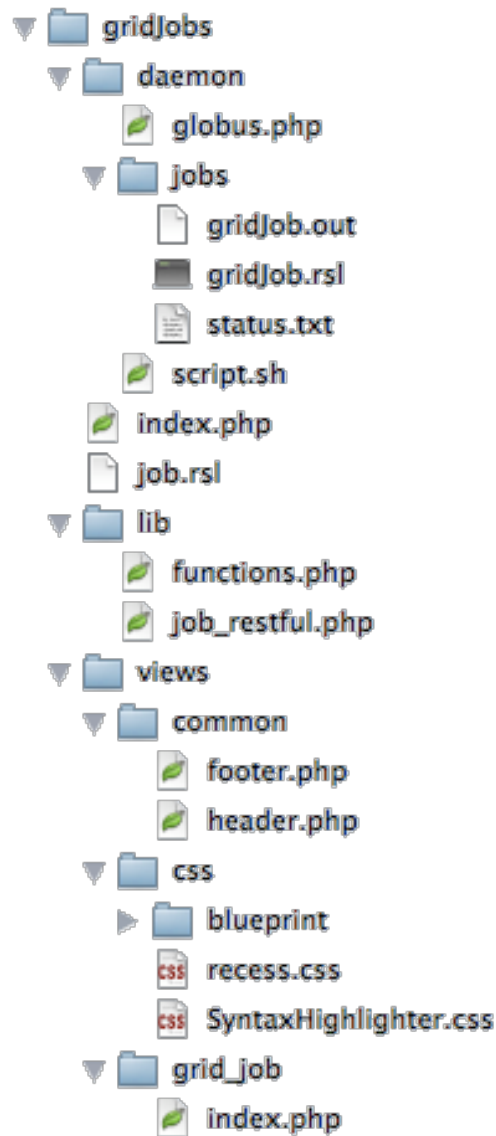
**Figura 5.** Esquema del prototipo propuesto

Después de definir el esquema sobre el que funcionará el prototipo se procede con la codificación, en la implementación se incluye una interfaz gráfica para hacer la simulación de la recepción del RSL vía POST, en una ventana del navegador.

En la estructura de archivos del programa se almacena el demonio en un directorio separado y allí también se guardará la descripción RSL, la salida del trabajo y los cambios de estados reportados por globusrun-ws.

En otro directorio se guardan los métodos básicos para construcción del comando a ejecutar y la clase principal del servidor RESTful.

En un tercer directorio están las vistas (la parte gráfica) de la interfaz para hacer la simulación.



**Figura 6.** Estructura de archivos

Una vez implementado el prototipo (ver link al código fuente al final del artículo) se configura en el ambiente de Instant-Grid, para esto se copia el programa -- directorio gridJobs -- en `/var/www`, también es importante para efectos de la prueba dar permisos 0777 y cambiar el propietario del directorio `daemon`.

Como root:

```
$ chmod -R 0777 /var/www/gridJobs/daemon
```

```
$ chown -R knoppix: /var/www/gridJobs/daemon
```

Ahora se puede ejecutar desde el navegador la pantalla de simulación:

<http://server/gridJobs>

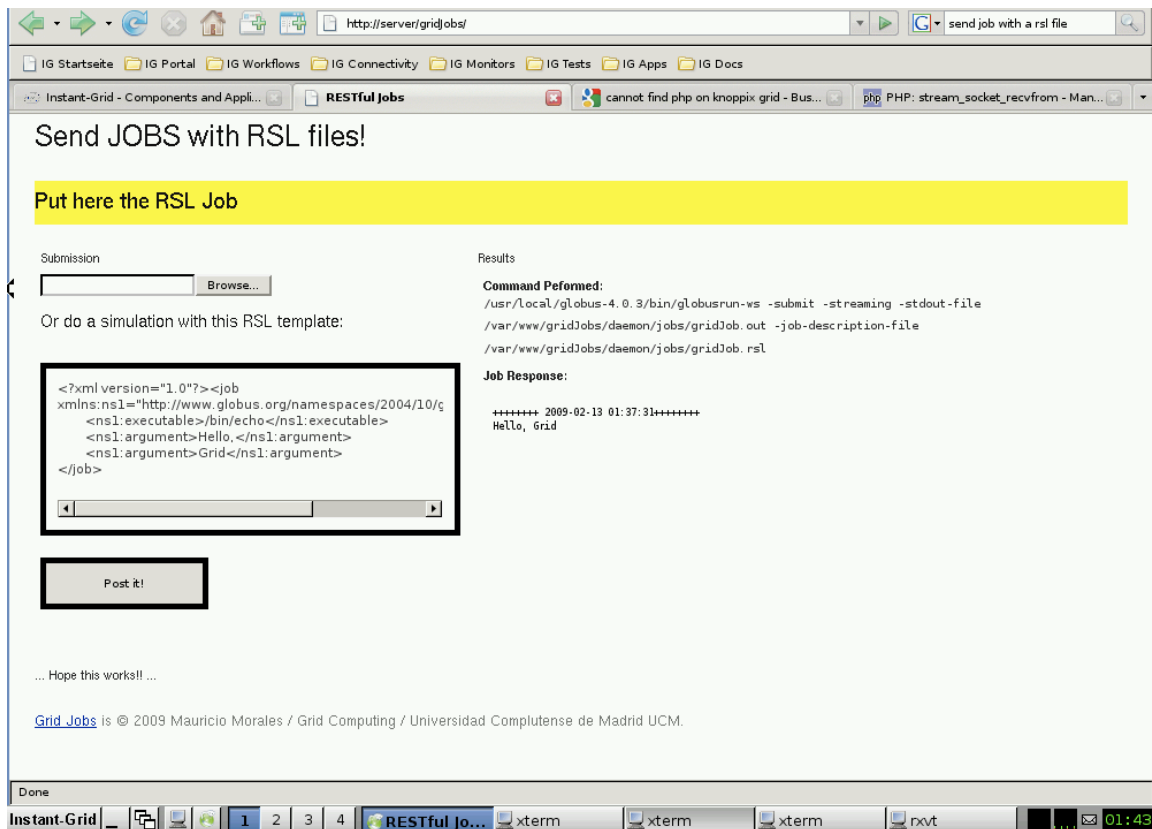


Figura 7. Pantalla de simulación

También se podría utilizar el método *submit* del Web Service para enviar un POST con la descripción del trabajo definida por medio de un RSL.

<http://server/gridJobs/?submit>

Este prototipo ofrece una funcionalidad básica de lanzar un trabajo a ejecución, una buena adición sería implementar el método *result* que reciba el ID del trabajo y retorne el resultado, para casos de procesos que tarden mucho su ejecución. Métodos para observar cambios y destruir trabajos

también serían útiles, en sí es muy interesante proveer de las opciones esenciales para la administración de trabajos.

A través de la consola de simulación se puede enviar un RSL de un trabajo:

```
<?xml version="1.0"?>
<job xmlns:ns1="http://www.globus.org/namespaces/2004/10/gram/job/description">
  <ns1:executable>/bin/echo</ns1:executable>
  <ns1:argument>Hello,</ns1:argument>
  <ns1:argument>UCM students!!!</ns1:argument>
</job>
```

Y se podrá ver una salida del sistema unos segundos después, se muestra el comando generado para la ejecución del trabajo y la salida que ha reportado globusrun-ws.

```
Command Performed:
/usr/local/globus-4.0.3/bin/globusrun-ws -submit -streaming -stdout-file
/var/www/gridJobs/daemon/jobs/gridJob.out -job-description-file
/var/www/gridJobs/daemon/jobs/gridJob.rsl

Job Response:

+++++++ 2009-02-13 01:45:48+++++++
Hello, UCM students!!!
```

**Figura 8.** Salida del sistema después del envío del RSL

Hasta este punto llega la implementación del prototipo, que como se ha visto podría extenderse y servir como una interfaz REST para la gestión de trabajos en Clouds y Grids computacionales.

## 7. Conclusiones, ventajas y desventajas

En este artículo me he enfocado en analizar los conceptos básicos del funcionamiento de GRAM y WSGRAM y el fin era conocer un poco la solución que Globus Toolkit provee para la gestión de recursos a través de Web Services.

Se ha propuesto una posible alternativa para la gestión de recursos, el servicio RESTful puede actuar como un objeto intermedio entre ordenadores locales y un Grid computacional.

La ventaja del prototipo se puede ver si analizamos un caso hipotético en el que un cliente (sin tener ningún tipo de software especial instalado) pueda ofertar por un recurso que se ejecutará en el Grid y que los ejecutables se encuentran también en el gestor (el servidor donde se encuentra el Web Service).

En este caso el cliente desde cualquier lugar podría enviar datos de autenticación y una solicitud para que se ejecute el proceso en el Grid y luego volver por la respuesta.

Suponiendo el caso en que se llegase a implementar el servicio completo en PHP con los métodos necesarios para delegar, lanzar, monitorear y finalizar trabajos, se podrían presentar inconvenientes al instalar el servicio en servidores administrados por terceros, pues las llamadas al sistema desde PHP están restringidas en muchos servidores por ser fuente de inseguridad.

Pienso que como una alternativa se podría llegar a una buena solución partiendo desde aquí.

### Descarga de Fuentes

Los fuentes se pueden descargar desde la siguiente dirección:

<http://maomorales.com/src/gridjobs.zip>

Si Usted está intentando configurar el ejemplo y tiene inquietudes o requiere colaboración, puede contactarme escribiendo a mi correo electrónico.

## Referencias

### **Globus Toolkit, Admin Guide.**

<http://www.globus.org/toolkit/docs/4.2/4.2.1/admin/quickstart/#q-prereq>

### **Key Concepts.**

<http://www.globus.org/toolkit/docs/4.0/execution/key/index.html>

### **WSGRAM Approach.**

[http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS\\_GRAM\\_Approach.html](http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Approach.html)

### **WSGRAM Developers Guide.**

<http://www.globus.org/toolkit/docs/4.0/execution/wsgram/developer-index.html>

### **Virtual Data Toolkit.**

<http://vdt.cs.wisc.edu/>

### **Instant-Grid.**

<http://www.instant-grid.org/>

### **Job Submission.**

<https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/JobSubmitWebServicesGram>

[http://www.bestgrid.org/index.php/Running\\_Jobs\\_via\\_Globus#Submitting\\_Jobs\\_via\\_Command\\_Line](http://www.bestgrid.org/index.php/Running_Jobs_via_Globus#Submitting_Jobs_via_Command_Line)

### **WSGRAM Java Scenarios.**

[http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS\\_GRAM\\_Java\\_Scenarios.html#s-wsgram-developer-scenarios-java-loadingjd](http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Java_Scenarios.html#s-wsgram-developer-scenarios-java-loadingjd)