Sistemas de Acceso a Grids

Grid Access Systems

Emilio Hernández
Universidad Simón Bolívar,
Departamento de Computación, Caracas 1080,
emilio@usb.ve

Resumen

Se presentan diversos esquemas relacionados con el acceso a los recursos de un grid. Se propone como un esquema muy flexible y poderoso el acceso a un grid a través de un servidor VPN, en vez de utilizar un User Interface estándar. En el esquema propuesto la funcionalidad de User Interface está localizada en la máquina del usuario y el grid gLite provee un servicio VPN para acceder a los demás componentes grid. El User Interface estándar es reemplazado por un User Interface Proxy de capa 2 (puente VPN). La ventaja principal de utilizar este esquema es la importante mejora en la calidad de la interacción, especialmente para los desarrolladores de aplicaciones, dado que la interacción con un grid a través de un User Interface tradicional está típicamente limitada a sesiones Web o ssh. Adicionalmente, el User Interface puede ser pre-empaquetado en una Máquina Virtual (VM), lo cual le evita al usuario final la carga de la instalación completa del User Interface.

Abstract

We present several schemes related to the access to grid resources. We propose as a highly flexible and powerful scheme the access to a grid through a VPN Server, instead of using a standard User Interface. In the proposed scheme the User Interface functionallity is located in the user machine and the gLite grid provides a VPN service for accesing grid components. The standard User Interface is replaced by a layer 2 (bridged VPN) User Interface Proxy. The main advantage of using such a scheme is the dramatic improvement in the interaction quality, especially for application developers, because the interaction with a grid through a traditional User Interface is typically limited to secure shell or web sessions. Additionally, the User Interface can be prepackaged in a Virtual Machine, which alleviates the burden of installing a complete User Interface to the user.

1. Introducción

La computación grid [1] ha evolucionado como una solución para compartir recursos heterogéneos, que pueden estar geográfica y organizacionalmente dispersos. Como plataforma, un grid es un sistema distribuido constituido por varios componentes situados en varias organizaciones. Estos componentes a su vez están ubicados en un perímetro virtual de seguridad, definido por las red subyacente (*overlay network*) que ellos conforman. El perímetro de seguridad del grid está implementado principalmente por la validación de máquinas y de usuarios, desarrollada durante cada conexión entre los componentes del grid. Lo definimos como virtual porque no se configura como un aislamiento delimitado físicamente, es decir, las máquinas pueden ser accesibles en la capa IP, pero sus servicios son inaccesibles sin los certificados digitales adecuados. Los usuarios típicamente se conectan desde una máquina ubicada fuera del perímetro de seguridad, normalmente desde una máquina local ubicada en sus sitios de trabajo.

En las interacciones comunes con las actuales versiones de producción de grid, tal como gLite [2] y GT4 [3], la identificación y autenticación de usuarios y máquinas en la Interfaz de Usuario (User Interface o UI) se hace a través de un interpretador de comandos seguro (ssh) o a través de portales de grid [4, 5]. Un servidor del interpretador de comandos seguro (ssh server) o un servidor web cumplen el rol de puerta de entrada al perímetro de seguridad del grid. Las sesiones bajo ssh limitan la interacción sólo a ejecuciones utilizando líneas de comandos. Los portales grid son mucho más útiles para desarrollar interfaces orientadas a aplicaciones específicas, que son más adecuadas para los

usuarios finales no especializados. Adicionalmente, los portales grid de propósito general han sido implementados y configurados con funcionalidades para la edición de archivos, compilación de programas y gestión de trabajos. Existen otras herramientas de interfaces con el grid, tales como Condor-G [6] y Netsolve/Gridsolve [7], entre muchas otras, las cuales normalmente requieren ser ejecutadas desde el perímetro de seguridad del grid. En gLite estos comandos están disponibles en una UI, donde el usuario se identifica y autentica previamente.

En esta propuesta, se considera que la interacción con un grid debería ser más transparente y más poderosa. En otras palabras, los usuarios generalmente quieren sentir que están interactuando directamente con aplicaciones que están corriendo en sus máquinas locales. Esto es especialmente útil para desarrolladores de aplicaciones, quienes necesitan utilizar una gran variedad de herramientas tales como depuradores gráficos y monitores de rendimiento, poco disponibles en la versión de aplicaciones web.

En este trabajo se presenta un forma diferente de interactuar con el grid, basada en el uso de Máquinas Virtuales estrechamente relacionadas con la plataforma grid y localizadas fuera de su perímetro de seguridad virtual. El esquema propuesto sobre un grid de tipo gLite, provee un servidor VPN, que denominamos UI Proxy, en modo puente (*bridged*) para conectar UI remotas, de modo que puedan acceder a los componentes grid. Usamos el término "proxy" en el sentido de que el servidor VPN realiza la representación de la máquina remota, pero lo hace al nivel de la capa 2, ya que el modo de conexión es *bridged*. Esta solución es fácil de instalar y de utilizar en una plataforma local, y es lo suficientemente segura para garantizar el acceso adecuado a los componentes grid.

El resto de este artículo está estructurado de la siguiente manera: en la sección 2 se describen las soluciones comunes hoy en día para acceder a los recursos de un grid. En la sección 3 se presenta una opción alternativa, y se discuten opciones de implementación. En la sección 4 se presenta una clasificación de las herramientas que podrían utilizarse con la solución de acceso propuesta y en la sección 5 se muestran algunos ejemplos ejecutados sobre el esquema propuesto. En la última sección se presentan las conclusiones y el trabajo futuro.

2. Esquemas de Interacción con un grid de tipo gLite

Existen dos esquemas principales utilizados para acceder a grids de tipo gLite. El más simple está basado en acceso a través de interpretadores de comandos seguros (ssh) hacia la interfaz de usuario grid (UI). Este esquema se muestra en la figura 1. Una vez que el usuario se ha identificado y autenticado en la UI, ésta puede ejecutar los comandos de grid, para la gestión de la seguridad, la gestión de trabajos, la transferencia de archivos y algunas otras funciones. Los requerimientos locales están limitados a tener un cliente ssh, dado que los comandos para las operaciones en el grid están instalados en la UI. Los archivos de programas deben ser enviados desde la máquina local hacia la UI. Los archivos de datos deben ser enviados hacia la UI o hacia un elemento de almacenamiento grid (Storage Element). Básicamente la ventaja principal de este método de acceso es que los usuarios finales no tienen que instalar los componentes grid en sus máquinas locales. Por otro lado, este esquema es muy restrictivo debido a que la interacción es sólo desarrollada a través de una consola ssh. Otra desventaja es que la UI debe tener suficiente espacio para un determinado número de cuentas de usuario, y eventualmente la UI podría sobrecargarse con las operaciones de compilación y transferencia de archivos. Desde el punto de vista de la seguridad, este esquema es muy básico, dado que el control de acceso está basado en la cuenta de usuario estándar en la UI, por ejemplo, en el par nombre de usuario/contraseña de la cuenta.

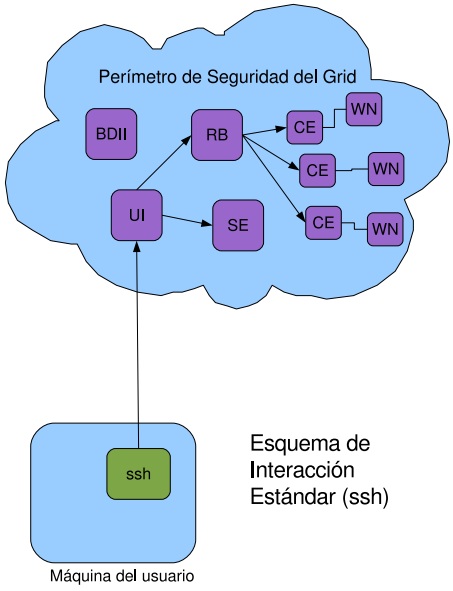

Figura 1. Un enfoque de acceso ssh

Otro esquema de interacción, mostrado e la figura 2, está basado en portales web, los cuales típicamente han sido desarrollados bajo una plataforma basada en portlets [4, 5]. Estas plataformas son útiles para el

desarrollo de aplicaciones interactivas seguras, accesibles por medio de navegadores web. El servidor web despliega los portlets, que a su vez invocan los comandos de grid para la ejecución y gestión de tareas. La interfaces pueden estar orientadas a las necesidades específicas de los usuarios finales.

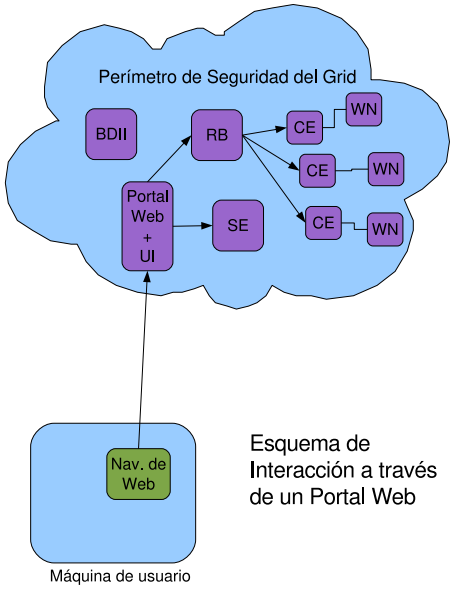

Figura 2. Un enfoque de acceso Portal Grid

Estos dos esquemas son razonablemente adecuados desde el punto de vista de la seguridad, porque las conexiones al UI, ubicado dentro del perímetro de seguridad del grid, son hechas a través de una conexión TCP simple, típicamente encriptada a través de SSL. Esto es útil para los controles de acceso a través de cortafuegos que protegen el sitio donde el punto de entrada al grid está instalado. Ambos esquemas normalmente tienen un método de autenticación simple, basado en el nombre de usuario y clave. Una vez que se auntentica dentro de una UI, la autenticación y autorización dentro del perímetro de seguridad del grid se hace con la ayuda de estrategias PKI y X.509.

2.1. Portales y Pasarelas de Grid

Un portal grid es una manera muy conveniente para acceder a los diversos recursos del grid a través de un navegador estándar. Debe proveer una interfaz sencilla y consistente de ambientes complejos de diversos sistemas, donde el usuario pueda obtener resultados para su trabajo rápidamente. Algunas veces se hace una diferencia entre Pasarelas de Grid (Grid gateways) y Portales de Grid (Grid Portals), significando que las primeras son más estáticas y normalmente orientadas a una área específica y los segundos son más dinámicos, configurables por el usuario final.

Las funcionalidades de un Portal de Grid deben incluir información sobre los recursos disponibles, la posibilidad de iniciar sesión en el sistema que permitirá tener acceso a datos específicos, información sobre la cuenta y los proyectos que se tienen, soporte para la creación y envío de trabajos, así como su modificación y eliminación, entre otros. Los portales se elaboran utilizando lenguajes de programación y herramientas estándar, basándose en herramientas (*kits*) de desarrollo: bibliotecas de funciones, aplicaciones y conjuntos de plantillas que facilitan la elaboración de portales de usuario y/o portales de aplicación.

2.2. Portlets

Los *portlets* son componentes que se pueden integrar en un portal web, similares a los *servlets*, aunque no se pueden direccionar con un URL ni definen páginas web completas. Fueron diseñados especialmente para ser agregados en el contexto de una página compuesta, capaz de procesar peticiones de los usuarios y devolver fragmentos de contenido (en lenguajes como HTML, WML, VoiceXML entre otros) que luego conforman la página de un portal. Son reutilizables y proporcionan acceso a contenido basado en Web, aplicaciones, materiales de contenido agrupado y otros recursos [8].

Bajo la perspectiva de un usuario, los *portlets* son las ventanas contenidas en el portal que proporcionan un determinado servicio o una determinada información, por ejemplo, material de agenda o noticias. Bajo una perspectiva de despliegue de aplicaciones, los *portlets* son módulos conectables diseñados para ejecutarse en el contenedor de portlets, que a su vez se ejecuta en el servidor del portal.

El contenedor de *portlets* proporciona un entorno de ejecución en el que se maneja el ciclo de vida de los portlets, creándoles instancias, utilizándolas y, por último, destruyéndolas. Adicionalmente, suministra servicios comunes y facilita mecanismos de persistencia a todos los portlets. Sin embargo, no es autónomo como lo sería un contenedor de *servlets*. Uno de los contenedores de portlets más utilizado es Gridsphere [9], que provee dos implementaciones distintas para portales. La primera es la JSR 168 portlet API estándar, que permite la interoperabilidad entre portlets y portales definiendo una serie de APIs para los portlets y haciendo estándar las preferencias, información del usuario, instalación o seguridad [10]. La segunda está basada en el IBM WebSphere Portlet API, que proporciona funciones esenciales de portal que le permiten construir

con rapidez portales ampliables.

Entre los ambientes de *portlets* de código abierto más importantes se puede mencionar OGCE (Open Grid Computing Environment) [4], hecho sobre la base de esfuerzos previos, como el GridPort Toolkit [11], originalmente hecho en el TACC (Texas Advanced Computing Center). El ambiente OGCE puede correr bajo los contenedores Gridsphere y uPortal. Implementa los portlets sobre el Cogkit de Java, lo que le ofrece portabilidad a través de varias versiones de Globus (GT2, GT3 y GT4). Entre los *portlets* ofrecidos se encuentran, de autenticación, basado en MyProxy, para acceder a los recursos del Grid usando GSI, de manejo de archivos, de envío de tareas vía GRAM/RSL, de envío de tareas a través de un agente Condor, de manejo de tareas enviadas y de descubrimiento de recursos, entre otros.

El ambiente GridPortlets [5] usa GridSphere y un API propietario para desarrollo de *portlets*. Estos APIs proveen una interfaz de alto nivel que proveen una marco uniforme para acceder a los servicios y recursos, así como una interfaz de alto nivel para acceder a la infraestructura Globus via el COG kit. En cuanto a los *portlets* suministrados para grid, provee unos portlets básicos para login, personalización de perfiles, adaptación del ambiente de interacción, creación de usuarios, grupos y manejo de *portlets*. Desde el punto de vista de acceso a los recursos del grid, las funcionalidades que provee GridPorlets son similares a las que provee OGCE.

3. Propuesta del Esquema de Interacción

Exploramos la posibilidad de utilizar esquemas alternativos de interacción con el grid, que permitan a los usuarios tener un ambiente de desarrollo, con todas las herramientas disponibles para la edición de programas, la compilación y envío de tareas hacia el grid, como también la ejecución de programas interactivos que utilicen al grid como base de cómputo. Los usuarios finales podrán levantar aplicaciones corriendo localmente y utilizando como base de cómputo la plataforma disponible a través del grid. Esto puede subsanar las limitaciones de la interacción a través de la consola ssh o del Portal Web.

La alternativa propuesta está basada en la instalación de un UI en una máquina virtual (VM), tal como Xen [12], VMplayer [13] o Qemu [14] y colocar esa máquina virtual fuera del perímetro de seguridad del grid. Como un caso particular, la VM puede ser colocada en la máquina del usuario final. Llamamos *VM-UI* a este nuevo componente de un grid gLite, localizado fuera del perímetro de seguridad estándar. La razón principal para utilizar una VM y no instalar la aplicación del UI directamente en la máquina local son: (1) el software de UI puede estar disponible para pocas plataformas, que pocas veces coincidirán con las plataformas de los usuarios y (2) el usuario puede descargar una versión preinstalada de un VM-UI, la cual tendría cargados todos los componentes necesarios para la funcionalidad de UI. Otra ventaja de utilizar la VM para la conexión al grid es que las interfaces de las nuevas aplicaciones podrán ser desarrolladas como una aplicación de escritorio (ejemplo: utilizar interfaces en forma de ventanas tales como Qt o GTK) en vez de utilizar una interfaz de Portal Web. Esto quiere decir que será más fácil para los usuarios desarrollar interfaces personalizadas. Este esquema simple se representa en la figura 3.

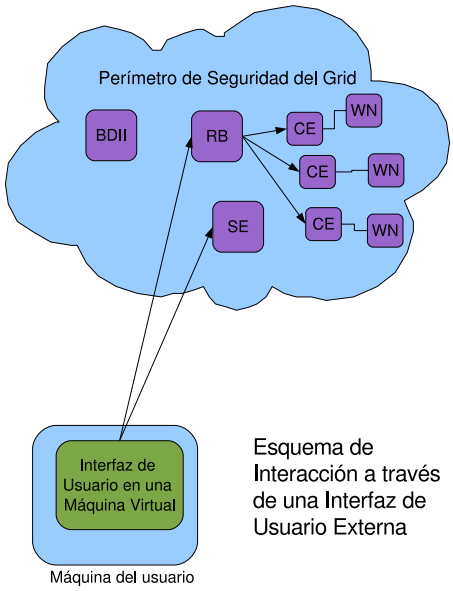

Figura 3. Interfaz de Usuario (UI) Externa

Sin embargo, bajo este esquema, los cortafuegos a cargo del perímetro de seguridad grid deberían estar configurados para permitir las conexiones desde una gran variedad de orígenes, directamente a varios componentes grid, tales como el coordinador de recursos (Resource Broker o RB) y los elementos de almacenamiento (SE). Adicionalmente, los servicios de conexión de la parte externa al perímetro deberían ser instalados en varios componentes grid. En consecuencia, este esquema puede ser mejorado a través de la colocación de un Proxy en la mitad de tales conexiones, el cual podría validar todas las conexiones al grid utilizando el mismo procedimiento que los UI estándar, y redireccionando las conexiones para alcanzar los componentes internos

del grid. Este esquema se presenta en la figura 4.

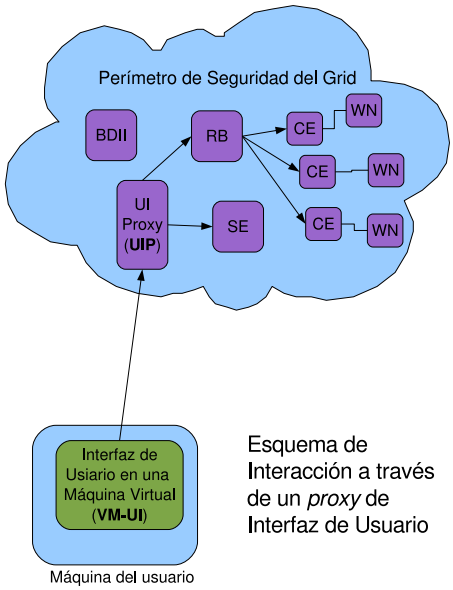

Figura 4. Proxy para Interfaces de Usuario

Se denomina User Interface Proxy (UIP) a este nuevo componente de un grid gLite, y tiene las siguientes funciones:

1. Permitir conexiones seguras (vía IPSec o VPN utilizando túneles SSL) desde componentes VM-UI, basándose en el mismo esquema de validación de certificados utilizados dentro del perímetro de seguridad.

2. Una vez que la conexión segura se ha validado, la VM-UI puede utilizar un modo de conexión tipo puente (*bridged*), la cual permitirá que se configura en la máquina donde se instale el UI una interfaz virtual con una dirección IP que puede ser enrutado a todos los posibles destinos dentro del grid, justamente para aquellos casos donde los componentes grid están situados en redes diferentes (tal es el caso de Internet 2)

Este esquema no sólo permite a los administradores de sistemas mantener el control de las conexiones entrantes, sino también tendrían otras ventajas, tales como:

1. La instalación y el mantenimiento de servidores IPSec o VPN se concentra en un punto único.

2. Un mecanismo NAT puede ser implementado para solventar el problema de las limitaciones existentes en cuanto a la numeración IP dentro del grid.

3. La UIP podría implementar políticas de calidad de servicio (QoS) a nivel de red.

4. Ambientes de Interacción

Trabajar directamente en una UI, o equivalentemente en una VM-UI, abre muchas posibilidades al desarrollo de aplicaciones para un grid. La razón es que se trabaja directamente en un ambiente gráfico, tipo KDE o Gnome, con la ayuda de todas las herramientas interactivas, de edición de aplicaciones y de desarrollo que ya están presentes en estas plataformas de trabajo.

4.1. Herramientas básicas

Las herramientas más directas para la elaboración de aplicaciones localmente, que utilizan recursos de Grid, son los *toolkits* que se instalan directemente en un UI. Los *toolkits* de mayor difusión han sido los derivados de la iniciativa denominada Globus [15], y se denominan *Globus Toolkits (GT)*, en versiones GT2, GT3 y GT4 (GT3 prácticamente en desuso), así como el *toolkit* llamado GLite, derivado de GT2, pero complementado con otras herramientas provenientes de diversos proyectos [2]. En general estos conjuntos de herramientes proveen comandos ejecutables a nivel de *shell*, que pueden utilizarse para hacer programas de tipo *script* (por ejemplo, en perl o python), y que normalmente tienen archivos de entrada que definen las tareas a ejecutarse en el grid (archivos en JDL, o Job Description Language). Los comandos de estos conjuntos de herramientas pueden dividirse en varios grupos, de acuerdo a la funcionalidad general con la que están vinculados: manejo de certificados digitales, manejo de tareas simples o compuestas y manipulación de datos y archivos, entre otros. Algunas implementaciones de portales hacen uso de los *toolkits* como mecanismo de interacción con el grid.

4.2. APIs

Más eficiente que usar los *toolkits* como base de la programación, es realizar la programación de aplicaciones sobre APIs definidos para el uso de los recursos de Grid. Los APIs pueden proveerse para una diversidad de lenguajes, aunque típicamente existen para Java y C++, y en menor medida para Python y Perl. Estos APIs existen para las diferentes plataformas (gLite, GT2, GT4) en forma de mecanismos básicos de envío y gestión de tareas (GRAM), gestión de seguridad (GSI), manejo de datos (GridFTP), etc. Estos APIs están más orientados a desarrolladores de aplicaciones que se ejecutarán en el Grid, aunque hay otros APIs orientados a

desarrolladores de componentes (*middleware*) del mismo grid, como APIs de acceso a datos de uso del grid (MDS, GIS), por ejemplo para implementar componentes internos del grid que realicen metaplanificación.

Los mecanismos tradicionales para la programación en sistemas distribuidos pueden utilizarse perfectamente en una VM-UI. Por ejemplo, el paradigma de llamada a procedimiento remoto (RPC) se ha propuestop y utilizado para la programación en estos ambientes. Un estándar llamado GridRPC [16], utilizado por plataformas como Gridsolve [7] y Ninf-G [17], tiene, además de los mecanismos estándares de ejecución remota bloqueante, mecanismos de ejecución no bloqueante, llamadas múltiples al mismo procedimiento (para invocación paralela en granjas de procesadores) y extensiones para adaptarse a la ejecución sobre sistemas batch (manejo de IDs de tareas).

Otros APIs de Grid intentan ofrecer una interfaz de más alto nivel, que permita ejecutar no solamente tareas en forma aislada, sino interpretar estructuras de control de tipo flujo de trabajo en forma de DAGs (Grafos Dirigidos Acíclicos, en sus siglas en inglés), aparte de otras abstracciones que facilitan ciertos tipos de trabajo a los usuarios, o a los desarrolladores de aplicaciones para ejecutar tareas en el grid. Tal es el caso de los CogKits de Java o Python [18].

4.3. Herramientas de Gestión de Tareas

El envío simple de tareas a un grid puede gestionarse con herramientas relativamente sencillas como Condor/G [6]. La secuenciación de comandos bajo un esquema de flujo de trabajo o *workflow*, que puede ser lineal o seguir una arquitectura de tipo DAG, también ha sido objeto de mucha investigación, y se han propuesto herramientas para definir workflows [19, 20, 21] entre otros esquemas y técnicas específicas, de mucha utilidad, en particular en entornos grid. DAGMan [22] es un metaplanificador centralizado que organiza tareas de Condor en forma de DAGs.

Otras herramientas ofrecen ayuda a los usuarios para ejecutar la misma aplicación muchas veces, con barrido de parámetros (*parameter sweeping*). Por ejemplo, Nimrod/G [23] es una herramienta que se enfoca en el manejo y planificación de procesos sobre recursos distribuidos, enfatizando un concepto que denominan economía computacional. Otro ejemplo, *APST* (AppLeS Parameter Sweep Template) [24], asigna aplicaciones con barrido de parámetros en un grid, basándose en *AppLeS*, una plataforma que permite al usuario implementar su propio planificador con base en información de la aplicación misma y en información dinámica, que recaba a tiempo de ejecución.

El envío de tareas a un Grid de un modo totalmente transparente (sin necesidad de crear descriptores de tareas en JDL, por ejemplo) también ha sido objeto de investigación. Por ejemplo, SUMA/G [25] es una plataforma para ejecuciópn de clases de Java de un modo transparente, donde las clases y los datos están en el directorio local del usuario y son cargados en la plataforma remota de ejecución por demanda. Los componentes de gestión de tareas de SUMA/G coexisten con los de Globus (no dependen de éstos) desde el punto de vista de que (1) utiliza los mismos mecanismos de autenticación y autorización que Globus (VOMS, GSI) y (2) acude a los sistemas de información de Globus (MDS, GIS) para registrar los componentes y para solicitar recursos ociosos para ejecución de tareas.

4.4. Ambientes Integrados de Desarrollo y Ejecución

Hay ambientes que permiten gestionar el envío de tareas y datos al grid, con interfaces más elaboradas, por ejemplo a través de ventanas. El ambiente denominado Migrating Desktop [26] tiene interfaces gráficas para definición de tareas, envío de tareas a ejecución, control de tareas enviadas, transferencia de datos, etc. Se ejecuta localmente como una aplicación que integra todas las herramientas que provee. Tener diferentes herramientas integradas entre sí para la interacción con los componentes del grid es similar a tener varios *portlets* integrados en un portal de grid. Una perspectiva interesante es desarrollar herramientas más independientes, pero integradas a través del ambiente gráfico (KDE o Gnome), como se propone en una herramienta llamada Grenade [27].

También existen ambientes de trabajo de tipo SDK (Software Development Kits) orientados a asistir a los usuarios desde el momento del desarrollo de las aplicaciones. Usualmente están integrados a mecanismos de utilización de los recursos del grid, por ejemplo, aquellos que permiten crear proyectos que contengan los datos administrativos del mismo (como descripción de las tareas en JDL), conectarse a los recursos del grid para saber el estado de los mismos, enviar las tareas (simples o compuestas) al grid y visualizar los resultados. Ejemplos de estos ambientes pueden ser g-Eclipse [28], el Grid Programming Environment de Intel [29].

5. Pruebas

Hemos probado varias VM-UI para acceder a una instalación de un grid gLite localizado en una red privada de alta velocidad entre centros de investigación, llamada *Reacciun 2* [30]. Las actuales VM-UIs tienen

instalados el sistema de operación Scientific Linux 3 y se conectan al grid gLite a través de una conexión OpenVPN en modo puente (*bridged*). El acceso se hace a través de un UIP de dominio de red dual (dual homed) con OpenVPN. La conexión se establece en tiempo de inicio del VM-UI, utilizando los certificados de máquina asignados para esa VM-UI, que corresponden con los certificados de máquina que usa el grid gLite.

Se han hecho varias pruebas de instalación, las cuales se intentaron combinar con soluciones de túneles conocidas, distintas plataformas VM y sistemas operativos huéspedes en las VM. Los resultados preliminares muestran que existen muchos puntos álgidos en cuanto a la interoperabilidad cuando se instalan las herramientas de túneles seguros sobre diferentes VM (Xen, VMPlayer o qemu) corriendo diferentes distribuciones de Linux, tales como Scientific Linux 3 o Debian (que son dos distribuciones de Linux compatibles con el software de gLite UI). Se espera que estos problemas serán resueltos en un futuro cercano.

En términos de funcionalidad, necesaria para el desarrollo de una solución que permita a los usuarios finales instalar una VM-UI en sus máquinas, se muestra un análisis cualitativo en la tabla 1. No obstante, debido a la naturaleza de la distribución de la VM-UI entre diversos usuarios, tendemos a favorecer una distribución con qemu, debido a que es una solución de licencia GPL.

(VM, Sist. Op. (Anfitrión)	**Facilidad de instalación**	**Portabilidad**
(VMPlayer, Linux)	Media	Buena
(VMPlayer, Windows)	Fácil	Buena
(qemu, Linux)	Fácil	Pobre
(qemu, Windows)	Media	Pobre

Cuadro 1. Comparación cualitativa entre las combinaciones (VM, Sistema Operativo)

Se realizaron algunos experimentos con respecto al consumo y rendimiento asociado al ancho de banda entre un VM-UI y un UIP. Se condujeron los experimentos en un ambiente controlado, una red local aislada. La UIP se ejecutó en un procesador Intel 3.4 Ghz con Ethernet de 100Mbps, corriendo Ubuntu 6.10, y el VM-UI estaba corriendo en un equipo con un procesador Intel Core Duo processor, 1.83 Ghz, Ethernet 100Mbps. El sistema operativo huésped en el equipo que ejecutó la VM-UI fue Ubuntu 6.10, y el sistema operativo de la VM-UI fue Scientific Linux 3.0. Se hizo un experimento simple, se transfirió un archivo de 140MB utilizando el comando gridFTP. Se usaron dos de los algoritmos de cifrado más utilizados hoy día, 3DES y Blowfish. Dependiendo del algoritmo de cifrado usado y a la máquina virtual usada, los tiempos de transmisión fueron afectados de diferentes formas. La tabla 2 muestra los resultados para qemu.

(VM, Host OS)	**3DES 192 bits**	**Blowfish 128 bits**
(qemu, Linux) Con acelerador	21,06 Mbps	22,90 Mbps
(qemu, Windows) Sin acelerador	5,22 Mbps	8,61 Mbps

Cuadro 2. Rendimiento de Ancho de Banda de (VM, Host OS) combinaciones sobre una red local a 100Mbps

Hemos hecho pruebas también con VMWare, que no se reportan debido a que en diferentes ejecuciones hemos obtenido velocidades de transmisión muy diferentes, evidenciando un problema de interoperabilidad entre el protocolo gridFTP (que abre varias conexiones simultáneas con el propósito de acelerar la transferencia), VMWare y OpenVPN. Suponemos que este problema será resuelto en un futuro cercano. En resumen, se puede inferir de estos resultados que hay importantes diferencias entre opciones, que se reflejan en la velocidad de transmisión. El uso del acelerador de qemu, denominado kqemu, tiene un impacto significativo en los tiempos de transmisión. La razón es que el impacto en la velocidad de transmisión varía, entre otras cosas, por la velocidad con que la máquina virtual ejecuta los algoritmos de encriptamiento involucrados en la transmisión.

Otras pruebas nos pueden ayudar a elegir el tipo de tunel que puede utilizarse para la conexión a una red privada virtual usando una Máquina Virtual. En la figura 3 pueden apreciarse las diferencias si se utiliza Openswan (una implementación de IPSec) y OpenVPN. Claramente OpenSwan es más rápido, porque trabaja a más bajo nivel, mientras que OpenVPN se conecta utilizando interfaces de sockets seguros. Se aprecia que las diferencias de velocidad pueden ser muy significativas, atribuibles en buena medida a la capacidad de las máquinas virtuales en procesar las comunicaciones cuando se utilizan mecanismos de comunicación o de E/S diferentes.

6. Conclusiones y Trabajo Futuro

En este trabajo mostramos un esquema para acceder a un grid bajo la plataforma de gLite a través de

(VM, Tunel)	Ancho de Banda (sin cifrador)
(VMPlayer, OpenVPN)	4.69 Mbps
(VMPlayer , Openswan)	23,30 Mbps
(qemu, OpenVPN)	3.20 Mbps
(qemu, Openswan)	15.26 Mbps

Cuadro 3. Transferencias entre combinaciones (VM,Tunel) en una red local, usando scp

un servidor VPN, que denominamod Interfaz Proxy del Usuario (UI Proxy), como alternativa al uso de la interfaz de usuario (UI) tradicional o de un Portal de Grid. En el esquema propuesto el UI está ubicado en la máquina del usuario, corriendo localmente en una instancia de una máquina virtual, sea ésta VMWare o qemu. La máquina virtual se conecta al UI Proxy a través de un túnel seguro, el cual se establece en tiempo de inicio del sistema operativo. La principal ventaja de utilizar este esquema es la mejora en las facilidades de interacción del usuario con el grid, ya que la interacción a través del esquema tradicional está limitada a las conexiones seguras utilizando interpretadores de comandos (shells) tal como ssh, o a través de la interacción con un navegador web. Nuestros resultados muestran que la interacción entre máquina virtual, sistema operativo anfitrión y mecanismo de encriptamiento del tunel seguro, deben tomarse en cuenta para la implementación de un sistema como el propuesto.

Estamos trabajando en la elaboración de una VM-UI en la que se distribuya, de forma preinstalada, un conjunto de herramientas de acceso al grid, como por ejemplo, interfaces de ventana, monitoreo del grid y APIs para envío y manejo de tareas, entre otros.

Referencias

[1] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3), 2001.

[2] gLite. Lightweight Middleware for Grid Computing, 2006. http://glite.web.cern.ch/glite/.

[3] Ian T. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006.

[4] D. Gannon, G. Fox, M. Pierce, B. Plale, G. von Laszewski, C. Severance, J. Hardin, J. Alameda, M. Thomas, and J. Boisseau. Grid portals: A scientist's access point for grid services (draft 1), September 2003. GGF working draft, http://forge.gridforum.org/projects/ggf-editor/document/GCE-Portal-working-draft/en/1/GCE-Portal-working-draft.pdf.

[5] . Gridsphere Portal Framework. http://www.gridsphere.org.

[6] The Condor Team. Condor High Throughput Project. http://www.cs.wisc.edu/condor/condorg.

[7] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Seymour, K. Sagi, Z. Shi, and S. Vadhiyar. Users' Guide to NetSolve V1.4.1. Innovative Computing Dept. Technical Report ICL-UT-02-05, University of Tennessee, Knoxville, TN, June 2002.

[8] Vieregger C. Develop Java Portlets, JavaWorld. http://www.javaworld.com/javaworld/jw-02-2003/jw-0207-iviews.html.

[9] J.Ñovotny, M. Russell, and O. Wehrens. GridSphere: a portal framework for building collaborations. *Concurrency - Practice and Experience*, 16(5):503–513, 2004.

[10] Abdelnur A., Hepper S. Portlet Specification version 1.0. http://www.jcp.org/en/jsr/detail?id=168.

[11] M. Thomas, J. Boisseau, S. Mock, M. Dahan, K. Mueller, and D. Sutton. The GridPort Toolkit Architecture for Building Grid Portals. In *Proc. of the 10th IEEE Internat. Symp. on High Performance Distributed Compumputing*, August 2001.

[12] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R.Ñeugebauer. Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, October 2003.

[13] Mick Bauer. Vmware workstation 5.5 for linux hosts. *Linux Journal*, 2006(145), May 2006.

[14] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *USENIX 2005 Annual Technical Conference, FREENIX Track*, pages 41–46, 2005.

[15] The Globus Alliance. The Globus Toolkit, 2006. http://www.globus.org/.

[16] Keith Seymour, Hidemoto Nakada, Satoshi Matsuoka, Jack Dongarra, Craig Lee, and Henri Casanova. Overview of GridRPC: A Remote Procedure

Call API for Grid Computing. In *Grid Computing - GRID 2002 : Proceedings of the Third International Workshop on grid Computing*, pages 274–278, Baltimore, USA, 2002.

[17] Y. Tanaka, H.Ñakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka. Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing. *Journal of Grid Computing*, 1(1):41–51, 2003.

[18] Gregor Laszewski and Mike Hategan. Workflow concepts of the java cog kit. *Journal of Grid Computing*, 3(3-4):239–258, September 2005.

[19] Soonwook Hwang and C. Kesselman. Grid workflow: a flexible failure handling framework for the grid. In *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*, pages 126–137, 2003.

[20] Junwei Cao, S. A. Jarvis, S. Saini, and G. R. Nudd. Gridflow: workflow management for grid computing. In *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, pages 198–205, 2003.

[21] K. Amin, G. von Laszewski, M. Hategan, N. J. Zaluzec, S. Hampton, and A. Rossi. Gridant: a client-controllable grid workflow system. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference*, pages 10 pp.+, 2004.

[22] The Condor Team. DAGMan (Directed Acyclic Graph Manager), 2007. http://www.cs.wisc.edu/condor/dagman/.

[23] Abramson D. Buyya R. and Giddy J. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. *in Proceedings of* the HPC ASIA 2000, the 4th. International Conference on High Performance Computing in Asia-Pacific Region, Beijing, China, IEEE Computer Society Press, USA, 2000.

[24] Henri Casanova, Graziano Obertelli, Francine Berman, and Rich Wolski. The apples parameter sweep template: User-level middleware for the grid. In *Proceedings of SuperComputing.*, 2000.

[25] Yudith Cardinale, Carlos Figueira, Emilio Hernández, Eduardo Blanco, and Jesús De Oliveira. Middleware support for Java applications on Globus-based grids. *Lecture Notes in Computer Science*, 4459:627–641, 2007. International Conference on Grid and Pervasive Computing(GPC'2007).

[26] M. Kupczyk, R. Lichwala, N. Meyer, B. Palak, M. Plociennik, M. Stroinski, and P. Wolniewicz. The migrating desktop as a gui framework for the .ªpplications on demandçoncept. *Lecture Notes in Computer Science*, 3036:91–98, 2004.

[27] J. Marsh, S. Pettifer, D. Hanlon, S. Pickles, J. MacLaren, and M. Foster. GRENADE: a Grid Enabled Desktop Environment. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2004.

[28] g-Eclipse team. g-Eclipse Project. http://www.geclipse.org, 2007.

[29] Intel Software. Grid Programming Environment, an Overview (White Paper).

[30] Ministerio de Ciencia y Tecnología de Venezuela. Red Académica de Centros de Investigación y Universidades: REACCIUN 2. http://www.reacciun2.edu.ve.