UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

Departamento de Arquitectura de Computadores y Automática

TESIS DOCTORAL

Arquitectura para el aprovisionamiento dinámico de recursos computacionales

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Constantino Vázquez Blanco

Directores

Eduardo Huedo Cuesta
Ignacio Martín LLorente

Madrid, 2013

Arquitectura para el Aprovisionamiento Dinámico de Recursos Computacionales

TESIS DOCTORAL

Constantino Vázquez Blanco

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Junio 2012

Arquitectura para el Aprovisionamiento Dinámico de Recursos Computacionales

Memoria que presenta para optar al título de Doctor en Informática

Constantino Vázquez Blanco

Dirigida por los Doctores

Eduardo Huedo Cuesta
Ignacio Martín Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Junio 2012

A mis padres y hermana, por estar ahí siempre
A mis amigos, por hacerlo llevadero
A Virginia, por todo.

Agradecimientos

En el transcurso del trabajo de esta tesis doctoral he tenido el placer y la suerte de encontrarme muchas personas que, gracias a su fe, apoyo y guía, han hecho posible que este proyecto se haya completado felizmente.

Quiero agradecer especialmente la confianza en mi demostrada por **Ignacio M. Llorente**, supervisor de esta tesis y director del grupo de investigación (**dsa-research.org**) en el cual se ha llevado a cabo todo el trabajo expuesto en este documento. Asimismo quiero reconocer el inmenso trabajo llevado a cabo por el **Eduardo Huedo**, también supervisor de esta tesis, así como la inspiración prestada y el ejemplo de la satisfacción del buen hacer que profesa, responsable de la calidad de este documento (en aquellos aspectos en los que ésta se manifieste en su cara positiva). Me gustaría hacer una mención especial para el tercer pilar en el que esta tesis se sustenta, el **Rubén Santiago**, por su ayuda en el día a día, por sus consejos a la hora de interpretar resultados y por sus explicaciones pacientes de los aspectos más intricados de la tecnología que se maneja en este trabajo.

Otros dos grandes culpables de que esta tesis se haya llevado a cabo son mis padres, **Pili y Tino**, así como mi hermana **Marta**. Sin este ambiente libre de grandes preocupaciones familiares que han (y hemos) conseguido crear no habría sido posible este trabajo, o al menos no en su actual forma. Gracias de corazón.

No podría dejar de agradecerle todo el apoyo (incondicional y desinteresado) brindado por la persona más importante de mi vida estos últimos años, **Virginia**. Gracias por estar ahí, por no quejarse nunca en las largas tardes de domingo en las que me enclaustraba, y sobre todo por animarme en todo momento a terminar esta carrera de fondo.

De especial importancia para el desarrollo de este trabajo ha sido todo el equipo de Globus. Quisiera agradecer a **Ian Foster** (el padre del Grid) su gran trabajo al frente de la alianza, y su apoyo a la herramienta GridWay, fruto del trabajo del grupo de investigación del que formo parte y que a su vez forma parte orgullosamente del standard de facto en middleware Grid: el Globus Toolkit. Gran cantidad de culpa de la excelente relación del proyecto Globus y el grupo **dsa-research.org** es de **Borja Sotomayor**, flamante nuevo profesor de la Universidad de Chicago. Tampoco me olvido del resto

de personas que hicieron y hacen posible Globus: **Stuart Martin, Martin Feller, Charles Bacon, Jennifer Schopf, etc**.

¿Y qué decir de de mis compañeros de trabajo? **Javier Fontán** merece una mención especial en estos agradecimientos, ya que ha contribuido invaluablemente al desarrollo de esta tesis. Él y **José Luis vázquez** son los dos compañeros más antiguos que tengo en el grupo de investigación, siendo éste último el encargado de darme la bienvenida al grupo de investigación, labor desempeñada a la perfección por este león social. A los demás compañeros (**Jaime Melis, Carlos Martin y Daniel Molina**) quiero agradecerles el ayudar a crear el ambiente de trabajo único que se vive en el algún día seguro que famoso 308b, el despacho donde la maquinaria de OpenNebula se ajusta día a día.

Con cariño mando un saludo también a mis amigos, especialmente los que residen en Madrid, como **Dani, Xoán, Pablo y Javi**, los cuales, aunque no lo sepan, han contribuido a este trabajo al poder descansar de él de vez en cuando ;). Gracias también a **Miguel Vázquez**, gurú de muchas cosas, siempre dispuesto a echar un cable.

Por último, me gustaría agradecer a los autores de **TeXiS**, la plantilla Latex para tesis doctorales diseñada por Marco Antonio y Pedro Pablo Gomez-Martin, la cual me ha resultado de gran ayuda para escribir este documento.

Acerca de este documento

Esta tesis doctoral se presenta como compendio de publicaciones editadas, de acuerdo con el epígrafe 4.4 de la Normativa de desarrollo de los artículos 11, 12, 13 y 14 del Real Decreto 56/2005, de 21 de Enero, por el que se regulan los estudios universitarios oficiales de postgrado de la Universidad Complutense (Aprobado en Consejo de Gobierno con fecha 13 de Junio de 2005 y publicado en el BOUC con fecha de 5 de Julio de 2005).

Los artículos que se aportan como parte de la tesis doctoral son los siguientes:

- C. Vázquez, E. Huedo, R.S. Montero and I.M. Llorente. *On the Use of Clouds for Grid Resource Provisioning*. **Future Generation Computer Systems**, 27 (5), 600-605, 2011.
- C. Vázquez, E. Huedo, R.S. Montero and I.M. Llorente. *Federation of TeraGrid, EGEE and OSG Infrastructures through a Metascheduler*. **Future Generation Computer Systems**, 26(5), 979-985, 2010.
- C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente. *Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers*. In Proceedings of Grid and Pervasive Computing Conference (GPC 2009), Workshop on Grids, Clouds and Virtualization (WGCV). **IEEE Computer Society**, 113-120, May 2009.
- C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente. *A Performance Model for Federated Grid Infrastructures*. In Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008). **IEEE Computer Society**, 188-192, 2008.
- C. Vázquez, J. Fontán, E. Huedo, R.S. Montero, I.M. Llorente. *Transparent Access to Grid-Based Compute Utilities*. In Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics (PPAM 2007). **Lecture Notes in Computer Science**, 4967, 817-824, 2008.
- C. Vázquez, E. Huedo, R.S. Montero and I.M. Llorente. *Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastruc-*

tures. In Proceedings of the 13th International Conference on Parallel Processing (Euro-Par 2007). **Lecture Notes in Computer Science**, 4641, 372-381, 2007.

Conforme a la normativa vigente, esta tesis doctoral presenta una introducción al trabajo de investigación llevado a cabo, así como una revisión del estado del arte del campo de la computación Grid y Cloud. A continuación, se describen las aportaciones del trabajo y se incluye una discusión integradora sobre los artículos presentados. Se concluye el documento con una sección dedicada a las conclusiones sobre el trabajo realizado y el trabajo futuro, y con otra sección conteniendo las referencias bibliográficas que complementan las ya incluidas en los artículos que muestran la investigación llevada a cabo en esta tesis doctoral.

Resumen

Esta tesis doctoral se centra en extender dos de los paradigmas de computación distribuidas más populares en la actualidad: la computación Grid y la computación Cloud. Se amplía el campo de la interoperabilidad de infraestructuras de computación y su acceso transparente, con el objetivo de proporcionar técnicas a la hora de construir infraestructuras accesibles por medio de un modelo de Computación como Servicio. Se describe asimismo un modelo de evaluación del rendimiento de infraestructuras federadas, aplicable al diseño de políticas de planificación y útil a la hora de evaluar la viabilidad de infraestructuras Grid agregadas. En esta misma línea se proponen las condiciones de federación de infraestructuras por medio de un metaplanificador, validándose la propuesta con experimentos en los cuales se consiguen federaciones complejas (jerárquicas y recursivas) entre grandes infraestructuras de producción. Complementariamente, y acorde con la situación actual de la computación distribuida, se hace una reflexión sobre la coexistencia y sinergias de las tecnologías Grid y Cloud, y se presenta el diseño de una arquitectura para el aprovisionamiento dinámico de infraestructuras Grid sobre una capa de virtualización.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

In the long run, history is the story of information becoming aware of itself.

James Gleick. *The Information: A History, a Theory, a Flood.*

En este capítulo de introducción se ofrece una visión general del origen y evolución de las tecnologías y paradigmas que conforman el marco en el que se ha desarrollado la investigación de esta tesis. De esta manera, se introducirá en la sección 1.1 el concepto seminal de *utility computing* y se hará un breve recorrido por la historia de la computación distribuida, haciendo especial mención a la computación Grid (o computación en malla) y a la computación Cloud (o computación en la nube). Asimismo, se hará una descripción de las tecnologías Grid en la sección 1.2, y una presentación de la noción de federación de infraestructuras Grid y su acceso transparente en la sección 1.3.

1.1. Evolución de la Computación Distribuida

El tema central de este trabajo, la provisión dinámica de recursos para servicios computacionales, tiene una historia dilatada en el mundo de las ciencias de la computación. El concepto, surgido en los años 60, de *utility computing*, o Computación como Servicio, sienta los requisitos para el aprovisionamiento de recursos computacionales de forma transparente para el proveedor de servicios, y, por supuesto, para el usuario final. En esencia, el *utility computing* es un modelo de aprovisionamiento de servicio que ofrece acceso a recursos computacionales (procesamiento y almacenamiento esencialmente) de una manera adaptativa, flexible y simple, posibilitando el pago por uso de forma similar al aplicado en otras comodidades como el agua o la electricidad. El objetivo de este paradigma es un escenario en el que la

computación se ofrece o se utiliza a través de los componentes que conforman una infraestructura de computación, de forma parecida al paradigma de productor/consumidor. Una de las principales aportaciones del *utility computing* es la de acortar de forma considerable el coste de adquirir nuevo hardware, ya que básicamente éste pasa a ser alquilado. Asimismo, se acorta el tiempo que se necesitaría para adquirir y ensamblar equipos físicos. Esta idea está plasmada acertadamente en el libro de Nicholas Carr [11]. De este concepto surgen sucesivamente hasta nuestros días distintas metodologías y paradigmas: el *time-sharing*, la computación distribuida, la computación Grid (ver sección 1.1.1) o, más recientemente, la computación Cloud (ver sección 1.1.2).

También en los años 60 tiene lugar el nacimiento de una tecnología crucial para el tema de esta tesis y para la construcción de infraestructuras computacionales hoy en día: la virtualización. Esta tecnología nace en IBM para llevar a la práctica el concepto de *time-sharing*. El problema al que se enfrentaba la compañía norteamericana era el de compartimentar estáncamente los potentes *mainframes* que habían fabricado para dar servicio a sus clientes, los cuales solo requerían una fracción de la potencia de cálculo de estos ordenadores, por lo cual darle acceso exclusivo a estas máquinas sería un derroche excesivo. De esta manera surgió el CP/CMS, un sistema operativo cuya primera prueba de concepto se materializó en el CP-40. El usuario final de un sistema CP/CMS tenía acceso a un ordenador simulado en el interior de uno de estos *mainframes*, de forma totalmente transparente. Este ordenador simulado tenía una fracción de los recursos hardware del ordenador físico en el que estaba corriendo, y acceso a los periféricos, interrupciones, conjunto de instrucciones y demás elementos a los que se podía acceder desde el ordenador principal, consiguiendo de esta forma la multiplexación de varios ordenadores en el interior de uno solo. Los conceptos y técnicas para conseguir esta virtualización fueron plasmados por Popek y Goldberg en un artículo seminal de virtualización [41]. Estos mismo conceptos son los que marcaron la tecnología de virtualización desde su origen hasta nuestros días. Anecdóticamente, el CP/CMS jugó un papel principal en el afianzamiento de la teoría sobre sistemas operativos, y también en la creación del movimiento de software libre, ya que su código fue liberado y de esa manera consiguió una amplia comunidad. Éste fue un movimiento clave para su éxito, ya que la comunidad proporcionaba soporte para sistema operativo al margen de IBM.

La aparición a principios de los 70 de redes de área local como Ethernet [37] propició la aparición de sistemas distribuidos. Estos sistemas consisten en múltiples computadores autónomos que se comunican a través de una red, ejecutando programas distribuidos que interactúan para conseguir un fin común. Los sistemas distribuidos supusieron una revolución en el mundo de la computación, tanto en tecnologías (arquitecturas cliente servidor,

comunicaciones punto a punto, etc.), como en paradigmas de computación. Intrínsecamente ligado a la aparición de los sistemas distribuidos, y con la creciente capacidad de los ordenadores y el ancho de banda disponible en las redes interconectándolos, surge el concepto de computación en clúster (cuando los sistemas son homogéneos) o en *intranet* (cuando los recursos son heterogéneos). Para sacar partido a estas infraestructuras, y de la creciente demanda de mayores potencias computacionales, surgen tecnologías que permiten la ejecución de trabajos por lotes. Estas tecnologías se conocen como LRMS (siglas en inglés de sistemas de gestión de recursos locales), y como ejemplos concretos podemos tomar Condor[1] (un precursor de la computación Grid) o el recientemente renombrado Oracle Grid Engine[2]. La aparición de varias de estas tecnologías y la incompatibilidad entre ellas fue uno de los motivos de la aparición de las tecnologías Grid.

1.1.1. Computación Grid

La computación Grid basa su propuesta en otro paradigma contemporáneo, la computación punto a punto (Peer-to-Peer, o P2P), popularizada a través de proyectos del calado de SETI@HOME [1]. Este paradigma promulga una tipo de comunicación entre ordenadores que no necesita un sistema centralizado de gestión de la computación, sino que se basa en la comunicación completamente descentralizada entre pares de componentes de computación de forma directa, sin necesidad de intermediario. De esta forma se aumenta tremendamente la escalabilidad (no hay un punto central por el que deben pasar todas las comunicaciones) y la robustez (se eliminan cuellos de botella) de las soluciones de computación distribuida. También surgen nuevos problemas, como por ejemplo la interoperabilidad de todos los componentes que conforman una infraestructura de computación, es decir, los diferentes clústeres manejados por LRMS en los que la computación distribuida estaba compartimentada.

El paradigma de la computación Grid [30] surge entrada la década de los 90, como una evolución natural de los sistemas distribuidos surgidos en las décadas anteriores. La propuesta Grid pasa por crear servicios específicos y bien definidos para ofrecer recursos computacionales por medio de la ejecución de trabajos en lote, y de esta forma asegurar la interoperabilidad de los clústeres. Se adopta así un modelo "reloj de arena" (ver Figura 1.1), en el cual el cuello estrecho del reloj de arena define un conjunto pequeño de abstracciones y protocolos al que diferentes componentes de alto nivel pueden ser conectados (la parte superior del reloj), y el cual a su vez puede ser conectado a diferentes tecnologías subyacentes (la base del reloj). Por definición, el número de protocolos definidos en el cuello debe ser pequeño.

[1] http://www.cs.wisc.edu/condor/
[2] http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html

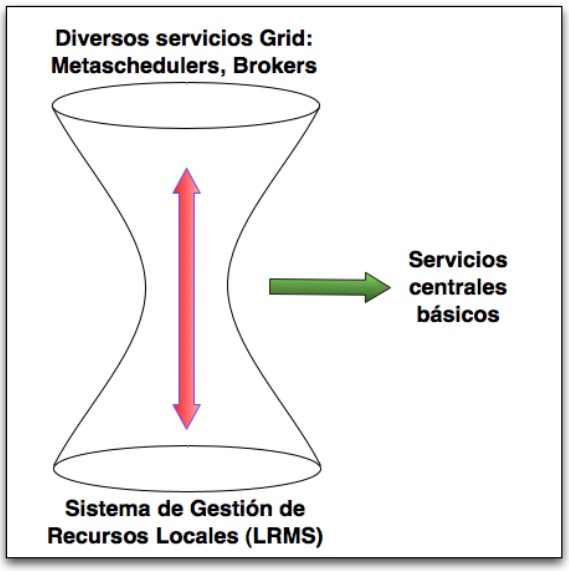


Figura 1.1: El modelo reloj de arena del Grid.

Vamos a proceder a definir los servicios a partir de los cuales pueden crearse infraestructuras Grid, comunes a todo middleware Grid funcional. Podemos definir un Grid como un sistema distribuido de computación que permite resolver problemas que requieren una gran potencia computacional por medio de la agregación de recursos computacionales no necesariamente cercanos geográficamente. Este sistema debe presentarse al usuario final de manera transparente, por lo cual se necesitan servicios que sean capaces de lidiar con la naturaleza distribuida de la computación Grid. Dichos servicios deben incluir mecanismos para:

- monitorizar los recursos disponibles en la infraestructura
- manejar el ciclo de vida de los trabajos de procesamiento por lotes que se pretenden ejecutar en la infraestructura
- transferir archivos de entrada salida de los trabajos, así como la aplicación en si, a los nodos de ejecución relevantes
- escoger automáticamente los nodos de ejecución óptimos para cada trabajo
- ofrecer un punto de acceso con la funcionalidad de la infraestructura, orquestando los servicios disponibles

1.1.2. Computación Cloud

La computación Cloud ha irrumpido recientemente con mucha fuerza como modelo de abstracción y aprovisionamiento de recursos computaciones. Como sistema novedoso, adolece de los problemas típicos de una tecnología joven. Un buen punto de partida para aclarar dudas sobre la computación en nube es el estudio de la Universidad de Berkeley [4].

Es muy importante separar de manera estanca las tres capas claramente diferenciadas de la computación Cloud, como puede verse en la Figura 1.2. La capa superior es normalmente referida como *Software como Servicio* (Software as a Service, SaaS). Esta capa está destinada al usuario final, y no deja de ser una encarnación de lo que supone internet: acceso a servicios software (léase Gmail, Facebook, Twitter) a través de un navegador, sin que el usuario necesite preocuparse por el sustrato físico en el que se almacenan y procesan los datos necesarios para ofrecer el servicio, ni en el tipo ni versión de software utilizado.

La capa intermedia de la computación Cloud está dirigida principalmente a los desarrolladores software y se conoce como *Plataforma como Servicio* (Platform as a Service, PaaS). Este sistema ofrece a un desarrollador una plataforma sobre la que ejecutar sus programas, despreocupándose de qué hardware lo sustenta, cuál es su arquitectura y, principalmente, la cantidad de recursos necesarios, ya que estos se ajustan automáticamente a la demanda. El beneficio de este modelo también se plasma en la posibilidad de centrarse en la programación del servicio y no en los problemas derivados de la escalabilidad, es decir, en la adquisición y mantenimiento de hardware e incluso de los servicios básicos de software que hacen falta para distribuir el servicio que se está programando. Existen varios ejemplos de estas plataformas, como Google App Engine[3] o Windows Azure[4], cuyo lema sintetiza perfectamente el concepto de *Plataforma como Servicio*: "Focus on your application. Not the infrastructure" ("Céntrate en tu aplicación. No en la infraestructura.").

Estas dos capas se sustentan sobre un sustrato físico de computación, que a su vez es también susceptible de ser ofrecido como un servicio. Esta acomodación del sustrato físico es lo que se conoce como *Infraestructura como Servicio* (Infrastructure as a Service, IaaS). El NIST [36] define el Cloud Computing (referido a infraestructura como servicio) como "un modelo para habilitar el acceso conveniente por demanda a un conjunto compartido de recursos computacionales configurables, por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios, que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios". Puede verse claramente como este modelo re-

[3] http://code.google.com/appengine/
[4] http://www.microsoft.com/windowsazure/

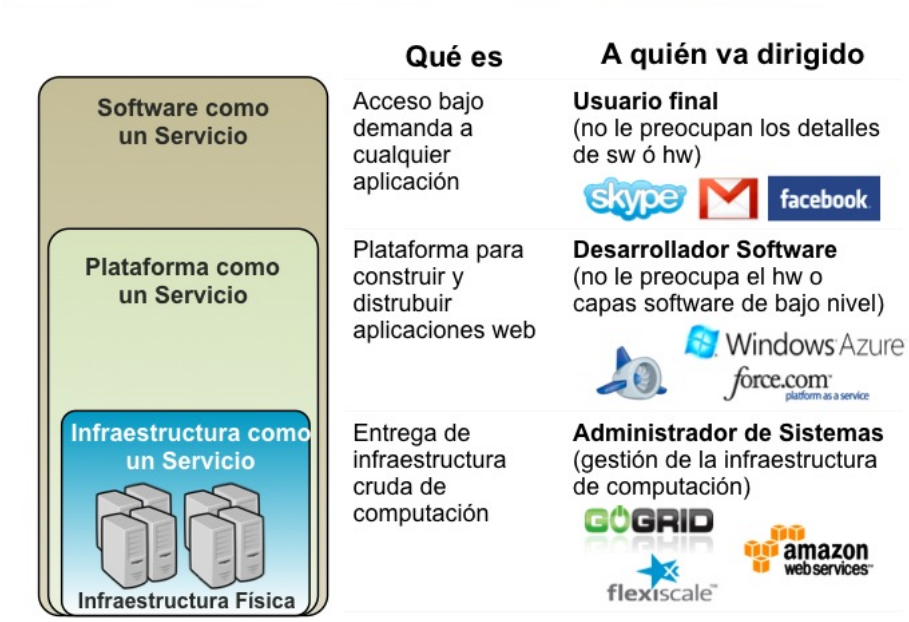


Figura 1.2: Las capas de la computación Cloud.

presenta un aprovisionamiento dinámico de recursos computacionales, y por tanto está alineado con los requisitos que representan el *utility computing*.

Una infraestructura Cloud puede clasificarse según el ámbito de sus recursos computacionales y de sus usuarios. De esta manera, podemos hablar de tres tipos principales de clouds:

- *Cloud privado*: En este tipo de cloud se usa la infraestructura local de una empresa o centro de procesamiento para uso propio. El objetivo es la flexibilización de los servicios a la hora de despliegue, mantenimiento y, sobre todo, escalabilidad. Este último aspecto es crucial a la hora de comparar la tecnología Cloud con otras soluciones.

- *Cloud híbrido*: En los casos en los que un cloud privado no pueda satisfacer los picos de demanda de computación generados desde la propia entidad que posee el cloud, surge como solución el *outsourcing* de recursos computacionales a otras empresas de cloud público, que obviamente supone el pago de los mismos. Este sistema tiene la ventaja de evitar los costes de mantener un conjunto de recursos de computación que podrían en su mayor parte del tiempo estar utilizados por debajo de sus posibilidades.

- *Cloud publico*: Este tipo de infraestructura tiene como objetivo ofertar los recursos computacionales no utilizados a través de un interfaz sencillo. Las ventajas de los clientes de este tipo de cloud coinciden con la

de los usuarios de los clouds híbridos (como hemos visto, un cloud híbrido usa un cloud público), y los que ofertan recursos de computación tienen como objetivo conseguir un beneficio económico a cambio de los ciclos no utilizados de sus servidores. El ejemplo más claro y temprano de cloud público y que de alguna manera propició el nacimiento de la computación cloud es Amazon EC2[5].

Con su reciente aparición, la tecnología Cloud se percibe como una evolución natural de la computación Grid, destinada a desplazarla. Como se puede ver en esta tesis estas dos tecnologías, aparte de poder beneficiarse una de otra en cuestiones técnicas (e incluso políticas), son perfectamente complementarias. En los artículos [55, 52] y en proyectos como StratusLab[6] puede verse como las infraestructuras Grid puede beneficiarse ampliamente de la tecnología Cloud, ya que ésta permite la construcción de infraestructuras Grid elásticas, que crecen o decrecen en función de su demanda.

1.2. Tecnologías y Componentes para la Computación Grid

Esta sección tiene como objetivo describir de una manera sucinta los componentes que conforman la capa de middleware Grid. De esta manera se introduce en esta Tesis una tecnología básica utilizada para realizar gran parte de la investigación: el metaplanificador GridWay[7] [26].

No podría entenderse la computación Grid como la concebimos hoy en día sin hacer referencia a un middleware Grid muy popular que implementa servicios necesarios para construir infraestructuras Grid y que se ha convertido en estándar de facto en la computación Grid: el Globus Toolkit [20], esfuerzo nacido en la Universidad de Chicago y dirigido por Ian Foster, considerado el padre del paradigma.

Existen numerosos ejemplos de middleware Grid: gLite[8], Unicore [17], Advance Resource Conector[9], Simple Grid Protocol[10], etc. Todos estos middlewares tienen en común el ofrecer paquetes software que interactúan y se disponen en capas, y que pueden ser accedidos por un API común que permite a los usuarios no tener que conocer la sintaxis y métodos de acceso de los componente subyacentes abstraídos por el middleware.

Debido a su status de estándar *de facto*, el middleware Globus se utiliza a continuación a modo de plantilla para describir la arquitectura típica de

[5] http://aws.amazon.com/ec2
[6] http://stratuslab.eu
[7] http://www.gridway.org
[8] http://glite.cern.ch
[9] http://www.nordugrid.org/middleware
[10] http://grid.bmk.com.au

un middleware Grid, por medio de la enumeración de las categorías en las que se pueden englobar los diferentes componentes de este software.

- *Componente común de tiempo de ejecución*, categoría en la que se incluyen las librerías y componentes básicos que se usan en la construcción de todos los demás servicios.
- *Seguridad*, componentes (que forman el Grid Security Infraestructure, GSI) que aseguran que todas las comunicaciones son fiables.
- *Data management*, categoría que engloba todas aquellas piezas de software que permiten la gestión (transferencia, almacenamiento) de conjuntos de datos que pueden ser potencialmente grandes.
- *Servicios de información*, componentes que constituyen el Monitoring and Discovery Services, MDS y que permiten extraer información de los elementos de computación (número de ranuras libres, potencia de cálculo, memoria disponible, etc).
- *Servicios de gestión de ejecución*, categoría que incluye todas las componentes encargadas de la iniciación, monitorización, planificación y coordinación de los programas ejecutables, llamados trabajos.

Se puede ver una correspondencia casi punto a punto de estos componentes con los servicios Grid identificados en la sección 1.1.1.

La Figura 1.3 muestra la correspondencia de estas categorías con los distintos servicios del Globus Toolkit, lo que da una idea de la riqueza de este middleware Grid. Se compone de servicios muy específicos y sencillos (siguen el principio Unix, herramientas sencillas para tareas sencillas) y variados (por ejemplo, la existencia de dos tipos de interfaces, uno anterior basado en interfaz propio y el moderno, basada en tecnologías de servicios Web). Su evolución y diferentes versiones son una fuente de heterogeneidad a la hora de construir federaciones de Grids.

Dada la cantidad de servicios ofrecidos por Globus se hace necesaria una componente que los orqueste. Un usuario puede hacer uso de estos servicios (Globus ofrece clientes para cada uno de ellos) para enviar sus trabajos al grid, pero la tarea se antoja farragosa: descubrir un elemento de computación con el menor tiempo de espera en cola usando MDS, la transferencia de archivos usando GridFTP o RFT y la ejecución final del trabajo usando GRAM. Existen herramientas que pueden realizar este trabajo de forma autónoma, como por ejemplo el metaplanificador GridWay, el cual ofrece una forma sencilla y transparente de enviar trabajos a infraestructuras Grid.

Podemos definir metaplanificador como "aquella componente del middleware Grid que descubre, evalúa y asigna recursos de computación a trabajos de ejecución por lotes mediante la coordinación de las actividades de los

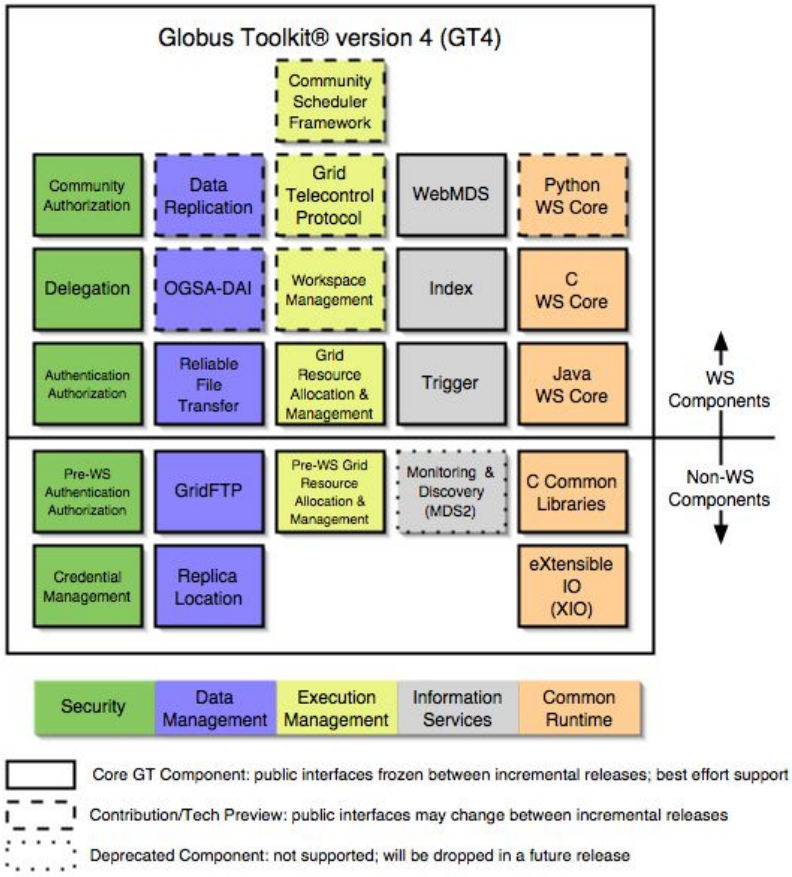


Figura 1.3: Servicios del middleware Grid Globus Toolkit. Fuente: Globus Alliance (http://www.globus.org)

.

diferentes planificadores a nivel local existentes" [59]. El metaplanificador GridWay presenta una arquitectura flexible que lo hace un buen candidato para desempeñar el papel de pieza central en la federación de infraestructuras Grid, ya que además de sus labores de metaplanificador, ofrece la posibilidad de ejecutar diferentes adaptadores para interactuar con distintos middleware (o con distintas versiones/configuraciones de un mismo middleware) Grid.

1.3. Federación y Acceso Transparente

La federación de diferentes infraestructuras Grid es el tema principal de esta tesis. De hecho, la hipótesis en la que se basa este trabajo es el "aprovisionamiento dinámico de recursos computacionales a través de la federación transparente de infraestructuras Grid, consiguiéndose dicho aprovisionamiento por medio de tecnologías puramente Grid y/o con la ayuda de

los desarrollos más recientes en computación Cloud". Esta hipótesis pretende satisfacer los requisitos expuestos por el concepto de *utility computing*, y que pueden sintentizarse en un acceso transparente a recursos computacionales con un modelo de pago por uso. Puede decirse por tanto que este trabajo tiene como objetivo un aporte original en el mundo del aprovisionamiento dinámico de recursos, basando este aporte en el estudio de la federación de infraestructuras y su conjugación con las novedosas tecnologías de computación en nube.

Los requisitos de la federación de grids (parte muy importante del problema que esta tesis pretende resolver) no han sido extraídos a través de razonamiento abstracto, si no que se llegó a la conclusión de que la mejor opción sería comprobar qué requisitos están presentes a día de hoy en la industria. La participación en el proyecto BEinGRID[11] fue aprovechada siguiendo esta idea, debido a que una lista de requisitos a tener en cuenta a la hora de crear infraestructuras Grid federadas fue identificada en un contexto en el cual estaban representados empresas reales, con problemas reales. La lista siguiente no pretende ser una lista exhaustiva de requisitos para federaciones de infraestructuras Grid, la cuales son dinámicas por naturaleza, y por lo tanto muy propensas a contaminaciones propias del dominio particular para el cual se hayan creado. Sin embargo, sí que se puede considerar una lista compacta de los requisitos más relevantes y comunes de la federación de infraestructuras Grid:

- Federación a través de diferentes dominios de administración.
- Capacidad de ejecución de trabajos de procesamiento por lotes de baja latencia y alta productividad.
- Acceso transparente para el usuario final.
- Imposición de políticas de planificación con conocimiento de la infraestructura federada.
- Seguridad.
- Disponibilidad de herramientas de contabilidad y facturación.

Llegados a este punto podemos empezar a definir el concepto de federación. Una posible definición de la federación sería la "agregación transparente de diferentes infraestructuras Grid". Es importante resaltar la palabra "diferente", debido a que encarna la tarea más complicada a la que se enfrenta la federación. Las diferencias no sólo se refieren a las diferentes topologías, capacidades, calidad o incluso localización de las infraestructuras Grid, pero también al hecho de que los recursos de computación están abstraídos por

[11] http://www.beingrid.eu/

diferentes *middlewares*. Esta característica de las infraestructuras Grid resulta irónica si tenemos en cuenta que el objetivo fundamental del paradigma Grid es transformar infraestructuras heterogéneas en homogéneas a través del uso de un mismo interfaz [19]. Sin embargo, es también obvio que el uso de diferentes *middlewares* es a su vez una fuente de heterogeneidad.

De esta condición de heterogeneidad impuesta se desprende la diferenciación entre *interoperabilidad* e *interoperación*. *Interoperabilidad* es la habilidad nativa de las infraestructuras Grid para interactuar directamente usando interfaces estándar, abiertos y comunes, mientras que *interoperación* representa el conjunto de técnicas orientadas a conseguir una solución más inmediata al problema de hacer trabajar de forma conjunta infraestructuras Grid de producción. Parte del trabajo de investigación [51] trata de demostrar que la interoperación es una condición necesaria, pero no suficiente, para conseguir federación. Una prueba de la importancia de la interoperabilidad de sistemas es el Real Decreto 4/2010 del 8 de enero[12], en el cual se expone la preocupación y los objetivos para asegurar una correcta interactuación entre los sistemas informáticos que dan soporte a la Administración Pública.

Hay multitud de ejemplos en la literatura sobre cómo conseguir federación entre infraestructuras Grid para construir el deseado World Wide Grid, centrándose en distintos aspectos de las infraestructuras. Unas líneas de investigación giran alrededor de la definición de los interfaces estándar para el desarrollo de aplicaciones, como por ejemplo SAGA [46], una API orientada a aplicaciones que asegura su compatibilidad con diferentes infraestructuras ofreciendo una capa extra de abstracción. En esta misma línea podemos considerar el trabajo del proyecto de middleware grid UNICORE [17], el API Java HiLA [23]. También es interesante considerar el esfuerzo de realizado para definir el *HPC Profile*, un conjunto de especificaciones no propietarias, refinamientos, e interpretaciones conceptuales que promueven la interoperabilidad. Ejemplos de especificaciones recomendadas en el *HPC Profile* incluyen el Job Submission Description Language (JSDL) [3], que ofrece una abstracción a la hora de especificar trabajos Grid, y el OGSA-BES (Open Grid Services Architecture Basic Execution Services[13] [22]), cuyo objetivo es la estandarización de las peticiones de inicio, monitorización y gestión de las actividades computacionales de una infraestructura Grid; ambos esfuerzos del Open Grid Forum[14]. Estos desarrollos y estándares se sitúan una capa de abstracción por encima de otros APIs destinados a aplicaciones para clústeres, como por ejemplo DRMAA [40], usado por la aplicaciones para interactuar programáticamente con los LRMS. Este estándar juega un papel importante en esta tesis, ya que se emplea para dar acceso transparente a infraestructuras a este tipo de aplicaciones desarrolladas para clúster, consiguiendo la compartición

[12] http://boe.es/boe/dias/2010/01/29/pdfs/BOE-A-2010-1331.pdf
[13] http://www.ogf.org/documents/GFD.108.pdf
[14] http://www.gridforum.org

de recursos en diferentes dominios de administración para aplicaciones que originalmente no fueron concebidas con esta característica.

Existen otras lineas de investigación destinadas a buscar la *interoperación*, usando las tecnologías desplegadas hoy en día en producción. Parte de estos esfuerzos están consideradas por el grupo del Open Grid Forum llamado Grid Interoperation Now (GIN) [43]. Entre los frutos más destacados del trabajo de este grupo surge la colaboración entre infraestructuras basadas en Globus y en UNICORE [7], así como otros métodos de colaboración basados en portales web y *workflows* [28]. También existen otros esfuerzos de *interoperación* como InterGrid [28], basada en la creación de InterGrid GateWays (ICG) para interconectar diferentes "islas" Grid, o como GridX1 [2], el cual encapsula una infraestructura para que pueda ser accedida como otro recurso del Grid Global de Computación LHC (Worldwide LHC Computing Grid, WLCG[15]). Existen otros trabajos basados en la idea de conseguir que colaboren infraestructuras usando los interfaces actuales que sugieren la introducción de un componente *meta broker* [29], para ayudar en el acceso de recursos situados fuera de la propia infraestructura.

El campo de la federación Grid es amplio y fecundo, es interesante ver otros problemas que comparten características comunes con el tema de esta tesis, como por ejemplo el de la calidad de servicio [42], o los esfuerzos de planificación basada en principios económicos en federación realizados por GRIDS Laboratory[16].

Como se ha comentado en la sección anterior, los esfuerzos de federación de infraestructuras realizados en este trabajo tienen una componente del ecosistema Grid como pieza fundamental, el metaplanificador GridWay. Esta componente es la encargada de ofrecer acceso transparente a la infraestructura federada que resulta de su habilidad para interactuar con las diferentes infraestructuras miembro, planificando y despachando trabajos en ellas. Como se puede ver en los siguientes capítulos y en los artículos adjuntos, GridWay es usado no únicamente como metaplanificador, sino como un orquestador de los diferentes servicios Grid para conseguir federación de diferentes tipos de infraestructuras a través de adaptadores que le permiten interactuar con distintas variedades de *middleware*. GridWay ofrece además un interfaz DRMAA [24], para el cual existen un número importante de aplicaciones, que de esta manera se pueden beneficiar de manera transparente (uno de los objetivos de esta tesis es precisamente el acceso transparente a la infraestructura federada) del salto que supone la ejecución en un entorno de clúster a una federación de infraestructuras Grid, beneficios medidos en términos de capacidad de computación principalmente. En un artículo en el que ha participado el autor [14] puede verse un ejemplo de aplicación (en este caso, laboratorio virtual), beneficiada de la capacidad ofrecida por GridWay

[15] http://lcg.web.cern.ch
[16] http://www.cloudbus.org/federation

a través del interfaz DRMAA. También en el marco de este trabajo se ha realizado un esfuerzo de integración de aplicaciones y de *frameworks* (véase el artículo [5] que forma parte de esta tesis), para capacitar su uso sobre infraestructuras federadas.

Capítulo 2

Modelo de Computación como Servicio Basado en la Federación de Infraestructuras Grid

...computing may someday be organized as a public utility just as the telephone system is a public utility...

John McCarthy. *MIT Centennial (1961)*

El modelo de Computación como Servicio se basa en el acceso transparente a infraestructuras de computación para facilitar el encapsulamiento de los detalles de implementación y uso de dichas infraestructuras. Este acceso transparente no es ofrecido únicamente a los usuarios finales sino también a aplicaciones paralelas que inicialmente no han sido concebidas para ser ejecutadas en entornos fuertemente distribuidos. En este capítulo se introduce el concepto de Computación como Servicio en la sección 2.1, para a continuación mostrar técnicas de encapsulamiento en la sección 2.2. Seguidamente se detallan en la sección 2.3 distintas técnicas para construir infraestructuras complejas de computación basadas en el acceso transparente, y se muestran ejemplos prácticos de arquitecturas basadas en este modelo e implementadas como parte de esta tesis. Se finaliza el capítulo presentando un modelo de evaluación del rendimiento de infraestructuras federadas en la sección 2.4.

2.1. La Computación como Servicio

Los modelos de computación Grid y Cloud hacen hincapié en un acceso de tipo *utility*. Este término está relacionado con un paradigma de aprovi-

sionamiento de la Tecnología de la Información (TI), el cual exhibe varios beneficios potenciales [39] para las organizaciones que hagan uso de infraestructuras de este tipo: reducción de costes fijos, el trato de coste variable los recursos de TI por medio de la provisión ilimitada de capacidad computacional, mejorando la flexibilidad; agilizando de esta manera el aprovisionamiento de recursos de una manera más adaptativa. Estos valiosos beneficios amenazan con acabar con el coste fijo de los recursos TI, donde la computación es llevada a cabo dentro de las organizaciones individuales o confiada a proveedores externos [10], aunque pueda verse en este *outsourcing* una etapa intermedia que apunta hacia un sistema de tipo *utility*, lo que se conoce actualmente como Computación como Servicio (Computation as a Service, *CaaS*).

El despliegue de una solución de tipo *CaaS* conlleva una separación completa entre el proveedor y el consumidor. El segundo requiere un acceso transparente, uniforme, seguro y robusto, mientras el primero requiere una infraestructura flexible, escalable y adaptativa para ofrecer el servicio.

Por tanto, vemos que entre las principales características de un sistema de tipo *CaaS* está la comodidad en el acceso a los recursos ofrecidos. Una de las aportaciones de esta tesis es la propuesta, diseño y evaluación de distintos mecanismos capaces de abstraer la complejidad y el uso de recursos computacionales a través de estándares, y aplicables a la hora de construir arquitecturas de tipo *CaaS*. Tres de estos mecanismos se presentan a continuación, el primero basado en el metaplanificador GridWay, el segundo construido a partir de un gestor local de recursos (LRMS), el Sun Grid Engine, y el tercero por medio del estándar de acceso clúster DRMAA.

Estos mecanismos tienen su origen en técnicas y patrones de diseño de arquitecturas de computación y redes distribuidas. Como puede verse en el en la sección 3.2 del siguiente capítulo, estos mecanismos pueden asociarse a técnicas usadas para conseguir interoperabilidad entre infraestructuras Grid, característica indispensable para su federación.

2.2. Interfaz Recursivo para el Acceso a Infraestructuras Grid

El proyecto *GridGateWay*[1] surge como parte del trabajo realizado en esta tesis. Esta componente permite el acceso a parte de las funcionalidades ofrecidas por el metaplanificador GridWay a través de un interfaz de tipo *web services*, concretamente el ofrecido por la componente GRAM del Globus Toolkit. GRAM se encarga de ofrecer un interfaz que permite a los usuarios localizar, enviar, monitorizar y cancelar trabajos remotos en una in-

[1] http://www.gridway.org/doku.php?id=ecosystem:gridgateway

fraestructura Grid[2]. La particularidad de *GridGateway* proviene de que los trabajos no son enviadosa un clúster gestionado por un único LRMS, sino que se envían a una instancia de GridWay que puede interactuar con varios de estos clústeres, o incluso con otras instancias de *GridGridWay*.

La ventaja de la técnica de tipo pasarela (ver sección 3.2.3) presente en la componente *GridGateWay* es la posibilidad de encapsular toda la complejidad y potencia de una infraestructura Grid a través de una interfaz concebida originalmente para la gestión de un único clúster, consiguiéndose de manera efectiva un acceso "transparente" al usuario, que no necesita conocer las particularidades del grid encapsulado para hacer uso de él. También ayuda en este sentido el acceso al grid usando un interfaz único y estándar (Globus GRAM), encapsulando y, por lo tanto ocultando, complejidades

Hay otra característica quizás más importante de esta componente, y es su habilidad para actuar de unión entre infraestructuras a la hora de diseñar grids federados, abriendo las puertas a diseños variados de grids jerárquicos. La particularidad de esta unión es que se realiza de forma recursiva, ya que la manera en que se consolida la unión permite tener un número arbitrario de estas uniones creando una cadena recursiva sin límites en cuanto a la profundidad que puede alcanzar. En este capítulo se presentan varios ejemplos prácticos diseñados con el patrón ofrecido por esta componente.

2.2.1. Arquitectura de GridGateWay

Como hemos visto, el acceso a la infraestructura Grid encapsulada se realiza por medio de un interfaz WS-GRAM. Debido a la naturaleza de este interfaz, no toda la funcionalidad de GridWay está accesible (como por ejemplo la habilidad de gestionar trabajos en *array* y *workflows*), pero a cambio se consigue que el usuario acceda con las herramientas clientes de Globus (por ejemplo, con el comando *globusrun-ws*), y también que todo el grid subyacente pueda ser manejado por herramientas de flujo de trabajo basadas en Globus, como por ejemplo las ofrecidas por java CoG (Karajan) y accesible por medio de las APIs de Globus. El acceso a los recursos, incluyendo la autenticación del usuario a través de los distintos grids está controlado por *GridGateWay* y es transparente a los usuarios.

Para conseguir el enlace de GridWay a través de GRAM se desarrolló un nuevo adaptador de planificación para GRAM, así como un generador de eventos de planificación. Se programó asimismo una componente para transmitir la información del número de nodos disponibles en el grid encapsulado, a través del servicio MDS de Globus. Puede apreciarse que la funcionalidad principal del GridGateWay es proporcionada por GridWay, aunque accedido a través de interfaces estándar del Globus Toolkit. De esta manera se saca partido, a través de estos interfaces, de las características de GridWay: so-

[2]http://dev.globus.org/wiki/GRAM

porte para múltiples usuarios, contabilidad, tolerancia a fallos, adaptadores para distintos servicios Grid, etc.

Las infraestructuras construidas a partir de una arquitectura que presente un *GridGateWay* ofrece una serie de ventajas en términos de seguridad y escalabilidad. Con respecto a la primera de estas características, la infraestructura solo se presentaría a internet por medio de *GridGateWay*, al cual aplican los mecanismos de seguridad del firewall de Globus. Además, es posible la restricción en la diseminación de los detalles de configuración que se publican al exterior, y el control de acceso y de uso de recursos es manejado y contabilizado por *GridGateWay*. También se pueden definir distintas políticas de asignación de certificados y el conjunto de autoridades de certificación (CAs) puede ser administrado de forma independiente en cada nivel. En lo que respecta a la escalabilidad, el problema de planificación está dividido entre las infraestructuras unidas por medio de *GridGateWays*, introduciendo diferentes niveles en los cuales las políticas de planificación pueden ser ajustadas de forma independiente. Esta componente también plantea un reto interesante, que es la de esconder detalles de configuración pero a la vez ofrecer información agregada que facilite la correcta planificación. Los parámetros r_∞ y $n_{1/2}$, introducidos en un trabajo previo del grupo [38] y explorados en la sección 2.4, podrían ser un buen punto de partida para conseguir esta planificación óptima con un conocimiento mínimo de las infraestructuras subyacentes.

La Figura 2.1 representa una jerarquía Grid en un modelo de Computación como Servicio. Una infraestructura Grid gestionada por el departamento de TI de una empresa tiene acceso a otra infraestructura Grid, gestionada por un proveedor de servicios externo. El grid externo provee capacidad de computación por medio de un modelo de pago por uso cuando los recursos locales están saturados, a la vez que la seguridad y la contabilidad son gestionadas localmente en *GridGateWay* que encapsula el grid externo, de manera transparente para el usuario final. Esta jerarquía Grid puede extenderse recursivamente para federar un mayor número de infraestructuras externas por medio de relaciones proveedor/consumidor, en una arquitectura que imita a la de internet (caracterizada por el paradigma punto a punto [9] y el modelo reloj de arena del protocolo IP [15]). Evidentemente, la componente *GridGateWay* no está exenta de penalizaciones de rendimiento, principalmente un aumento de latencias, cuantificados como parte del trabajo de esta tesis (ver subsección 2.2.2).

Esta componente es el primer paso para conseguir escenarios como los explorados a continuación en este capítulo, ya que se necesitan otras componentes para garantizar Service License Agreements (SLAs), contabilidad y facturación para desplegar soluciones en producción.

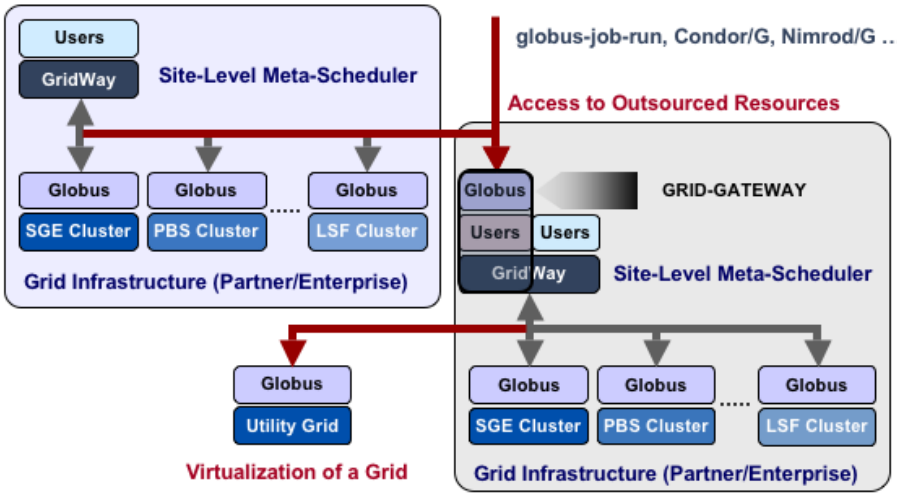


Figura 2.1: Encapsulamiento de una infraestructura Grid por medio de GridGateWay.

```
HID OS              ARCH   MHZ %CPU  MEM(F/T)      DISK(F/T)     N(U/F/T) LRMS                HOSTNAME
0   Scientific Linu i686  1001    0   513/513           0/0    3/103/224 jobmanager-lcgpbs    lcg02.ciemat.es
1   ScientificSL3.0 i686   551    0   513/513           0/0       0/3/14 jobmanager-lcgpbs    ce2.egee.cesga.es
2   Scientific Linu i686  1000    0 1536/1536          0/0       0/2/26 jobmanager-lcgpbs    lcgce01.jinr.ru
3   Scientific Linu i686  2800    0 2048/2048          0/0       0/0/98 jobmanager-lcgpbs    lcg06.sinp.msu.ru
4   Scientific Linu i686  1266    0 2048/2048          0/0      0/26/58 jobmanager-pbs       ce1.egee.fr.cgg.com
5   Scientific Linu i686  2800    0 2048/2048          0/0   0/135/206 jobmanager-lcgpbs    node07.datagrid.cea.fr
6   ScientificSL3.0 i686  3000    0 2048/2048          0/0   0/223/352 jobmanager-lcgpbs    fal-pygrid-18.lancs.ac.u
7   Scientific Linu i686  2400    0 1024/1024          0/0   0/139/262 jobmanager-lcgpbs    heplnx201.pp.rl.ac.uk
8   Scientific Linu i686  3000    0 2048/2048          0/0      0/0/60 jobmanager-pbs       cluster.pnpi.nw.ru
9   Scientific Linu i686  1098    0 3000/3000          0/0       0/5/16 jobmanager-lcgpbs    grid002.jet.efda.org
10  Scientific Linu i686  2800    0 1024/1024          0/0      0/30/32 jobmanager-lcgpbs    ce.hep.ntua.gr
11  Scientific Linu i686  2000    0   492/492          0/0        0/2/7 jobmanager-lcgpbs    ce.epcc.ed.ac.uk
```

Figura 2.2: Recursos disponibles en la infraestructura Grid de EGEE, VO de *Fusion*.

2.2.2. Evaluación del Rendimiento de la Infraestructura Encapsulada

También como parte de esta tesis [48] se ha desarrollado un experimento destinado a medir la latencia introducida por la componente *GridGateWay*. Para ello, se ha comparado el acceso directo a recursos del grid de EGEE (de la misma organización virtual de *Fusion*) con el acceso a estos mismo recursos por medio de *GridGateWay*. En el primer escenario, una instancia de GridWay se configura con los recursos del grid de EGEE (ver Figura 2.2), y se le envían 100 trabajos de procesamiento por lotes. En el segundo, los mismos trabajos son enviados a una instancia de GridWay con un *GridGateWay* como único recurso, encapsulando las mismas máquinas de la Figura 2.2.

En la Figura 2.3 puede verse el rendimiento de los recursos accedidos di-

rectamente frente a su rendimiento cuando son accedidos a través del *GridGateWay*. Un rendimiento de 284.8 trabajos/hora se consiguen en el escenario directo, frente a los 253.9 trabajos/hora que se obtienen a través de *GridGateWay*, de lo que se infiere una pérdida de eficiencia del 10.85 %. Hay que tener en cuenta que esta penalización se obtiene con trabajos de muy corta duración (alrededor de diez segundos, por supuesto dependiendo del hardware). Debido a que la penalización es independiente del tiempo total requerido por la aplicación, supondrá un menor porcentaje del tiempo total cuando se use el *GridGateWay* para aplicaciones más exigentes.

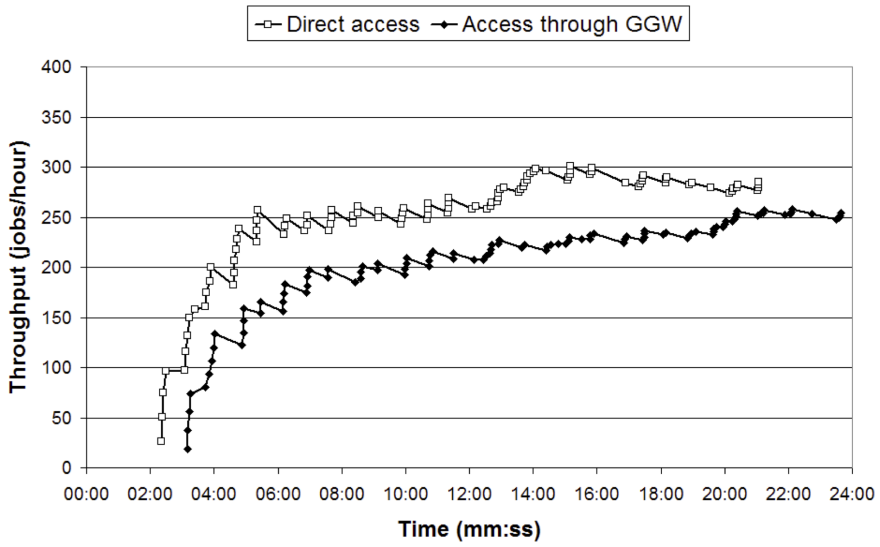


Figura 2.3: Rendimiento de los recursos Grid de EGEE (VO de *Fusion*) conseguido al ser accedidos directamente o a través de *GridGateWay*.

2.2.3. Federación Jerárquica y Recursiva de Infraestructuras Grid

El siguiente experimento fue llevado a cabo como parte de esta tesis [49] para probar la viabilidad de la componente *GridGateWay* en una infraestructura real. La componente se instala en el servidor *cepheus*, y es la encargada de encapsular un grid externo, en este caso la organización virtual *Fusion* de la infraestructura Grid del proyecto Enabling Grids for E-SciencE o EGEE[3] (introducido en más detalle en 3.4). El escenario del experimento (reflejado en la Figura 2.4) se compone de una infraestructura Grid federada con un único punto de entrada a través del metaplanificador GridWay, el cual gestiona un Grid local localizado en la UCM y formado por distintos LRMS:

[3]http://public.eu-egee.org/

```
HID OS              ARCH   MHZ %CPU MEM(F/T)     DISK(F/T)   N(U/F/T) LRMS HOSTNAME
0   Linux2.6.16-2-6 x86   3216    0 831/2027 114644/118812      0/0/1 Fork cygnus.dacya.ucm.es
1   Linux2.6.16-2-a x86_6 2211  100 671/1003   76882/77844      0/2/2 SGE  aquila.dacya.ucm.es
2   Linux2.6.16-2-6 x86   3215    0 153/2027 114541/118812      0/0/1 Fork draco.dacya.ucm.es
3   Linux2.6.16-2-a x86_6 2211  100 674/1003   76877/77844      0/2/2 PBS  hydrus.dacya.ucm.es
4   NULLNULL        NULL     0    0      0/0           0/0 6/665/1355 GW   cepheus.dacya.ucm.es
```

Figura 2.4: Recursos de la UCM mostrados por el comando `gwhost`.

cygnus y *draco* presentan uno de tipo Fork (en el cual los trabajos simplemente se ejecutan en el servidor que recibe la petición, es decir, conforma un clúster de un único nodo que a la vez es el punto de entrada), *hydrus* es un clúster de dos nodos manejado por PBS/Torque[4], y *aquila* es otro clúster de dos nodos gestionado por Sun Grid Engine. Como puede observarse en la misma figura, *cepheus* consta como entrada a un clúster gestionado por el "LRMS" GW (GridWay), el cual en realidad es otra infraestructura Grid encapsulada por medio de un *GridGateWay*. Puede observarse como la única información mostrada se refiere al número de nodos totales, usados y libres de la infraestructura subyacente, notablemente superiores a los otros clúster, ya que se accede a toda la potencia de computación de la VO de *Fusion* del grid de EGEE); mientras detalles de configuración y localización de dicha infraestructura son abstraídos al usuario de *cepheus*. Estos detalles podrían ser ofrecidos de forma agregada (memoria total disponible, disco utilizado, etc.), para lo cual habría que desarrollar una componente proveedora de dicha información.

La Figura 2.5 muestra de forma gráfica el rendimiento de la infraestructura que compone el experimento. Se ha usado una aplicación de tipo barrido de parámetros, fuertemente paralelizable, compuesta de 100 tareas, cada una de las cuales requieren 10 segundos de ejecución en un Pentium 4 a 3.2GHz. Los recursos locales de la UCM ejecutan 73 de estos trabajos, dejando los restante 27 para la infraestructura de EGEE. La razón de esta disparidad es el mayor tiempo de respuesta de los recursos de EGEE frente a los locales de la UCM (el primer trabajo se ejecuta en el grid de EGEE al cabo de 7 minutos 45 segundos después del inicio del experimento). Esto es debido en parte a la latencia de red (los recursos locales muestran una menor latencia) y en parte a que los recursos del grid externo están fuertemente saturado con trabajos de producción, y, por lo tanto, los trabajos externalizados sufren de un mayor tiempo de espera en las colas de ejecución de los gestores locales. UCM y EGEE contribuyen respectivamente 297.96 y 121.65 trabajos/hora al rendimiento agregado de 408.16 trabajos/hora. Con el método de evaluación de infraestructuras evaluadas introducido en la sección puede apreciarse cómo esta configuración consigue un rendimiento cercano al rendimiento óptimo.

[4]http://www.adaptivecomputing.com/products/torque.php

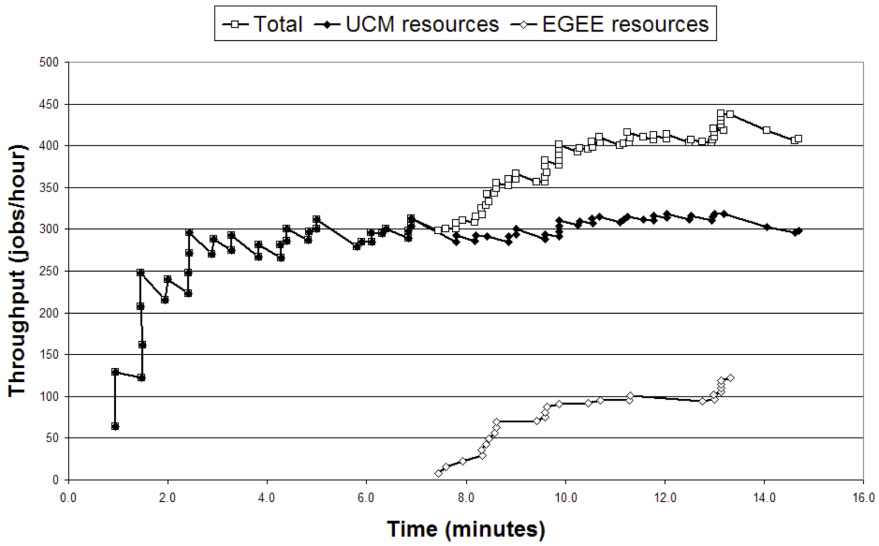


Figura 2.5: Rendimiento de la infraestructura federada (total), frente a la infraestructura local (UCM) y la infraestructura externa (EGEE).

2.3. Acceso Transparente a Servicios de Computación

El acceso transparente a infraestructuras se basa en la capacidad de procesamiento de trabajos por parte de un sistema por medio de su desvío a otras infraestructuras de computación débilmente acopladas. Este concepto conlleva a explorar nuevas posibilidades integrando diferentes componentes para dotarlas de acceso a infraestructuras Grid. Esta integración se consigue por medio del uso de componentes pasarela (como *GridGateWay*) o de interfaces (como DRMAA) que posibilitan el acceso transparente. En esta sección se muestran dos trabajos de integración usando estas técnicas.

2.3.1. Colas de Transferencia

Las colas de transferencia de Sun Grid Engine (SGE) son un tipo especial de cola de procesamiento de trabajos que, en lugar de estar asignadas a un nodo local de ejecución como es tradicional en los sistema de tipo clúster, se encaminan hacia un recurso Grid de tipo GRAM. Esto abre la puerta a configuraciones en las cuales aplicaciones que hagan uso de un interfaz directo a un LRMS como SGE puedan hacer uso transparente de recursos de otro clúster, a través de un interfaz Grid como GRAM. Por lo tanto, estas colas de transferencia pueden ser usadas como componente para lograr acceso transparente a infraestructuras.

La idea de extender esta componente con el uso de un *GridGateWay* surge

de manera inmediata, y las ventajas son obvias: pasamos de encapsular un clúster a encapsular un grid, potencialmente compuesto de varios de estos clústeres. La Figura 2.6 representa gráficamente una infraestructura con un único punto de acceso agregando diferentes recursos a través de interfaces distintos, basada en colas de transferencia de SGE y *GridGateWay*. En la capa superior nos encontramos la visión del SGE, con sus colas de trabajos locales y una cola especial de transferencia Grid, que permite al SGE enviar trabajos a un servicio GRAM. Este servicio puede, a su vez, encapsular un clúster o una instancia de GridWay (un *GridGateWay*), dando acceso a una infraestructura externa. La cola de transferencia puede ser configurada para estar disponible únicamente bajo ciertas condiciones, como por ejemplo un escenario de pico de carga de trabajo, o en caso de fallo de los recursos locales.

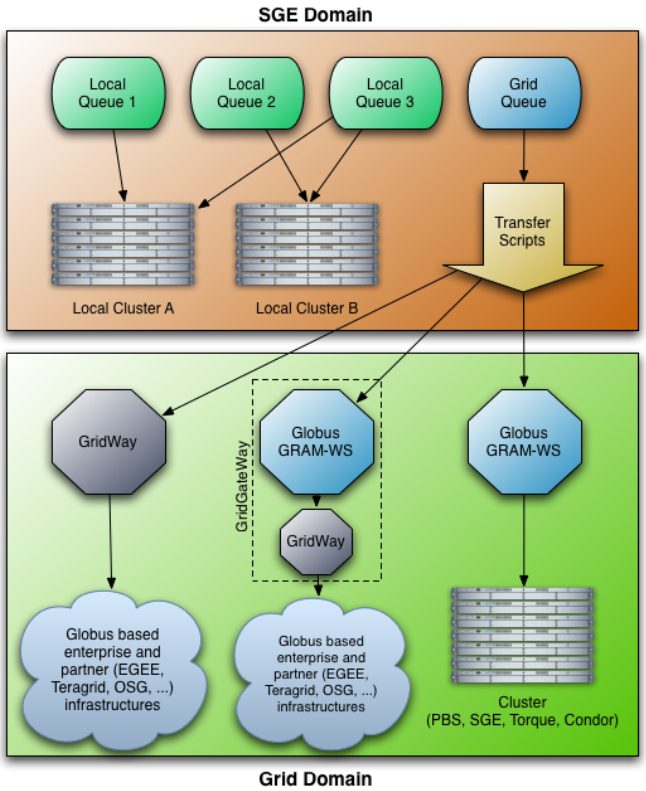


Figura 2.6: Acceso transparente a una infraestructura Grid por medio de colas de transferencia de SGE.

Esta integración, aunque directa, no ha estado exenta de problemas. El más obvio es la no disponibilidad de colas de transferencia para la versión *web service* de GRAM, si no para la versión anterior, empaquetada en Globus Toolkit 2.0. Las colas de transferencia consta de un conjunto de *scripts*

para manejar el ciclo de vida de un trabajo: lanzamiento, suspensión, reanudación y terminación; con lo que hubo que preparar un conjunto nuevo para interactuar con el nuevo GRAM, bajo el cual se había implementado el *GridGateWay* en un trabajo previo. El sistema resultante de la integración utiliza técnicas de adaptadores y pasarelas (ver sección 3.2)

2.3.2. El Interfaz DRMAA

DRMAA es un interfaz de programación estándar pensado para aplicaciones paralelas que se ejecuten en un clúster de computación. Tiene su origen en el Open Grid Forum, y surge como un esfuerzo para estandarizar el acceso de las aplicaciones a infraestructuras de tipo clúster. De esta manera, el programador describe las dependencias entre los procesos y datos de la aplicación, y orquesta la ejecución paralela de subtareas por medio de directivas DRMAA. Estas directivas ofrecen la posiblidad de describir ficheros de entrada y salida, de potencia de cálculo necesaria, y de otras funcionalidades que permiten al programador abstraerse de los detalles de la infraestructura (como por ejemplo, la interactuación con el LRMS).

Este potencialidad de encapsulamiento, de ocultar detalles, del interfaz DMRAA puede ser explotada más allá de lo que originariamente ha sido concebido para hacer, como el acceso transparente a infraestructuras de computación de tipo Grid en lugar de tipo clúster.

Como ejemplo práctico que demuestra la viabilidad de esta técnica puede tomarse la integración de GridWay y GRID superscalar (GRIDs). GRIDs es un entorno de desarrollo que encarna un paradigma de programación para aplicaciones que posibilita su ejecución en infraestructuras de tipo clúster. El entorno consta de un interfaz de programación y de un sistema de ejecución encargado de resolver las dependencias de las tareas en las que se descompone el programa construido en el entorno. Dicho programa está es descrito de forma secuencial, y GRIDs se encarga de convertirlo en un conjunto de aplicaciones paralelas que son ejecutadas en diferentes servidores. La implementación del interfaz DRMAA le permite ejecutar tareas en clústeres basados en LRMS que implementen el estándar: Condor, PBS, SGE, etc. En la integración de GRIDs con GridWay, llamada GridAD [5], se ha aprovechado la implementación DRMAA por parte del metaplanificador [24] para proporcionar acceso transparente a una infraestructura Grid usando un protocolo pensado para sistemas clúster. Esta integración, en la que ha participado el autor de esta tesis, ha tenido lugar en el contexto del proyecto BEinGRID, el cual surge como respuesta de la Unión Europea ante la falta de casos de uso de referencia del paradigma Grid en el sector industrial; y es un ejemplo de uso de un interfaz común (ver sección 3.2.1) para el acceso transparente no únicamente a una infraestructura distinta pero de la misma clase, si no a una infraestructura de un tipo distinto, en concreto un orden de magnitud mayor, ya que una infraestructura Grid es un conglomerado de

clústeres.

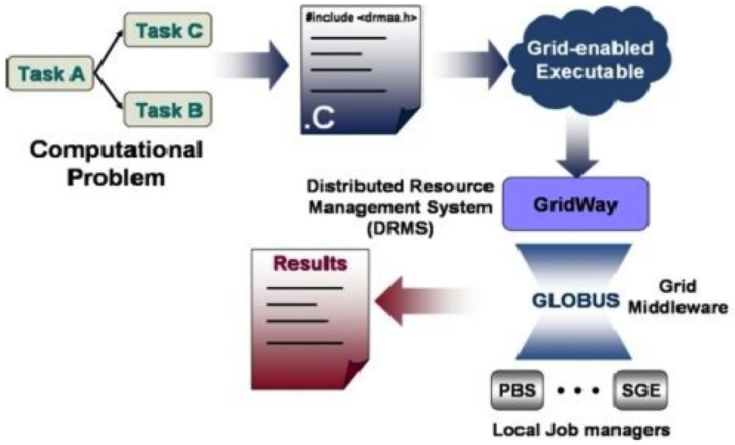


Figura 2.7: Integración de GridWay y DRMAA.

La Figura 2.7 muestra cómo funciona la integración entre GridWay y DRMAA. Aplicaciones compiladas usando las directivas DRMAA son ejecutadas por GridWay en la infraestructura que metaplanifica. Usando un ejemplo de aplicación fuertemente paralelizable (CD-HIT, más detalles en la sección 3.4.5) se han implementado varias infraestructuras federadas por medio de GridWay (con técnicas presentadas en esta tesis) a las que la aplicación CD-HIT se accede de forma transparente, y se ha presentado este sistema en varias conferencias y foros internacionales:

- Demostración en TeraGrid 07 Conference titulada "GridWay - A Metascheduler for Globus based Grids" [53].
- Presentación en EGEE 2007 titulada "GridWay Interfaces for on-demand Access to EGEE" [35].
- Presentación en Open Grid Forum 22 titulada "Scaling DRMAA codes to the Grid: A Demonstration on EGEE, TG and OSG" [54].

2.4. Modelo de Evaluación del Rendimiento de Infraestructuras Grid Federadas

A la hora de definir arquitecturas Grid federadas es de vital importancia disponer de un mecanismo para evaluar de manera fiable el rendimiento de la infraestructura resultante. Por lo tanto, se ha desarrollado en esta tesis un modelo de evaluación de rendimiento de infraestructuras Grid federadas, siguiendo el trabajo de caracterización de grids [27] empezado por el grupo de investigación al cual pertenece el autor.

Se ha partido de la metodología de evaluación de grids computacionales introducida en [38], en la cual se sugiere que un grid puede ser considerado, desde el punto de vista computacional, como un *array* de procesadores heterogéneos. De esta manera, la siguiente ecuación expresa el número de tareas completadas en función del tiempo:

$$n(t) = \sum_{i \in G} N_i \left\lfloor \frac{t}{T_i} \right\rfloor \tag{2.1}$$

donde N_i el el número de procesadores del recurso i en un grid G que puede computar una tarea en T_i segundos.

La mejor caracterización de un grid puede obtenerse teniendo en cuenta el comportamiento del sistema en promedio. La siguiente fórmula permite representar *n(t)* en función de los parámetros r_∞ y $n_{1/2}$, definidos por Hockney and Jesshope [25]:

$$n(t) = r_\infty t - n_{1/2} \tag{2.2}$$

Estos parámetros pueden definirse de la siguiente manera:

- Rendimiento asintótico (r_∞): el máximo ratio de rendimiento, entendido como tareas ejecutadas por segundo. En el caso de *arrays* de N procesadores homogéneos con un tiempo de ejecución por tarea de T, tenemos que $r_\infty = N/T$.
- Longitud de la mitad del rendimiento ($n_{1/2}$): el número de tareas requeridas para obtener la mitad del rendimiento asintótico. Este parámetro es también una medida de la cantidad de paralelismo en un sistema desde el punto de vista de la aplicación.

Medidas de estos parámetros pueden obtenerse a través de comparativas (o "benchmarking") intrusivas o no intrusivas [38]. La simplicidad de este modelo permite la caracterización de un grid utilizando únicamente los dos parámetros mencionados anteriormente.

Una aproximación de estos parámetros en el caso de infraestructuras Grid federadas sería el sumatorio de los valores de los parámetros de las infraestructuras componentes de la federación. Asumiendo que FG es el conjunto de grid federados, cada uno caracterizado por su modelo de rendimiento (i.e. r^i_∞ y $n^i_{1/2}$, $\forall i \in FG$):

$$\begin{aligned} r_\infty &= \sum_{i \in FG} r^i_\infty \\ n_{1/2} &= \sum_{i \in FG} n^i_{1/2} \end{aligned} \tag{2.3}$$

Este modelo de caracterización de grids federados es lo suficientemente flexible y fiable para aplicarse a problemas que necesiten de una expresión

computacional del comportamiento del grid. Por ejemplo, como se indica en esta tesis [50], se puede utilizar para calcular el número óptimo de trabajos que deberían ser enviados a una infraestructura Grid particular para obtener el tiempo mínimo de computación (o, expresado de otra forma, minimizar el tiempo máximo de ejecución completa de trabajos), y de esta manera ajustar políticas de planificación adecuadamente, ya que el modelo captura fehacientemente el comportamiento de la infraestructura Grid. Esta aproximación de metaplanificación ha sido explorada en más detalle por el grupo de investigación dsa-research.org [31, 32]

2.4.1. Validación del Modelo

Para validar este modelo se han ejecutado varios experimentos, de los cuales se presenta a continuación el más representativo. En él, se configura una infraestructura Grid federada en la cual una instancia de GridWay se utiliza para acceder a recursos locales (en este caso, recursos de la Universidad Complutense de Madrid) y a una *pasarela* por medio de la componente GridGateWay, bajo la cual se encapsula un Grid con un interfaz GRAM. Esta *pasarela* encapsula en este experimento recursos de la organización virtual de *Fusion*[5] de EGEE. A esta infraestructura federada se envían dos iteraciones de 100 trabajos cada una, cuya función es calcular el número π (aunque la finalidad del trabajo es irrelevante para el experimento), del cual se conoce el tiempo de transferencia (que incluye ficheros de entrada y de salida) y de ejecución. En este experimento estamos interesados en ver si el modelo de evaluación del rendimiento para predecir la *productividad* se ajusta a los valores obtenidos en la infraestructura federada. En la Figura 2.8 puede apreciarse el número de trabajos ejecutados por hora en la infraestructura federada, mientras que en la Figura 2.9 pueden verse estos datos contrastados con las predicciones del modelo.

Puede verse como en el experimento el modelo agregado es prácticamente idéntico al obtenido por medio de la medición directa sobre la infraestructura federada.

[5] http://grid.bifi.unizar.es/egee/fusion-vo/

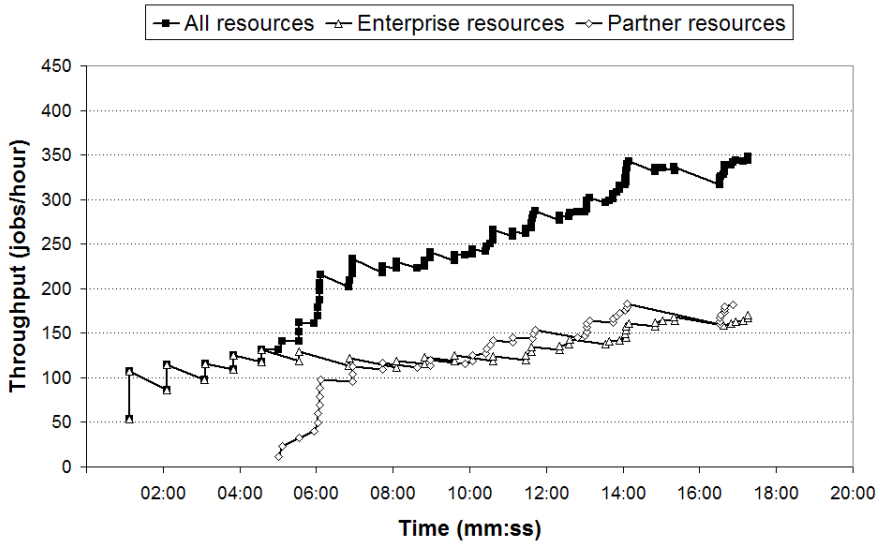


Figura 2.8: *Productividad* alcanzada en la ejecución del experimento.

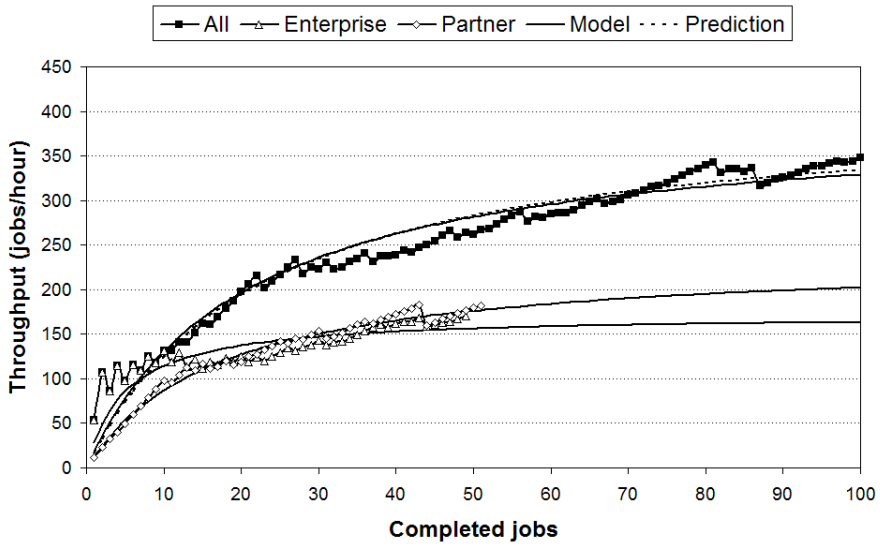


Figura 2.9: Modelo de evaluación del rendimiento aplicado al experimento.

Capítulo 3

Interoperabilidad de Infraestructuras Grid

Empty your mind, be formless, shapeless - like water. Now you put water into a cup, it becomes the cup, you put water into a bottle, it becomes the bottle, you put it in a teapot, it becomes the teapot. Now water can flow or it can crash. Be water, my friend.

Bruce Lee. *Tao of Jeet Kune Do*

La interoperabilidad de infraestructuras Grid es uno de los temas centrales de esta tesis. En este capítulo se introducen las condiciones necesarias para que se de la federación en la sección 3.1, así como técnicas de interoperabilidad en la sección 3.2 que permiten dicha federación, y la componente usada para llevar a cabo la misma, el metaplanificador, como se recoge en la sección 3.3. Se finaliza el capítulo en la sección 3.4, en la cual se presenta un caso práctico donde se aplican estas técnicas para conseguir una federación de las tres mayores infraestructuras Grid existentes.

3.1. Condiciones para la Federación

Podemos definir el concepto de federación como la unión comprendida de un número de entidades parcialmente auto gobernadas, unidas por medio de un gobierno central o "federal". En nuestro caso, las entidades referidas en la definición se pueden concretar en infraestructuras Grid. Existen diferentes esfuerzos dirigidos a federar distintas infraestructuras, como por ejemplo el trabajo desarrollado para que cooperen el WLCG y el grid canadiense HEPGrid [56]. Este trabajo se centra en el uso de una capa central de metaplanificadores como encarnación del gobierno federal central.

Una parte fundamental a la hora de proponer una estrategia y una arquitectura para la federación de infraestructuras Grid es el estudio de las condiciones necesarias y suficientes para conseguir dicha federación, que para cuestiones prácticas podemos expresar como la habilidad de un usuario concreto de ejecutar sus trabajos de computación en cualquiera de los recursos presentes en las partes que componen el sistema federado.

Un análisis general del problema revela que la mayoría de estas condiciones caen fuera del ámbito puramente técnico, y se restringen a áreas centradas en cuestiones políticas (por ejemplo, cuándo y cómo un usuario debería o no usar una infraestructura determinada, en relación a los intereses de sus propietarios), o en áreas meramente operacionales (por ejemplo, qué software está instalado en los nodos de ejecución), las cuales tampoco afectan necesariamente a la estrategia de federación desde un punto de vista de la interoperabilidad del middleware Grid.

Considerando únicamente los aspectos puramente técnicos, nos encontramos con dos condiciones absolutamente necesarias para poder hablar de una federación efectiva. La primera es la interoperabilidad del middleware Grid, la cual permitiría el envío de trabajos a cualquier componente de la federación. Es tentador pensar en la presencia de *interfaces comunes* (ver sección 3.2) como condición necesaria y *suficiente* para la federación Grid, pero como expone esta tesis [51], no es condición que garantice la federación, ya que en ocasiones son necesarios técnicas adicionales de interoperabilidad cuando los modelos difieran (por ejemplo, en casos en que los modelos de almacenamiento implementados por las infraestructuras sean muy diferentes, aunque los interfaces ofrecidos sean idénticos). De esta manera, podemos concluir que la interoperación es una condición necesaria para la federación, y que dicha condición supone un requisito más fuerte que la interoperabilidad. Habiendo aclarado esto, es conveniente indicar que normalmente las palabras *interoperabilidad* e *interoperación* se utilizan como sinónimos en la literatura, siendo más frecuente el uso de la primera. Para evitar confusión, se utilizará a partir de ahora este término para referirse a la interactuación entre dos infraestructuras Grid, sin discriminar la naturaleza de dicha interactuación.

De cualquier forma, el hecho de considerar la interoperabilidad como condición suficiente para la federación deja fuera de la ecuación uno de los factores principales. Este factor es la *escalabilidad*, es de vital importancia considerar si el componente encargado de federar realmente escala. Entendemos como *escalabilidad* la habilidad del sistema de seguir manteniendo una nivel de rendimiento aceptable aunque el tamaño del sistema se incremente, principalmente en lo que se refiere a recursos de computación o infraestructuras federadas. El estudio de escalabilidad que se ha llevado a cabo en esta tesis (ver sección 3.3), muestra la escalabilidad del metaplanificador GridWay que se ha usado como actor principal de la federación. Llegados a este

punto podemos establecer que, manteniendo a un lado los aspectos no puramente técnicos, la verdadera federación surge la de la conjunción de dos factores clave: *interoperabilidad* y *escalabilidad*.

Otro aspecto a considerar a la hora de diseñar una arquitectura federada está relacionado con la experiencia del usuario final de la infraestructura. Es importante ofrecer una visión homogénea del sistema resultante, a pesar de la potencial heterogeneidad sobre la que se construye la federación. Podemos extraer por tanto otra condición necesaria de federación, que se puede expresar como el *acceso transparente* a la infraestructura. En esta tesis [5] se ha tenido en cuenta este aspecto, a través principalmente del soporte por parte del metaplanificador GridWay del interfaz DRMAA para LRMS, permitiendo el acceso a una infraestructura federada a aplicaciones pensadas originalmente para su ejecución en clústeres locales, sin que sea necesaria su modificación.

3.2. Técnicas de Interoperabilidad

Existen diferentes técnicas de interoperabilidad disponibles a la hora de diseñar la arquitectura de una una infraestructura Grid federada, presentadas en esta sección.

3.2.1. Interfaces Comunes

El uso de *interfaces comunes* es la forma más ideal, directa y simple (y, como todo aquella forma que se puede tildar de ideal, prácticamente utópica) de conseguir interoperabilidad de infraestructuras Grid. La técnica necesaria es la aplicación literal de su definición, es decir, la presencia en cada infraestructura miembro de la federación de interfaces compartidos, con lo cual la federación surge de manera casi espontánea de la homogeneidad presente entre todas las partes. La manera más lógica y aparentemente directa de conseguir esta homogeneidad en los interfaces es por medio de estándares. Existen una multitud de estándares que cubren prácticamente todo el espectro de funcionalidad ofrecido por los distintos middleware Grid, caben destacar al respecto los esfuerzos del OGF (Open Grid Forum[1]). A continuación se muestra un listado no exhaustivo de funcionalidades Grid que cuentan con uno o varios estándares:

- **Ejecución de trabajos**. El estándar más popular para describir trabajos de procesamiento por lotes para su envío a infraestructuras Grid es JSDL (Job Submission Description Language[2]). También en esta

[1] http://www.gridforum.org/
[2] http://www.gridforum.org/documents/GFD.56.pdf

categoría podemos encontrar estándares para la iniciación, monitorización y gestión de actividades computacionales, como es el caso del OGSA-BES.

- **Servicios de información**. Esta categoría se refiere a aquellos estándares encargados de describir los aspectos de una infraestructura Grid que produzcan información consumible, como por ejemplo el estado de la infraestructura en términos de memoria RAM disponible, cantidad de trabajos ejecutados, etc. El esquema GLUE (Grid Laboratory for a Uniform Environment[3]) es un ejemplo de estándar de esta categoría.

- **Transferencia de Ficheros** La transferencia de ficheros en los sistemas Grid requieren un trato distinto a las transferencias convencionales debido al volumen de los datos manejados. Estándares como el GridFTP[4] lidian con este aspecto.

- **Autenticación y Autorización** La seguridad en sistemas Grid es de vital importancia, sobre todo si tenemos en cuenta su aspecto de compartición de recursos entre empresas podemos ver cuán importante es poder asegurar quién es quién y qué puede hacer. Gracias a su adopción por el Globus Toolkit, el uso de certificados y *proxies* X.509 para realizar estas tareas es un estándar en el mundo Grid.

Una vez satisfecha la condición de *interfaces comunes*, la interoperabilidad es simplemente una cuestión administrativa relacionada con la autorización de los usuarios y la imposición de políticas de uso de recursos. En este tipo de federación se puede plantear un único punto de acceso (por ejemplo, un portal), teniendo en cuenta que cada parte puede ser accedida usando los mismos mecanismos. No debe asumirse la presencia del mismo middleware Grid como garantía de interfaz común, ya que, como se desprende de varios experimentos realizados en esta tesis, distintas configuraciones del mismo middleware supone diferencias a la hora de utilizar los interfaces expuestos.

Desafortunadamente, en el panorama actual de software middleware Grid nos encontramos con implementaciones incompatibles de estándares e incluso con middleware tempranos que no los implementan. Las razones por las cuales no se unifican los interfaces son variadas (razones políticas, falta de coordinación entre infraestructuras, diferencias en el tiempo en los que se establecieron, migraciones o actualizaciones en progreso), y están fuera del ámbito de este trabajo, aunque probablemente la razón más importante para esta falta de acuerdo sea la falta de un estándar aceptado de middleware Grid. Todo esto nos lleva a la necesidad de contar con técnicas adicionales de interoperabilidad. En concreto existen dos técnicas, *adaptadores* y *pasarelas*, prestadas del dominio de la gestión de redes de computación, suficientemente

[3] http://forge.ogf.org/sf/projects/glue-wg
[4] http://www.ogf.org/documents/GFD.20.pdf

generales para que sean de aplicación en otros dominios a la hora de diseñar arquitecturas software. En el dominio particular de la computación Grid, estas técnicas se usan para conseguir la interactuación de infraestructuras Grid heterogéneas, es decir, que no comparten interfaces comunes.

3.2.2. Adaptadores

Es frecuente encontrar diferentes middleware Grid desplegados en las infraestructuras candidatas a la federación. Una posible solución a esta situación es el uso de diferentes componentes (para el envío de trabajos, recogida de información sobre el estado de la infraestructura, transferencia de ficheros, etc) para interactuar con las diferentes infraestructuras. Estas componentes, llamadas *adaptadores*, están específicamente diseñadas para una capa particular de middleware Grid, o incluso para diferentes versiones de un mismo middleware. En su acepción más general, un *adaptador* es un "dispositivo que permite a un sistema conectarse e interactuar con otro". Un ejemplo de esta técnica son los distintos drivers de GridWay (introducidos en la sección 1.2), utilizados precisamente para acceder a infraestructuras Grid con diferentes middlewares Grid desplegados. Su principal inconveniente es la necesidad de modificar las herramientas clientes de los servicios Grid para la inserción de los adaptadores.

3.2.3. Pasarelas

Otra técnica para conseguir federación de infraestructuras Grid consiste en encapsular uno o más infraestructuras en un único recurso accedido a través de un único interfaz. De esta manera, el problema de federación se puede reducir al escenario de *interfaces comunes*. Este encapsulamiento actúa traduciendo las peticiones del único punto de acceso (como por ejemplo un portal web) en peticiones que las infraestructuras encapsuladas puedan entender, y transmitiendo de vuelta los resultados. Esta técnica es conceptualmente similar a las pasarelas utilizadas en las redes de computación, también llamadas "gateways", y está especialmente indicada para aquellas situaciones en las que no es posible la modificación de los servicios Grid de alto nivel. El uso de pasarelas tiene la ventaja de ofrecer un acceso homogéneo al grid encapsulado, con lo que las herramientas clientes del grid no necesitan ser modificadas. Por contra, son un potencial punto singular de fallo, así como un posible cuello de botella a la hora de escalar [18].

3.3. Interoperabilidad a Nivel de Metaplanificador

Como se vio en la sección 1.1.1, existen distintos servicios Grid que se encargan de funciones específicas bien definidas, que encapsulan aspectos diferentes de la infraestructura. Una de ellas es el servicio de metaplanificación,

encargado de escoger los nodos de ejecución óptimos para cada trabajo. Un ejemplo de metaplanificador es GridWay, el cual además de ejercer labores de metaplanificación también orquesta distintos servicios Grid para ofrecer una visión conjunta de la infraestructura al usuario final.

La arquitectura de GridWay (Figura 3.1) está diseñada no únicamente para planificar, si no también para orquestar los servicios de transferencia de ficheros, de información del estado de la infraestructura y trabajos en ejecución, así como los aquellos dedicados a controlar el ciclo de vida de los trabajos de procesamiento por lotes. Otro aspecto que se ha tenido también en cuenta en el diseño de este metaplanificador es el de la interoperabilidad (ver sección 1.3). La interoperabilidad ha sido tenida en cuenta implementando estándares para interactuar con los distintos servicios Grid (los estándares enumerados en la sección 3.2 son utilizados por GridWay). La interoperabilidad también está especialmente presente como motivo conductor en la arquitectura de GridWay. Los componentes encargados de interactuar con servicios Grid (llamados Middlware Access Drivers, o MADs) actúan como adaptadores (ver sección 3.2), permitiendo su reemplazo o incluso uso simultáneo para acceder a infraestructuras con diversos interfaces basados en diferentes estándares. También es interesante el uso del interfaz de ejecución de trabajos GRAM[5] interactuando con GridWay, lo que se conoce como GridGateWay [49], para encapsular infraestructuras Grid (potencialmente federadas), ofreciéndose al usuario final como una infraestructura uniforme, o incluso como recurso a otro GridWay, abriendo el camino para infraestructuras Grid organizadas de forma jerárquica, como las *infraestructuras Grid con capa múltiple de metaplanificadores*.

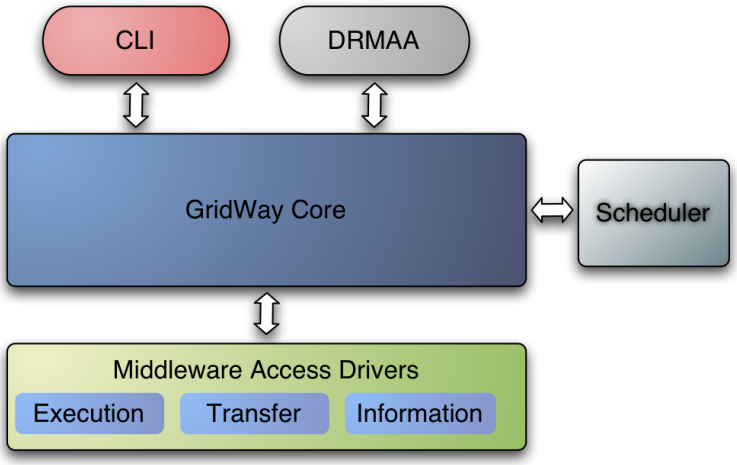


Figura 3.1: Una arquitectura para la interoperabilidad: GridWay.

La robustez y la escalabilidad han sido también tenidos en cuenta en

[5]http://dev.globus.org/wiki/GRAM

el diseño del metaplanificador. Parte del trabajo de esta tesis [51] ha sido mejorar y analizar el comportamiento de GridWay ante situaciones de estrés. El trabajo se ha centrado en conseguir un uso óptimo de recursos por parte de todas las componentes de GridWay. En las Figuras 3.2 y 3.3 podemos ver gráficas de consumo de memoria y procesador respectivamente, ante escenarios de gestión de trabajos en volúmenes del orden de decenas de miles. También se han realizado esfuerzos en el consumo óptimo de espacio de almacenamiento y en la velocidad de ejecución y robustez del servicio GridWay.

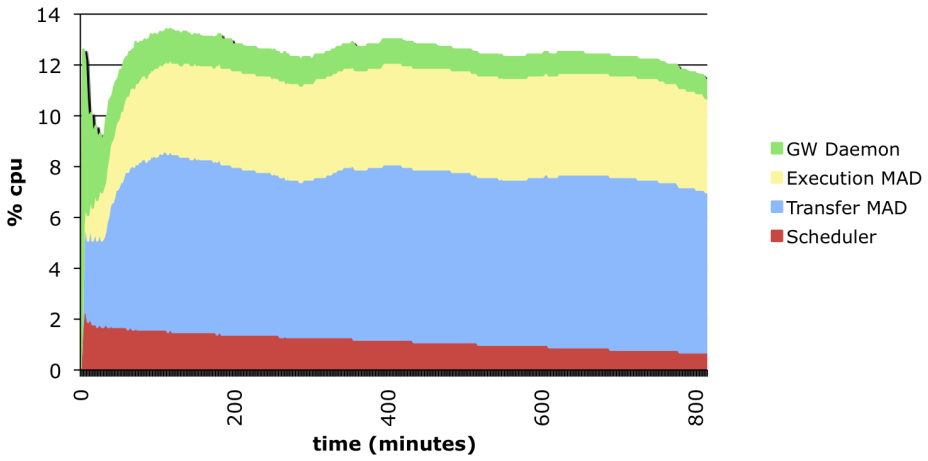


Figura 3.2: Consumo de procesador del metaplanificador.

3.4. Federación de EGEE, TeraGrid y Open Science Grid

Uno de los resultados prácticos del presente trabajo de investigación ha sido la federación efectiva de tres de la mayores infraestructuras Grid en el mundo, el grid del proyecto EGEE (europea), TeraGrid y Open Science Grid (americanas). El acceso a dichas infraestructuras ha sido posible dado el interés en el estudio sobre la compatibilidad de éstas con el metaplanificador GridWay, ya que con la inclusión del metaplanificador en el Globus Toolkit[6], GridWay pasa a ser parte del standard "de facto" en los middleware Grid. En esta sección se introducen y se describen las principales características de cada una de las ellas.

[6]http://dev.globus.org/wiki/GridWay

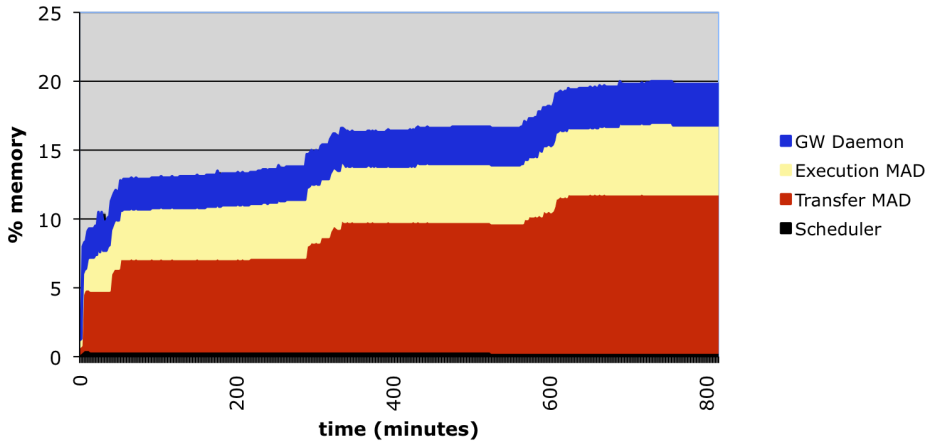


Figura 3.3: Consumo de memoria del metaplanificador.

3.4.1. Enabling Grids for E-SciencE

El proyecto Enabling Grid for E-SciencE (EGEE) tenía como objetivo proporcionar a investigadores de academia e industria acceso a recursos de computación de altas prestaciones, independientemente de su localización geográfica. El proyecto tiene tres objetivos principales:

- construir una infraestructura Grid consistente y robusta que atraiga recursos de computación adicionales.
- mantener y mejorar el middleware Grid sobre el que se basa la infraestructura para proveer de un servicio fiable.
- atraer nuevos usuarios y proporcionarles la formación y el soporte necesario.

La infraestructura Grid del proyecto EGEE se sustenta sobre la red de computación GÉANT[7], y se basa en el middleware de computación Grid gLite[8]. Su aparición fue financiada por la Unión Europea, y propiciada inicialmente por la alta demanda de computación prevista para el acelerador de partículas existente en el CERN, pero se ha ido diversificando para dar servicio a una miríada de comunidades científicas, como la de aplicaciones biomédicas, químicas, hidrología, climatología, geofísica, etc. Estas comunidades se reúnen en las llamadas "organizaciones virtuales" (VOs), las cuales cuentan con cuotas de uso de los distintos tipos de recursos disponibles.

[7] http://www.geant.net/
[8] http://glite.cern.ch/

El middleware gLite ha sido desarrollado expresamente para el EGEE. Propone unos requisitos fuertes (su instalación sólo se puede llevar a cabo sobre una distribución linux particular, Scientific Linux), con el fin de ofrecer cotas altas de disponibilidad y compatibilidad a la hora de instalarse y actualizarse. Actualmente, gLite es mantenido por el proyecto EMI (European Middleware Initiative[9]).

El proyecto EGEE finalizó en abril del 2010, pero la continuidad de la infraestructura Grid EGEE está asegurada por el proyecto EGI (European Grid Initiative[10]).

Un detalle de la configuración de esta infraestructura es la no compartición de directorios entre los servidores encargados de recibir peticiones y los nodos encargados de procesarlas, por lo que el *script* de empaquetado de trabajos que utiliza GridWay tuvo que ser modificado para ejercer también de gestor de transferencias de archivos.

3.4.2. TeraGrid

La infraestructura Grid de TeraGrid está orientada a la investigación científica, que combina recursos de alta capacidad de once centros de computación para crear un grid persistente e integrado. Usando conexiones de red de alto rendimiento, el TeraGrid integra computadores de altas prestaciones, recursos de datos y herramientas para la investigación, consiguiendo una capacidad de computación de 1 petaflop, así como más de 30 petabytes de almacenamiento. Esta infraestructura está financiada por el grupo de infraestructuras Grid de NSF (National Science Foundation), y su coordinación grid está centralizada en el GIG (Grid Infraestructure Group), que opera en la Universidad de Chicago.

Los recursos de TeraGrid están integrados a través de una arquitectura orientada a servicio, en la que cada recursos ofrece un "servicio" definido en términos de interfaz y operación. Esto es posible a través de un conjunto de paquetes software que conforman los servicios y software cooirdinados de TeraGrid (Coordinated TeraGrid Software and Services, CTSS). El propósito de CTSS es el de ofrecer un entorno de usuario familiar, permitiendo a los científicos ejecutar su código en cualquiera de los dominios de administración asociados al TeraGrid. Asimismo, CTSS provee funciones integradas como la autenticación singular, el envío remoto de trabajos, el soporte para flujos de trabajo, herramientas para la transferencia de datos, etc; e incluye software Grid y de computación en clúster como el Globus Toolkit y Condor.

Los adaptadores de GridWay, especialmente el encargado de la transferencia de archivos, tuvo que ser modificado para poder interactuar con los servidores de almacenamiento de TeraGrid, ya que tradicionalmente suelen

[9] http://www.eu-emi.eu/
[10] http://www.egi.eu/

estar situados en el mismo servidor que el encargado de aceptar peticiones de trabajos, pero esta infraestructura los presenta en dos servidores diferentes. Además, se ofrecen dos versiones distintas del Globus Toolkit (la pre web services y la actual) en diferentes infraestructuras miembro.

3.4.3. Open Science Grid

Open Science Grid (OSG[11]) es un consorcio de software, servicios, proveedores de servicio e investigadores de universidades, laboratorios nacionales y centros de computación a lo largo de Estados Unidos, centrados en ofrecer una infraestructura Grid común a través de una red de investigación y un conjunto común de middleware. El proyecto está financiado por la NSF y el Departamento de Energía (DoE), y ofrece a las comunidades de investigación participantes acceso a más recursos computacionales de a los que podrían tener acceso por separado. Entre los miembros más destacados y que más recursos aportan podemos señalar FermiLab y SLAC National Accelerator Laboratory.

Uno de los objetivos de OSG es la de trabajar conjuntamente con otras infraestructuras de computación, a nivel internacional, para conseguir un sistema interoperable a nivel mundial, como por ejemplo el WLCG para los experimentos planeados en el CERN.

OSG basa su middleware común (el Virtual Data Toolkit[12], o VDT) de grid en el Globus Toolkit, aunque también incluye componentes de gLite, de Condor y de Virtual Organization Management Service (VOMS). Con este middleware es posible que las comunidades de investigación formen "organizaciones virtuales", compartiendo recursos y a la vez aportando las herramientas software necesarias al VDT para poder ser usados en todas las infraestructuras adheridas al OSG.

El metaplanificador GridWay soporta el acceso a infraestructuras Grid basadas en el Globus Toolkit, pero debido a peculiaridades de la configuración de OSG (como por ejemplo, el uso de puertos no estándares), los adaptadores del metaplanificador tuvieron que ser alterados para lidiar con estas particularidades.

3.4.4. Experimentos de Interoperabilidad

Como ejemplo de las distintas infraestructuras que se pueden crear a partir de utilizar el metaplanificador GridWay como herramienta de federación, se han realizado distintos experimentos para demostrar la viabilidad de las técnicas de interoperabilidad descritas en la sección 3.2. En los más representativos se utilizan recursos de grandes infraestructuras Grid (EGEE, TeraGrid, Open Science Grid), de forma combinada con recursos locales de

[11] http://www.opensciencegrid.org
[12] http://vdt.cs.wisc.edu

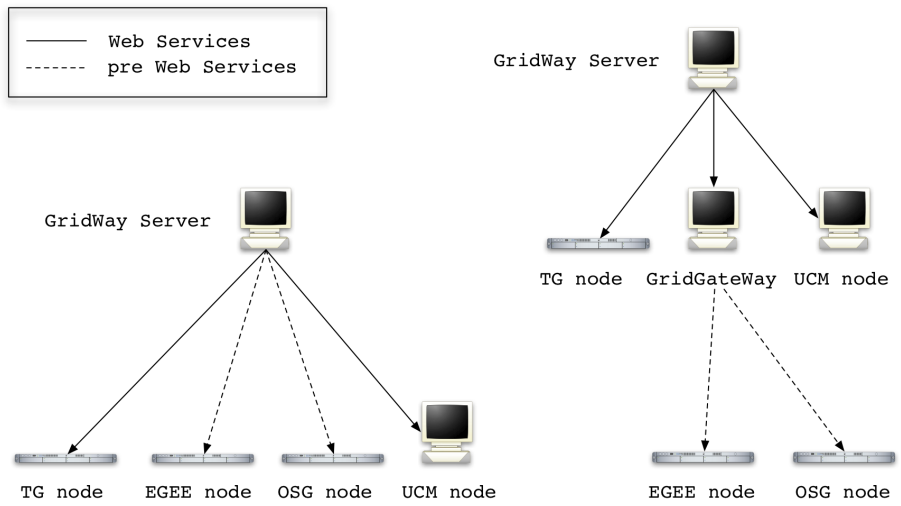


Figura 3.4: Experimentos de federación: Adaptadores (izquierda) y Pasarelas (derecha).

la Universidad Complutense de Madrid. En la Figura 3.4 puede verse la configuración de dos infraestructuras para sendos experimentos en los que se ejercitan los *adaptadores* y las *pasarelas*. En el escenario de *adaptadores* podemos observar un servidor GridWay configurado para acceder por medio de interfaces Globus web services y pre web services a recursos de cuatro infraestructuras. La interactuación con los distintos interfaces por parte de GridWay se realiza utilizando distintos adaptadores (o MADs en la terminología de GridWay), cada uno desarrollado para intectuar con una versión particular de Globus. El núcleo de GridWay envía a los adaptadores intrucciones sobre las acciones a realizar, que los adaptadores traducen a peticiones adecuadas a los servicios de ejecución. Por otra parte, el escenario de *pasarelas* presenta una arquitectura jerárquica. Los mismos cuatro recursos que en el escenario anterior son accedidos, pero los recursos que presentan un interfaz pre web services son encapsulados por medio de un *GridGateWay*, el cual ofrece un interfaz web services. De esta manera se divide la infraestructura en dos capas jerárquicamente organizadas, la superior ofreciendo un interfaz web services, y la inferior uno pre web services.

Sobre ambas infraestructuras se ejecutan tres iteraciones de doscientos trabajos cada una. La *productividad* (medido en número de trabajos satisfactoriamente completados por hora) de cada escenario se recoge en las tablas 3.1 y 3.2, para demostrar la viabilidad de las federaciones propuestas.

Puede verse en la Figura 3.5 la distribución de trabajos en las tres iteraciones del experimento de *Adaptadores*, mientras que la manera en que se reparten los trabajos por las distintas infraestructuras en el experimento de *pasarelas* está plasmado gráficamente en la Figura 3.6. En esta última puede

Iteración #	Total	UCM	TG WS	EGEE	OSG preWS
1	125.78	37.10	31.44	12.57	44.65
2	144.92	39.85	41.30	14.49	49.27
3	122.7	38.65	35.58	13.49	34.96

Tabla 3.1: Productividad conseguida (total y por infraestructura) en el escenario de *Adaptadores*.

Iteration #	Total	UCM	TG WS	EGEE	OSG preWS
1	132.45	37.75	37.08	13.90	43.70
2	126.58	40.51	39.87	14.55	31.64
3	121.21	40.61	36.97	13.33	30.30

Tabla 3.2: Productividad conseguida (total y por infraestructura) en el escenario de *Pasarelas*.

observarse que la columna que representa al *GridGateWay* (GGW) presenta el agregado de trabajos computados por las infraestructuras del proyecto EGEE y de OSG. La razón por la que el número de trabajos de la infraestructura Grid del proyecto EGEE es bajo si lo comparamos con las demás que conforman el experimento es la elección de recursos de comparativamente menor rendimiento.

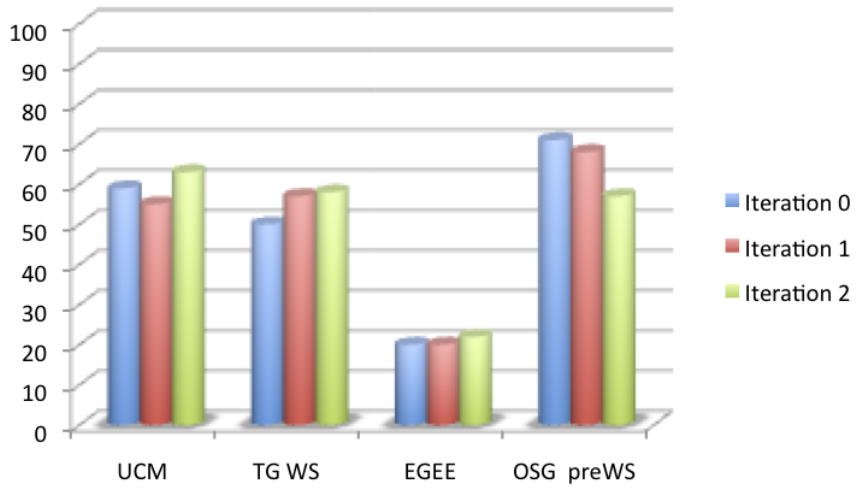


Figura 3.5: Distribución de trabajos a través de las infraestructuras en el experimento de *adaptadores*.

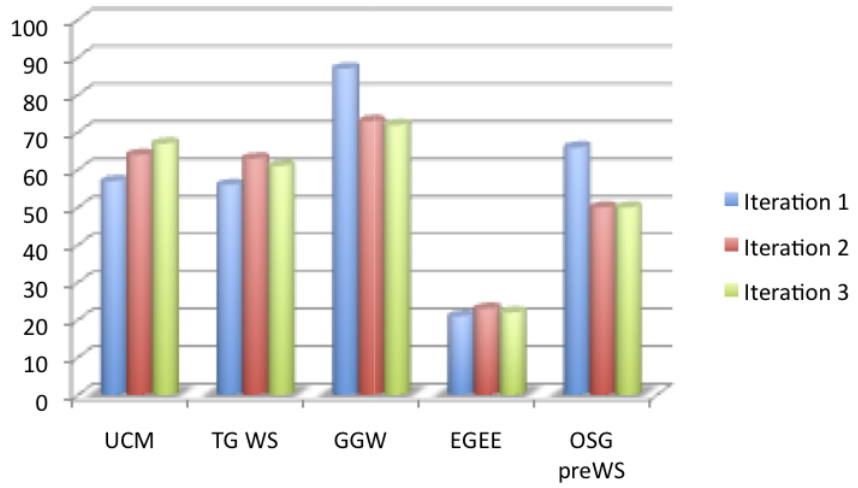


Figura 3.6: Distribución de trabajos a través de las infraestructuras en el experimento de *pasarelas*.

3.4.5. Experimentos de Uso del Intefaz DRMAA en Infraestructuras Federadas a Gran Escala

Se han realizado como parte del trabajo de esta tesis dos experimentos adicionales en los cuales se utilizan infraestructuras federadas usando las técnicas de *adaptadores* sobre las se ejecutan aplicaciones diseñadas para interactuar por medio del interfaz DRMAA.

El primer experimento fue presentado en la vigésimo segunda edición del Open Grid Forum, bajo el título de "Scaling DRMAA codes to the Grid: A Demonstration on EGEE, TG and OSG" [54]. La infraestructura federada se contruye a través de GridWay interactuando sobre recursos de TeraGrid, EGEE, Open Science Grid y de la Universidad Complutense, sobre la que se ejecutan varias iteraciones del alogritmo CD-HIT [33] (ver Figura 3.7). Este algoritmo permite la reducción de redundancia en una base de datos de proteínas, y es un candidato perfecto para su uso en infraestructuras de computación distribuida dado su alto grado de paralelismo. La aplicación que utiliza el algoritmo CD-HIT se vale del interfaz DRMAA para enviar de forma transparente trabajos a GridWay, el cual utiliza los recursos de las infraestructuras federadas para ejecutar las tareas dictadas por CD-HIT.

El segundo experimento fue presentado en el evento TeraGrid07, titulado "GridWay - A Metascheduler for Globus based Grids" [53]. Puede verse en la Tabla 3.9 un listado de los recursos accedidos por GridWay en este experimento, en la que constituyó probablemente la infraestructura federada más compleja y de más número de miembros creada a lo largo de esta Tesis,

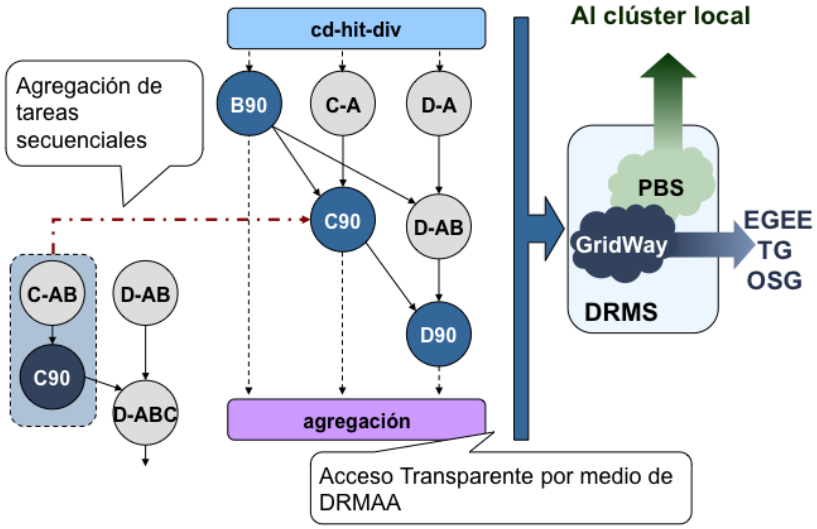


Figura 3.7: Ejecución del algoritmo CD-HIT sobre una infraestructura Grid federada a través de GridWay.

ya que para no complicar innecesariamente los experimentos de la sección anterior sólo se han escogido para ellos nodos representativos de cada infraestructura. Las infraestructuras utilizadas en la demostración presentada en TeraGrid07 son: recursos del grid del proyecto EGEE (HID: 0-4), del grid de OSG (HID: 6-9), un clúster de TeraGrid (HID:10) y recursos locales de la universidad Complutense (HID: 14-18). Sobre esta infraestructura federada se llevaron a cabo también varias iteraciones del algoritmo CD-HIT. La Figura 3.8 muestra los estados por los que pasan los trabajos que ejecutan las tareas realizadas en este experimento.

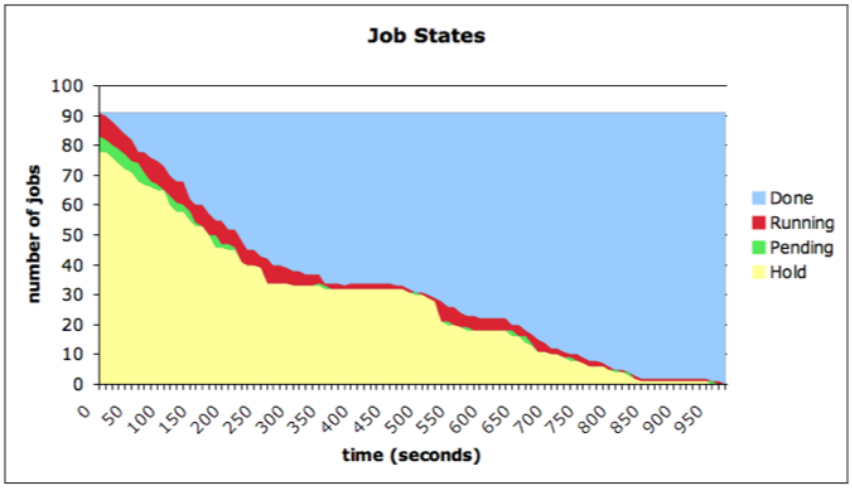


Figura 3.8: Estados de los trabajos del algoritmo CD-HIT ejecutados en una infraestructura Grid federada.

HID	PRI	OS	ARCH	MHZ	%CPU	MEM(F/T)	DISK(F/T)	N(U/F/T)	LRMS	HOSTNAME
0	20	Linux2.4.21-32.	i686	2665	189	964/2006	62787/73964	0/2/1	jobmanager-fork	atlas.dpcc.uta.edu
1	20	Linux2.4.21-37.	athlo	1991	110	1214/2009	62680/71679	0/1/1	jobmanager-condor	ce01.cmsaf.mit.edu
2	20	Linux2.4.21-20.	i686	2665	388	1246/2006	70846/99795	0/1/1	jobmanager-condor	fiupg.ampath.net
3	20	Linux2.4.27-1-3		1595	95	381/884	6350/7525	0/1/1	jobmanager-condor	grid.physics.purdue.edu
4	20	Linux2.6.9-42.0	x86_6	2592	200	3041/3901	12040/13770	0/1/1	jobmanager-condor	osg.hpcc.nd.edu
5	75							0/89/300	PBS	tg-grid.uc.teragrid.org
6	1	ScientificSLSL	i686	1200	0	1024/1024	0/0	0/51/75	jobmanager-pbs	lcg2ce.ific.uv.es
7	1	ScientificSL4	i686	866	0	513/513	0/0	0/22/22	jobmanager-lcgpbs	ramses.dsic.upv.es
8	1	ScientificSL4	i686	2800	0	1024/1024	0/0	0/152/158	jobmanager-lcgpbs	lcg-ce.usc.cesga.es
9	1	ScientificSLBer	i686	3000	0	1024/1024	0/0	0/49/108	jobmanager-lcgpbs	ce2.egee.cesga.es
10	1	ScientificSLBer	i686	4000	0	1024/1024	0/0	0/0/24	jobmanager-lcgpbs	ifaece01.pic.es
11	25							0/214/219	Condor	nest.phys.uwm.edu
12	25							0/6/11	Condor	osg-itb.ligo.caltech.edu
13	90	Linux2.6.17-2-6	x86	3216	0	45/2027	70824/118812	0/0/2	Fork	cygnus.dacya.ucm.es
14	90	Linux2.6.17-2-6	x86	3216	181	681/2027	98861/11881	0/2/2	Fork	draco.dacya.ucm.es
15	90	Linux2.6.18-4-a	x86_6	2211	100	954/1003	77081/77844	0/3/4	PBS	hydrus.dacya.ucm.es
16	90	Linux2.6.18-4-a	x86_6	2211	100	776/1003	76428/77844	0/5/5	SGE	aquila.dacya.ucm.es

Figura 3.9: Recursos accedidos por el metaplanificador GridWay en la demonstración de TeraGrid07.

Capítulo 4

Aprovisionamiento Dinámico desde Infraestructuras Cloud

The history of the commercial application of IT has been characterized by astounding leaps, but nothing that has come before - not even the introduction of the personal computer or the opening of the Internet - will match the upheaval that lies just over the horizon.

Nicholas Carr. *The End of Corporate Computing*

Este capítulo introduce las diferencias y similitudes de las dos tecnologías de computación distribuida reinantes hoy en día, como se puede ver en la sección 4.1, basándonos en las características de las aplicaciones nativas de cada tipo de infraestructura, descritas en la sección 4.2. Seguidamente, se presentan una solución en la sección 4.3 para aprovechar los puntos fuertes de ambos paradigmas, y se finaliza demostrando en la sección 4.4 la viabilidad de la solución propuesta.

4.1. Coexistencia de Tecnologías Grid y Cloud

Es un hecho que recientemente el paradigma de computación distribuida conocido como computación Cloud o computación en la nube está tomando fuerza. Hay numerosos proyectos de investigación dedicados al estudio de la nube, centrados en el diseño e implementación de infraestructuras que presenten los beneficios prometidos por la tecnología Cloud (flexibilidad y coste, principalmente) en los sistemas computacionales académicos, gubernamentales y de la industria. El autor ha participado en varios de estos proyectos,

como el proyecto europeo RESERVOIR[1], destinado al diseño de un middleware Cloud para implementación de clouds privados sin perder de vista una eventual federación, o como el proyecto BonFIRE[2], cuya meta es la construcción de una infraestructura Cloud europea para su uso por parte de la comunidad científica.

A pesar de que las tecnologías Cloud puedan verse como una evolución de las tecnologías Grid, también es cierto que cada tecnología tiene definido su campo de actuación, e incluso puede considerarse como una tecnología específica al dedicarse a la resolución de la problemática de un área en concreto. El propósito de este capítulo es mostrar no sólo que una coexistencia de las dos tecnologías es posible, sino que además pueden usarse conjuntamente creando infraestructuras que se benefician de las ventajas de ambas aproximaciones. Esta coexistencia se hace posible dado el diferente dominio de las aplicaciones que se ejecutan: aplicaciones masivamente paralelas, no interactivas, con grandes demandas de tiempo de procesador en las infraestructuras Grid; frente a aplicaciones de ámbito más general, destinadas al usuario final e interactivas en las infraestructuras Cloud. Existen varios ejemplos en la literatura sobre las distintas direcciones en las que se pueden explorar distintas colaboraciones entre estos dos tipos de infraestructuras, pero quizás la más prometedora y la que ciertamente está alineada con la investigación de esta tesis es la de construir clústeres virtuales que se adapten dinámicamente a la demanda, siguiendo la linea del software COD (Cluster On Demand) [12].

La idea de aprovisionar grids (o sus componentes, los clústeres) a través de clouds se entiende si tenemos en cuenta que las características de un cloud incluyen flexibilidad a la hora de ejecutar cualquier aplicación. Las ventajas de ejecutar una infraestructura Grid sobre un cloud va más allá de las ventajas de la simple virtualización:

- Consolidación y aprovisionamiento rápido de recursos al grid
- Soporte a varias organizaciones virtuales, aprovechando las características de aislamiento que ofrecen las infraestructuras Cloud
- Crecimiento/decrecimiento elástico de la infraestructura Grid
- Posibilidad de usar recursos computacionales de un proveedor Cloud público
- Mayor flexibilidad en el soporte de aplicaciones desde los nodos de ejecución, al poderse aprovisionar nodos heterogéneos fácilmente a través de la virtualización

[1] http://www.reservoir-fp7.eu
[2] http://bonfire-project.eu

Existen también desventajas en el uso de grids sobre infraestructuras Cloud, siendo probablemente la más notoria la pérdida de eficiencia debido a la introducción de otra capa computacional, la virtualización. Sin embargo, ciertos estudios [57, 58] señalan que realmente la capa de virtualización no tiene un impacto significativo en la eficiencia de ejecución de aplicaciones intensivas en memoria y/o procesador (aplicaciones HPC) que se ejecutan tradicionalmente en las infraestructuras Grid. Sin embargo, es indudable que las ventajas superan a los inconvenientes, como prueba la existencia de proyectos europeos como StratusLab, el cual explora precisamente la ejecución de Grids sobre tecnología Cloud. Son interesantes a este respecto también los planes de futuro de la European Grid Initiative[3], en concreto los workshops de virtualización, lo cual ya supone una acercamiento en europa del mundo Grid a la tecnología Cloud.

4.2. Caracterización de Aplicaciones Grid y Cloud

Las *tecnologías Grid* están especializadas en la resolución de trabajos por lotes. Se puede argumentar que nacen para resolver problemas que demandan una alta capacidad de computación, lo que se conoce como High Performance Computing o HPC [16]. Este paradigma de computación se centra en aplicaciones con código fuertemente acoplado, sincronizado, paralelo y optimizado, las cuales se ejecutan en sistemas dedicados de altas prestaciones. Una variante de este paradigma es el High Troughput Computing o HTC [34], en el cual no se imponen condiciones sobre cuándo se termina de ejecutar una tarea, y por el contrario se prima la eficiencia entendida como la ejecución del mayor número de tareas en un período de tiempo. Es importante destacar el hecho de que estas aplicaciones se ejecutan de forma desatendida (el usuario envía los trabajos y se "olvida" - *submit and forget*), y su ejecución no implica ninguna interactividad con el usuario final, ya que están centradas en el procesamiento masivo de grandes cantidades de datos. Un ejemplo de aplicación típica de una infraestructura Grid es aquella que tiene como característica fundamental un alto grado de paralelismo (aplicaciones "embarrisingly distributed"), lo cual las hace un candidato ideal para ser ejecutadas en un clúster manejado por un LRMS, componente fundamental de una infraestructura Grid. En este ejemplo [14] de aplicación no interactiva se ha ayudado a implementar una aplicación de trazado de rayos para conseguir un tratamiento radiológico eficiente. Los resultados se esperan para empezar el tratamiento pero no se necesitan en tiempo real, el cálculo, que necesita una gran potencia de procesamiento, puede descomponerse en partes más pequeñas a partir de las cuales puede construirse fácilmente la solución final.

Por otra parte, el conjunto de *tecnologías Cloud* (es importante insistir otra vez en la vertiente de infraestructura en la que se utiliza este término a

[3] http://www.egi.eu

lo largo de este trabajo) puede dar soporte a servicios más generales en donde la interactuación con el usuario final es más dinámica y requiere un tiempo de respuesta corto. En contraposición a los problemas con los que trata típicamente una infraestructura Grid, las infraestructuras Cloud se centran principalmente en dar soporte a servicios interactivos, si bien no necesariamente en tiempo real, si orientados a usuario final, en el que el tiempo de respuesta corto es fundamental para hacerlos útiles. Una de las claves para comprender las tecnologías Cloud es poder ver un servicio como un conglomerado de componentes, cada uno residiendo en una máquina virtual, y comunicándose a través de redes virtuales. Una característica interesante de esta descripción es la variabilidad en el número de nodos de ejecución, denominada la elasticidad del servicio. Es precisamente la elasticidad una de las ventajas principales que ofrece la tecnología Cloud a la hora de ejecutar servicios; la responsabilidad de su manejo recae en un componente llamado *service manager*, o gestor de servicios. En la Figura 4.1 puede verse una arquitectura general para un sistema Cloud, en la que se basan las infraestructuras Cloud existentes hoy en día. Un ejemplo de un tipo de servicio que se puede beneficiar de ejecutarse en un cloud es el de un servidor web con balanceo de carga, con una máquina virtual actuando de punto de entrada (*front-end*), la cual se encarga de encaminar el tráfico a un número variable de nodos de ejecución que a su vez se conectan a una base de datos y potencialmente a otros componentes.

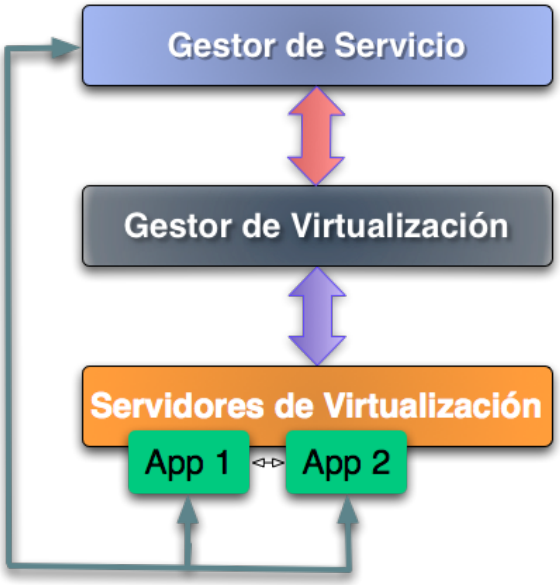


Figura 4.1: Arquitectura típica de un sistema Cloud.

De lo expuesto anteriormente se puede concluir que, desde el punto de

Tipo de Aplicación	Interactividad	Elasticidad	Eficiencia	Flexibilidad
Grid	No	No	Si	No
Cloud	Si	Si	No	Si

Tabla 4.1: Características de aplicaciones Grid vs Cloud.

vista de la aplicación (la cual condiciona enormemente las características que se demandan de la infraestructura), existen importantes diferencias que se pueden resumir sucintamente en la Tabla 4.1.

La elasticidad no suele ser un problema acuciante en las aplicaciones Grid, debido principalmente a su no interactividad, lo cual hace que su demanda de computación sea como mínimo predecible, si no constante. Asimismo, la eficiencia de las aplicaciones Grid está garantizada en tanto en cuanto se ejecutan sin capas adicionales como la impuesta por la virtualización en las tecnologías Cloud (es decir se ejecutan sobre *bare metal*), las cuales suelen tener un impacto en el tiempo de entrada/salida (I/O) de las aplicaciones.

Una característica ciertamente deseable en cualquier aplicación es la flexibilidad, referida esta a la disponibilidad de distintas arquitecturas, con distintos sistemas operativos y herramientas necesarias para ejecutar trabajos. Las infraestructuras Cloud ciertamente ofrecen esta característica, ya que se puede ejecutar cualquier máquina virtual con prácticamente cualquier arquitectura, contextualizada cada máquina para satisfacer los requisitos de aplicaciones concretas. Es mucho más difícil conseguir esto en un grid, ya que cada nodo de ejecución es una máquina física, con un sistema operativo y unos paquetes software en particular cuyo cambio, actualización o reemplazo no es tarea trivial, en contraposición a la forma dinámica que se consigue en el Cloud, configurando y lanzando una o varias máquinas virtuales nuevas.

4.3. Arquitectura para el Aprovisionamiento Dinámico

El planteamiento que se ofrece en esta tesis es una arquitectura de aprovisionamiento dinámico basada en el metaplanificador GridWay, en el cual se combinan técnicas de ambos mundos, Grid y Cloud, para conseguir una infraestructura adaptable a la demanda, flexible en el tipo de aplicaciones que pueda ejecutar y con capacidad de crecimiento a proveedores Cloud públicos. La infraestructura que se propone se basa en la flexibilidad de GridWay para acceder a distintas infraestructuras Grid, como se expuso en la sección anterior. Para incrementar todavía más la flexibilidad a la hora de gestionar recursos, se usan también técnicas de clústeres, como es el acceso ssh a nodos sin necesidad de un interfaz Grid. En este modo de funcionamiento, GridWay puede verse como un LRMS.

De lo anterior se desprende que la infraestructura propuesta está orientada a la ejecución de trabajos por lotes, ya que se trata de una infraestructura Grid construida sobre una fábrica Cloud. Las ventajas de este tipo de despliegue son las de proporcionar la flexibilidad del Cloud a las infraestructuras Grid: velocidad de despliegue, capacidad elástica de respuesta a demandas de computación variable, facilidad de provisión de nuevos servicios Grid, de distintas librerías software, etc.

Un ejemplo de la potencia de este planteamiento puede verse si aplicamos esta infraestructura al problema de las organizaciones virtuales que existe en en la infraestructura Grid del proyecto EGEE. Estas organizaciones virtuales son agrupaciones de recursos que prestan servicio a una comunidad científica, como por ejemplo *BioMed* para aplicaciones de bioinformática, o *Fusion*, para físicos especializados en altas energías. Si se sirviesen a estas comunidades con una infraestructura como la propuesta, se podría incrementar o disminuir el número de nodos dedicados a cada comunidad de una manera rápida (apagando o encendiendo máquinas virtuales con las librerías software utilizadas por cada comunidad), ofreciendo mejor servicio a cada una de ellas que con una infraestructura estática.

La arquitectura de la infraestructura propuesta puede verse en la Figura 4.2. La figura del metaplanificador (GridWay en nuestro caso) sigue siendo central en la arquitectura, proporcionando acceso transparente e interoperando infraestructuras Grid con distintos interfaces de una forma escalable usando adaptadores, es decir, federando. También se puede observar cómo en esta arquitectura se introduce también un híbrido de tecnologías Grid y clúster, ya que GridWay accede también directamente a nodos de computación sin interfaz Grid, usando las técnicas de clúster tradicionales, como puede ser el acceso a través de ssh.

Es importante destacar la componente encargada de la gestión de servicios, el *service manager*, la cual monitoriza un servicio y le añade o quita potencia de computación dinámicamente según la carga de trabajo que tenga que satisfacer el servicio. En este caso, el servicio que se ejecuta en la infraestructura Cloud es una infraestructura Grid, con lo cual el *service manager* debe monitorizar el nivel de carga de la infraestructura para hacer decisiones sobre la elasticidad requerida, es decir, sobre si la infraestructura Grid debe crecer o decrecer. La monitorización entonces se realiza sobre el metaplanificador, concretamente en la cola de trabajos de GridWay. En una primera versión, el *service manager* contiene heurísticas sencillas, en las cuales se detecta si la cola sobrepasa un cierto umbral calculado sobre el número de nodos totales disponibles en la infraestructura Grid. Cuando este umbral se sobrepasa, el *service manager* utiliza adaptadores Cloud para interactuar con hipervisores presentes en los servidores físicos de la infraestructura local y lanzar más máquinas virtuales que conformarían una nueva infraestructura Grid o nodos clúster accesibles por ssh, comunicando la disponibilidad

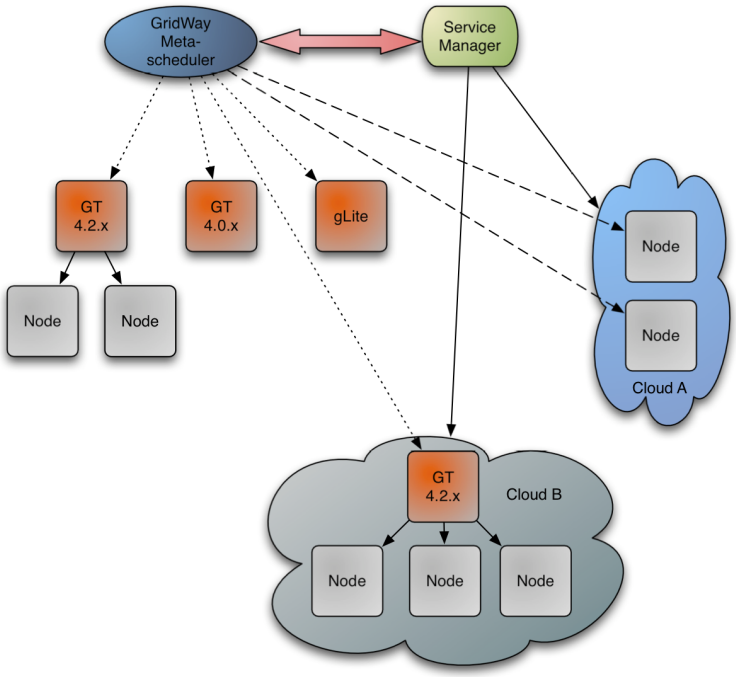


Figura 4.2: Provisión de Grids usando infraestructuras Cloud.

de estos nodos a GridWay, que de esta manera puede empezar a enviarles trabajos. De manera inversa, si la cola de trabajos desciende de un cierto umbral, el *service manager* apaga máquinas virtuales que estén actualmente en desuso (es decir, que no estén procesando trabajos), y los elimina de la lista de recursos disponibles para GridWay.

El *service manager* también utiliza adaptadores Cloud para interactuar con servidores de Cloud público (como por ejemplo Amazon EC2[4] o ElastiCHosts[5]), para poder utilizar otros recursos aparte de los disponibles en la infraestructura física, con un modelo de pago bajo demanda. Esta técnica se conoce con el nombre de *cloudbursting*.

4.4. Experimentos de Validación

Se han llevado a cabo varios experimentos para verificar la viabilidad de la arquitectura propuesta. En esta sección se presentan los más relevante, diseñados para evaluar el coste operativo (*overhead*) incurrido en la gestión de un nuevo nodo de ejecución añadido (a discreción de un *service manager*) a un clúster virtual basado en el LRMS Sun Grid Engine (SGE[6]) ejecutado

[4]http://aws.amazon.com/ec2/
[5]http://www.elastichosts.com/
[6]http://gridscheduler.sourceforge.net/

sobre el gestor de virtualización Globus Nimbus[7]. Este clúster virtual forma parte de una infraestructura Grid planificada por una instancia de GridWay.

Con el propósito de esta evaluación, se ha desplegado un clúster virtual manejado por Globus Nimbus ya en ejecución, y se pretende medir el coste en tiempo de añadir un nuevo nodo al clúster virtual a través de todas las capas de arquitectura, su apagado, su detección por parte de SGE, el servicio de información de Globus (MDS) y GridWay, y la penalización de procesamiento incurrida por el nodo por estar virtualizado. El experimento ha sido realizado en un cloud privado local desplegado en el laboratorio de *dsa-research.org*. La Figura 4.3 muestra una descripción gráfica del experimento y también de parte de los resultados.

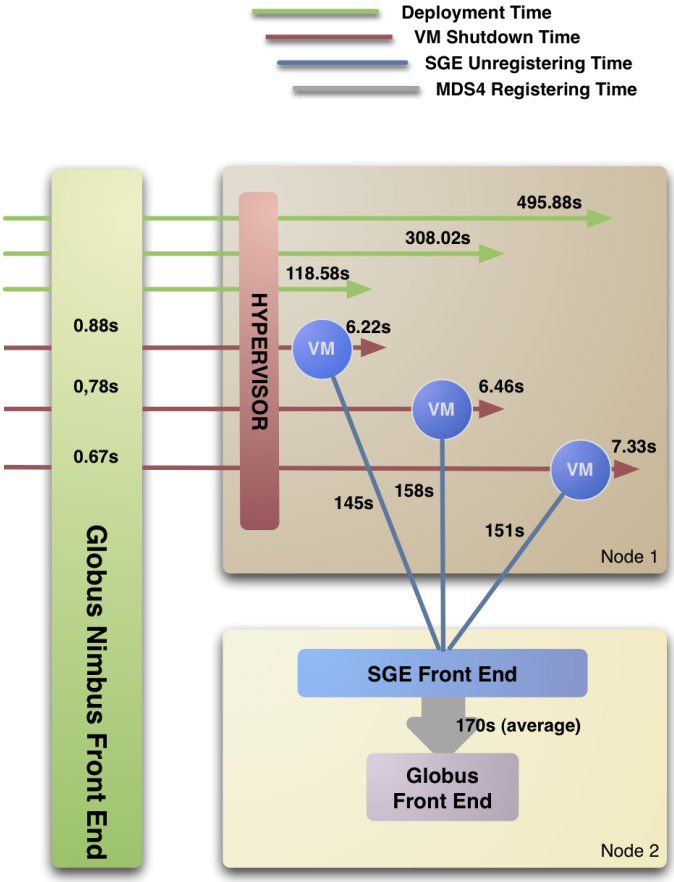


Figura 4.3: Penalizaciones de virtualización gestionando una infraestructura Grid sobre una infraestructura Cloud.

Para comprender mejor dónde se encuentran los costes de computación adicionales es importante poder seguir la cadena de acciones que se lleva a

[7]http://www.nimbusproject.org/

cabo cuando el *service manager* decide añadir un nuevo nodo de ejecución al cluster virtual, y, por lo tanto, a la infraestructura Grid a la cual pertenece.

- En primer lugar el *service manager* informa a Nimbus de la necesidad de añadir un nuevo nodo al clúster, y éste (Nimbus) determina el mejor nodo físico para ejecutar la máquina virtual que contiene el nodo clúster, basando la elección en los requisitos computacionales demandados por dicha máquina virtual.

- A continuación, si la imagen física de la máquina virtual no reside en el nodo físico, se accede al servidor de imágenes a través de un protocolo adecuado (que puede ser GridFTP) para obtener una copia.

- Una vez que la imagen es transferida, el servidor DHCP se configura para establecer la IP y el nombre de la nueva máquina virtual.

- Al finalizar estas operaciones, la máquina virtual se inicia y el nodo virtual se registra al SGE, pasando a formar parte del clúster virtual. Pasado un tiempo, el servicio de información Grid (MDS) detecta el nuevo nodo y lo publica.

- Finalmente, GridWay actualiza la información disponible en los recursos de la infraestructura Grid y detecta el nuevo nodo de ejecución, con lo cual, y de acuerdo con las políticas de planificación con las que esté configurado, podrá enviar trabajos en el nuevo nodo por medio del interfaz GRAM de ejecución de trabajos.

En la infraestructura descrita anteriormente se han realizado distintas medidas para calcular los distintos costes de ejecución. El primer coste considerado es el causado por el despliegue de una máquina virtual bajo diferentes condiciones. El experimento realizado consistió en el despliegue de una, dos y tres máquinas virtuales en el mismo servidor físico. Estos tiempos están anotados en la Figura 4.3 como *deployment time*. Puede observarse como este tiempo depende del número de máquinas virtuales que se despliegan, ya que en este caso se utiliza un único servidor físico.

Una vez que la máquina virtual se inicia, transcurre un tiempo hasta que se enrola en la infraestructura Grid. El tiempo total desde que se pide un nodo virtual y éste está completamente inicializado y dispuesto para procesar trabajos es de unos 2 minutos aproximadamente (114 segundos para su registro en el *service manager*, 2 segundos para su registro en el LRMS SGE). Es destacable el hecho de que los tiempos de penalización introducidos por la capa del SGE al registrar nuevos nodos sean insignificantes (menores del uno por ciento) si se comparan con el tiempo total de inicialización. El tiempo que se necesita para registrar el nodo en el servicio de información de Grid (MDS4 en este experimento) es de 170 segundos. Es interesante indicar

que el tiempo de publicación en el MDS4 es mayor que el de iniciación de una máquina virtual más el tiempo de registro en el SGE, con lo cual la inicialización de varias máquinas de manera secuencial hace que estos tiempos se solapen y se pueda, por tanto, ahorrar tiempo adicional. El coste de registro en MDS y GridWay pueden ser a su vez reducidos ajustando su intervalo de monitorización.

En el caso de la penalización por el apagado de una máquina virtual (ver Tabla 4.2) se han tomado tres valores relevantes. *Comando recibido* mide el tiempo que transcurre desde que se realiza la petición a Globus Nimbus hasta que llega al hipervisor. *MV destruida* expresa el tiempo que pasa hasta que la máquina virtual (MV) se apaga completamente desde que el hipervisor recibe la orden de apagado. Ambos tiempos están consolidados en la figura como *VM shutdown time*. En este caso el tiempo es constante independientemente del número de máquinas virtuales que se apagan. También se necesita tiempo para desacoplar el nodo de la infraestructura. Estos tiempos son mucho menores que el caso de *deployment time* (de 114 segundos a solo 7 de media)

La penalización introducida por el SGE al apagar una máquina virtual (*SGE unregistering time* en la figura) puede ser minimizado reduciendo el tiempo entre intervalos de monitorización del SGE. Por defecto son 300 segundos, así que la media de tiempo que tarda el SGE en ser notificado de la nueva situación es de 150 segundos. El lado negativo de este reducción de tiempo de monitorización es un aumento del consumo de red, con lo cual hay que alcanzar un compromiso. Estos 150 segundos suponen un sustancial incremento si los comparamos con los 2 que tarda el SGE en registrar un nuevo nodo. Es interesante considerar que el metaplanificador puede mandar trabajos a una MV que ya está apagada, pero no desacoplada del LRMS. En estas situaciones, el metaplanificador debería reenviar el trabajo a otro recurso cuando este falle.

Número	Comando Recibido	MV Destruida	SGE	Total
1	0.78s	6.22s	145s	152s
2	0.88s	6.46s	158s	165.33s
3	0.67s	7.33s	151s	159s

Tabla 4.2: Penalización en segundos por el apagado de nodos virtuales.

La tecnología de virtualización impone una penalización en la eficiencia debido a la capa adicional entre el hardware y el sistema operativo virtualizado, penalización que depende de varios factores: el hardware usado, la tecnología de virtualización y el tipo de aplicaciones que se estén ejecutando. Como demuestran algunos estudios [6, 13], Xen se comporta particularmente

bien en cualquier tipo de tarea, con una pérdida de eficiencia entre el 1 y el 15 %. VMware también consigue una eficiencia casi nativa en aplicaciones intensivas en procesador, pero experimenta una severa penalización (hasta el 88 %) en aplicaciones intensivas en entrada/salida. El equipo de desarrollo de Nimbus refiere [21] una pérdida de solo el 5 % en una aplicación de meteorología en un grid virtualizado. Resultados previos [45, 44] indican que, en general, la virtualización no tiene un impacto significativo en la ejecución de aplicaciones intensivas en procesador y memoria, típicas de clústeres HPC.

Capítulo 5

Principales Aportaciones y Trabajo Futuro

5.1. Principales Aportaciones

El trabajo de esta tesis se desarrolla simultáneamente con la transición del paradigma Grid al paradigma Cloud. Aunque es cierto que la tecnología Grid supuso un salto importante a la hora de aglutinar recursos para construir infraestructuras con cada vez mayor potencia de cálculo, es cada vez más patente que el paradigma Cloud está llamado a revolucionar verdaderamente el mundo de las tecnologías de la información (TI) como lo conocemos, desde el organigrama clásico que se maneja en las empresas (los departamentos de TI se enfrentan a su posible extinción como se conocen actualmente) hasta los entornos de desarrollo de aplicaciones, que cada vez más necesitan tener en cuenta conceptos de la *nube*, como escalabilidad, deslocalización, pago por uso, etc; pasando por todos los componentes (hardware y software) que sustentan la oferta de servicios informáticos y que conforman los Centros de Procesamiento de Datos (CPDs).

Desde un inicio, el mundo Grid se ha visto amenazado por el mundo Cloud de una manera directa, sea esta vista por una disminución de las ayudas a programas de investigación Grid por su desviación a investigación en infraestructuras y modelos Cloud o, de manera general, por el cambio de foco de uno a otro paradigma. Según se defiende en esta tesis, ambos paradigmas son complementarios, y tanto el mundo académico como el empresarial pueden beneficiarse de la colaboración de ambas tecnologías. Es interesante considerar en este aspecto los esfuerzos realizados por la industria para ofrecer soporte computacional a través de infraestructuras Cloud a las grandes instituciones científicas europeas (EMBL[1],ESA[2], CERN[3]), las cuales hacen

[1] http://www.embl.de
[2] http://www.esa.int
[3] http://cern.ch

uso intensivo hoy en día de aplicaciones Grid de procesamiento por lotes. Un ejemplo de estos esfuerzos es el proyecto *Helix Nebula*, un consorcio formado por la industria e instituciones científicas para formar una plataforma europea de cloud computing para proporcionar computación Grid. Estas sinergias hacen posible una transición suave entre paradigmas, o incluso el aprovechamiento de los beneficios de ambos, sin enfrentarse a cambios inmediatos y drásticos en sus aplicaciones Grid tradicionales.

Una de las principales aportaciones de esta tesis es la descripción, diseño, implementación y verificación de varias arquitecturas de sistemas distribuidos, destinadas a la construcción de infraestructuras capaces de ofrecer recursos computacionales como un servicio. Una parte importante del trabajo se ha centrado en la interoperabilidad entre sistemas informáticos, de vital importancia en cualquier ámbito de la computación distribuida. También se han desarrollado metodologías de evaluación de rendimiento para demostrar la viabilidad de las soluciones propuestas. Se han sugerido maneras de aprovechar las bondades de ambas tecnologías en infraestructuras híbridas como la presentada en el capítulo 4. A modo de resumen, las principales aportaciones de esta tesis son:

- Modelo de Computación como Servicio basado en la federación de infraestructuras Grid (ver capítulo 2) [49] [48].
- Modelo de evaluación del rendimiento de grids federados (ver sección 2.4) [50].
- Descripción, diseño y aplicación de técnicas de interoperabilidad de infraestructuras Grid (ver capítulo 3) [51].
- Diseño e implementación de técnicas de aprovisionamiento dinámico de infraestructuras Grid usando tecnología Cloud (ver capítulo 4) [52] [55].

Como ya se ha mencionado, vivimos en una etapa de transición. Parece razonable pensar que tenemos por delante una época de simbiosis entre ambas tecnologías, empezando por las más directa, el uso de infraestructuras Cloud como sustento a aplicaciones Grid. Seguramente habrá intentos (de hecho, ya se están dando los primeros pasos) para construir un cloud global, del mismo modo que se ha concebido el proyecto fallido de construir un grid global y que tenemos un internet global. Ya se está viendo una fuerte tendencia a la fragmentación en la oferta de infraestructuras Cloud, con distintos interfaces y filosofías. Los problemas a los que se enfrentan los tempranos proyectos de federación Cloud son viejos conocidos del mundo Grid: autenticación y autorización (¿quién tiene permiso para utilizar los recursos, y cuáles y cuánto de ellos?), compartición del almacenamiento (¿dónde están mis datos?), confianza (¿están seguros mis datos?), facturación (¿cuánto están usando nuestros recursos, y cuánto cuesta?), etc. Es importante que

no se haga borrón y cuenta nueva, y que se aprovechen los vastos esfuerzos hechos en la última década en el ecosistema Grid para afrontar estos mismos problemas.

5.2. Trabajo Futuro

El actual es un momento muy interesante y excitante en el campo de la computación distribuida. El momento de crisis que se vive actualmente en la gestión de infraestructuras abre un gran número de posibilidades a la hora de buscar campos de investigación y de diseño e implementación de tecnología.

Un campo muy prometedor es el de la gestión de infraestructuras virtuales, también denominados cloud privados. El autor de esta tesis está involucrado en el proyecto OpenNebula[4], que desarrolla una solución para la construcción de distintos tipos de Cloud. Esta participación le ha permitido ser un miembro activo de la comunidad científica dedicada al estudio de la computación como servicio, así como desarrollar literatura sobre el paradigma de computación Cloud a través de la participación en el desarrollo de libros sobre la materia [8, 47].

El campo del aprovisionamiento dinámico de recursos presenta varios problemas. Uno de ellos es la importancia de la interoperabilidad, es decir, ser capaz de crecer una infraestructura usando cualquier tipo de recurso, por medio de interfaces comunes u otras técnicas que se han visto en este trabajo. Su aplicación al mundo Cloud es de gran interés y un prometedor campo de trabajo futuro. Otra pregunta es cuándo el crecimiento (o su contrario) es necesario y cómo se lleva a la práctica. En este aspecto sería interesante la investigación y desarrollo del componente de *service manager*. En concreto parece de vital importancia tener en cuenta el aspecto del coste de uso de recursos, a la hora de planificar el crecimiento dinámico automático de una infraestructura, este campo parece fértil para su estudio futuro, sin descuidar otros aspectos como el cumplimiento de Service License Agreements (SLAs) para garantizar una satisfactoria calidad de servicio.

[4] http://opennebula.org

Apéndice A

Artículos Adjuntos

A.1. Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures

Cita completa

C. Vázquez, E. Huedo, R.S. Montero and I.M. Llorente. *Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures.* In Proceedings of the 13th International Conference on Parallel Processing (Euro-Par 2007). **Lecture Notes in Computer Science**, 4641, 372-381, 2007.

Indicios de calidad

Este congreso está en la categoría CORE A según el ERA Conference Ranking. En la edición de 2007 recibió 333 envíos de los que se aceptaron 89, lo cual supone un ratio de aceptación de aproximadamente 26 %. Ha sido citado 22 veces según Google Scholar.

Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures*

Tino Vázquez, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid, Spain
{tinova,ehuedo}@fdi.ucm.es, {rubensm,llorente}@dacya.ucm.es

Abstract. Utility computing is a service provisioning model which will provide adaptive, flexible and simple access to computing resources, enabling a pay-per-use model for computing similar to traditional utilities such as water, gas or electricity. On the other hand, grid technology provides standard functionality for flexible integration of diverse distributed resources. This paper describes and evaluates an innovative solution for utility computing, based on grid federation, which can be easily deployed on any infrastructure based on the Globus Toolkit. This solution exhibits many advantages in terms of security, scalability and site autonomy, and achieves good performance, as shown by results, mainly with compute-intensive applications.

1 Introduction

Utility is a computing term related to a new paradigm for an information technology (IT) provision which exhibits several potential benefits for an organization [1]: reducing fixed costs, treating IT as a variable cost, providing access to unlimited computational capacity and improving flexibility, thereby making resource provision more agile and adaptive. Such valuable benefits may bring the current fixed-pricing policy of IT provision to an end, where computing is carried out within individual corporations or outsourced to external service providers [2].

The deployment of a utility computing solution involves a full separation between the provider and the consumer. The consumer requires a uniform, secure and reliable functionality to access the utility computing service and the provider requires a scalable, flexible and adaptive infrastructure to provide the service. Moreover, the solution should be based on standards and allow a gradual deployment in order to obtain a favorable response from application developers and IT staff.

* This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, by Ministerio de Educación y Ciencia, through research grant TIN2006-02806, and by European Union, through BEinGRID project EU contract IST-2005-034702.

A.-M. Kermarrec, L. Bougé, and T. Priol (Eds.): Euro-Par 2007, LNCS 4641, pp. 372–381, 2007.

In a previous work [3], we have proposed a solution for federating grids which can be deployed on a grid infrastructure based on the Globus Toolkit (GT). Such a solution demonstrates that grid technology overcomes utility computing challenges by means of its standard functionality for flexible integration of diverse distributed resources.

A similar approach for the federation of grid infrastructures has been previously applied to meet LCG and GridX1 infrastructures [4], hosting a GridX1 user interface in a LCG computing element. However, this solution imposes software, middleware and network requirements on worker nodes. The Globus project is also interested in this kind of *recursive* architectures, and is working on Bouncer [5], which is a Globus job forwarder initially conceived for federating TeraGrid and Open Science Grid infrastructures. There are other approaches to achieve middleware interoperability, for example between UNICORE and Globus [6] and between gLite and UNICORE [7].

On the other hand, MOAB Grid Suite from Cluster Resources[1] is a grid workload management solution that integrates scheduling, management, monitoring and reporting of workloads across independent clusters. Moreover, Condor's Flocking and GlideIn mechanisms [8] provide similar functionality, allowing job transfers across Condor pools' boundaries or the deployment of remote Condor daemons, respectively. Nevertheless, these solutions are not based on standards and require the same workload manager to be installed in all resources. In this sense, the Open Grid Forum's Grid Scheduling Architecture research group[2] is working on a standard architecture for the interaction between different metaschedulers.

Finally, other projects, like Gridbus [9] or GRIA [10], are developing components for accounting, negotiation and billing, in order to provide end-to-end quality of service driven by computational economy principles.

In this work, we use a technology for the federation of grid infrastructures to build a utility model for computing, which provides full metascheduling functionality, and it is flexible, scalable and based on standards. The rest of this paper is as follows. Section 2 presents a solution for building utility grid infrastructures based on the Globus Toolkit and the GridWay Metascheduler by means of GridGateWays. The architecture of a GridGateWay is described and evaluated in Section 3, while Section 4 shows some experimental results. Finally, Section 5 presents some conclusions and our plans for future work.

2 Utility Grid Infrastructures

A grid infrastructure offers a common layer to integrate non-interoperable computational platforms by defining a consistent set of abstraction and interfaces for access to, and management of, shared resources [11]. Most current grid infrastructures are based on the Globus Toolkit [12], that implements a collection of high level services at the grid infrastructure layer. These services include,

[1] http://www.clusterresources.com
[2] http://forge.gridforum.org/projects/gsa-rg

among others, resource monitoring and discovery (Monitoring and Discovery Service, MDS), resource allocation and management (Grid Resource Allocation and Management, GRAM), file transfer (Reliable File Transfer, RFT, and GridFTP) and a security infrastructure (Globus Security Infrastructure, GSI). The Globus layer provides a uniform interface to many different Distributed Resource Manager (DRM) systems, allowing the development of grid workload managers that optimize the use of the underlying computing platforms.

The technological feasibility of the utility model for computing services is established by using a novel grid infrastructure based on Globus Toolkit components and the GridWay Metascheduler[3] [13]. Globus Toolkit services provide a uniform, secure and reliable interface to heterogeneous computing platforms managed by different DRM systems. The main innovation of our model is the use of Globus Toolkit services to recursively interface to the services available in a federated Globus based grid.

A set of Globus Toolkit services hosting a GridWay Metascheduler, what we call a *GridGateWay*, provides the standard functionality required to implement a gateway to a federated grid. Such a combination allows the required virtualization technology to be created in order to provide a powerful abstraction of the underlying grid resource management services and meets the requirements imposed by a utility computing solution [3]. The GridGateWay acts as the utility computing service, providing a uniform standard interface based on Globus interfaces, protocols and services for the secure and reliable submission and control of jobs, including file staging, on grid resources.

Application developers, portal builders, and ISVs may interface with the utility computing service by using the Distributed Resource Management Application API (DRMAA)[4] [14]. DRMAA is an Open Grid Forum (OGF)[5] standard that constitutes a homogeneous interface to different DRM systems to handle job submission, monitoring and control, and retrieval of finished job status.

The grid hierarchy in our utility computing model is clear. An enterprise grid, managed by the IT Department, includes a GridGateWay to an outsourced grid, managed by the utility computing service provider. The outsourced grid provides pay-per-use computational power when local resources are overloaded. This hierarchical grid organization may be extended recursively to federate a higher number of partner or outsourced grid infrastructures with consumer/provider relationships. Figure 1 shows one of the many grid infrastructure hierarchies that can be deployed with GridGateWay components. This hierarchical solution, which resembles the architecture of the Internet (characterized by the end-to-end argument [15] and the IP hourglass model [16]), involves some performance overheads, mainly higher latencies, which will be quantified in Section 4.

The access to resources, including user authentication, across grid boundaries is under control of the GridGateWay service and is transparent to end users. In fact, different policies for job transfer and load balancing can be defined in

[3] http://www.gridway.org
[4] http://www.drmaa.org
[5] http://www.ogf.org

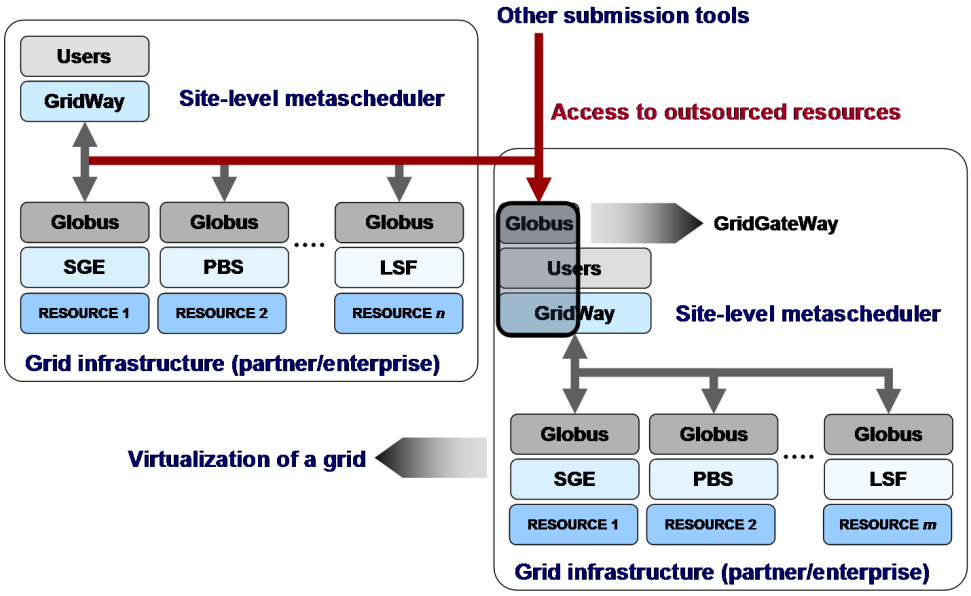


Fig. 1. Utility computing solution based on the Globus Toolkit and the GridWay Metascheduler

the GridGateWay. The user and resource accounting and management could be performed at different aggregation levels in each infrastructure.

On the other hand, the cultural and business model changes required for adopting the utility model should be gradual, starting with access to a local workload manager, followed by an in-house enterprise grid and finally moving onto outsourced services. These adoption steps are transparent to the end user in the proposed solution, since he always interfaces with a given GridGateWay using DRMAA standard.

3 Architecture of the GridGateWay

To interface GridWay through GRAM, a new scheduler adapter has been developed along with a scheduler event generator. Also, a scheduler information provider has been developed in order to feed MDS with scheduling information. Therefore, the main functionality of the GridGateWay is provided by the GridWay Metascheduler, although accessed through the standard interfaces provided by the Globus Toolkit. Moreover, the GridGateWay takes advantage of recent improvements in the GridWay Metascheduler, like multiple user support, accounting, client and resource fault-tolerance, and new drivers to access different grid services. It is expected that new functionality related to the GridGateWay will be added to GridWay, like new scheduling policies for enterprise, partner and utility grid infrastructures, security and certificate management policies, billing, etc.

The presented architecture has a number of advantages in terms of security and scalability. Regarding security, only the GridGateWay should be accessible from the Internet, and only Globus firewall requirements are imposed. Moreover, it is possible to restrict the dissemination of system configuration outside

site boundaries, and resource access and usage is controlled and accounted in the GridGateWay. Finally, different certificate mapping policies can be defined at each level, and the set of recognized Certificate Authorities (CA) can also be different. Regarding scalability, with this architecture the scheduling process is divided into different levels, where a different scheduling policy can be applied. Also, there is no need to disseminate everywhere all system monitoring information. For example, resource information can be aggregated and then published by means of the r_∞ and $n_{1/2}$ parameters, as proposed previously by the authors [17].

4 Experiments

4.1 Measurement of GridGateWay Overheads

In order to measure the overheads imposed by the GridGateWay architecture, we set up a simple infrastructure where a client runs an instance of the GridWay Metascheduler interfacing with a GridGateWay (running another instance of the GridWay Metascheduler accessed through Globus Toolkit services) which, in turns, interfaces with a Globus resource managed by the Fork job manager. The application used was a simple `/bin/echo` (using the executable in the remote resource), so the required computational time was negligible and the file transfer costs were minimum (basically the standard input/output/error streams).

The average Globus overhead in both the GridGateWay and the resource was 6 seconds. The average scheduling overhead in the client machine was 15 seconds, since the scheduling interval was set to 30 seconds, while in the GridGateWay it was only 5, since the scheduling interval was set to 10 seconds in order to improve the response time. Therefore, the difference in execution time between a direct execution and an execution through a GridGateWay can be as low as 11 seconds. This difference could be higher when more file transfers are involved.

4.2 Enterprise Grid Resource Provision from a Partner Grid Through a GridGateWay

Now, in order to evaluate the behavior of the proposed solution, we set up a more realistic infrastructure where a client run an instance of the GridWay Metascheduler interfacing local resources in an enterprise (UCM, in this case), based on GT4 Web Services interfaces, and a GridGateWay that gives access to resources from a partner grid (*fusion* VO of EGEE, in this case), based on GT pre-Web Services interfaces. The simultaneous use of different Middleware Access Drivers (MAD) to access multiple partner grid infrastructures has been demonstrated before [18,19]. In fact, that could be an alternative for the coexistence of different grid infrastructures, although based on distinct middleware (GT2, GT4, LCG, gLite...).

In this configuration, draco is the client machine, providing access to the enterprise grid, and cepheus is the GridGateWay, providing access to the partner grid (see Figure 2). Notice that, in this case, the GridGateWay is hosted in the

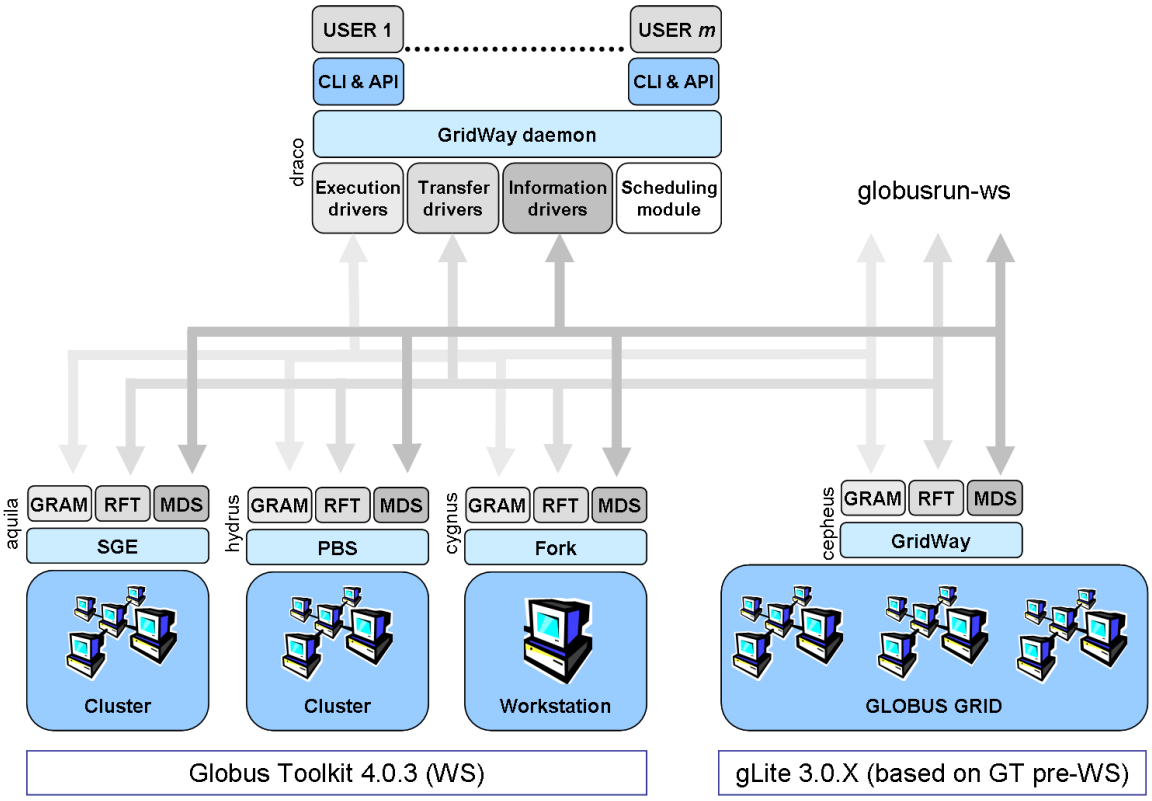


Fig. 2. Experimental environment

```
ehuedo@draco:~$ gwhost
HID OS              ARCH    MHZ %CPU  MEM(F/T)      DISK(F/T)    N(U/F/T) LRMS            HOSTNAME
0   Linux2.6.16-2-6 x86    3216    0  831/2027 114644/118812       0/0/1 Fork            cygnus.dacya.ucm.es
1   Linux2.6.16-2-a x86_6 2211   100  671/1003   76882/77844       0/2/2 SGE             aquila.dacya.ucm.es
2   Linux2.6.16-2-6 x86    3215    0  153/2027 114541/118812       0/0/1 Fork            draco.dacya.ucm.es
3   Linux2.6.16.13- x86    3200  200    10/512 148855/159263       0/0/2 SGE             ursa.dacya.ucm.es
4   Linux2.6.16-2-a x86_6 2211   100  674/1003   76877/77844       0/2/2 PBS             hydrus.dacya.ucm.es
5   NULLNULL        NULL      0    0       0/0            0/0 6/665/1355 GW            cepheus.dacya.ucm.es
```

Fig. 3. Resources in enterprise grid (UCM). Notice that cepheus, acting as a GridGateWay, appears as another resource with GW (GridWay) as LRMS (Local Resource Management System) and provides no information about its configuration, but only about free and total nodes.

enterprise. However, in a typical business situation it would be hosted in the partner infrastructure. The characteristics of resources from UCM are shown in Figure 3, as provided by GridWay command `gwhosts`, while those from EGEE (*fusion* VO) are shown in Figure 4.

Notice also that, with the GridGateWay architecture, it is possible to access resources with non WS (Web Services) interfaces, like those present in the LCG computing element of gLite 3, using WS interfaces, like those present in the new services provided by GT4, which are based on the Web Services Resource Framework (WSRF). Therefore, the interface the client sees is more homogeneous. Moreover, the access pattern of EGEE resources requires the client to start a GridFTP or GASS server. Nevertheless, GridWay doesn't need such server to access WS-based resources. With the proposed architecture, there is a GridFTP server already started in the GridGateWay, so there is no need to

```
ehuedo@cepheus:~$ gwhost
HID OS              ARCH   MHZ %CPU  MEM(F/T)  DISK(F/T)   N(U/F/T) LRMS               HOSTNAME
0   Scientific Linu i686  1001    0   513/513        0/0  3/103/224 jobmanager-lcgpbs lcg02.ciemat.es
1   ScientificSL3.0 i686   551    0   513/513        0/0     0/3/14 jobmanager-lcgpbs ce2.egee.cesga.es
2   Scientific Linu i686  1000    0 1536/1536        0/0     0/2/26 jobmanager-lcgpbs lcgce01.jinr.ru
3   Scientific Linu i686  2800    0 2048/2048        0/0     0/0/98 jobmanager-lcgpbs lcg06.sinp.msu.ru
4   Scientific Linu i686  1266    0 2048/2048        0/0    0/26/58 jobmanager-pbs    ce1.egee.fr.cgg.com
5   Scientific Linu i686  2800    0 2048/2048        0/0 0/135/206 jobmanager-lcgpbs node07.datagrid.cea.fr
6   ScientificSL3.0 i686  3000    0 2048/2048        0/0 0/223/352 jobmanager-lcgpbs fal-pygrid-18.lancs.ac
7   Scientific Linu i686  2400    0 1024/1024        0/0 0/139/262 jobmanager-lcgpbs heplnx201.pp.rl.ac.uk
8   Scientific Linu i686  3000    0 2048/2048        0/0     0/0/60 jobmanager-pbs    cluster.pnpi.nw.ru
9   Scientific Linu i686  1098    0 3000/3000        0/0     0/5/16 jobmanager-lcgpbs grid002.jet.efda.org
10  Scientific Linu i686  2800    0 1024/1024        0/0    0/30/32 jobmanager-lcgpbs ce.hep.ntua.gr
11  Scientific Linu i686  2000    0   492/492        0/0      0/2/7 jobmanager-lcgpbs ce.epcc.ed.ac.uk
```

Fig. 4. Resources in partner grid (*fusion* VO of EGEE). Notice that the access to these resources is configured in cepheus, acting as a GridGateWay.

open incoming ports in the client, and the firewall requirements results solely in allowing outgoing ports. This is very important when the access to grid resources is generalized and performed from ISV applications, using the DRMAA standard.

In the case of EGEE resources, and in order to not saturate the testbed (which is supposed to be at production level) with our tests, we limited the number of running jobs in the same resource to 10, and the number of running jobs belonging to the same user to 30.

First of all, we want to compare the direct access to EGEE resources and the access through a GridGateWay. In order to do that, in a first experiment we submitted the jobs directly to the GridWay instance running on cepheus. And, in a second experiment, we submitted the jobs to the GridWay instance running on draco with cepheus, acting as a GridGateWay, as the unique resource.

In this case, the application used was the distributed calculation of the π number as $\int_0^1 \frac{4}{1+x^2} dx$. Each task computes the integral in a separate section of the function and all results are finally added to obtain the famous 3.1415926435... Therefore, the required computational time is now higher (about 10 seconds on a 3.2 GHz Pentium 4 for each task) and file transfer costs include the executable and the standard input/output/error streams (about 10 KB per task).

Figure 5 shows the throughput achieved in EGEE (*fusion* VO) resources when accessed directly and when they are accessed through the GridGateWay. The number of tasks submitted was 100. As expected, there are differences in latency (response time) and throughput when directly accessing the resources and when using the GridGateWay. A throughput of 284.8 jobs/hour was achieved with direct access, versus 253.9 jobs/hour with the access through the GridGateWay. Therefore, the use of a GridGateWay supposes a performance loss of only 10.85%. Notice that this performance loss has been obtained with an application requiring only 10 seconds to execute. Since the overheads are independent on the computational time required by the application, they will suppose a smaller fraction of the total time when more demanding applications are used.

Figure 6 shows the throughput achieved in UCM when provisioning partner resources from EGEE (*fusion* VO) through a GridGateWay. The number of tasks submitted was 100 again. In this case, there are also differences in latency between in-house and partner resources. Besides network connection and the use

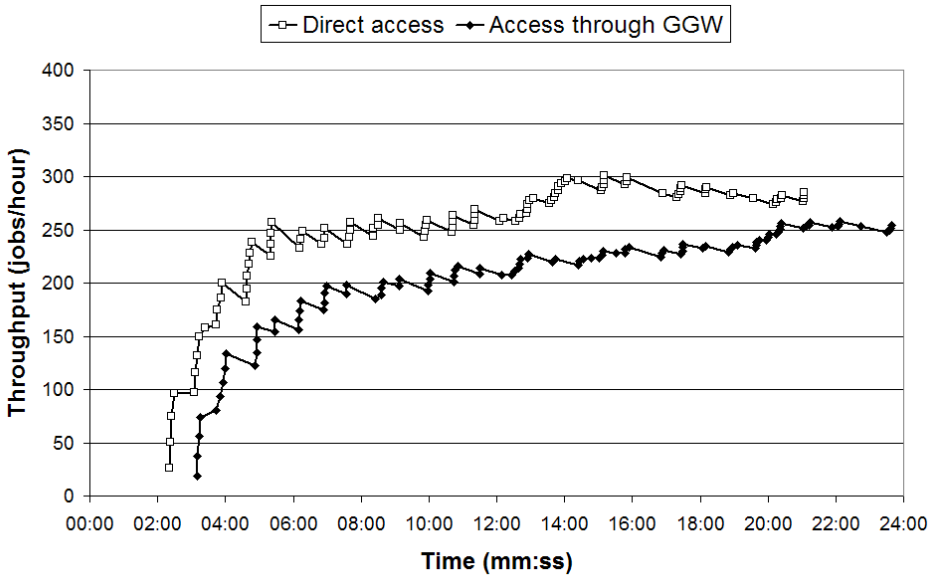


Fig. 5. Throughput achieved in EGEE (*fusion* VO) when accessed directly and when accessed through a GridGateWay

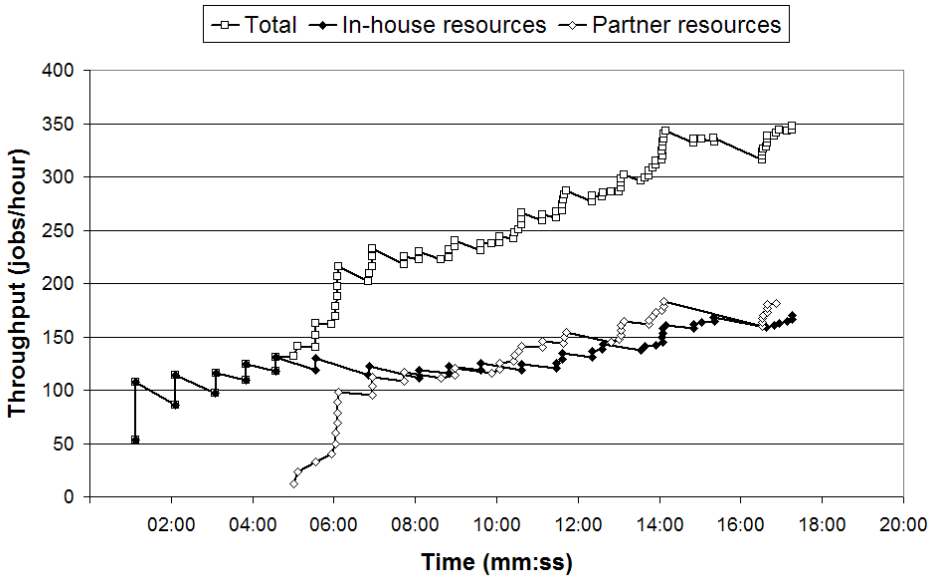


Fig. 6. Throughput achieved in UCM when provisioning resources from EGEE (*fusion* VO) through a GridGateWay

of a GridGateWay, it is also due to the production status of partner resources, as they are under heavy usage. The aggregated throughput achieved was 347.5 jobs/hour. In-house and partner resources contributed almost equally (170.3 and 181.4 jobs/hour, respectively) to the aggregated throughput.

5 Conclusions and Future Work

We have presented and evaluated a solution that meets the requirements for utility computing, namely: (i) a uniform, secure and reliable functionality to

access the utility; (ii) a scalable, flexible and adaptive infrastructure to provide the service; and (iii) a solution based on standards and gradually deployable. Moreover, the presented performance results are promising.

This innovative utility solution for computing provision, which can be deployed on a grid infrastructure based on existing Globus Toolkit components and related tools, will allow companies and research centers to access their in-house, partner and outsourced computing resources via automated methods using grid standards in a simpler, more flexible and adaptive way. Moreover, the proposed solution has many advantages in terms of security, scalability and site autonomy. Initial results show that the performance loss is low (about 10% for very short tasks), and it would be even lower with applications requiring more computational time.

Future work will include the fine tuning of components to reduce latency and increase throughput, as well as the development of scheduling policies considering these factors, latency and throughput, in order to reduce the total execution time of a whole workload, and also taking into account resource ownership to reduce the associated cost (in terms of real money, resource usage or partner satisfaction). New components for negotiation, service level agreement, credential management, and billing are currently being developed in the context of the Grid4Utility project[6].

Acknowledgments

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org.

We want to acknowledge all institutions belonging to the *fusion* VO[7] of the EGEE project for giving us access to their resources.

References

1. Murch, R.: Introduction to Utility Computing: How It Can Improve TCO. Prentice-Hall, Englewood Cliffs (2005)
2. Carr, N.G.: The End of Corporate Computing. MIT Sloan Management Review 46(3), 67–73 (2005)
3. Llorente, I.M., Montero, R.S., Huedo, E., Leal, K.: A Grid Infrastructure for Utility Computing. In: Proc. 15th IEEE Intl. Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), pp. 163–168. IEEE Computer Society Press, Los Alamitos (2006)

[6] http://www.grid4utility.org
[7] http://grid.bifi.unizar.es/egee/fusion-vo

4. Agarwal, A., Ahmed, M., Berman, A., et al.: GridX1: A Canadian Computational Grid. Future Generation Computer Systems 23(5), 680–687 (2007)
5. Baumbauer, C., Goasguen, S., Martin, S.: Bouncer: A Globus Job Forwarder. In: Proc. 1st TeraGrid Conf (TeraGrid 2006) (2006)
6. Breuer, D., Wieder, P., van den Berghe, S., von Laszewski, G., MacLaren, J., Nicole, D., Hoppe, H.C.: A UNICORE-Globus Interoperability Layer. Computing and Informatics 21, 399–411 (2002)
7. Mallmann, D., Riedel, M., Twedell, B., Streit, A.: Interoperability Plan for UNICORE. Technical Report MSA3.3, EGEE-II (2006)
8. Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing 5(3), 237–246 (2002)
9. Buyya, R., Venugopal, S.: The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. In: Proc. 1st IEEE Intl. Workshop on Grid Economics and Business Models (GECON 2004), pp. 19–36. IEEE Computer Society Press, Los Alamitos (2004)
10. Surridge, M., Taylor, S., Roure, D.D., Zaluska, E.: Experiences with GRIA Industrial Applications on a Web Services Grid. In: Proc. 1st Intl. Conf. e-Science and Grid Computing (e-Science05), pp. 98–105. IEEE Computer Society Press, Los Alamitos (2005)
11. Foster, I., Tuecke, S.: Describing the Elephant: The Different Faces of IT as Service. Enterprise Distributed Computing 3(6), 26–34 (2005)
12. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005)
13. Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Software - Practice and Experience 34(7), 631–651 (2004)
14. Herrera, J., Huedo, E., Montero, R.S., Llorente, I.M.: GridWay DRMAA 1.0 Implementation – Experience Report. Document GFD.E-104, DRMAA Working Group – Open Grid Forum (2007)
15. Carpenter, E.B.: Architectural Principles of the Internet. RFC 1958 (June 1996)
16. Deering, S.: Watching the Waist of the Protocol Hourglass. In: 6th IEEE International Conference on Network Protocols (1998)
17. Montero, R.S., Huedo, E., Llorente, I.M.: Benchmarking of High Throughput Computing Applications on Grids. Parallel Computing 32(4), 267–279 (2006)
18. Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. J. Parallel and Distributed Computing 66(5), 763–771 (2006)
19. Huedo, E., Montero, R.S., Llorente, I.M.: A Modular Meta-Scheduling Architecture for Interfacing with Pre-WS and WS Grid Resource Management Services. Future Generation Computer Systems 23(2), 252–261 (2007)

A.2. Transparent Access to Grid-Based Compute Utilities

Cita completa

C. Vázquez, J. Fontán, E. Huedo, R.S. Montero, I.M. Llorente. *Transparent Access to Grid-Based Compute Utilities*. In Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics (PPAM 2007). **Lecture Notes in Computer Science**, 4967, 817-824, 2008.

Indicios de calidad

Este congreso está en la categoría CORE C según el ERA Conference Ranking. En la edición de 2007 recibió 230 envíos de los que se aceptaron 141, lo cual supone un ratio de aceptación de aproximadamente 55 %. Ha sido citado 1 vez según Google Scholar.

Transparent Access to Grid-Based Compute Utilities*

Constantino Vázquez, Javier Fontán, Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid, Spain
{tinova, jfontan, ehuedo}@fdi.ucm.es, {rubensm, llorente}@dacya.ucm.es

Abstract. There is normally the need for different grid architectures depending on the constrains: from enterprise infrastructures, to partner or outsourced ones. We present the technology that enables the creation and combination of different grid architectures, offering the possibility to create federated grids that can be easily deployed as compute utilities on grid infrastructures based on the Globus Toolkit. Then, taking advantage of the uniform and standard interface of the proposed grid-based compute utilities, we present a way to enable transparent access to virtualized grids from local computing infrastructure managed by Sun Grid Engine, although similar approaches could be followed for other resource managers.

1 Introduction

Grid computing is concerned with "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations", as stated by Ian Foster in [1]. Therefore, due to its multi-institutional nature, a grid is built on top of potentially heterogeneous resources.

Furthermore, due to socio-political landscape, there is normally the need for different grid architectures depending on the constraints. We can differentiate three kinds of arrangements. There are, firstly, enterprise grids, contained within the boundaries of one organization. They improve internal collaboration and enables a better return from IT investment, coming from a more optimal resource utilization. Secondly, we have partner grids, created using the resources of two or more organizations and /or scientific communities. Partner grids allow the access to higher computing performance to satisfy peak demands and also provide support to face collaborative projects. As a third architecture we find

* This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, by Ministerio de Educación y Ciencia, through research grant TIN2006-02806, and by European Union, through BEinGRID project EU contract IST-2005-034702.

R. Wyrzykowski et al. (Eds.): PPAM 2007, LNCS 4967, pp. 817–824, 2008.

outsourced grids, although not a reality right now, it has been predicted that dedicated service providers will organize and maintain cyberinfrastructures to supply resources on demands over the Internet. This business model will be feasible when growing network capacity is enough to allow business and consumers to draw their computing resources from outsourced grids apart from enterprise and partner grids [2].

These different grid perspectives need the technology to ensure their deployment in a way that preserves the autonomy of each organization, while offering at the same time overall security (critical if organizations are planning to outsource sensitive data) and scalability (an inherent requirement of grid infrastructures).

In this paper we present the technology that enables the creation and combination of different grid architectures, i.e. we offer the possibility to create federated grids, that can be easily deployed as compute utilities on grid infrastructures based on the Globus Toolkit (GT). Furthermore, taking advantage of the uniform and standard interface of these grid-based compute utilities, we present a way to enable transparent access to virtualized grids from local computing infrastructure managed by Sun Grid Engine, although similar approaches could be followed for other resource managers.

The rest of this paper is organized with the following structure. Section 2 introduces and explains the "grid for utility" concept. Section 3 presents the GridGateWay solution for building federated grid infrastructures based on the Globus Toolkit and the GridWay Metascheduler. It also describes a testbed for a federated grid and provides some results. In Section 4, we present a complementary component to enable access to virtualized grids using the Sun Grid Engine *Transfer Queue to Globus*. To finish, Section 5 presents some conclusions and our plans for future work.

2 Grid for Utility

Originally, the concept of grid was developed from the power grid. In short, the idea behind a grid is to be able to plug a computing resource so it gets added to the total computer power with the minimum possible configuration. It also will be able to use the grid computing capability as a single computer, transparently to the user, the same way an electric device can be plugged to a power socket to draw electricity from it.

Utility computing is a service provisioning model which will provide adaptive, flexible and simple access to computing resources, enabling a pay-per-use model for computing similar to traditional utilities such as water, gas or electricity. It pursues an scenario where computing power is given or taken (consumer/producer paradigm) from the different components that conform to the grid based on the demand and offer of this computing power at a given time. This posses different problems, ranging from accounting and billing to scalability and security.

One desired outcome of the "grid for utility" idea is to allow the creation of the outsourced grids identified above. It will indeed provide means to *sell*

computing power to anyone interested on it. Obviously, security is essential, as sensitive data may be outsourced. Several efforts have been undergone to tackle this problem [3,4].

3 The GridGateWay Solution for Grid Federation

In order to leverage the practical implementation of the concepts of the previous section we provide means to build grid federations, although admittedly only in a somewhat still crude form.

A GridGateWay provides the standard functionality required to implement a gateway to a federated grid. The main innovation of our model is the use of Globus Toolkit services to recursively interface to the services available in a federated Globus-based grid. Such a combination allows the required virtualization technology to be created in order to provide a powerful abstraction of the underlying grid resource management services. The GridGateWay acts as a utility computing service, providing a uniform standard interface based on Globus protocols and services for the secure and reliable submission and control of jobs, including file staging, on grid resources.

A GridGateWay offers the possibility of encapsulating a virtualized grid inside a WS (Web Service) GRAM (Globus Resource Allocation and Management), using GridWay as the underlying LRMS (Local Resource Management System). To interface GridWay through GRAM, a new scheduler adapter has been developed along with a scheduler event generator. Also, a scheduler information provider has been developed in order to feed MDS (Monitoring and Discovery Service) with scheduling information.

The grid hierarchy in our federation model is clear. An enterprise grid, managed by the IT Department, includes a GridGateWay to an outsourced grid, managed by the service provider. The outsourced grid provides on-demand or pay-per-use computational power when local resources are overloaded. This hierarchical grid organization may be extended recursively to federate a higher number of partner or outsourced grid infrastructures with consumer/provider relationships. Figure 1 shows one of the many grid infrastructure hierarchies that can be deployed with GridGateWay components. This hierarchical solution, which resembles the architecture of the Internet (characterized by the end-to-end argument [5] and the IP hourglass model [6]), involves some performance overheads, mainly higher latencies, which have been quantified elsewhere [7]. The access to resources, including user authentication, across grid boundaries is under control of the GridGateWay service and is transparent to end users. In fact, different policies for job transfer and load balancing can be defined in the GridGateWay. The user and resource accounting and management could be performed at different aggregation levels in each infrastructure.

There are several characteristics of this architecture that we identify as advantages. First one, it is completely based on standard protocols and uses a *de facto* standard grid middleware as Globus Toolkit is. Secondly, it doesn't require for extra deployment for clients as it uses a GRAM interface. And, as a third

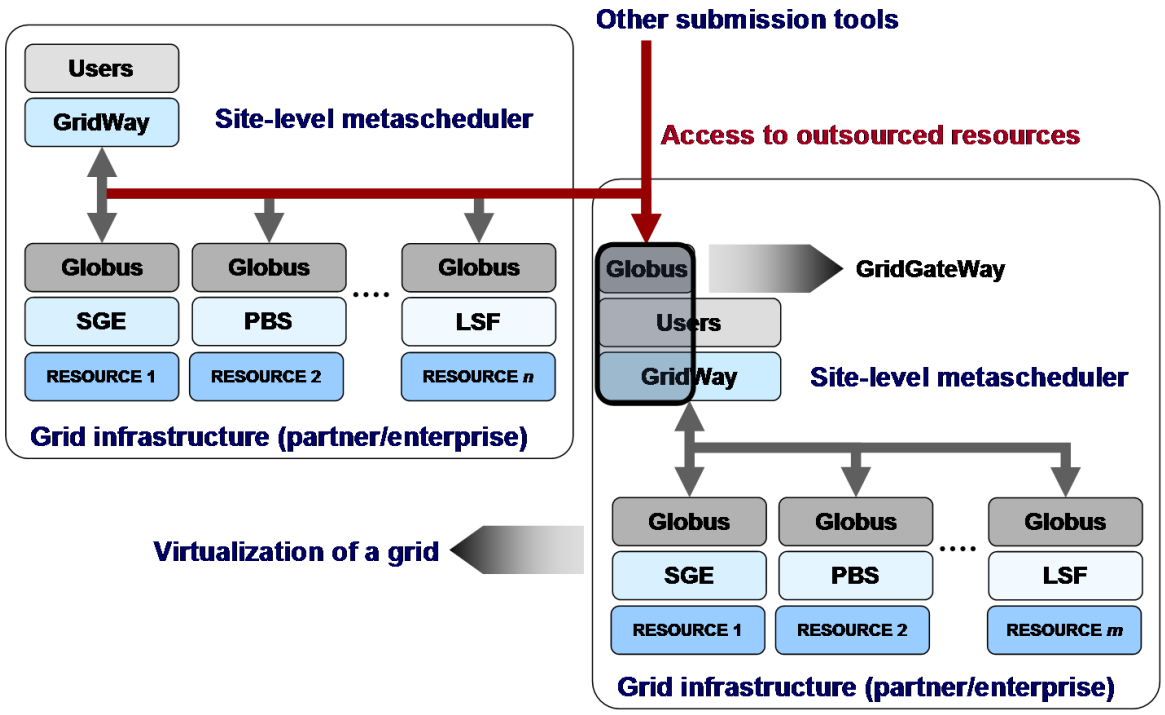


Fig. 1. GridWay encapsulated under a Globus WS GRAM service

advantage, also being its main objective, it allows for a complex grid federation, based on recursive architecture.

In order to test the GridGateWay we used the following testbed. On one server (cepheus) we have an instance of GridGateWay that virtualizes a partner grid, in this case is going to be the *fusion* Virtual Organization (VO) of EGEE. Also, we have a local cluster managed by GridWay, that also has access to the EGEE partner grid. As it can be seen in Figure 2, cepheus appears as having a GW (GridWay) LRMS and it just displays the number of total and free nodes, and no more configuration details, it is in fact virtualizing the partner grid. The users doesn't know the details of the grid behind cepheus Then again, he really doesn't need to know, but if needed a host information provider could be developed.

Figure 3 shows the throughput achieved with this configuration. We used a small parameter sweep application composed of 100 tasks, each one needing about 10 seconds to execute on a 3.2 GHz Pentium 4 and about 10 KB of

```
tinova@draco:~$ gwhost
HID OS              ARCH   MHZ %CPU MEM(F/T)     DISK(F/T)   N(U/F/T) LRMS HOSTNAME
0   Linux2.6.16-2-6 x86    3216    0 831/2027 114644/118812      0/0/1 Fork cygnus.dacya.ucm.e
1   Linux2.6.16-2-a x86_6 2211  100 671/1003   76882/77844      0/2/2 SGE  aquila.dacya.ucm.e
2   Linux2.6.16-2-6 x86    3215    0 153/2027 114541/118812      0/0/1 Fork draco.dacya.ucm.es
3   Linux2.6.16-2-a x86_6 2211  100 674/1003   76877/77844      0/2/2 PBS  hydrus.dacya.ucm.e
4   NULLNULL        NULL      0    0      0/0           0/0 6/665/1355 GW   cepheus.dacya.ucm.
```

Fig. 2. Resources in UCM (enterprise grid) as provided by GridWay `gwhost` command. cepheus hosts a GridGateWay and appears as another resource with GW (GridWay) as LRMS (Local Resource Management System), only providing information about its status (used, free and total nodes).

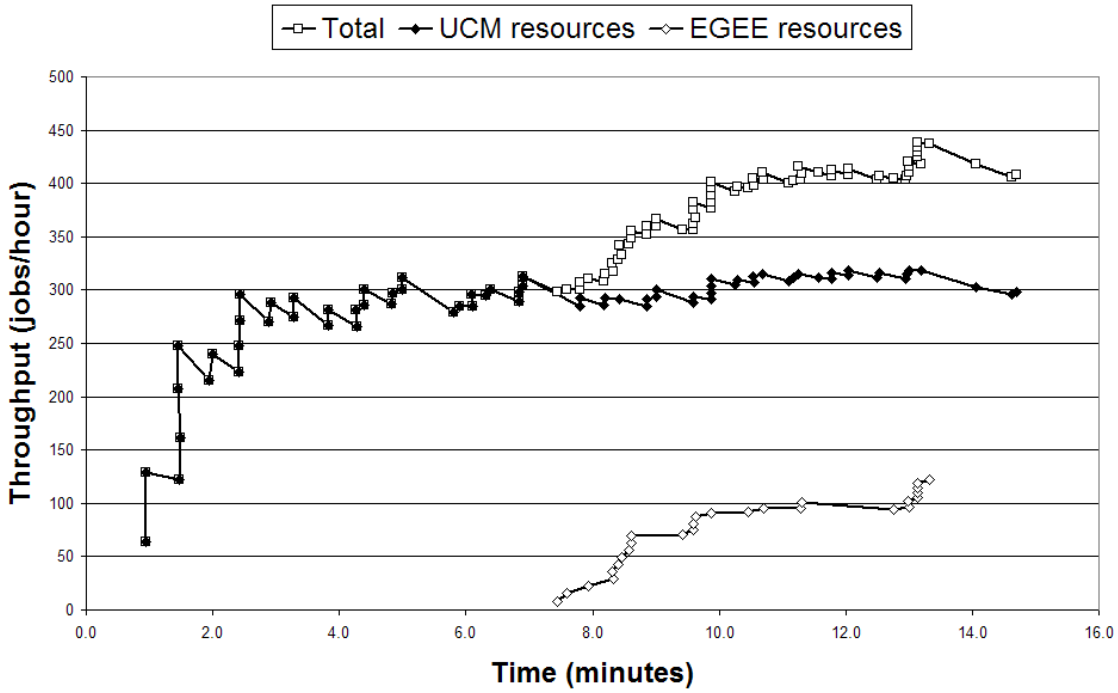


Fig. 3. Throughput achieved in UCM when provisioning resources from EGEE (*fusion* VO) through a GridGateWay

input/output data. UCM resources executed 73 jobs, thus EGEE resources executed only 27 jobs, because their response time was much higher than that of UCM resources (the first job executed on EGEE completes 7.45 minutes after the experiment starts). This was in part due to network latencies and the use of the GridGateWay, but mainly because EGEE resources were heavily loaded with production jobs and, therefore, the submitted jobs suffered from high queue wait times. UCM and EGEE respectively contributed with 297.96 and 121.65 jobs/hour to the aggregated throughput of 408.16 jobs/hour. Results with this configuration has been proved to give a close to optimal performance, according to predictions made using a performance model for federated grid infrastructures [8].

4 Sun Grid Engine *Transfer Queue to Globus*

An interesting application of the GridGateWay solution is the outsourcing of computing power to sustain peak demands. This can be done using a pure GridWay based solution, where a local GridWay instance manages the local resources and uses a GridGateWay to satisfy computing power demand when the local cluster/grid is overloaded.

We are aware that in the real, non academic, world (and even there) there is a huge inertial component that makes difficult to assume changes. With that in mind, we are committed to provide solutions for a wide range of the current possible configurations, so as to allow our concept of grid federation without any major change in the available infrastructure (i.e. both the grid middleware or the

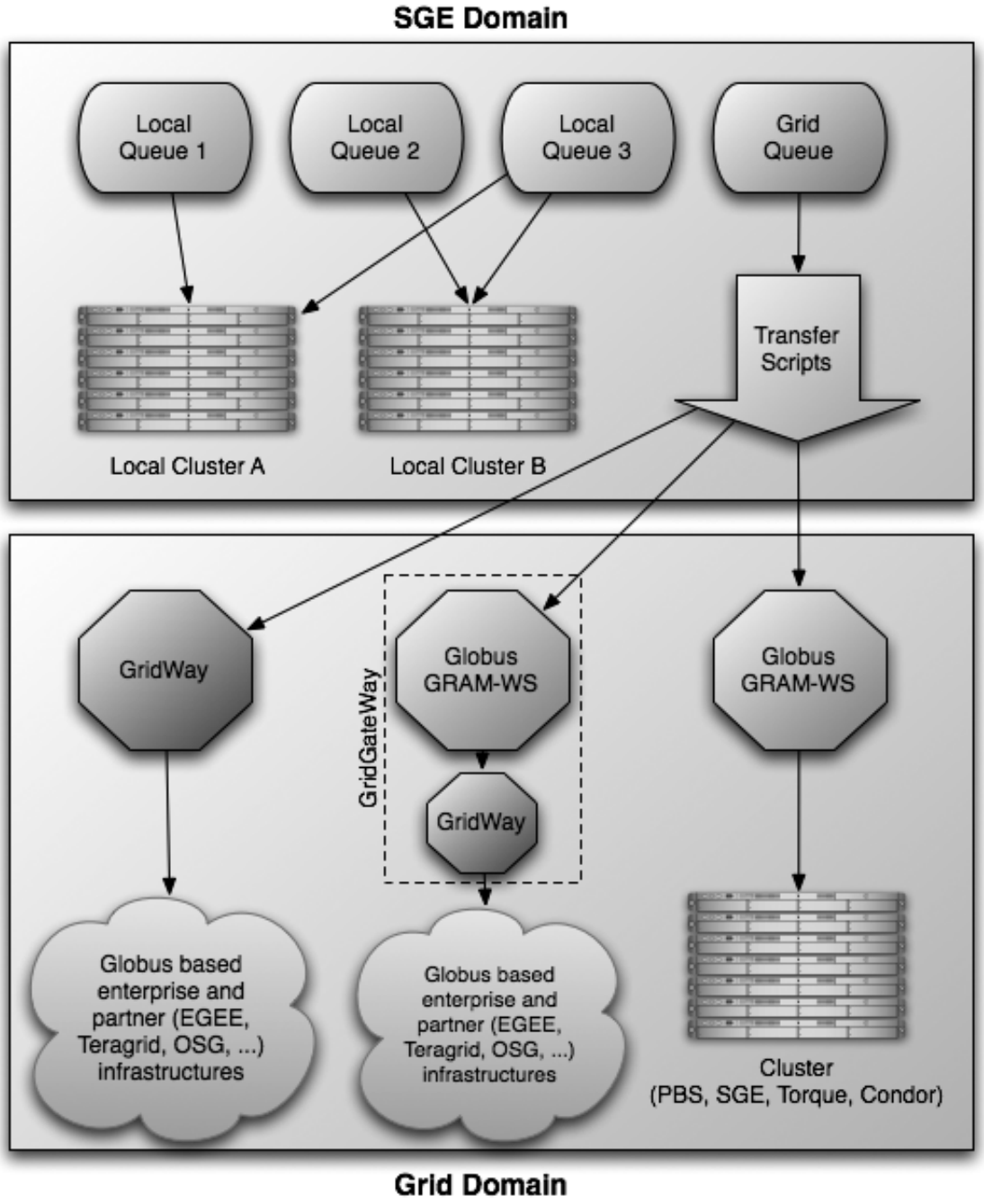


Fig. 4. Grid federation achieved through GridGateWay and accessible through a local LRMS, in this case, SGE

underlying LRMS). GridWay already offers the technology to construct a grid out of heterogeneous sub-grids, based on different flavors of Globus (pre-WS and WS) [9,10]. On top of that, we now offer the possibility of accessing a federated grid not only from GridWay, but also from a LRMS like Sun Grid Engine (SGE).

Sun Grid Engine[1] is an open source batch-queuing system, supported by Sun Microsystems[2]. This LRMS offers the possibility to define custom queues,

[1] http://gridengine.sunsource.net
[2] http://www.sun.com

through the development of scripts to submit, kill, pause and resume a job. It is already available from the vendor a *Transfer Queue to Globus* [11], but just for GT2, with pre-WS interfaces. As stated above, GridGateWay uses WS GRAM interface, so a Transfer Queue to GT4 has been developed[3]. What we obtain from this component is the ability to submit jobs to a virtualized grid (utility computing service) from a local, well known, LRMS in a transparent way for the user.

Figure 4 shows SGE with one transfer queue configured. In the SGE domain we find the normal SGE local queues and a special grid queue, that enables SGE to submit jobs to a WS GRAM service or to GridWay. This WS GRAM service can, in turns, virtualize a cluster (with several known LRMS) or it can encapsulate a GridWay (thus conforming a GridGateWay) that gives access to a Globus-based outsourced infrastructure. The special grid queue can be configured to be available only under certain special conditions, like peak demands or resource outage.

5 Conclusions and Future Work

In this paper we have shown a utility computing solution that can be deployed on any grid infrastructure based on existing Globus Toolkit components. This solution enables the creation of enterprise, partner and outsourced grids using automated methods on top of *de facto* grid standards in a simple, flexible and adaptive way. Moreover, we have developed components that intend to make the deployment as simple as possible, ideally requiring the minimum changes to the existing local infrastructure. As of now, we just support one LRMS, but is in the scope of the Grid4Utility project[4] to provide a wider range of components to handle the broader set of pre-existing configurations as possible. Other aims of the project are the development of new components for scheduling, negotiation, service level agreement, credential management and billing.

Acknowledgments

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org.

We want to acknowledge all institutions belonging to the *fusion* VO[5] of the EGEE project for giving us access to their resources.

[3] Available at http://www.grid4utility.org/#SGE
[4] http://www.grid4utility.org
[5] http://grid.bifi.unizar.es/egee/fusion-vo

References

1. Foster, I.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Intl. J. High Performance Computing Applications 15(3), 200–222 (2001)
2. Llorente, I.M., Montero, R.S., Huedo, E., Leal, K.: A Grid Infrastructure for Utility Computing. In: Proc. 15th IEEE Intl. Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), IEEE CS Press, Los Alamitos (2006)
3. Humphrey, M., Thompson, M.: Security Implications of Typical Grid Computing Usage Scenarios. Cluster Computing 5(3), 257–264 (2002)
4. Azzedin, F., Maheswaran, M.: Towards Trust-Aware Resource Management in Grid Computing Systems. In: 2nd IEEE/ACM International Symp. Cluster Computing and the Grid (CCGrid 2002), pp. 452–457 (2002)
5. Carpenter, E.B.: Architectural Principles of the Internet. RFC 1958 (June 1996)
6. Deering, S.: Watching the Waist of the Protocol Hourglass. In: 6th IEEE International Conference on Network Protocols (1998)
7. Vázquez, T., Huedo, E., Montero, R.S., Llorente, I.M.: Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641, Springer, Heidelberg (2007)
8. Vázquez, T., Huedo, E., Montero, R.S., Llorente, I.M.: A Performance Model for Federated Grid Infrastructures. In: The 16th Euromicro International Conference on Parallel, Distributed and network-based Processing (PDP 2008) (in press) (Submitted)
9. Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. J. Parallel and Distributed Computing 66(5), 763–771 (2006)
10. Huedo, E., Montero, R.S., Llorente, I.M.: A Modular Meta-Scheduling Architecture for Interfacing with Pre-WS and WS Grid Resource Management Services. Future Generation Computer Systems 23(2), 252–261 (2007)
11. Seed, T.: Transfer-queue Over Globus (TOG): How To. Technical Report EPCC-SUNGRID-TOG-HOW-TO 1.0, EPCC, The University of Edinburgh (July 2003), Available at: `http://gridengine.sunsource.net/download/TOG/tog-howto.pdf`

A.3. A Performance Model for Federated Grid Infrastructures

Cita completa

C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente. *A Performance Model for Federated Grid Infrastructures*. In Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008). **IEEE Computer Society**, 188-192, 2008.

Indicios de calidad

Este congreso está en la categoría CORE C según el ERA Conference Ranking. En la edición de 2008 recibió 140 envíos de los que se aceptaron 56, lo cual supone un ratio de aceptación de aproximadamente 40 %. Ha sido citado 6 veces según Google Scholar.

16th Euromicro Conference on Parallel, Distributed and Network-Based Processing

A Performance Model for Federated Grid Infrastructures *

Constantino Vázquez Eduardo Huedo Rubén S. Montero
Ignacio M. Llorente
Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid, Spain

E-mail: {tinova, ehuedo}@fdi.ucm.es, {rubensm, llorente}@dacya.ucm.es

Abstract

A performance model, previously proposed to characterize the performance of grid infrastructures, is extended to evaluate federations of grids by aggregating their performance parameters. These parameters can then be used to develop scheduling policies based on them. The new model can be used to take scheduling decisions based on them and hence to aid in the development of scheduling policies. The model has been validated using the performance results obtained in the execution of a high throughput computing application on an enterprise grid composed of Globus Toolkit Web Service resources and a GridGateWay giving access to gLite resources from the EGEE infrastructure.

1. Introduction

The deployment of *enterprise grids* enables diverse resource sharing to improve internal collaboration and achieve a better return from IT investment. On the other hand, *partner grids* of several scales are being mainly deployed within the context of different research projects, whose final goal is to provide large-scale, secure and reliable sharing of resources among partner organizations and supply-chain participants. Such partner grids allow access to a higher computing performance to satisfy peak demands and also provide support to face collaborative projects.

Different studies suggest that growing network capacity will allow businesses and consumers to draw their computing resources from outsourced grids apart from enterprise and partner grids. Therefore, future grid infrastructures will be composed of several enterprise, partner and outsourced grids, showing a hierarchical architecture, which preserves the autonomy of each organization and improves both security and scalability.

In a previous work [20], we have proposed a solution for federating grids that can be deployed on a grid infrastructure based on the Globus Toolkit (GT). A similar approach for the federation of grid infrastructures has been previously applied to meet LCG and GridX1 infrastructures [3], hosting a GridX1 user interface in a LCG computing element. However, this solution imposes software, middleware and network requirements on worker nodes. The Globus project is also interested in this kind of *recursive* architectures, and is working on Bouncer [4], which is a Globus job forwarder initially conceived for federating TeraGrid and Open Science Grid infrastructures. There are other approaches to achieve middleware interoperability, for example between UNICORE and Globus [5] and between gLite and UNICORE.

A lot of work has been done to benchmark, model, predict or even control the performance of grid infrastructures. Previous works consider the characteristics of grid infrastructures and applications, like dynamism, heterogeneity or adaptation, but few of them deal with the aggregation of performance models (or their parameters) to model the performance of federated grid infrastructures [12, 13].

In this work we extend a performance model, previously proposed, to characterize federated grid infrastructures. Then we apply it to a federation of grid based on GridGateWays technology.

The rest of this paper is as follows. Section 2 describes the performance model ans its extensions to deal with federated grids. Section 3 presents a solution for building federated grid infrastructures based on the Globus Toolkit and the GridWay Metascheduler by means of GridGateWays. Section 4 presents some experimental results to validate the performance model and, finally, Section 5 provides some conclusions and plans for future work.

*This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, by Ministerio de Educación y Ciencia, through research grant TIN2006-02806, and by European Union, through BEinGRID project EU contract IST-2005-034702.

0-7695-3089-3/08 $25.00 © 2008 IEEE
DOI 10.1109/PDP.2008.42

2. Performance Model

In the execution of High-Throughput Computing applications, a grid can be considered, from the computational point of view, as an array of *heterogeneous* processors [8]. Therefore, the number of tasks completed as function of time is given by the following equation:

$$n(t) = \sum_{i \in G} N_i \left\lfloor \frac{t}{T_i} \right\rfloor \quad (1)$$

where N_i is the number of processors of resource *i* in a grid *G* that can compute a task in T_i seconds.

The best characterization of a grid can be obtained if we take the average behavior of the system. The next formula represents *n(t)* using the r_∞ and $n_{1/2}$ parameters defined by Hockney and Jesshope [16]:

$$n(t) = r_\infty t - n_{1/2} \quad (2)$$

These parameters are called:

- Asymptotic performance (r_∞): the maximum rate of performance in tasks executed per second. In the case of an homogeneous array of N processors with an execution time per task T, we have $r_\infty = N/T$.
- Half-performance length ($n_{1/2}$): the number of tasks required to obtain the half of asymptotic performance. This parameter is also a measure of the amount of parallelism in the system as seen by the application. In the homogeneous case we obtain $n_{1/2} = N/2$, so $2 \cdot n_{1/2}$ represents the apparent number of –homogeneous– processors (thus, equal to N in the homogeneous case).

These parameters can be obtained through intrusive or non-intrusive benchmarking, as proposed in [8].

The following equation defines the performance of the system (tasks completed per time) on actual applications with a finite number of tasks based on the linear relation of Eq. 2:

$$r(n) = n(t)/t = \frac{r_\infty}{1 + n_{1/2}/n} \quad (3)$$

The simplicity of this model allows the characterization of grid infrastructures by using a very reduced set of metrics (just the r_∞ and $n_{1/2}$ parameters), while fully capturing their behavior. Moreover, it is straightforward to obtain these parameters in the case of federated grid infrastructures by just adding the parameters of the grid infrastructures building up the federation. Let assume that FG is the set of federated grids, each one characterized by its linear performance model (i.e. r^i_∞ and $n^i_{1/2}, \forall i \in FG$):

$$r_\infty = \sum_{i \in FG} r^i_\infty$$
$$n_{1/2} = \sum_{i \in FG} n^i_{1/2} \quad (4)$$

The suitability of this approach will be evaluated in section 4 with the GridGateWay architecture. It is worth mentioning that this model can be applied to the federation architecture.

As another application of this model, it is possible to obtain the optimum number of jobs that should be submitted to each infrastructure in order to achieve the minimum computational time. We will formulate the general problem as follows. There are J jobs to be processed, each job can be processed in any grid of the federation FG. We assume that each grid can process j_i jobs simultaneously. We are interested in optimize the maximum completion time criterion (*makespan*), C_{max}. The time needed for the grid i to process the jobs assigned to it, can be easily derived from Eq. 2, therefore the makespan yields:

$$C_{max} = \max_{i \in FG} \frac{j_i + n^i_{1/2}}{r^i_\infty} \quad (5)$$

This problem can be formulated as an integer linear programming problem as follows:

$$\begin{aligned} &\textbf{minimize} \\ &\quad C_{max} \\ &\textbf{subject to} \\ &\quad \sum_{i \in FG} j_i - J = 0; \\ &\quad j_i \geq 0 \ \ \forall i \in FG \end{aligned} \quad (6)$$

We will apply this optimization problem in the experiments section.

3. Federation of Grid Infrastructures through GridGateWays

The Globus Toolkit [18] provides a uniform, secure and reliable interface to many different DRM (Distributed Resource Manager) systems, allowing the development of grid workload managers that optimize the use of the underlying computing platforms.

Our solution uses Globus Toolkit services to recursively interface to the services available in a federated Globus based grid. A set of Globus Toolkit services hosting a GridWay Metascheduler[1] [19], what we call a *GridGateWay*, provides the standard functionality required to implement a gateway to a federated grid. Such a combination allows the required virtualization technology to be created in order to provide a powerful abstraction of the underlying grid resource management services. The GridGateWay acts as a computing service, providing a uniform standard interface based on Globus interfaces, protocols and services for the

[1] http://www.gridway.org

secure and reliable submission and control of jobs, including file staging, on grid resources.

The grid hierarchy in our federation model is clear. An enterprise grid, managed by the IT Department, includes a GridGateWay to an outsourced grid, managed by the service provider. The outsourced grid provides on-demand or pay-per-use computational power when local resources are overloaded. This hierarchical grid organization may be extended recursively to federate a higher number of partner or outsourced grid infrastructures with consumer/provider relationships. This solution involves some performance overheads, mainly higher latencies, which have been quantified before [20].

The access to resources, including user authentication, across grid boundaries is under control of the GridGateWay service and is transparent to end users. In fact, different policies for job transfer and load balancing can be defined in the GridGateWay. The user and resource accounting and management could be performed at different aggregation levels in each infrastructure.

4. Experiments

We set up a federated grid infrastructure where a client runs an instance of the GridWay Metascheduler interfacing local resources in an enterprise grid (UCM, in this case), based on GT4 Web Services interfaces, and a GridGateWay that gives access to resources from a partner grid (*fusion* VO of EGEE, in this case), based on GT pre-Web Services interfaces found on gLite 3.0. The simultaneous use of different adapters to access multiple partner grid infrastructures has been demonstrated before [22]. In fact, that could be an alternative for the coexistence of different grid infrastructures, although based on distinct middleware (GT2, GT4, LCG, gLite...).

In this configuration, draco is the client machine, providing access to the enterprise grid, and cepheus is the GridGateWay, providing access to the partner grid (see Figure 1 (top)). Notice that, in this case, the GridGateWay is hosted in the enterprise. However, in a typical business situation it would be hosted in the partner infrastructure, enforcing its security and usage policies and providing services for accounting and billing.

In the case of EGEE resources, and in order to avoid saturation of the testbed (which is supposed to be at production level) with our tests, we limited the number of running jobs in the same resource to 10, and the number of running jobs belonging to the same user to 30. This kind of usage policies can be defined by the service provider in the instance of GridWay running in the GridGateWay.

The application used was the distributed calculation of the π number as $\int_0^1 \frac{4}{1+x^2} dx$. Each task computes the integral in a separate section of the function and all results are finally added to obtain a good approximation of π. The required computational time is about 10 seconds per task on a 3.20GHz Pentium 4; and file transfer cost, including the executable and the standard input/output/error streams, is about 10KB per task. Is worth noting here that the fact that the π number calculation is a toy program doesn't affect it's execution and transfer times, so it doesn't affect the validity of the model.

Table 1. Parameters of the performance model for each experiment.

	Experiment 1		Experiment 2	
	r_∞	$n_{1/2}$	r_∞	$n_{1/2}$
UCM	171.75	5.01	322.16	2.93
fusionVO	236.83	17.24	224.61	24.65
Experimental	395.63	20.27	458.77	12.32
Estimated	408.58	22.25	546.77	27.57

Figure 2 (bottom) shows the dynamic throughput achieved in UCM when provisioning partner resources from EGEE (*fusion* VO) through a GridGateWay. The number of tasks submitted was 100. Besides network connection and the use of a GridGateWay, there are differences in latency between enterprise and partner resources due to the production status of partner resources, as they are under heavy usage. The aggregated throughput achieved in the first experiment was 347.5 jobs/hour. Enterprise and partner resources executed almost the same number of jobs (49 and 51 jobs, respectively) and contributed almost equally (170.3 and 181.4 jobs/hour, respectively) to the aggregated throughput. In the second experiment, the aggregated throughput achieved was higher, 408.16 jobs/hour, due to a better response of enterprise resources, which executed 73 jobs contributing with 297.96 jobs/hour to the total throughput. Therefore, a lower usage of partner resources was needed, executing only 27 jobs and contributing with 121.65 jobs/hour to the final throughput.

Figure 2 shows the result of applying the performance model to the above experiments, while Table 1 summarize the r_∞ and $n_{1/2}$ parameters obtained.

In the first experiment, the aggregated model is almost equal to the one obtained with the experimental data. In the second experiment, they are not so equal, because the model doesn't capture well the behavior of the partner infrastructure. Due to the low number of jobs submitted to the partner grid, the steady state of testbed saturation is not reached and, thus, the testbed seems to have a bigger number of more heterogeneous resources (notice the high value of $n_{1/2}$, almost equal to the number of submitted jobs). This makes the aggregated model to have a higher r_∞, but also a higher $n_{1/2}$. Nevertheless, the problem is in the input data,

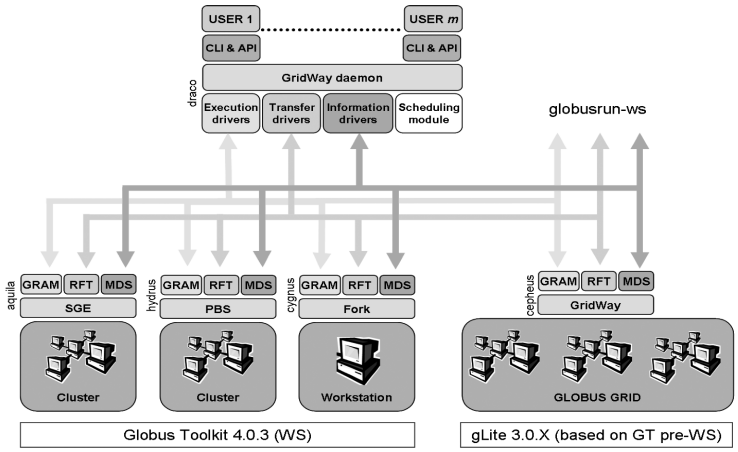


Figure 1. Experimental environment.

Table 2. Optimum and actual number of jobs submitted to each infrastructure.

	Optimum		Actual	
	UCM	fusionVO	UCM	fusionVO
Exp. 1	54	46	51	49
Exp. 2	72	28	73	27

not in the performance model.

A comparison between the optimum number of jobs needed to minimize the optimum makespan predicted by Eq. 6 (with $FG = \{UCM, fusionVO\}$) and the actual number of jobs submitted by GridWay to each infrastructure is shown on Table 2. It can be seen that GridWay numbers in both experiments are very close to the optimum ones.

5. Conclusions and Future Work

We have shown the suitability of the proposed performance model for obtaining a straightforward characterization of federating grid infrastructures by applying it to our solution based on GridGateWays. However, we have identified a limitation of the model when there are not enough samples.

We have applied the performance model to data obtained before-hand in order to validate the model, but the final aim is to automatically compute the parameters of the model (e.g. by periodically executing a given benchmark) in order to take scheduling decisions based on them. Therefore, future work includes the development of scheduling policies considering these parameters to reduce the total execution time of a whole workload, while also taking into account resource ownership, to maximize the use of local resources and so reduce costs.

Finally, we want to extend the experimental scenario with more enterprise, partner and outsourced grids. For the latter, economic models should be proposed and, due to the complexity of such an infrastructure, the use of simulation tools will be of great help. These new ideas, as well as new components for scheduling, negotiation, service level agreement, credential management, and billing, are currently being developed in the context of the Grid4Utility project[2].

6. Acknowledgments

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org.

We want to acknowledge all institutions belonging to the *fusion* VO[3] of the EGEE project for giving us access to their resources.

References

[1] Vázquez, T., Huedo, E., Montero, R.S., Llorente, I.M.: Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures. In: 13th Intl Conference Parallel and Distributed Computing (August 2007), LNCS (2007)

[2] http://www.grid4utility.org

[3] http://grid.bifi.unizar.es/egee/fusion-vo

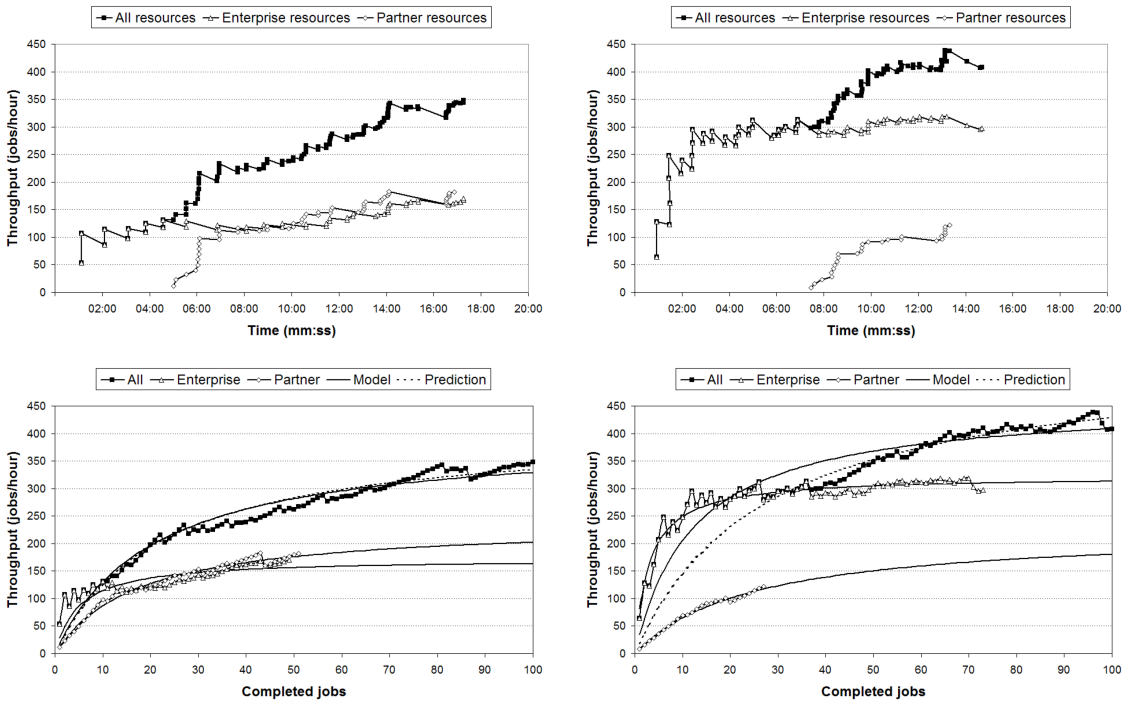


Figure 2. Dynamic throughput achieved in the first (left) and second (right) experiment (top). Performance model and estimation applied to both experiments (bottom).

[2] Walker, R., Vetterli, M., Impey, R., Mateescu, G., Caron, B., Agarwal, A., Dimopoulos, A., Klektau, L., Lindsay, C., Sobie, R., Vanderster, D.: Federating Grids: LCG Meets Canadian HEPGrid. In: Proc. Computing in High Energy and Nuclear Physics (CHEP 2004). (2004)

[3] Baumbauer, C., Goasguen, S., Martin, S.: Bouncer: A Globus Job Forwarder. In: Proc. 1st TeraGrid Conf. (2006)

[4] Breuer, D., Wieder, P., van den Berghe, S., von Laszewski, G., MacLaren, J., Nicole, D., Hoppe, H.C.: A UNICORE-Globus Interoperability Layer. Computing and Informatics **21** (2002) 399–411

[5] Nasri, W., Steffenel, L.A., Trystram, D.: Adaptive Performance Modeling on Hierarchical Grid Computing Environments. In: 7th IEEE Intl Symp on Cluster Computing and the Grid, CCGRID. (2007) 505–512

[6] Cao, J., Kerbyson, D., Nudd, G.: Performance evaluation of an agent-based resource management infrastructure for grid computing. In: 1st IEEE/ACM Intl Symp on Cluster Computing and the Grid, CCGRID. (2001) 311–318

[7] Montero, R.S., Huedo, E., Llorente, I.M.: Benchmarking of High Throughput Computing Applications on Grids. Parallel Computing **32**(4) (April 2006) 267–279

[8] Hockney, R., Jesshope, C.: Parallel Computers 2:Architecture Programming and Algorithms. Adam Hilgee Ltd. (1998)

[9] Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: Proc. IFIP Intl. Conf. Network and Parallel Computing (NPC 2005)

[10] Huedo, E., Montero, R.S., Llorente, I.M.: A Framework for Adaptive Execution on Grids. Software - Practice and Experience **34**(7) (2004) 631–651

[11] Huedo, E., Montero, R.S., Llorente, I.M.: A Modular Meta-Scheduling Architecture for Interfacing with Pre-WS and WS Grid Resource Management Services. Future Generation Computer Systems **23**(2) (February 2007) 252–261

A.4. Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers

Cita completa

C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente. *Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers*. In Proceedings of Grid and Pervasive Computing Conference (GPC 2009), Workshop on Grids, Clouds and Virtualization (WGCV). **IEEE Computer Society**, 113-120, May 2009.

Indicios de calidad

Este congreso está en la categoría CORE C según el ERA Conference Ranking. En la edición de 2009 recibió 48 envíos de los que se aceptaron 21, lo cual supone un ratio de aceptación de aproximadamente 43 %. Ha sido citado 10 veces según Google Scholar.

Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers *

Constantino Vázquez Eduardo Huedo Rubén S. Montero
Ignacio M. Llorente
Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid, Spain

E-mail: {tinova, ehuedo}@fdi.ucm.es, {rubensm, llorente}@dacya.ucm.es

Abstract

Grid computing involves the ability to harness together the power of computing resources. In this paper we push forward this philosophy and show technologies enabling federation of grid infrastructures regardless of their interface. The aim is to provide the ability to build arbitrary complex grid infrastructure able to sustain the demand required by any given service. In this very same line, this paper also addresses mechanisms that potentially can be used to meet a given quality of service or satisfy peak demands this service may have. These mechanisms imply the elastic growth of the grid infrastructure making use of cloud providers, regardless of whether they are commercial, like Amazon EC2 and GoGrid, or scientific, like Globus Nimbus. Both these technologies of federation and dynamic provisioning are demonstrated in two experiments. The first is designed to show the feasibility of the federation solution by harnessing resources of the TeraGrid, EGEE and Open Science Grid infrastructures through a single point of entry. The second experiment is aimed to show the overheads caused in the process of offloading jobs to resources created in the cloud.

1. Introduction

In high performance computing there is the necessity to attend fluctuating and peak demands. For instance, a supercomputing center is subject to this needs, since projects can demand resources punctually for experiments, and they can do so even simultaneously. Moreover, meeting a Service Level Agreement (SLA) defining a Quality of Service (QoS) with the center's available resources can be challenging at certain times. The logical provisioning model will be to use the local infrastructure to attend all the existing demand, if possible (i.e., using their enterprise grid). If that is not enough, the excess load can be delegated to a partner grid, with which a previous arrangement has been made. This partner grid does not need to provide the same interface as the enterprise grid, but still a single point of access is desirable to the whole federated infrastructure. If the computing demand still overflows the existing resources, the center will need to use a cloud provider to perform temporary increase in its computing power. A unified point of access is even valuable as more heterogenous resources are added to the grid infrastructure.

In this paper we will propose and evaluate an architecture to build a grid infrastructure in a very flexible way, being able to hire computer resources with potentially different interfaces; and we will show an experiment to prove its feasibility. This will be done with technologies available nowadays rather than the intention of developing standards for the future, i,e, using interoperation rather than interoperability [1]. Furthermore, we will show a framework for monitoring service capacity and for growing grid infrastructures when it comes close to saturation, using cloud providers like Amazon EC2 [1] and GoGrid [2] (commercial) or Globus Nimbus [3] (scientific); and again we will show empirical data on an experiment showing this dynamic growth. This will provide with the necessary components to build a grid infrastructure with a single point of access that can be adapted as in the aforementioned though experiment of the supercomputing center. One interesting characteristic of using the virtualization that conform clouds is the abil-

*This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, by Ministerio de Educación y Ciencia, through the research grant TIN2006-02806, and by the European Union through the research grant RESERVOIR Contract Number 215605

[1] http://www.amazon.com/ec2
[2] http://www.gogrid.com
[3] http://workspace.globus.org

978-0-7695-3677-4/09 $25.00 © 2009 IEEE
DOI 10.1109/GPC.2009.22

ity to provide resources with certain characteristics, that is, particular software libraries or specific configuration can be asked for in the requested virtual machines (VMs) to better fulfill the demands of the grid infrastructure.

The aim of this paper is therefore twofold, and it contributes with solutions to two related but different challenges:

- The interoperation of different grid infrastructures.
- The dynamic provision of resources using cloud providers.

The structure of this paper is as follows. Section 2 unfolds work related with this paper, while Section 3 unfolds a general architecture of the solution for the desired adapting grid infrastructure. Section 4 deals with the problem of grid interoperation, showing an experiment harnessing the computing power of resource pertaining to the TeraGrid, EGEE and Open Science Grid infrastructures. Section 5 presents a solution to dynamically grow an existing grid infrastructure using resources from cloud providers. Finally, Section 6 states plans for future work and summarizes the conclusions of this paper.

2. Related Work

Interoperation is an intrinsic characteristic of grid technologies, which basically consist on the aggregation of heterogenous resources in which interoperation obviously plays a crucial factor. Over the time, grid middlewares have been evolved without agreement between parts, resulting in incompatible interfaces for grid infrastructures. There are a numerous efforts focused on what is available nowadays, trying to provide federation solutions for the short term. This is exactly the aim of the Grid Interoperation Now [2] (GIN) group of the OGF. We can also remark efforts to interoperate different grid middlewares (for example, Globus and UNICORE [3]). It is also interesting to see efforts like InterGrid [4], proposing the creation of InterGrid Gateways (IGGs) to interconnect the existing different grid islands. Moreover, there are various works enabling interoperability between existing metaschedulers [5]. There is even an OGF group devoted to this research line, the Grid Scheduling Architecture (GSA) research group. Our solution takes advantage of the modular architecture of the GridWay metascheduler to use different adapters to access grid infrastructures with different interfaces, allowing to do so with a single point of access. Interoperation efforts are particularly important in the context of the EGEE infrastructure with respect to its National Grid Infrastructures (NGIs). Basically, each state contribute to the whole of the EGEE infrastructure providing resources grouped in a NGI. Therefore, we an think of the EGEE as a federation of NGIs, so interoperation becomes a key aspect between NGIs.

On the other hand, regarding on-demand provision of computational services, different approaches have been proposed in the literature. Traditionally, these methods consist in overlaying a custom software stack on top of an existing middleware layer, like for example the MyCluster Project [6] or the Falkon system [7]. These approaches essentially shifts the scalability issues from the application to the overlaid software layer, whereas the proposed solution transparently scales both the application and the computational cluster.

The idea of a virtual cluster which dynamically adapts its size to the workload is not new. Jeffrey Chase et al., from Duke University, describe [8] a cluster management software called COD (Cluster On Demand), which dynamically allocates servers from a common pool to multiple virtual clusters. Although the goal is similar, the approach is completely different. COD worker nodes employ NFS to mount different software configurations. In our case, the solution is based on VMs, and the system is studied from a grid perspective. The use of virtualization to provide on-demand clusters has been also studied in the context of the Globus Nimbus [9]. Globus Nimbus provides a WSRF interface to launch heterogeneous clusters on a cloud. However, these clusters can not be easily integrated with the local resources nor can be supplemented with other cloud providers.

Finally, Amazon Elastic Computing Cloud provides a remote VM execution environment. It allows to execute one or more "Amazon Machine Images" on their systems, providing a simple web service interface to manage them. Users are billed for the computing time, memory and bandwidth consumed. This service greatly complements our development, offering the possibility of potentially unlimited computing resources. It would be possible to employ a grid-enabled Amazon Machine Image, and create as many instances as needed, getting on-demand resources in case the physical hardware cannot satisfy a peak demand. UnivaUD is an example of a company offering solutions to virtual provisioning by monitoring applications, gathering and analyzing detailed performance metrics, and then driving appropriate provisioning actions based on these metrics to meet established customer SLAs [4].

Our approach uses the concept of a dynamically adapting grid infrastructure, but more aligned with the concept of a Service Manager like Hedeby [5], although using VMs with specific configurations rather than configuring physical servers. In this aspect, our solution is similar to that provided by RightScale [6], but it differs from it since it is not a completely virtualized solution, but rather a way to extend

[4] http://www.univaud.com/reliance/use-cases/provisioning.php
[5] http://hedeby.sunsource.net
[6] http://www.rightscale.com

a physical infrastructure by the punctual use of virtualized resources.

3. Architecture

Dynamic provisioning posses various problems. One problem in this field is the importance of interoperability, i.e., being able to grow using any type of given resource, independently of what interface may be offering. Another problem is answering the question of when this dynamic growth is necessary and how to actually perform it. An even a third issue can be the enforcement of a budget on this decision, taking into account expected CPU and network usage.

In this section, these problems are addressed by means of a grid infrastructure that can be flexibly built, that is aware of its load, featuring a single point of access and that can incorporate new resource temporarily in an automatic fashion to satisfy heavy demands. Figure 1 sketches an architecture of such a solution. We can see that one of the building blocks is the GridWay metascheduler.

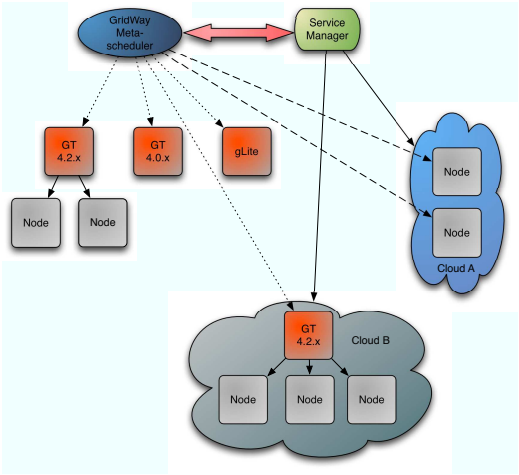


Figure 1. Architecture for an elastic grid infrastructure.

The flexible architecture of this metascheduler allows the use of adapters. Although sharing the same interfaces is the ideal way to achieve interoperation, sometimes there are different middlewares deployed in the sites to be federated, and it is unfeasible to unify them for a variety of reasons (politics, time constraints or ongoing migration or upgrades). This can be seen as the consequence of not having an accepted standard, and therefore, lack of interoperation. One possible solution to federate these different sites is to build a portal that uses different components (to submit jobs, gather information, transfer files, etc) to interface these sites. These components are designed specifically for a particular middleware stack or even version, and we can call them adapters.

GridWay already has a number of adapters, called Middleware Access Drivers (MADs), that enable access to different production grid infrastructures. In Section 4, an experiment shows the metascheduler accessing the EGEE, TeraGrid and Open Science Grid, featuring different interfaces. Moreover, there are plans to provide SSH MADs, so access to local resources can be achieved with decreased overhead, and avoiding the need to have installed and configured in the nodes any grid software as, for instance, the Globus Toolkit.

Once described the federation approach, lets see the proposed architecture for dynamic provisioning, which principal component is the the Service Manager. It is used to monitor the GridWay metascheduler, and when the load of the system excesses a threshold, detected using heuristics, it is responsible to grow the available grid infrastructure using specific adapters to access different cloud providers. This growth can be accomplished in two ways. The first one is by requesting a number (calculated with the aid of said heuristics) of single hosts. These need to have a previously defined software configuration that will then help them enroll in the available grid infrastructure. This corresponds to the use of Cloud A in Figure 1.

Another possibility is to deploy a full virtualized cluster, with a front-end controlling a number of slave nodes. This front-end can then enroll itself to the existing grid infrastructure, adding its capacity. The GridWay metascheduler features mechanisms to dynamically discover new hosts or sites which can be used for this purpose. This corresponds to the use of Cloud B in the figure.

Therefore, the provisioning model we envision is twofold. The first mode adds one single computing resource to the grid infrastructure. This computing resource can be accessible through a GRAM interface, meaning that the VM that is going to be awaken needs to have the Globus Toolkit installed and correctly configured. An even more practical approach will be to use just SSH access to perform job execution in this kind of nodes, the GridWay metascheduler already has a prototype of such SSH drivers. In this way, machines from cloud providers can be used out-of-the-box, with little to non configuration needed, basically SSH access is the only requirement. On the other hand, a second mode of growing the existing grid infrastructure would be to use these cloud providers in a slightly different fashion. Negotiation with the cloud provider will grant access to a virtual cluster, accessible through GRAM and controlled by a LRMS like for example PBS or SGE. This cluster will then be added to the federated grid infrastructure the same way as one of the physical sites we saw in the last section. Future work is planned to enrich the flexibility of the grid infrastructure by removing the GRAM layer, enabling Grid-

Way to access the cluster by talking directly to the local resource management system (LRMS)

Moreover, this solution has another advantage. Not only it can dynamically increase the size of the grid infrastructure, but the added computing nodes can be waked with different configurations. In other words, if the Service Manager can be built in such a way that it won't only concern itself with the need to increase the computing capacity, but it can do so in a service oriented way. If there is one specific service which is suffering from the peak demand, the Service Manager can decide to increase the number of nodes prepared to satisfy such a service. For instance, if the service is an application that requires specific mathematic libraries, virtual machines images containing that specific libraries be chosen to be awaken as nodes to increase the grid infrastructure capacity.

4. Provisioning from Heterogeneous Grid Infrastructures

We can think of *interoperation* as an immediate solution for the collaboration between two or more heterogeneous grids. On the other hand, *interoperability* focuses on the big picture and tries to bring together technologies that implement the grid infrastructure by means of standardization (like, for example, the Simple API for Grid Applications, SAGA, or the Basic Execution Service, BES, do). It is clear that this can not be achieved without a significant amount of effort and, more important to the point being made here, time. Thus, the need to provide interoperation and the justification of the GIN group within the OGF.

Since most common open standards to provide grid interoperability are still being defined and only a few have been consolidated, grid interoperation techniques, like for instance adapters, are needed. An adapter is, according to different dictionaries of computer terms, a device that allows one system to connect to and work with another. The aim of this section is to show the feasibility of the adapter-based approach to interoperation.

GridWay's architecture is flexible enough to allow for the use of adapters to achieve interoperation between infrastructures exposing different interfaces. The experiment shown later in this section proves the feasibility of this approach. Notwithstanding, this solution in turn posses a new problem that requires the development of new heuristics for the optimal scheduling of jobs across resources of different infrastructures. Lets see an example in order to clarify what these heuristics will have to decide. For clarity's sake, we are going to assume that GridWay is configured to access two different infrastructures, one with 30 free nodes and another with 20; and GridWay receives a job array of 18 elements. Currently, GridWay scheduler doesn't take into account the notion of array for scheduling, so it will probably begin scattering jobs across both infrastructures. Nonetheless, it is intuitively clear that we would want to send the whole array to, for instance, the 20-node infrastructure, making space for a possible next 30 job array. Some work has been done in this direction [10], and it can be taken as a good starting point for new heuristics for metascheduling.

GridWay's adapters are the Middleware Access Drivers (MADs), enabling the metascheduler to access different infrastructures simultaneously and seamlessly. GridWay has evolved over time to take advantage of the adapters technique, it is interesting to see its evolution, from a first tentative effort to harness both EGEE and the IRISGrid infrastructures [11], to the seminal work that produced its full blown current modular architecture [12], motivated by the interoperation between grid resource management services provided by Globus. There are three types of adapters: execution, transfer and information MADs.

4.1 Description of the Experiment

GridWay metascheduler was configured to access four different infrastructures. Not only different interfaces were the problem, but also different versions of the same middleware posed their own issues. Furthermore, different configurations of even the same version of the same middleware stacks are troublesome for the correct interoperation of the four infrastructures. GridWay was set to use different adapters especially configured to access the following infrastructures:

- *Open Science Grid*: This infrastructure offers two versions of the Globus Toolkit, deployed using VDT[7]: the pre Web Services (preWS) and Web Services (WS) versions.
- *TeraGrid*: Again, two versions of Globus are offered. Configuration for the file transfer was different than in other infrastructures since the Storage Element (SE) was a separate machine (sharing homes with the Computing Element).
- *EGEE*: This infrastructure uses the gLite middleware stack, which is based on Globus preWS. The preWS MADs are used, with a special file staging configuration. An information MAD for the Berkely Database Information Index (BDII) using the Glue scheme is used for host monitoring.
- *UCM:* This is dsa-research.org group local infrastructure at Universidad Complutense de Madrid. Probably due to the fact that the Globus WS MADs were developed against this very cluster, no extra configuration was needed for GridWay to access it.

[7] http://vdt.cs.wisc.edu/

Three iterations of two hundred jobs each were sent to the GridWay metascheduler, which in turn distribute them along the four infrastructures, in order to show the federation feasibility, using the Embarrassingly Distributed (ED) benchmark from the NAS Grid Benchmark suite.

Figure 2 depicts the set-up of the experiment. Resources are accessed by different adapters (dotted arrows represent preWS interfaces, solid arrows represent Web Services, WS, interfaces). For illustrative purposes, one resource from each infrastructure was chosen for this paper experiments, so they could be performed in a more controlled environment. A complete resource listing for the adapters scenario can be seen in Figure 3. To show different adapters in action, WS MADs were used to access the TeraGrid (with Host Identifier , HID, 0 in the figure) and the UCM (HID 2) infrastructure, while the preWS MADs were chosen to access the Open Science Grid (HID 1) and the EGEE (HID 3) infrastructure. It is worth noting that WS versions 4.0.x and 4.2.x of the Globus Toolkit are incompatible, meaning that clients from one version cannot access servers from another. GridWay is able to use MADs from toolkits of both versions, and therefore access seamlessly Globus containers of both versions, solving neatly the incompatibility between them and allowing a smooth upgrading process. In this experiment, GridWay accesses version 4.0.x of the Globus Toolkit to use resources from the TeraGrid, and version 4.2.x to use UCM resources.

4.2 Analysis of Results

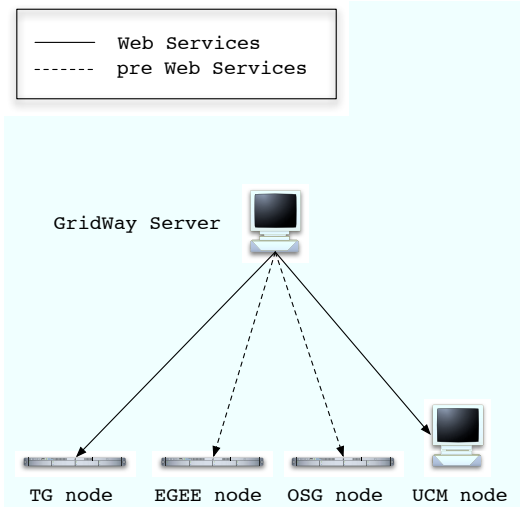


Figure 2. Interoperation across infrastructures

Figure 4 shows the number of jobs against the infrastructures where the jobs were executed. There is an approximately even distribution between our local cluster (*UCM*), the Open Science Grid preWS resource (*OSG preWS*) and the TeraGrid WS one (*TG WS*), while the *EGEE* resource shows a smaller ratio of jobs completed. This is due to the chosen EGEE site for the EGEE, ramses.dsic.upv.es, having lower computing power (both CPU and memory) than the resource chosen for this experiment in the other infrastructures.

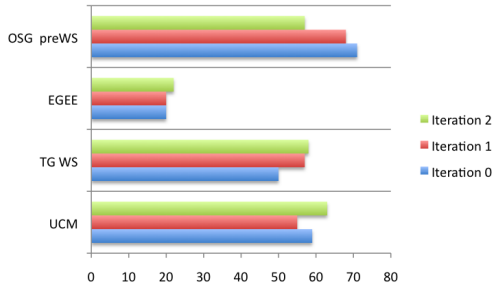


Figure 4. Job distribution across infrastructures

The capability to add new resources with potentially different interfaces to an existing grid infrastructure is therefore shown by this experiment, maintaing a single point of access, in this case the GridWay metascheduler.

5. Extension to Cloud Providers

Last section model provides a single point of access to a grid infrastructure that can be extended using different providers with potentially different interfaces. In this section it is presented a technique to guarantee pre accorded QoS and, therefore, meet SLAs even in cases of high saturation of the grid infrastructure. Also, it gives a solution for peak demands that occur without enough time for planning the extension through federation. This solution involves the elastic growth of the computing infrastructure by means of a cloud provider, being that commercial (Amazon EC2, GoGrid) or scientific (Globus Nimbus), being the charge model the only difference between these two type of cloud providers.

To enable our grid infrastructure to be able to meet a given QoS and so satisfying predefined SLAs we need a component that is aware of the load of said infrastructure. Our solution consists of a Service Manager component that monitors the metascheduler in order to find when it should elastically grow (or, conversely shrink) the available resources by waking up nodes or entire clusters (or shutting them down). In short, this component is responsible for adapting the grid infrastructure to dynamic computing demands.

```
HID PRI  OS              ARCH   MHZ %CPU MEM(F/T)   DISK(F/T)  N(U/F/T)  LRMS              HOSTNAME
0   1                                                  0/5/5  PBS               tg-grid.uc.teragrid.org
1   1    Linux2.4.21-32. i686  2665  189 964/2006 62787/73964   0/10/10  jobmanager-condor cmsgrid01.hep.wisc.edu
2   1    Linux2.6.24-17  x86_6 1995  100   18/499 10169/21817     0/5/5  SGE               aquila.dacya.ucm.es
3   1    ScientificSL4   i686   866    0  513/513         0/0   0/21/22  jobmanager-lcgpbs ramses.dsic.upv.es
```

Figure 3. Resources as provided by the `gwhost` command.

This solution takes advantage of the chosen GridWay metascheduler. It employs a dynamic scheduling system and therefore it can detect when a new machine has been added or removed from a grid infrastructure, and redistribute the work load. It also features fault detection & recovery capabilities. Transparently to the end user, it is able to detect and recover from any of the Grid elements failure, outage or saturation conditions.

The Service Manager is in charge of monitoring the metascheduler and to detect and excess of load for the available resources. In order to detect this excess, a set of heuristics have to be defined, so they define the threshold of number of pending jobs waiting for resources, and the load present on the available resources. A second set of heuristics is needed to decide which cloud provider is going to be used to elastically grow the cluster. These heuristics should be based on economics criteria, minimizing the total cost of CPU and network usage. In this line, a good starting point would be the work done in budget constrained cost-time optimization algorithms for scheduling [13].

Optionally, there is even a third set of rules that the Service Manager need to employ in case that it is aware of what service demands need to be satisfied. This rules will be used to decide which type of VM is going to be awaken to satisfy the excess of demand. For instance, in the context of a supercomputing center, VMs with a certain Virtual Organization (VO) configuration can be the ones chosen to be up, if that VO has suddenly increased its demand for computing power.

5.1 Description of the Experiment

This experiment is designed to evaluate the overhead incurred in the management of a new worker node in a virtualized cluster being executed in Globus Nimbus. Hence, for the purpose of this experiment, the virtual cluster is already being executed, and we are measuring the overheads between the different layers of our architecture in the process of adding a new worker node, i.e., the overheads caused by awaking the worker node, shutting it down, being it detected by the SGE, the MDS and GridWay, and the processing overhead incurred by this node since it is virtualized. This experiment is performed on a local cloud deployed in our laboratory at dsa-research.org, using Globus Nimbus as the cloud manager.

In order to better understand the chain of events and where the overheads occur lets see the actions that takes place when the Service Manager decides to add a new worker node to the grid infrastructure. First, the Service Manager requests a new VM to Nimbus, which determines the best node to run the virtual machine, based on the resources requested (e.g memory). Afterwards, if the VM image (appliance) is not local to the host system, it accesses the image server via a suitable protocol (e.g. GridFTP) and obtains a copy. Once the image has been transferred, the physical node's DHCP server configuration file is altered in order to establish VM's IP and hostname. When these operations conclude, the VM is booted. When the VM has been deployed and is running, it registers on LRMS front-end as an execution host. After a given time, the Grid information system (MDS) detects the new node and publishes it. Finally, GridWay the metascheduler will refresh the information of the available Grid resources, and detect the new worker node. Then, according to the scheduling policies, it will allocate jobs on this new resource by interfacing with the Grid execution service (GRAM). The behavior of the previous deployment strategy will be analyzed on a testbed based on Globus Toolkit 4.0.3 and GridWay 5.2.1.

5.2 Evaluation of Overheads

Several experiments where run in the testbed in order to analyze the overheads caused by virtualization and the software layers corresponding to the proposed architecture.

The first overhead considered is caused by the *deployment* of a VM under several conditions. The experiment consisted in the deployment of one, two and three VMs con the same physical machine. Respectively, total times of deployment in seconds were 118.58, 308.02, 495.88. We would like to remark that deploying more than 3 VMs simultaneously derived most of the times in error situations. Also, note that the overhead induced by the SGE layer is negligible (the time to register a worker node in the cluster is less than a 1% of the total deployment time).

In the case of *shutting down* a VM (Table 1), we have measured three relevant values. In this case the time is constant regardless of the number of VMs being shut down. Overhead introduced by SGE when shutting down a VM can be minimized by reducing the polling time. Default value is 300 seconds, so it takes an average of 150 seconds to detect the new situation. Reducing polling time limits the overhead, although increments network usage. System ad-

ministrator must tune this value according to the number of nodes in the cluster and average VM uptime.

Number	Command Received	VM Destroyed	SGE	Total
1	0.78	6.22	145	152
2	0.88	6.46	158	165.33
3	0.67	7.33	151	159

Table 1. Times (in seconds) when a worker node is shut down

Once the VM is being booted, time is needed until it is enrolled for use in the original grid infrastructure, and complementarily, time is also needed to plug the node out of the infrastructure. These overheads can be called *grid integration* overheads. The time to start a virtual worker node (time since the Service Manager requests a worker node, 114 sec., till it is registered in the LRMS, 2 sec.) is roughly 2 minutes. The time to register the new slot in the Grid Information system (MDS4) is about 170 seconds. It is worth pointing out that MDS publishing time is greater than the time employed on deploying one VM plus SGE register time. Therefore, when sequentially deploying several VMs both times overlap, producing an additional time saving. The MDS and GridWay overhead can be limited by adjusting their refresh polling intervals. When shutting down, the same steps are accomplished. In this case, the time until the operation is accomplished at the machine layer is greatly reduced, from 114 to 7 seconds. However, time until LRMS detects the lack of the VM is incremented, from 2 to about 150 seconds. It is interesting to note that the metascheduler could assign jobs to the cluster during the worker node shutting down time. In this case the metascheduler should be able to re-schedule this job to another resource.

Virtualization technology imposes a performance penalty due to an additional layer between the physical hardware and the guest operating system. This penalty (that we can call the *processing* overhead) depends on the hardware, the virtualization technology and the kind of applications being run. Two good performance comparisons of VMware and Xen were conducted by the computer science departments at University of Cambridge [14]. and Clarkson University [15]. On these studies, Xen performed extremely well in any kind of tasks, with a performance loss between 1 and 15%. VMware also achieves near-native performance for processor-intensive tasks, but experiences a significant slow-down (up to 88%) on I/O bound tasks. Nimbus development team measured its performance in a real-world grid use case [16], a climate science application, achieving about a 5% performance loss. Previous results [17, 18] indicate that, in general, the virtualization platform has no significant impact on the performance of memory and CPU-intensive applications for HPC clusters.

6. Conclusions and Future Work

We have shown an architecture to build any type of arbitrary complex grid infrastructures with a single point of access, which is able to dynamically adapt its size (and therefore, capacity) using a cloud provider to react to peak demands and/or meet SLAs.

This paper opens the path for a number of research lines to be followed and developments to be done. Regarding the provisioning from heterogenous infrastructures, there is room for the development of new adapters to access a greater number of different types of grid infrastructures. Currently, there are already two new drivers being developed, one for the new job execution service from gLite, called CREAM, and one set of drivers to interoperate with the UNICORE middleware. Complementary to these developments, it will be interesting to add SSH adapters to the GridWay metascheduler so it is able to interface with "raw" machines, i.e., without the need of installing a grid middleware layer. This will suit perfectly the use of VMs from cloud providers such as Amazon EC2, since the only configuration to be done would be to obtain SSH access. Also, adapters for direct access to LRMS like SGE or PBS will add a great flexibility in the grid infrastructures that can be built using this architecture. One good option for this adapter will be to develop it against Distributed Resource Management Application API (DRMAA), since then it will be possible to plug it to the multiple LRMS that supports this API.

Heuristics for the problem posed by the need to schedule workloads across several grid infrastructures need to be develop. Moreover, there is also work to be done in the heuristics needed to tune the Service Manager. From an economic model of the cloud provider to a set of rules to aid in the decision of adding resources to the grid infrastructure, all is needed to properly develop a Service Manager that is aware of what service is being offered and what is needed to properly satisfy its demand.

References

[1] Field, L.: Getting Grids to work together. CERN Computer Newsletter **41**(5) (Nov-Dec 2006) 8–9

[2] Riedel, M., et al.: Interoperation of World-Wide Production e-Science Infrastructures. Concurrency and Computation: Practice and Experience (2008) (in press).

[3] Breuer, D., Wieder, P., van den Berghe, S., von Laszewski, G., MacLaren, J., Nicole, D., Hoppe,

H.C.: A UNICORE-Globus Interoperability Layer. Computing and Informatics **21** (2002) 399–411

[4] Assunção, M.D., Buyya, R., Venugopal, S.: Intergrid: A Case for Internetworking Islands of Grids. Concurrency and Computation: Practice and Experience **20**(8) (July 2008) 997–1024

[5] Bobroff, N., Fong, L., Kalayci, S., Liu, Y., Martinez, J.C., Rodero, I., Sadjadi, S.M., Villegas, D.: Enabling Interoperability among Meta-Schedulers. In: Proceedings of 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid-2008). (2008) 306–315

[6] Walker, E., Gardner, J., Litvin, V., Turner, E.: Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment. In: In Proceedings of the IEEE Challenges of Large Applications in Distributed Environments. (2006) 95–103

[7] Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I., Wilde, M.: Falkon: a Fast and Light-weight tasK executiON farmework. In: In Proceedings of the IEEE/ACM SuperComputing. (November 2007)

[8] Chase, J., Irwin, D., Grit, L., Moore, J., Sprenkle, S.: Dynamic Virtual Clusters in a Grid Site Manager. In: Twelfth IEEE Symposium on High Performance Distributed Computing (HPDC), Seattle, Washington (June 2003)

[9] Freeman, T., Keahey, K.: Flying Low: Simple Leases with Workspace Pilot. In: Euro-Par, 2008

[10] Leal, K., Huedo, E., Llorente, I.M.: Dynamic Objective and Advance Scheduling in Federated Grid. In: International Conference on Grid computing, high performance and Distributed Applications. Volume 5331. (2008) 711–725

[11] Vázquez-Poletti, J.L., Huedo, E., Montero, R.S., Llorente, I.M.: Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. Journal of Parallel and Distributed Computing **5**(65) (May 2005) 763–771

[12] Huedo, E., Montero, R.S., Llorente, I.M.: A Modular Meta-scheduling Architecture for interfacing with pre-WS and WS Grid Resource Management Services. Future Generation Computing Systems **23**(2) 252–261

[13] Buyya, R., Murshed, M., Abramsin, D., Venugopal, S.: Scheduling Parameter Sweep Applicatoin on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm. International Journal of Software: Practice and Experience (SPE) **5** (2005) 491–512

[14] Barham, P., Dragovid, B., Fraser, K., Hand, S., Ahrris, T., Ho, R.A., Pratt, I., Warfield, A.: Xen and the Art of Virtualization. In: Symposium on Operating Systems Principles. (October 2003) 164–177

[15] Clark, B., Deshane, T., Dow, E., Evanchik, S., Herne, M., Matthews, J.: Xen and the Art of Repeated Search. In: USENIX Annual Technical Conference. (2004) 47–47

[16] Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayor, B., Zhang, X.: Virtual clusters for grid communities. In: Proceedings of the Sixth IEEE International Symposium on Cluster Computer and the Grid (CCGRID 06), IEEE Computer Society (2006) 513–520

[17] Rubio-Montero, A.J., Montero, R.S., Huedo, E., Llorente, I.M.: Management of Virtual Machines on Globus Grids Using GridWay. In: In Proceedings of the 4th High-Performance Grid Computing Workshop, in conjunction with 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS-07). 1–7

[18] Rodriguez, M., Tapiador, D., Fontan, J., Huedo, E., Montero, R.S., Llorente, I.M.: Dynamic Provisioning of Virtual Clusters for Grid Computing. In: In Proceedings of the 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 08), in conjuction with EuroPar08. (2008)

A.5. Federation of TeraGrid, EGEE and OSG Infrastructures through a Metascheduler

Cita completa

C. Vázquez, E. Huedo, R.S. Montero and I.M. Llorente. *Federation of TeraGrid, EGEE and OSG Infrastructures through a Metascheduler*. **Future Generation Computer Systems**, 26(5), 979-985, 2010.

Indicios de calidad

Este artículo ha sido publicado en la revista Future Generations Computer Systems, con un índice de impacto de 2,365 en JCR 2010, ocupando el primer tercio (10/97) de la categoría de "Computer Science, Theory and Methods". Ha sido citado 5 veces según Google Scholar.

Future Generation Computer Systems 26 (2010) 979–985

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Federation of TeraGrid, EGEE and OSG infrastructures through a metascheduler☆

Constantino Vázquez*, Eduardo Huedo, Rubén S. Montero, Ignacio M. Llorente

Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, Spain

ARTICLE INFO

Article history:
Received 16 December 2008
Received in revised form
26 March 2010
Accepted 16 April 2010
Available online 24 April 2010

Keywords:
Grid
Interoperability
Federation
Scalability
Metascheduler

ABSTRACT

Since its conception, Grid technology concerned itself with interoperability between heterogeneous computers by providing a middleware layer to abstract underlying resource characteristics. Unfortunately there is no agreement today on a common set of standards and, therefore, different conceptions lead to different middleware implementations. That effectively rendered interoperability between grid infrastructures to be a complex issue, from which originally sprang the idea of Grid. In this paper we present technologies to achieve interoperation (i.e. interoperability not based on standards) between sites and show its feasibility, aiming to provide a mid-term solution to the federation problem until the promise of interoperability through standards becomes a reality. Using interoperation techniques and bringing them together in a common federation component, the GridWay metascheduler, we are able to offer common access to well known grid infrastructures. This approach is demonstrated in the performance evaluation of the execution of one benchmark of the NAS Grid Benchmark suite.

1. Introduction

A growing number of grid infrastructures and middleware are being developed and deployed, favored by the high expectations raised by Grid [1,2] computing. Some of the most successful infrastructures (e.g. TeraGrid,[1] EGEE[2] or Open Science Grid[3]) are used at the production level, enabling resource sharing and collaboration between scientific institutions geographically apart. Nevertheless, if we want to move forward and aggregate resources from these infrastructures, we face several problems.

It is not really surprising finding here the same problems found in the early stages of grid technology. The goal is still the same: harness resources independently of their software stack, geographical disposition or administration domains; albeit scaled up the ladder a bit: we are trying to interoperate grids. These grids have potentially different middleware stacks which make them incompatible, and even those who use the same middleware have slightly different deployment characteristics that makes them non-interoperable.

There are many examples in the literature on how to achieve federation between grids with different interfaces in order to achieve the so-longed-for world wide grid. Interoperability through the use of standard and open interfaces is one of the defining characteristics of grid technologies [3]. There are ongoing projects that address the definition of these standard interfaces, for example SAGA [4], an initiative to provide grid applications with a common standard API so they can be built in a middleware-agnostic way. This is an example of a solution for interoperability from the application point of view. There are standardization efforts like the Job Submission Description Language (JSDL), designed to describe jobs to be run on the grid. Others, like the Basic Execution Service (BES), focus on how to send requests to initiate, monitor, and manage computational activities. These two standards (JSDL and BES) are used in the HPC Basic Profile [4]. Both SAGA and HPC Profile efforts stem from the Open Grid Forum (OGF[4]), an organization that promotes the creation of standards for interoperability.

But while other needed standards are being defined or current ones consolidated, there is another set of efforts that focuses on what is available nowadays and tries to provide federation solutions for the short term. This is exactly the aim of the Grid Interoperation Now [5] (GIN) group of the OGF, in which some of the projects mentioned from now onwards participate. Following this interoperation philosophy, we can note efforts to interoperate different grid middleware (for example, Globus and UNICORE [6]) and approaches based on portals and workflows [7]. In this line, it is also interesting to see efforts like InterGrid [8], which propose the

☆ This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through MEDIANET Research Program S2009/TIC-1468, by Ministerio de Ciencia e Innovación, through the research grant TIN2009-07146, and by the European Union through the research grant EGEE-III Contract Number 22667.

* Corresponding author. Tel.: +34 639150939; fax: +34 913947527.
E-mail addresses: tinova@fdi.ucm.es (C. Vázquez), ehuedo@fdi.ucm.es (E. Huedo), rubensm@dacya.ucm.es (R.S. Montero), llorente@dacya.ucm.es (I.M. Llorente).

1 http://www.teragrid.org/.

2 http://www.eu-egee.org/.

3 http://www.opensciencegrid.org/.

4 http://www.ogf.org/.

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.04.004

creation of InterGrid Gateways (IGGs) to interconnect the existing different grid islands, or GridX1 [9], which can be accessed as another resource of LCG. We can find other works based on the same idea of interoperation using current interfaces, for instance by introducing the concept of a meta-broker component [10], which will aid the access to resources outside one grid domain. Moreover, and complementary to the direction taken by this paper, there are various works enabling interoperability between existing metaschedulers [11]. There is even an OGF group devoted to this research line, the Grid Scheduling Architecture (GSA) research group.

In this paper, we will show how to uniformly access grid resources from different infrastructures and how the tool used for this solution (the GridWay metascheduler [12]) is production ready, i.e., it scales. We will use *federation* to describe this aggregation of resources, and show how it is defined by two intrinsic characteristics: *interoperation* and *scalability*, although there are other, less technical issues out of the scope of this paper. Our solution offers unprecedented levels of flexibility for constructing any kind of grid infrastructure.

The remainder of this paper conforms with the following structure. Section 2 presents a strategy to achieve federation, while Section 3 describes the tool involved in the interoperation process and Section 4 shows how it scales and therefore is suitable for federation. Afterwards, Section 5 explains the output of the different federation experiments (coming from different federation approaches) and comments on their differences. Finally, Section 6 outlines the main conclusions that can be drawn from this work and sets plans for future work on this matter.

2. Interoperation for federation

Reaching agreement between infrastructures on which interface to use to achieve interoperability takes time [13]. Driven by this factor, the OGF makes a clean distinction between *interoperability*, or the native ability of grids and grid technologies to interact directly via common open standards, and *interoperation*, or the set of techniques to get production grid infrastructures to work together in the short term.

Hence, we can think of *interoperation* as a more immediate solution for the collaboration between two or more heterogeneous grids. On the other hand, *interoperability* focuses on the big picture and tries to bring together technologies that implement the grid infrastructure by means of standardization (like, for example, SAGA or BES do). It is clear that this is not a feat that can be achieved without a significant amount of effort and, more important to the point being made here, time. Thus, the need to provide interoperation and the justification of the GIN group within the OGF.

Since most common open standards to provide grid interoperability are still being defined and only a few have been consolidated, grid interoperation techniques, like adapters and gateways, are needed. An adapter is, according to different dictionaries of computer terms, a device that allows one system to connect to and work with another. On the other hand, a gateway is conceptually similar to an adapter, but it is implemented as an independent service, acting as a bridge between two systems. The main drawback of adapters is that grid middleware or tools must be modified to insert the adapters. In contrast, gateways can be accessed without changes on grid middleware or tools, but they can become a single point of failure or a scalability bottleneck [13].

Taking these techniques into account, there are three different ways in which interoperation can be achieved:

- *Common interfaces*. This is the most straightforward way of interoperation. It requires that all sites involved in the federation share the same interfaces, so interoperation between them becomes just an administrative task regarding authorization and other policies. A single point of entry (i.e. a portal) can be easily built, since it can be used to access all the sites using the same mechanism, although, as will be seen in Section 5, sharing the same middleware stack or the same interfaces does not guarantee the ability of having exactly the same way to access them, due to, for example, different configurations that the middleware is subject to.
- *Adapters*. Although sharing the same interfaces is the ideal way to achieve interoperation, sometimes there different middleware is deployed in the sites to be federated, and it is unfeasible to unify them for a variety of reasons (politics, time constraints or ongoing migration or upgrades). This can be seen as the consequence of not having an accepted standard, and therefore, lack of interoperability. One possible solution to federate these different sites is to build a portal that uses different components (to submit jobs, gather information, transfer files, etc) to interface these sites. These components are designed specifically for a particular middleware stack or even version, and we can call them adapters.
- *Gateways*. Another way to achieve interoperation consists in encapsulating one or more sites under one single resource accessed through one interface. The interoperation problem can then be reduced to the *Common interfaces* scenario. This encapsulation acts by translating the requests from the portal to requests that the underlying sites can understand, and conveying the results to the originator of the job request. This technique is conceptually similar to network *gateways* and is especially suited for situations where it is not possible to modify higher level services.

Therefore we must wonder whether interoperation is a necessary and *sufficient* condition for grid federation. We argue that this is not the case, since there are even non-technical issues that we are not taking into account, like for example political matters (when and how a user should use or not use a given infrastructure) and operational issues (which software is installed in the execution nodes, for example). Nevertheless, in Section 5 it is shown how two grid infrastructures with common interfaces (as TeraGrid and Open Science Grid having the same execution service interface, Globus GRAM) are not necessarily federated, but interoperation is needed because their execution model is different (for example, their storage model is very dissimilar). Thus interoperation is a necessary condition for federation.

Considering interoperation enough for federation leaves the key factor out of the equation. This factor is indeed scalability, it is important to know if the component that performs the federation actually scales. This question is tackled in Section 4, where we are going to measure the scalability of the component we are using for federation: the metascheduler. Reaching this point we claim that true federation is achieved, among other less technical issues, by the conjunction of two key factors: *interoperation* and *scalability*.

3. The GridWay metascheduler

The GridWay metascheduler enables large-scale, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers…), managed by different LRM (Local Resource Management) systems, within a single organization (enterprise grid) or scattered across several administrative domains (partner or global grid).

GridWay is composed of several modules as seen in Fig. 1. First we have the GridWay daemon, which is the core that coordinates the whole job life cycle process by means of a state machine. It uses a set of Middleware Access Drivers (MADs), basically independent processes that talk to the daemon using the standard I/O streams. They are used to perform execution (for job submission), transfer (for data staging) and information (for

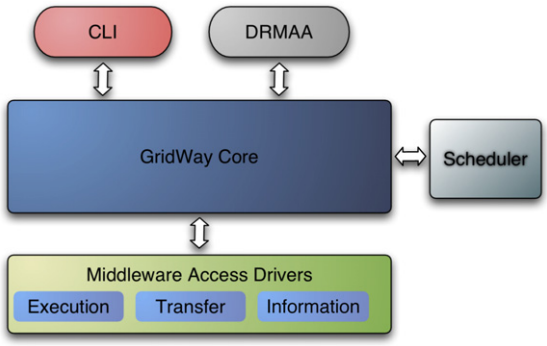


Fig. 1. GridWay high level architecture.

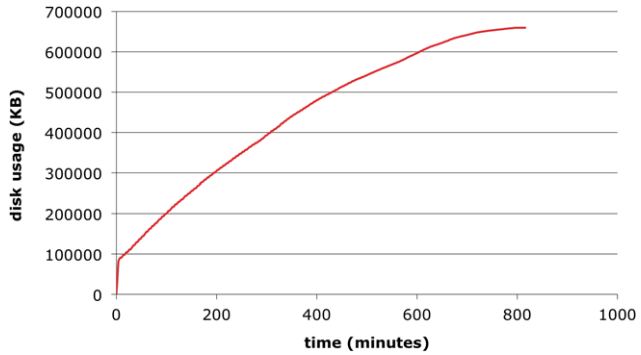


Fig. 2. Disk consumption by all GridWay components, including logs and accounting information.

resource discovery and monitoring) tasks; several MADs of each type can be loaded and used simultaneously. The last component in which we can decompose GridWay is the scheduler, which also lives in a separate process, also communicating with the daemon through the standard streams, and is in charge of the job allocation in the resources.

GridWay exhibits two ways of interfacing with the core. One is a powerful command line interface (CLI), featuring commands to manage jobs and computing resources. But GridWay can be used programmatically with the Distributed Resource Management Application API (DRMAA), which is an OGF standard. In this way, applications can be built abstracted from the heterogeneous resources where it can be executed.

4. Experiments: metascheduler scalability

In this section we will show experiments designed to demonstrate GridWay's scalability and, hence, its ability to perform the role of federation component. To measure the scalability of GridWay, we set up an experiment that tries to saturate all GridWay components by submitting 10,000 jobs.

The resources used for the testbed are located in the Universidad Complutense de Madrid (UCM) local grid, so issues like network saturation can be more controlled. These resources are Intel Pentium 4's (3.2 GHz) with 2 GB of RAM that communicate with each other over a 100 LAN network, running Debian Etch. The testbed for the scalability test uses four computing elements with the aforementioned specifications, served by GRAM pre-Web Services (preWS). The GridWay client is also a machine with the same characteristics.

GridWay was configured using a static information provider. The latter allowed the configuration of the four hosts for GridWay as if having two hundred free slots each managed by a Fork job manager. The performance benefit of using the static information MAD as opposed to a dynamic one is negligible due to the small number of hosts. GridWay scalability for discovering and monitoring is quite good, particularly with the new multithreaded MAD for the Monitoring and Discovery Service of the Globus Toolkit 4.0 (MDS4). Moreover, GridWay implements throttling techniques to avoid saturation that could occur if all the known hosts were to be monitored simultaneously.

The GridWay instance was configured so it submits jobs at the following rate:

- In each iteration, the scheduler submits a maximum of 60 jobs (DISPATCH_CHUNK = 60).
- Each iteration of the scheduler happens with a 3 min interval (SCHEDULING_INTERVAL = 180).
- There is no limit on how many jobs per user can be running at the same time (MAX_RUNNING_USER = 0).
- There is no limit on the maximum number of jobs that the scheduler submits to a given resource (MAX_RUNNING_RESOURCE = 0), so ideally all the two hundred slots would be used per host.

Jobs being sent in this experiment consist basically of a task sleeping from 1800 to 5400 min. Each job gets its arguments according to a piecewise linear function, defined in Eq. (1)

$$f(i) = 1800 + (20i)\%(5400 - 1800), \tag{1}$$

where i is the job index given by GridWay, ranging from 0 to 9999.

This task does not make use of the CPU, since we are not interested in measuring a computer's saturation but rather GridWay's. The file staging carried by each job involves the creation of a directory in the prolog state and the removal of that directory in the epilog state, and also two set of files, which amount to 12 KB for the transfer of input files and 16 KB for the transfer of output files.

The first interesting result of the test is the time that GridWay needs to accept, sequentially, 10,000 jobs. The average time in seconds for this is 226 s, i.e., a bit less than four minutes. This contrasts strongly with the 2 h needed by the gLite Workload Management System (WMS) to accept 500 jobs [14].

We can see in Fig. 2 the disk consumption of GridWay. For ten thousand *completed* jobs the disk consumption gets to 660,084 KB, and that gives an average of approximately 66 KB per job. It is worth noting that this number includes all the logging that GridWay is capable of producing, including information for debugging, the actual production value being significantly less.

Fig. 3 is more revealing as it shows the percentage of memory consumption for all GridWay components and the total sum between all of them. We can see how the Transfer MAD gets most of the memory, but it is important to note how the GridWay daemon and the scheduler are less demanding in memory consumption. Two sudden increases in Transfer MAD's memory consumption can be seen around minutes 300 and 600, probably due to memory leaks triggered by the handling of job errors.

Fig. 4 shows the CPU consumption. Here we can see a similar trend, where the MADs take the most CPU time and the GridWay daemon and the scheduler do not take that much. It is worth noting how the scheduler CPU consumption reduces when there are fewer pending jobs to be scheduled. It is therefore clear from these two last figures that GridWay's CPU and memory consumption are particularly low, if we take into account that it is managing 10,000 jobs.

Fig. 5 depicts how the jobs get themselves from the pending to the done state, maintaining an acceptable average rate of 800 jobs running. It is interesting to see how there are two backward movements in the number of completed tasks around minutes 300 and 600, matching the CPU and memory consumption increase of the transfer MAD. This is coherent with the interpretation of error conditions around those times (the increase of pending jobs indicates that running jobs did fail), and shows the robustness of GridWay in how it managed to complete all the jobs successfully.

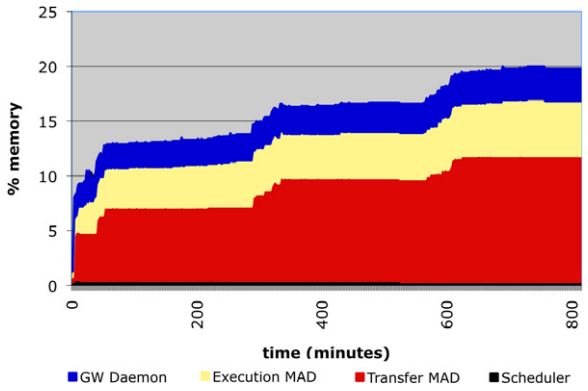


Fig. 3. Memory consumption by GridWay split across its different components.

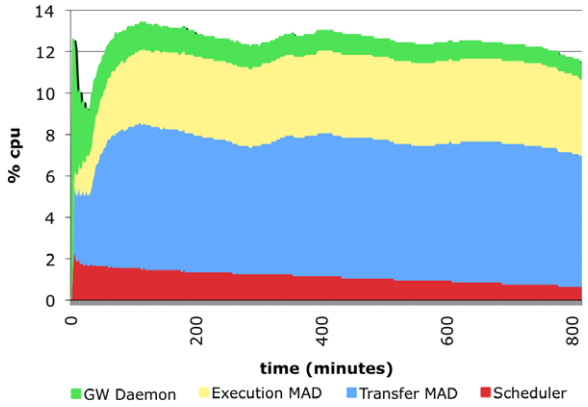


Fig. 4. CPU consumption by GridWay's components.

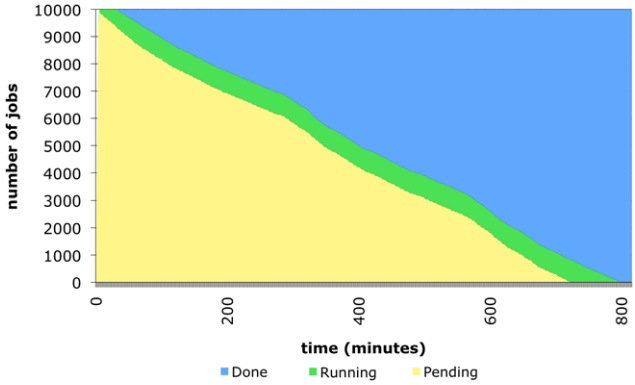


Fig. 5. Job states distribution against time.

5. Experiments: interoperation

In order to demonstrate the feasibility of both the adapters and the gateway solutions for interoperation, two different scenarios were deployed, and jobs were sent using GridWay to a set of resources gathered from three major grid infrastructures: Open Science Grid, TeraGrid and EGEE. Additionally, resources from a local UCM cluster were used as well. Three iterations of two hundred jobs each were submitted in each of the two scenarios in order to show their feasibility. For both these experiments we used the Embarrassingly Distributed (ED) benchmark from the NAS Grid Benchmark suite [15].

Fig. 6 depicts the architecture of both experiment interfaces. The adapters scenario presents a flat line-up of the available resources, all of them accessed by different adapters (dotted arrows represent pre Web Services (preWS) interfaces, solid arrows represent Web Services (WS) interfaces). On the other hand, the gateway scenario distributes resources in two layers, separating resources by their accessing adapters: all resources in the upper level are accessed using common interfaces, as are those on the lower level, albeit this time the interfaces are different. The GridWay configuration needed to set up the scenarios and the results obtained are presented in the following two subsections.

5.1. Adapters scenario

To address the adapters scenario, the GridWay metascheduler was configured to access four different infrastructures. Not only different interfaces were the problem, but also different versions of the same middleware posed their own issues. Furthermore, different configurations of even the same version of the same middleware stacks are troublesome for the correct interoperation of the four infrastructures.

GridWay's adapters are the Middleware Access Drivers (MADs), enabling the metascheduler to access different infrastructures simultaneously and seamlessly. GridWay has evolved over time to take advantage of the adapters technique; it is interesting to see its evolution, from a first tentative effort to harness both EGEE and the IRISGrid infrastructures [16], to the seminal work that produced its full blown current modular architecture [17], motivated by the interoperation between grid resource management services provided by Globus. There are three types of adapters: execution, transfer and information MADs. Details about them and their configuration follow:

- *Open Science Grid*: This infrastructure offers two versions of the Globus Toolkit, deployed using VDT.[5] The particularity of the OSG configuration-wise with respect to a standard Globus install is that it uses the non standard 9443 port to offer the WS created by the Globus container, so GridWay MADs have to be configured accordingly.[6]
- *TeraGrid*: Again, two versions of Globus are offered. Configuration for the file transfer was different than in other infrastructures since the Storage Element (SE) was a separate machine (sharing homes with the Computing Element), thus the SE_HOSTNAME attribute of GridWay was used to direct staging to the SE node.[7]
- *EGEE*: This infrastructure uses the gLite middleware stack, which is based on Globus preWS. The preWS MADs are used, but there are a few particularities of the EGEE that requires a different method of file staging. Mainly, worker nodes of EGEE do not share their home folders with the front-end. Instead, each has an outbound connection. Thus, the weight of the file staging is left to the wrapper script, which is in charge of staging the files and executing the job.[8] An information MAD for the Berkeley Database Information Index (BDII) (which is basically a LDAP server running on port 2170) using the GLUE scheme is used for host monitoring.
- *UCM:* This is dsa-research.org group local cluster at Universidad Complutense de Madrid. Probably due to the fact that the Globus WS MADs were developed against this very cluster, no extra configuration was needed for GridWay to access it.

A testbed was prepared in order to perform the experiments in the adapters scenario. A demonstration in TeraGrid07 featured GridWay accessing several resources from the three infrastructures and resources from UCM, as shown in Fig. 7. For illustrative purposes, one resource from each infrastructure was chosen for the

[5] http://vdt.cs.wisc.edu/.
[6] http://www.gridway.org/documentation/howtos/osghowto.pdf.
[7] http://www.gridway.org/documentation/howtos/tghowto.pdf.
[8] http://www.gridway.org/documentation/howtos/egeehowto.pdf.

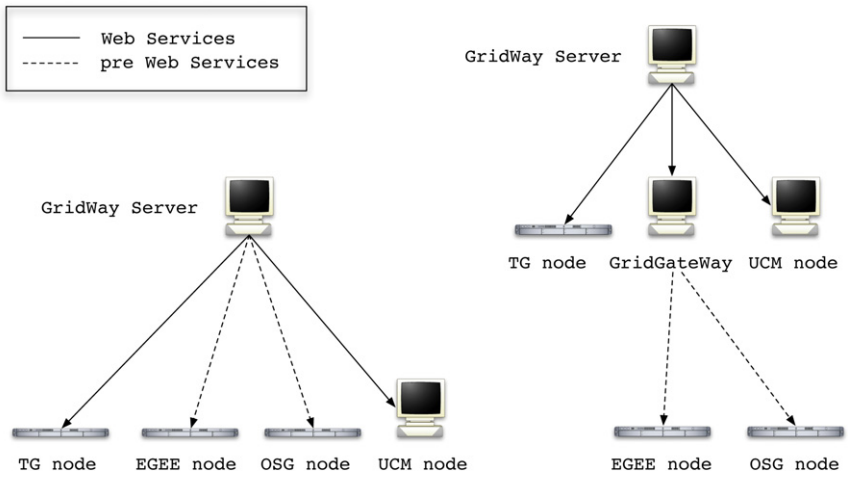


Fig. 6. Adapters scenario vs. gateway scenario.

```
HID PRI OS              ARCH    MHZ %CPU  MEM(F/T)    DISK(F/T)  N(U/F/T) LRMS             HOSTNAME
0   20  Linux2.4.21-32. i686  2665  189  964/2006  62787/73964     0/2/1 jobmanager-fork   atlas.dpcc.uta.edu
1   20  Linux2.4.21-37. athlo 1991  110 1214/2009  62680/71679     0/1/1 jobmanager-condor ce01.cmsaf.mit.edu
2   20  Linux2.4.21-20. i686  2665  388 1246/2006  70846/99795     0/1/1 jobmanager-condor fiupg.ampath.net
3   20  Linux2.4.27-1-3       1595   95   381/884    6350/7525     0/1/1 jobmanager-condor grid.physics.purdue.edu
4   20  Linux2.6.9-42.0 x86_6 2592  200 3041/3901  12040/13770     0/1/1 jobmanager-condor osg.hpcc.nd.edu
5   75                                                        0/89/300 PBS             tg-grid.uc.teragrid.org
6   1   ScientificSLSL  i686  1200    0 1024/1024          0/0   0/51/75 jobmanager-pbs    lcg2ce.ific.uv.es
7   1   ScientificSL4   i686   866    0   513/513          0/0   0/22/22 jobmanager-lcgpbs ramses.dsic.upv.es
8   1   ScientificSL4   i686  2800    0 1024/1024          0/0 0/152/158 jobmanager-lcgpbs lcg-ce.usc.cesga.es
9   1   ScientificSLBer i686  3000    0 1024/1024          0/0  0/49/108 jobmanager-lcgpbs ce2.egee.cesga.es
10  1   ScientificSLBer i686  4000    0 1024/1024          0/0    0/0/24 jobmanager-lcgpbs ifaece01.pic.es
11  25                                                    0/214/219 Condor           nest.phys.uwm.edu
12  25                                                      0/6/11 Condor            osg-itb.ligo.caltech.edu
13  90  Linux2.6.17-2-6  x86  3216    0   45/2027 70824/118812     0/0/2 Fork             cygnus.dacya.ucm.es
14  90  Linux2.6.17-2-6  x86  3216  181  681/2027  98861/11881     0/2/2 Fork             draco.dacya.ucm.es
15  90  Linux2.6.18-4-a x86_6 2211  100  954/1003  77081/77844     0/3/4 PBS              hydrus.dacya.ucm.es
16  90  Linux2.6.18-4-a x86_6 2211  100  776/1003  76428/77844     0/5/5 SGE              aquila.dacya.ucm.es
```

Fig. 7. Resources accessed by GridWay in the Teragrid07 demo.

```
HID PRI  OS              ARCH    MHZ %CPU MEM(F/T)   DISK(F/T)  N(U/F/T)  LRMS             HOSTNAME
0   1                                                    0/5/5  PBS              tg-grid.uc.teragrid.org
1   1    Linux2.4.21-32. i686  2665  189 964/2006 62787/73964   0/10/10  jobmanager-condor cmsgrid01.hep.wisc.edu
2   1    Linux2.6.24-17  x86_6 1995  100   18/499 10169/21817     0/5/5  SGE              aquila.dacya.ucm.es
3   1    ScientificSL4   i686   866    0  513/513         0/0   0/21/22  jobmanager-lcgpbs ramses.dsic.upv.es
```

Fig. 8. Resources for the adapters scenario, as provided by the `gwhost` command.

experiments reported in this paper, so they could be performed in a more controlled environment. A complete resource listing for the adapters scenario can be seen in Fig. 8. To show different adapters in action, WS MADs were used to access the TeraGrid (with Host Identifier, HID, 0 in the figure) and the UCM (HID 2) infrastructure, while the preWS MADs were chosen to access the Open Science Grid (HID 1) and the EGEE (HID 3) infrastructure. GridWay is able to use MADs from toolkits of both version 4.0.x and 4.2.x of the Globus Toolkit, neatly solving the incompatibility between them and allowing a smooth upgrading process.

Fig. 9 shows the number of jobs against the infrastructures where the jobs were executed. There is an approximately even distribution between our local cluster (*UCM*), the Open Science Grid preWS resource (*OSG preWS*) and the TeraGrid WS one (*TG WS*), while the *EGEE* resource shows a smaller ratio of jobs completed. This is due to the chosen EGEE site for the EGEE, ramses.dsic.upv.es, having lower computing power (both CPU and memory) than the resource chosen for this experiment in the other infrastructures.

Average suspension times can be seen in Table 1, where suspension time is the time a job spends waiting on the remote queue to be executed, and they show in this case the dynamic nature of the grid. Since resources are not completely devoted to our experiment, we thus experience disparity of the measured times between iterations. Here we can see the *OSG preWS* resource rendering different suspension times and how they directly affect its amount of taken jobs. From the comparatively lower times shown by the EGEE resource we can deduce that it is probably less occupied than the

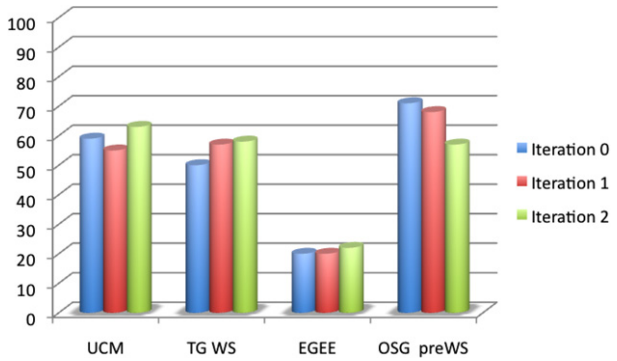


Fig. 9. Job distribution across infrastructures: adapters scenario.

Table 1
Average suspension times as experienced by jobs from adapters scenario.

Iteration #	UCM	TG WS	EGEE	OSG preWS
1	208.59	35.84	86.05	90.47
2	218.43	16.4	85.65	107.16
3	225.52	16.39	84.14	194.85

```
HID PRI OS              ARCH    MHZ %CPU  MEM(F/T)   DISK(F/T) N(U/F/T) LRMS  HOSTNAME
0   1                                                  0/5/5 PBS   tg-grid.uc.teragrid.org
1   1   Linux2.6.24-17  x86_6 1995  100    18/499 10169/21817    0/5/5 SGE   aquila.dacya.ucm.es
2   1                                                0/26/32 GW    cepheus.dacya.ucm.es
```

Fig. 10. Resources for gateway scenario upper level, as seen by `gwhost` in the GridWay server.

```
HID PRI OS              ARCH   MHZ %CPU  MEM(F/T)   DISK(F/T) N(U/F/T) LRMS             HOSTNAME
0   1   Linux2.4.21-32. i686  2665  189  964/2006 62787/73964  0/10/10 jobmanager-condor cmsgrid01.hep.wisc.edu
1   1   ScientificSL4   i686   866    0   513/513         0/0  0/16/22 jobmanager-lcgpbs ramses.dsic.upv.es
```

Fig. 11. Resources for gateway scenario lower level, as seen by `gwhost` in *cepheus*.

Table 2
Productivity achieved (total and per resource) in the adapters scenario.

Iteration #	Total	UCM	TG WS	EGEE	OSG preWS
1	125.78	37.10	31.44	12.57	44.65
2	144.92	39.85	41.30	14.49	49.27
3	122.7	38.65	35.58	13.49	34.96

OSG preWS resource, but its comparative slowness prevents it from increasing the ratio of completed jobs.

Productivity in Table 2 measures the amount of jobs taken by each infrastructure per hour. As expected, the *EGEE* resource shows a much lower productivity, again due to the lower performance of the chosen resource, while the other three infrastructures show more even productivity. It is interesting to see how the suspension time affects the productivity in the case of the *OSG preWS* resource, and in turn affects the amount of jobs taken shown in Fig. 9. This can be explained if we take into account that an increase in the suspension time of a particular resource causes its designated jobs to wait more time in the queue, hence the productivity decreases (less jobs completed by the hour) and thus the amount of completed jobs also decreases.

5.2. Gateway scenario

The gateway scenario is configured in a hierarchical manner. In the upper level, GridWay accesses resources for the TeraGrid and the UCM local cluster. Configuration details for these two infrastructures are identical as in the adapters scenario. A special case is *cepheus*, a resource encapsulating another instance of GridWay through a preWS or WS GRAM interface. This is what we call a GridGateWay [18]. Resources for this upper level can be seen in Fig. 10. Information corresponding to *cepheus* only accounts for the aggregation of free and used nodes of all the resources being encapsulated in the lower level.

At a lower level we have an encapsulated GridWay accessing both the Open Science Grid and EGEE infrastructures. Both of them are accessed using preWS, but offered to the GridWay in the upper level as a WS GRAM interface. For resources on this level, refer to Fig. 11.

With this setup, we achieve a common interfaces approach for the upper level, where both TeraGrid and the local UCM cluster are accessed through the same WS interfaces (albeit with different configurations and versions for each of them). Also, at this same level, a GridGateWay is accessed using the same WS interfaces. At the lower level we find that the encapsulated infrastructures are both accessed using different adapters for preWS interfaces than those of the upper level. In this way, we show the three forms of interoperation in the same scenario: common interfaces in each level, gateways to join both levels and adapters to address the two sets of common interfaces.

Fig. 12 provides a visual representation of job distribution in the two levels across the three iterations. The infrastructure labeled as *GGW* is really the server running the GridGateWay, so all its jobs are really being forwarded to one of *EGEE* or *OSG preWS* resources. Therefore the middle column of the figure offers a compound view of the *OSG preWS* and *EGEE* resources that the gateway

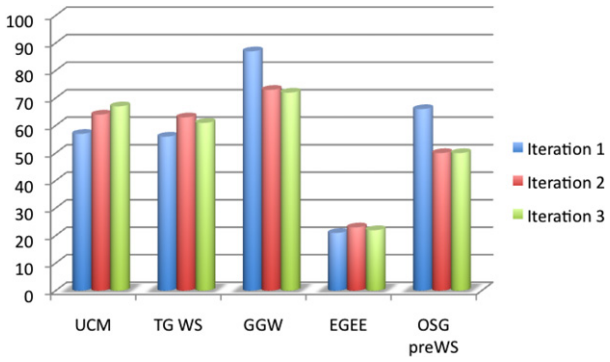


Fig. 12. Job distribution across infrastructures: gateway scenario.

Table 3
Average suspension times as experienced by jobs from the gateway scenario.

Iteration #	UCM	TG WS	EGEE	OSG preWS
1	220.28	18.73	84.85	20.62
2	215.09	17.76	82.69	39.2
3	201.44	16.32	88.95	38.26

Table 4
Productivity achieved (total and per resource) in the gateway scenario.

Iteration #	Total	UCM	TG WS	EGEE	OSG preWS
1	132.45	37.75	37.08	13.90	43.70
2	126.58	40.51	39.87	14.55	31.64
3	121.21	40.61	36.97	13.33	30.30

encapsulates. We can still see the expected short amount of jobs taken by the *EGEE* infrastructure and an almost even distribution among the other three. We draw attention to the slight increase in jobs taken by both *UCM* and *TG WS* resources with respect to the adapters scenario, since the overhead caused by the GridGateWay increased the jobs diverted to the two upper level infrastructures.

Average suspension times can be seen in Table 3. The *OSG preWS* resource shows a much lower suspension time than in the adapters scenario, and it can be seen how this affects the amount of jobs taken by this infrastructure between iterations. This can be read as less jobs completed whenever they have to wait more in the queue. The other three resources show much more even values, whether comparing between iterations or even between scenarios.

Table 4 shows the productivity for the resources in this scenario. It is a good reflection of the effect caused by the overhead of the gateway, since we can see a slight decrease in the productivity of the encapsulated *OSG preWS* resource, if we compare it with the productivity measured in the previous scenario, even considering the notable decrease in the suspension time. The *EGEE* resource apparently is not affected by this overhead, since its productivity is similar across both scenarios. The most likely explanation is that the reduced performance imposes a stronger condition on the productivity so that the overhead is not shown. The decrease in productivity of the encapsulated resources shows the overhead caused by the gateway and hence the penalty imposed by the use of a hierarchical structure in federation. There is no overall decrease

in productivity due to the better behavior of the *UCM* and *TG WS* resources, which are directly accessed.

6. Conclusions and future work

With this work we have shown that a metascheduler can be used as a tool for federation. Although of course it does not solve all the problems alone, it helps to model a solution, becoming an important component to achieve the desired federation. It was argued in this paper that interoperability is not a sufficient condition to produce federation, but other issues have to be taken into account. One of the key conditions of the federation solution has to be its scalability.

We claim that the GridWay metascheduler is a valuable tool for building different types of grid infrastructures. In order to show that our proposed solutions are fit for federation and therefore to sustain our claim, scalability of the metascheduler had to be tested. As a conclusion of that study, GridWay showed great stability, robustness and scalability during this test; where it scheduled 10,000 jobs and kept track of them, showing remarkable robustness and responsiveness. Perhaps the best indicator of this assertion is the fact that it managed to admit 10,000 jobs in the pending state in less than four minutes. However, there is work to be done in the transfer MAD, to prevent the memory leaks observed in the scalability test.

As future work in this direction, we want to keep GridWay's value as a tool for federation. Hence, there are plans to develop new MADs for other infrastructures, more concretely, MADs for UNICORE and for CREAM (the new gLite lightweight service for resource management) are already being developed while also plans for new and more complete scalability and robustness tests are being prepared. New policies for job scheduling are also areas to be explored, to take into account the different topologies of federation described in this paper. In a similar line, the ability to negotiate SLAs in order to automatically acquire resources from an infrastructure provider is being explored, and integration with existing SLA components is currently being study. Another field where is room for improvement is the GridGateWay, for example by reducing its latency to minimize the overhead caused and to be able to improve the productivity achieved by the resources being encapsulated by this component.

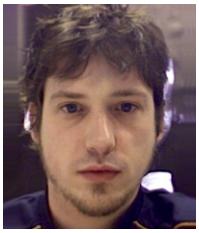
Acknowledgements

This work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-222667) through the Seventh Framework Programme. EGEE provides a seamless Grid infrastructure available to the European research community 24 h a day. Full information is available at http://www.eu-egee.org.

Also, we are grateful for resources lent by both the TeraGrid and the Open Science Grid for this experiment.

References

[1] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, The International Journal of High Performance Computing Applications 15 (3) (2001) 200–222.
[2] I. Foster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., 1999.
[3] I. Foster, What is the Grid? A Three Point Checklist, GRIDToday 1 (6) (2002).
[4] C. Smith, T. Kielmann, S. Newhouse, M. Humphrey, The HPC basic profile and SAGA: standardizing compute grid access in the open grid forum, Concurrency and Computation: Practice and Experience 21 (8) (2009) 1053–1068.
[5] M. Riedel, et al., Interoperation of world-wide production e-science infrastructures, Concurrency and Computation: Practice and Experience Journal (2008) OGF Special Issue.
[6] D. Breuer, P. Wieder, S. van den Berghe, G. von Laszewski, J. MacLaren, D. Nicole, H.C. Hoppe, A UNICORE-globus interoperability layer, Computing and Informatics 21 (2002) 399–411.
[7] P. Kacsuk, T. Kiss, G. Sipos, Solving the grid interoperability problem by P-GRADE portal at workflow level, Future Generation Computer Systems 24 (7) (2008) 744–751.
[8] M.D. Assunção, R. Buyya, S. Venugopal, Intergrid: a case for internetworking islands of grids, Concurrency and Computation: Practice and Experience 20 (8) (2008) 997–1024.
[9] A. Agarwal, et al., GridX1: a Canadian computational grid, Future Generation Computing Systems 23 (5) (2007) 680–687.
[10] A. Kertesz, P. Kacsuk, Grid meta-broker architecture: towards an interoperable grid resource, in: CoreGRID Workshop on Grid Middleware in Conjunction with Euro-Par 2006, 2006.
[11] N. Bobroff, L. Fong, S. Kalayci, Y. Liu, J.C. Martinez, I. Rodero, S.M. Sadjadi, D. Villegas, Enabling interoperability among meta-schedulers, in: Proceedings of 8th IEEE International Symposium on Cluster Computing and the Grid, CCGrid-2008, 2008, pp. 306–315.
[12] E. Huedo, R.S. Montero, I.M. Llorente, A framework for adaptive execution on grids, Software—Practice and Experience 34 (7) (2004) 631–651.
[13] L. Field, Getting grids to work together, CERN Computer Newsletter 41 (5) (2006) 8–9.
[14] I. Sfiligoi, B. Holzman, Workload management systems evaluation and integration, Tech. rep., Fermilab, March 2007.
[15] M.A. Frumkin, R.F. Van der Wijngaart, NAS grid benchmarks: a tool for grid space exploration, Journal of Cluster Computing 5 (3) (2002) 247–255.
[16] J.L. Vázquez-Poletti, E. Huedo, R.S. Montero, I.M. Llorente, Coordinated harnessing of the IRISGrid and EGEE testbeds with gridway, Journal of Parallel and Distributed Computing 5 (65) (2005) 763–771.
[17] E. Huedo, R.S. Montero, I.M. Llorente, A modular meta-scheduling architecture for interfacing with pre-WS and WS grid resource management services, Future Generation Computing Systems 23 (2) (2007) 252–261.
[18] C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente, Evaluation of a utility computing model based on federation of grid infrastructures, in: 13th International Euro-Par Conference, in: Lecture Notes in Computer Science (LNCS), vol. 4641, 2007, pp. 372–381.



Constantino Vázquez received his M.E. in Computer Science (2002) from the Universidad Complutense de Madrid (UCM) and M.Sc. in Computing & Internet Systems (2003) from King's College, London. He is currently doing his Ph.D. in Computer Architecture and is part of the Distributed Systems Architecture Group at UCM. He is involved in several open source projects such as the GridWay metascheduler and the OpenNebula virtual infrastructure engine. His research interests include Grid federation, Utility Computing and Virtualization.

Eduardo Huedo received his M.E. in Computer Science (1999) and Ph.D. in Computer Architecture (2004) from the Universidad Complutense de Madrid (UCM). He is an Associate Professor of Computer Architecture and Technology at UCM. Previously, he worked as a Postdoctoral Researcher at the Advanced Computing Laboratory at Centro de Astrobiología (CSIC-INTA), associated to the NASA Astrobiology Institute. His research areas are Performance Management and Tuning, Parallel and Distributed Computing and Grid and Virtualization Technology.

Rubén S. Montero, Ph.D. is an associate professor in the Department of Computer Architecture at Universidad Complutense de Madrid. He has published more than 70 scientific papers in the field of High-Performance Parallel and Distributed Computing, and contributed to more than 20 research and development programmes. His research interests lie mainly in resource provisioning models for distributed systems: Grid resource management and scheduling, distributed management of virtual machines and cloud computing. He is also actively involved in several open source grid initiatives such as the Globus Toolkit and the GridWay metascheduler. Currently, he is co-leading the research and development activities of OpenNebula, a distributed virtual machine manager.

Ignacio M. Llorente has a graduate degree in Physics (B.S. in Physics and M.S. in Computer Science), a Ph.D. in Physics (Program in Computer Science) and an Executive Master in Business Administration. He has about 15 yrs of research experience in the field of High-Performance Parallel and Distributed Computing. Currently, he is a Full Professor in Computer Architecture and Technology at Universidad Complutense de Madrid, where he leads the Distributed Systems Architecture Research Group. His research interests are Service Oriented Architectures and Infrastructures; Grid, Cloud and Virtualization Technologies, High Performance Computing and Utility Computing.

A.6. On the Use of Clouds for Grid Resource Provisioning

Cita completa

C. Vázquez, E. Huedo, R.S. Montero and I.M. Llorente. *On the Use of Clouds for Grid Resource Provisioning*. **Future Generation Computer Systems**, 27 (5), 600-605, 2011.

Índice de impacto

Este artículo ha sido publicado en la revista Future Generations Computer Systems, con un índice de impacto de 2,365 en JCR 2010, ocupando el primer tercio (10/97) de la categoría de "Computer Science, Theory and Methods". Ha sido citado 6 veces según Google Scholar.

Future Generation Computer Systems 27 (2011) 600–605

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

On the use of clouds for grid resource provisioning☆

Constantino Vázquez *, Eduardo Huedo, Rubén S. Montero, Ignacio M. Llorente

Departamento de Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense de Madrid, 28040, Spain

ARTICLE INFO

Article history:
Received 26 February 2010
Received in revised form
27 September 2010
Accepted 12 October 2010
Available online 20 October 2010

Keywords:
Grid
Cloud
Adaptable architectures
Automation
Dynamic provision

ABSTRACT

Cloud computing is being built on top of established grid technology concepts. On the other hand, it is also true that cloud computing has much to offer to grid infrastructures. The aim of this paper is to provide the ability to build arbitrary complex grid infrastructures able to sustain the demand required by any given service, taking advantage of the pay-per-use model and the seemingly unlimited capacity of the cloud computing paradigm. It addresses mechanisms that potentially can be used to meet a given quality of service or satisfy peak demands this service may have. These mechanisms imply the elastic growth of the grid infrastructure making use of cloud providers, regardless of whether they are commercial, like Amazon EC2 and GoGrid, or scientific, like Globus Nimbus. This technology of dynamic provisioning is demonstrated in an experiment, aimed to show the overheads caused in the process of offloading jobs to resources created in the cloud.

1. Introduction

Cloud computing was arguably first popularized in 2006 by Amazon's Elastic Compute Cloud, which started offering virtual machines (VMs) for $0.10/h using both a simple web interface and a programmer-friendly API. Although not the first to propose a utility computing model, Amazon EC2 contributed to the popularization of the Infrastructure as a Service (IaaS) paradigm, which became closely tied to the notion of cloud computing. An IaaS cloud enables on-demand provisioning of computational resources, in the form of VMs deployed in a cloud provider's datacenter (such as Amazon's), minimizing or even eliminating associated capital costs for cloud consumers, allowing capacity to be added or removed from their IT infrastructure in order to meet peak or fluctuating service demands, while only paying for the actual capacity used.

Over time, an ecosystem of providers, users, and technologies has coalesced around this IaaS cloud model. More IaaS cloud providers, such as GoGrid, FlexiScale, and ElasticHosts have emerged. A growing number of companies base their IT strategy on cloud-based resources, spending little or no capital to manage their own IT infrastructure. Other providers offer products that facilitate working with IaaS clouds, such as rPath's rBuilder, which allows dynamic creation of software environments to run on a cloud.

In general, an IaaS cloud consists of three main components, namely: a virtualization layer on top of the physical resources including network, storage and compute; the virtual infrastructure manager (VIM) that control and monitor the VMs over the distributed set of physical resources; and a cloud interface that provides the users with a simple abstraction to manage VMs. In recent years a constellation of technologies that provide one or more of these components has emerged. Therefore, a variety of hypervisors have been developed and greatly improved, most notably KVM, Xen and VMWare. Also, several VIM technologies that cover the functionality outlined above have appeared, like Platform VM Orchestrator, VMware DRS, or Ovirt. On the other hand, projects like Globus Nimbus,[1] Eucalyptus[2] or OpenNebula,[3] or products like VMware vSphere, that can be termed cloud toolkits, can be used to transform the existing infrastructure into an IaaS cloud with cloud-like interfaces.

Cloud computing has emerged as a very promising paradigm to simplify and improve the management of current IT infrastructures of any kind and, in particular, grid ones. Clouds, in their IaaS form, have opened up avenues in this area to ease the maintenance, operation and use of grid sites, and to explore new resource sharing

☆ This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through MEDIANET Research Program S2009/TIC-1468, by Ministerio de Ciencia e Innovación, through the research grant TIN2009-07146, and by the European Union through the research grant RESERVOIR Contract Number 215605.

* Corresponding author. Tel.: +34 639150939; fax: +34 913947527.
E-mail addresses: tinova@fdi.ucm.es (C. Vázquez), ehuedo@fdi.ucm.es (E. Huedo), rubensm@dacya.ucm.es (R.S. Montero), llorente@dacya.ucm.es (I.M. Llorente).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.10.003

[1] http://www.nimbusproject.org.
[2] http://open.eucalyptus.com.
[3] http://opennebula.org.

models that could simplify in some cases the porting and development of grid applications. The first works about the joint use of clouds and grids are exploring two main approaches, namely:

- The use of virtual machines and clouds as an effective way to provide grid users with custom execution environments. Therefore the same grid site can easily support Virtual Organizations (VOs) with different (or conflicting) configurations.
- The access to grid resources in a cloud-way. So, the users will access raw computing capacity by-passing the classical grid middleware stack. This approach is also being considered as a natural way to attract business users to our current e-infrastructures.

The problem we are trying to address in this paper is the necessity to attend to fluctuating and peak demands in high performance computing. For instance, a supercomputing center is subject to this need, since projects can demand resources punctually for experiments, and they can do so even simultaneously. Moreover, meeting a Service Level Agreement (SLA) defining a Quality of Service (QoS) with the center's available resources can be challenging at certain times. The logical provisioning model will be to use the local infrastructure to attend to all the existing demand, if possible (i.e., using their enterprise grid). If that is not enough, the excess load can be delegated to a partner grid, with which a previous arrangement has been made. This partner grid does not need to provide the same interface as the enterprise grid, but still a single point of access is desirable to the whole federated infrastructure. If the computing demand still overflows the existing resources, the center will need to use a cloud provider to perform a temporary increase in its computing power. A unified point of access is even valuable as more heterogenous resources are added to the grid infrastructure.

In this paper we will show a framework for monitoring service capacity and for growing grid infrastructures when it comes close to saturation, using cloud providers like Amazon EC2[4] and GoGrid[5] (commercial) or Globus Nimbus[6] (scientific); and we will show empirical data on an experiment showing this dynamic growth. This will provide the necessary components to build a grid infrastructure with a single point of access that can be adapted as in the aforementioned thought experiment of the supercomputing center. One interesting characteristic of using the virtualization that conform clouds is the ability to provide resources with certain characteristics, that is, particular software libraries or specific configurations can be asked for in the requested virtual machines (VMs) to better fulfill the demands of the grid infrastructure.

The aim of this paper is to contribute with a solution to the challenge of the dynamic provision of resources using cloud providers. The structure of this paper is as follows. Section 2 references work related to this paper, while Section 3 unfolds a general architecture of the solution for the desired adapting grid infrastructure. Section 4 presents a solution to dynamically grow an existing grid infrastructure using resources from cloud providers. Finally, Section 5 states plans for future work and summarizes the conclusions of this paper.

2. Related work

In the last decade we have witnessed the consolidation of several transcontinental grid infrastructures that have achieved unseen levels of resource sharing. In spite of this success, current grids suffer from several obstacles that limit their efficiency, namely:

- An increase in the cost and length of the application development and porting cycle. New applications have to be tested in a great variety of environments where the developers have limited configuration capabilities.
- A limitation on the effective number of resources available to each application. Usually different VOs require different software configurations, so an application can only be executed on those sites that support the associated VO. Moreover, the resources devoted to each VO within a site are usually static and cannot be adapted to the VO's workload.
- An increase in the operational cost of the infrastructure. The deployment, maintenance and distribution of different configurations requires specialized, time consuming and error prone procedures. Even worse, new organizations joining a grid infrastructure need to install and configure an ever-growing middleware stack.

This situation often leads to a struggle between the users, who need more control of their execution environments, and grid operators, who want to limit the heterogeneity of the infrastructure. As a result several alternatives to reconcile both positions have been explored in the past. For example, the SoftEnv project[7] is a software environment configuration system that allows the users to define the applications and libraries they need. Another common solution is to use a custom software stack on top of the existing middleware layer, usually referred to as pilot-jobs. For example, the MyCluster Project [1] creates a Condor or Sun Grid Engine (SGE) cluster on top of TeraGrid services; and similarly over other middleware we may cite: DIRAC [2], glideinWMS [3], PanDa[8] or the Falkon system [4]. These approaches essentially shift the scalability issues from the application to the overlaid software layer, whereas the proposed solution transparently scales both the application and the computational cluster.

The idea of a virtual cluster which dynamically adapts its size to the workload is not new. It can seen being applied for instance in the cluster management software called COD (Cluster On Demand) [5], which dynamically allocates servers from a common pool to multiple virtual clusters. Although the goal is similar, the approach is completely different. COD worker nodes employ NFS to mount different software configurations. Similarly, the VIOcluster [6] project enables dynamically adjusting the capacity of a computing cluster by sharing resources between peer domains.

Another area explored nowadays is the use of the existing grid infrastructure to build autonomic clouds [7]. This is achieved by creating virtual environments, with their associated benefits like flexibility and adaptation, across multiple physical domains, using a decentralized private overlay network system called IPOP (IP over P2P). In this fashion, a massive number of grid infrastructure could be turned into adaptable clouds in a non-disruptive manner for the existing grid services, although they will be suitable for serving high throughput, high latency tolerant services.

However the most promising technology to provide VO with custom execution environments is virtualization. The dramatic performance improvements in hypervisor technologies made it possible to experiment with virtual machines (VM) as basic building blocks for computational platforms. Several studies [8,9] reveal that the virtualization layer has no significant impact on the performance of memory and CPU-intensive applications for HPC clusters. The first works in this area integrated resource management systems with VMs to provide custom execution environments on a per-job basis. For example Dynamic Virtual Clustering [10] and

4 http://www.amazon.com/ec2.
5 http://www.gogrid.com.
6 http://workspace.globus.org.
7 http://www.teragrid.org/userinfo/softenv.
8 https://twiki.cern.ch/twiki/bin/view/Atlas/Panda.

XGE [11] for MOAB and SGE job managers respectively. These approaches only overcome the configuration limitation of physical resources because VMs are bounded to a given resource and only exist during job execution. A similar approach has been implemented [12] at the grid level using the Globus GridWay Metascheduler. GridWay allows the definition of an optional phase before the actual execution phase to perform advanced job configuration routines. In this phase, the availability of the requested VM image in the cluster node is checked, transferring it from a GridFTP repository if needed. Then, in the execution phase, a script starts or restores the VM on a worker node and waits for its activation by periodically probing its services, and executes the user program after the VM is ready. The program copies all the input files needed to the VM, and executes the user program. This strategy does not require additional middleware to be deployed and is not tied to a given virtualization technology. However, since the underlying local resource management system is not aware of the nature of the job itself, some of the potential benefits offered by the virtualization technology (e.g. server consolidation) are not fully exploited.

More general approaches involve the use of virtual machines as workload units, which implies a change in paradigm from building grids out of physical resources to virtualized ones. For example, the VIOLIN [13] project proposes a novel alternative to application-level overlays based on virtual and isolated networks created on top of an overlay infrastructure. Also the VMPlant service [14] provides the automated configuration and creation of VMs that can be subsequently cloned and instantiated to provide homogeneous execution environments across distributed grid resources. The InterGrid system uses as well VMs as building blocks to construct execution environments that span multiple computing sites [15]. Such environments can be created by deploying VMs on different types of resources, like local data centers, grid infrastructures or cloud providers. InterGrid uses OpenNebula as a component for deploying VMs on a local infrastructure. On the other hand, the In-VIGO [16] project adds some virtualization layers to the classical grid model, to enable the creation of dynamic pools of virtual resources for application-specific grid computing. Also in this line of work, several studies have explored the use of virtual machines to provide custom (VO-specific) cluster environments for grid computing. In this case, the clusters are usually completely build up of virtualized resources, as in the Globus Nimbus project [17], or the Virtual Organization Clusters (VOC) [18]. The latter allows us, by means of real-time monitoring of LRMS queues, to launch VO-specific instances of Virtual Machines in order to serve the jobs belonging to the Virtual Organization that is in need of computing power. Once the jobs are completed, the Virtual Machines are powered down so the physical resources can be re-claimed. Aligned with this research direction, our research group has explored a hybrid model, so the cluster combines physical, virtualized and cloud resources [19].

It is important to note that the previous works also highlight that the use of virtualization in grid environments can greatly improve the efficiency, flexibility and sustainability of current production grids. Not only by extending the classical benefits of VMs for constructing clusters, e.g. consolidation or rapid provisioning of resources [20,21]; but also grid-specific benefits, e.g. support to multiple VOs, isolation of workloads and the encapsulation of services. Some EU projects like StratusLab[9] are studying a cloud-like provisioning model for grid-sites. In this way, a grid site exposes a typical cloud interface to instantiate and control virtual machines. This would allow accessing grid computing resources as cloud providers, complementing the existing grid middleware since the aim is for the cloud layer to be fully transparent to the layers above, and thus effectively taking benefit from the adaptability of service provisioning given by clouds applied to the batch processing goodness of grid infrastructures.

Our approach uses the concept of a dynamically adapting grid infrastructure, but more aligned with the concept of a Service Manager like Hedeby,[10] although using VMs with specific configurations rather than configuring physical servers. In this aspect, our solution is similar to that provided by RightScale,[11] but it differs from it since it is not a completely virtualized solution, but rather a way to extend a physical infrastructure by the punctual use of virtualized resources.

Not just the ability to fire VMs is enough for grid cloudbursting, also crucial is the ability of the grid infrastructure to federate with other resources. For this, we used the GridWay metascheduler to architect this solution. It features a modular design, with adapters (Middleware Access Drivers in GridWay speak) that enables interoperation of different grid middleware stacks through the metascheduler, therefore making federation of grid resources feasible. This is explored in the work presented in [22], where adapters are used to access simultaneously grid infrastructures exposing different flavors of the globus grid middleware (pre-Web Services and Web Services). Moreover, it can integrate resources being run by a local resource management system (LRMS) to the existing grid infrastructure cluster or it even has the potential to integrate with single computers via SSH.

3. Architecture

Dynamic provisioning poses various problems. One problem in this field is the importance of interoperability, i.e., being able to grow using any type of given resource, independently of what an interface may be offering. Another problem is answering the question of when this dynamic growth is necessary and how to actually perform it. A third issue can be the enforcement of a budget on this decision, taking into account expected CPU and network usage.

In this section, these problems are addressed by means of a grid infrastructure that can be flexibly built, that is aware of its load, featuring a single point of access and that can incorporate a new resource temporarily in an automatic fashion to satisfy heavy demands. Fig. 1 sketches an architecture of such a solution. We can see that one of the building blocks is the GridWay metascheduler.

The flexible architecture of this metascheduler allows the use of adapters. Although sharing the same interfaces is the ideal way to achieve interoperation, sometimes there are different middlewares deployed in the sites to be federated, and it is unfeasible to unify them for a variety of reasons (politics, time constraints or ongoing migration or upgrades). This can be seen as the consequence of not having an accepted standard, and therefore, lack of interoperability. One possible solution to federate these different sites is to build a portal that uses different components (to submit jobs, gather information, transfer files, etc.) to interface these sites. These components are designed specifically for a particular middleware stack or even version, and we can call them adapters.

GridWay already has a number of adapters (MADs), that enable access to different production grid infrastructures. Moreover, it also provides SSH MADs, so access to local resources can be achieved with decreased overhead, avoiding the need to have installed and configured in the nodes any grid software such as, for instance, the Globus Toolkit.

[9] http://www.stratuslab.eu.

[10] http://hedeby.sunsource.net.

[11] http://www.rightscale.com.

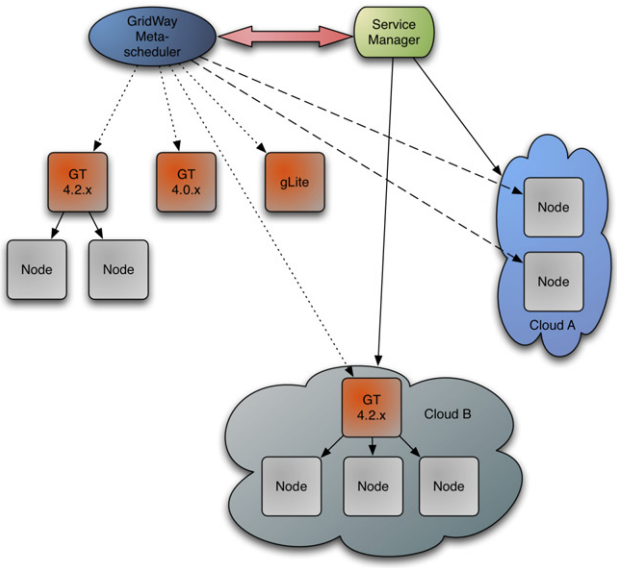


Fig. 1. Architecture for an elastic grid infrastructure.

Having described the federation approach, let's see the proposed architecture for dynamic provisioning, whose principal component is the Service Manager. This component is used to monitor the GridWay metascheduler, and when the load of the system exceeds a threshold, detected using heuristics, it is responsible to grow the available grid infrastructure using specific adapters to access different cloud providers. This growth can be accomplished in two ways. The first one is by requesting a number (calculated with the aid of said heuristics) of single hosts. These need to have a previously defined software configuration that will then help them enroll in the available grid infrastructure. This corresponds to the use of Cloud A in Fig. 1.

Another possibility is to deploy a full virtualized cluster, with a front-end controlling a number of slave nodes. This front-end can then enroll itself in the existing grid infrastructure, adding its capacity. The GridWay metascheduler features mechanisms to dynamically discover new hosts or sites which can be used for this purpose. This corresponds to the use of Cloud B in the figure.

Therefore, the provisioning model we envision is twofold. The first mode adds one single computing resource to the grid infrastructure. This computing resource can be accessible through a GRAM interface, meaning that the VM that is going to be awakened needs to have the Globus Toolkit installed and correctly configured. An even more practical approach will be to use just SSH access to perform job execution in this kind of nodes; the GridWay metascheduler already has a prototype of such SSH drivers. In this way, machines from cloud providers can be used out-of-the-box, with little to no configuration needed; basically SSH access is the only requirement. On the other hand, a second mode of growing the existing grid infrastructure would be to use these cloud providers in a slightly different fashion. Negotiation with the cloud provider will grant access to a virtual cluster, accessible through GRAM and controlled by an LRMS like for example PBS or SGE. This cluster will then be added to the federated grid infrastructure the same way as one of the physical sites we saw in the last section. Future work is planned to enrich the flexibility of the grid infrastructure by removing the GRAM layer, enabling GridWay to access the cluster by talking directly to the LRMS.

Moreover, this solution has another advantage. Not only can it dynamically increase the size of the grid infrastructure, but the added computing nodes can be awakened with different configurations. In other words, the Service Manager can be built in such a way that it won't only concern itself with the need to increase the computing capacity, but it can do so in a service oriented way. If there is one specific service which is suffering from the peak demand, the Service Manager can decide to increase the number of nodes prepared to satisfy such a service. For instance, if the service is an application that requires specific mathematical libraries, virtual machine images containing that specific libraries be chosen to be awakened as nodes to increase the grid infrastructure capacity.

4. Grid resource provisioning using clouds

A technique to guarantee pre-accorded QoS and, therefore, meet SLAs even in cases of high saturation of the grid infrastructure is presented in this section. Also, it gives a solution for peak demands that occur without enough time for planning the extension through federation. This solution involves the elastic growth of the computing infrastructure by means of a cloud provider, being that commercial (Amazon EC2, GoGrid) or scientific (Globus Nimbus), being the charge model is the only difference between these two types of cloud providers.

To enable our grid infrastructure to be able to meet a given QoS and so satisfying predefined SLAs we need a component that is aware of the load of the said infrastructure. Our solution consists of a Service Manager component that monitors the metascheduler in order to find when it should elastically grow (or, conversely, shrink) the available resources by waking up nodes or entire clusters (or shutting them down). In short, this component is responsible for adapting the grid infrastructure to dynamic computing demands.

This solution takes advantage of the chosen GridWay metascheduler. It employs a dynamic scheduling system and therefore it can detect when a new machine has been added or removed from a grid infrastructure, and redistributes the work load. It also features fault detection and recovery capabilities. Transparently to the end user, it is able to detect and recover from any of the grid elements failure, outage or saturation conditions.

The Service Manager is in charge of monitoring the metascheduler and to detect an excess of load for the available resources. In order to detect this excess, a set of heuristics have to be defined, so they define the threshold of the number of pending jobs waiting for resources, and the load present on the available resources. A second set of heuristics is needed to decide which cloud provider is going to be used to elastically grow the cluster. These heuristics should be based on economic criteria, minimizing the total cost of CPU and network usage. In this line, a good starting point would be the work done in budget constrained cost-time optimization algorithms for scheduling [23].

Optionally, there is even a third set of rules that the Service Manager needs to employ in case that it is aware of what service demands need to be satisfied. This rule will be used to decide which type of VM is going to be awakened to satisfy the excess of demand. For instance, in the context of a supercomputing center, VMs with a certain Virtual Organization (VO) configuration can be the ones chosen to be up, if that VO has suddenly increased its demand for computing power.

4.1. Description of the experiment

This experiment is designed to evaluate the overhead incurred in the management of a new worker node in a virtualized cluster being executed in Globus Nimbus. Hence, for the purpose of this experiment, the virtual cluster is already being executed, and we are measuring the overheads between the different layers of our architecture in the process of adding a new worker node, i.e., the overheads caused by awaking the worker node, shutting it down, being it detected by the SGE, the MDS and GridWay, and the processing overhead incurred by this node since it is virtualized.

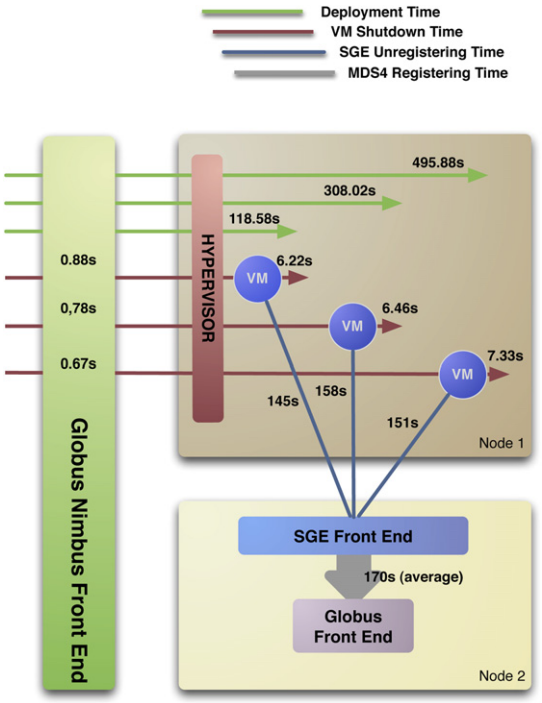


Fig. 2. Grid resource provisioning using cloud overhead experiment.

This experiment is performed on a local cloud deployed in our laboratory at dsa-research.org, using Globus Nimbus as the cloud manager.

In order to better understand the chain of events and where the overheads occur let's see the actions taking place when the Service Manager decides to add a new worker node to the grid infrastructure. A graphical depiction of the experiment, with measured times, can be seen in Fig. 2. First, the Service Manager requests a new VM to Nimbus, which determines the best node to run the virtual machine, based on the resources requested (e.g. memory). Afterwards, if the VM image (appliance) is not local to the host system, it accesses the image server via a suitable protocol (e.g. GridFTP) and obtains a copy. Once the image has been transferred, the physical node's DHCP server configuration file is altered in order to establish VM's IP and hostname. When these operations conclude, the VM is booted. When the VM has been deployed and is running, it registers on the LRMS front-end as an execution host. After a given time, the grid information system (MDS) detects the new node and publishes it. Finally, the GridWay metascheduler will refresh the information of the available grid resources, and detect the new worker node. Then, according to the scheduling policies, it will allocate jobs on this new resource by interfacing with the grid execution service (GRAM). The behavior of the previous deployment strategy will be analyzed on a testbed based on Globus Toolkit 4.0.3 and GridWay 5.2.1.

4.2. Evaluation of overheads

Several experiments were run in the testbed in order to analyze the overheads caused by virtualization and the software layers corresponding to the proposed architecture.

The first overhead considered is caused by the *deployment* of a VM under several conditions. The experiment consisted in the deployment of one, two and three VMs within the same physical machine. Respectively, total times of deployment in seconds were 118.58, 308.02, and 495.88 (*Deployment Time* in the figure). We would like to remark that deploying more than 3 VMs simultaneously resulted in most of the times in error situations. Also, note that the overhead induced by the SGE layer is negligible (the time to register a worker node in the cluster is less than 1% of the total deployment time).

Table 1
Times (in seconds) when a worker node is shut down.

Number	Command received	VM destroyed	SGE	Total
1	0.78	6.22	145	152
2	0.88	6.46	158	165.33
3	0.67	7.33	151	159

In the case of *shutting down* a VM (Table 1), we have measured three relevant values. *Command received* measures the time passed between the request being made to the Globus Nimbus front end and its arrival to the underlying hypervisor. *VM destroyed* expresses the time it takes to completely shut down a VM since the shutdown order arrives to the hypervisor. Both times are consolidated in figure's *VM Shutdown Time*. In this case the time is constant regardless of the number of VMs being shut down. Overhead introduced by SGE when shutting down a VM (*SGE Unregistering Time* in the figure) can be minimized by reducing the polling time. Default value is 300 s, so it takes an average of 150 s to detect the new situation. Reducing polling time limits the overhead, although it increments network usage. The system administrator must tune this value according to the number of nodes in the cluster and average VM uptime.

Once the VM is being booted, time is needed until it is enrolled for use in the original grid infrastructure, and complementarily, time is also needed to plug the node out of the infrastructure. These overheads can be called *grid integration* overheads. The time to start a virtual worker node (time since the Service Manager requests a worker node, 114 s, till it is registered in the LRMS, 2 s) is roughly 2 min. The time to register the new slot in the Grid Information System (MDS4) is about 170 s (*MDS4 Registering Time* in the figure). It is worth pointing out that the MDS publishing time is greater than the time employed on deploying one VM plus SGE register time. Therefore, when sequentially deploying several VMs both times overlap, producing an additional time saving. The MDS and GridWay overhead can be limited by adjusting their refresh polling intervals. When shutting down, the same steps are accomplished. In this case, the time until the operation is accomplished at the machine layer is greatly reduced, from 114 to 7 s. However, the time until the LRMS detects the lack of the VM is incremented, from 2 to about 150 s. It is interesting to note that the metascheduler could assign jobs to the cluster during the worker node shutting down time. In this case the metascheduler should be able to re-schedule this job to another resource.

Virtualization technology imposes a performance penalty due to an additional layer between the physical hardware and the guest operating system. This penalty (that we can call the *processing* overhead) depends on the hardware, the virtualization technology and the kind of applications being run. As some studies suggest [24,25], Xen performs extremely well in any kind of tasks, with a performance loss between 1 and 15%. VMware also achieves near-native performance for processor-intensive tasks, but experiences a significant slow-down (up to 88%) on I/O bound tasks. The Nimbus development team measured its performance in a real-world grid use case [26], a climate science application, achieving about a 5% performance loss. Previous results [12,27] indicate that, in general, the virtualization platform has no significant impact on the performance of memory and CPU-intensive applications for HPC clusters.

5. Conclusions and future work

We have shown an architecture to build any type of arbitrary complex grid infrastructures with a single point of access, which is

able to dynamically adapt its size (and therefore, capacity) using a cloud provider to react to peak demands and/or meet SLAs.

This paper opens the path for a number of research lines to be followed and developments to be done. On the one hand, there is room for the development of new adapters for the GridWay to access a greater number of different types of grid infrastructures. On the other hand, the problem posed by the need to schedule workloads across several grid infrastructures needs heuristics to be developed. The said heuristics need to be implemented in the metascheduler, and use information gathered by it to decide which infrastructure to send the job to, taking into account their workload, their hardware characteristics, job execution history, and even user or group quotas.

Moreover, there is also work to be done in the heuristics needed to tune the Service Manager. This component needs to be aware of the characteristics of the service being offered to correctly adjust its capacity in the case of a demand increase. Based on history, the Service Manager could even anticipate a peak demand and readies the service for it, adding resources to the grid infrastructure. It is also crucial that the heuristics work upon an economic model that knows different cloud provider characteristics to reach a trade off between service efficiency and price.

References

[1] E. Walker, J. Gardner, V. Litvin, E. Turner, Creating personal adaptive clusters for managing scientific jobs in a distributed computing environment, in: Proceedings of the IEEE Challenges of Large Applications in Distributed Environments, 2006, pp. 95–103.

[2] A. Tsaregorodtsev, V. Garonne, I. Stokes-Rees, DIRAC: a scalable lightweight architecture for high throughput computing, in: Fifth IEEE/ACM International Workshop on Grid Computing, GRID'04, 2004, pp. 19–25.

[3] I. Sfiligoi, Making science in the grid world: using glideins to maximize scientific output, in: Nuclear Science Symposium Conference Record, 2007, NSS'07, IEEE 2, Honolulu, HI, USA, 2007, pp. 1107–1109.

[4] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde, Falkon: a fast and light-weight tasK executiON framework, in: Proceedings of the IEEE/ACM SuperComputing, November 2007.

[5] J. Chase, D. Irwin, L. Grit, J. Moore, S. Sprenkle, Dynamic virtual clusters in a grid site manager, in: Twelfth IEEE Symposium on High Performance Distributed Computing, HPDC, Seattle, Washington, June 2003.

[6] P. Ruth, X. Jiang, D. Xu, S. Goasguen, Virtual distributed environments in a shared infrastructure, IEEE Computer (2005) Virtualization Technologies.

[7] Michael Murphy, Linton Abraham, Michael Fenn, Sebastien Goasguen, Autonomic clouds on the grid, Journal of Grid Computing 8 (1) (2010) 1–18. Springer Netherlands.

[8] L. Youseff, R. Wolski, B. Gorda, C. Krintz, Paravirtualization for HPC systems, in: Workshop on XEN in HPC Cluster and Grid Computing Environments, XHPC, Held in Conjunction with the International Symposium on Parallel and Distributed Processing and Application, ISPA 2006, December 2006.

[9] L. Youseff, K. Seymour, H. You, J. Dongarra, R. Wolski, The Impact of paravirtualized memory hierarchy on linear algebra computational kernels and software, in: High Performance Distributed Computing, HPDC, Boston, June 2008.

[10] W. Emeneker, D. Jackson, J. Butikofer, D. Stanzione, Dynamic virtual clustering with Xen and Moab, in: Proc. of the Frontiers of High Performance Computing and Networking, ISPA 2006 Workshops, in: Lecture Notes in Computer Science, vol. 4331, 2006, pp. 440–451.

[11] N. Fallenbeck, H.J. Picht, M. Smith, B. Freisleben, Xen and the art of cluster scheduling, in: First International Workshop on Virtualization Technology in Distributed Computing, VTDC, 2006.

[12] A.J. Rubio-Montero, R.S. Montero, E. Huedo, I.M. Llorente, Management of virtual machines on globus grids using gridway, in: Proceedings of the 4th High-Performance Grid Computing Workshop, Conjunction with 21st IEEE International Parallel and Distributed Processing Symposium, 2007, pp. 1–7.

[13] X. Jiang, D. Xu, Violin: virtual internetworking on overlay infrastructure, in: Proceedings of the 2nd International Symposium on Parallel and Distributed Processing and Applications, December 2004.

[14] I. Krsul, A. Ganguly, J. Zhang, J.A.B. Fortes, R.J. Figueiredo, VM-plants: providing and managing virtual machine execution environments for grid computing, in: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, 2004.

[15] A. di Costanzo, M.D. de Assuncao, R. Buyya, Harnessing cloud technologies for a virtualized distributed computing infrastructure, IEEE Internet Computing 13 (5) (2009) 24–33.

[16] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, X. Zhu, From virtualized resources to virtual computing grids: the in-VIGO system, Future Generation Computer Systems 21 (6) (2005) 896–909.

[17] T. Freeman, K. Keahey, Flying low: simple leases with workspace pilot, in: Euro-Par, 2008.

[18] Michael A. Murphy, Sebastien Goasguen, Virtual organization clusters: self-provisioned clouds on the grid, Future Generation Computer Systems 26 (8) (2010) 1271–1281.

[19] I.M. Llorente, R. Moreno-Vozmediano, R.S. Montero, Cloud computing for on-demand grid resource provisioning, in: High Speed and Large Scale Scientific Computing, in: Advances in Parallel Computing, vol. 18, IOS Press, 2009, pp. 177–191.

[20] H. Nishimura, N. Maruyama, S. Matsuoka, Virtual clusters on the fly—fast, scalable, and flexible installation, in: CCGRID 2007, Seventh IEEE International Symposium on Cluster Computing and the Grid, May 2007.

[21] W. Emeneker, D. Stanzione, Dynamic virtual clustering, in: IEEE Cluster 2007, Austin, TX, September 2007.

[22] E. Huedo, R.S. Montero, I.M. Llorente, A modular meta-scheduling architecture for interfacing with pre-WS and WS grid resource management services, Future Generation Computer Systems 23 (2) (2007) 252–261.

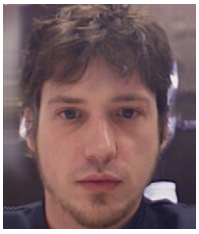
[23] R. Buyya, M. Murshed, D. Abramsin, S. Venugopal, Scheduling parameter sweep application on global grids: a deadline and budget constrained cost-time optimisation algorithm, International Journal of Software: Practice and Experience (SPE) 5 (2005) 491–512.

[24] P. Barham, B. Dragovid, K. Fraser, S. Hand, T. Ahrris, R.A. Ho, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Symposium on Operating Systems Principles, October 2003, pp. 164–177.

[25] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Herne, J. Matthews, Xen and the art of repeated search, in: USENIX Annual Technical Conference, 2004, p. 47–47.

[26] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayor, X. Zhang, Virtual clusters for grid communities, in: Proceedings of the Sixth IEEE International Symposium on Cluster Computer and the Grid, CCGRID 06, IEEE Computer Society, 2006, pp. 513–520.

[27] M. Rodriguez, D. Tapiador, J. Fontan, E. Huedo, R.S. Montero, I.M. Llorente, Dynamic provisioning of virtual clusters for grid computing, in: Proceedings of the 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing, VHPC 08, Conjunction with EuroPar08, 2008.



Constantino Vázquez received his M.E. in Computer Science (2002) from the Universidad Complutense de Madrid (UCM) and M.Sc. in Computing & Internet Systems (2003) from King's College, London. He is currently doing his Ph.D. in Computer Architecture and is part of the Distributed Systems Architecture Group at UCM. He is involved in several open source projects like the GridWay metascheduler and the OpenNebula virtual infrastructure engine. His research interests include Grid federation, Utility Computing and Virtualization.

Eduardo Huedo received his M.E. in Computer Science (1999) and Ph.D. in Computer Architecture (2004) from the Universidad Complutense de Madrid (UCM). He is an Associate Professor of Computer Architecture and Technology at UCM. Previously, he worked as a Postdoctoral Researcher at the Advanced Computing Laboratory at Centro de Astrobiolog'a (CSIC-INTA), associated to the NASA Astrobiology Institute. His research areas are Performance Management and Tuning, Parallel and Distributed Computing and Grid and Virtualization Technology.

Rubén S. Montero, Ph.D. is an associate professor in the Department of Computer Architecture at the Universidad Complutense de Madrid. He has published more than 70 scientific papers in the field of High-Performance Parallel and Distributed Computing, and contributed to more than 20 research and development programmes. His research interests lie mainly in resource provisioning models for distributed systems: Grid resource management and scheduling, distributed management of virtual machines and cloud computing. He is also actively involved in several open source grid initiatives like the Globus Toolkit or the GridWay metascheduler. Currently, he is co-leading the research and development activities of OpenNebula, a distributed virtual machine manager.

Ignacio M. Llorente has a graduate degree in Physics (B.S. in Physics and M.S. in Computer Science), a Ph.D. in Physics (Program in Computer Science) and an Executive Master in Business Administration. He has about 15 years of research experience in the field of High-Performance Parallel and Distributed Computing. Currently, he is a Full Professor in Computer Architecture and Technology at the Universidad Complutense de Madrid, where he leads the Distributed Systems Architecture Research Group. His research interests are Service Oriented Architectures and Infrastructures; Grid, Cloud and Virtualization Technologies, High Performance Computing and Utility Computing.

Bibliografía

Quienes imaginan la Biblioteca sin límites, olvidan que los tiene el número posible de libros. Yo me atrevo a insinuar esta solución del antiguo problema: la Biblioteca es ilimitada y periódica.

La biblioteca de Babel. Jorge Luís Borges.

[1] D. Aderson, J. Cobb, E. Korpela, L. Matt, and D. Werthimer. SETI@HOME: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[2] A. Agarwal, M. Ahmed, A. Berman, B. Caron, A. Charbonneau, D. Deatrich, R. Desmarais, A. Dimopoulos, I. Gable, et al. GridX1: A Canadian computational grid. *Future Generation Computer Systems*, 23(5):680 – 687, 2007.

[3] A. Anjomshoaa, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) Specification, Version 1.0. *Document GFD-R.056, Open Grid Forum*, November 2005. URL `http://www.gridforum.org/documents/GFD.56.pdf`.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, et al. Above the clouds: A Berkeley view of Cloud Computing, Feb 2009. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`.

[5] R. Badia, D. Du, E. Huedo, A. Kokossis, I. Llorente, R. Montero, M. de Palol, R. Sirvent, and C. Vazquez. Integration of GRID Superscalar and Gridway Metascheduler with the DRMAA OGF Standard. In *Proceedings of the 14th International Conference on Parallel Processing (Euro-Par 2008)*, volume 5168 of *Lecture Notes in Computer Science*, pages 445–455. Springer, 2008.

[6] P. Barham, B. Dragovid, K. Fraser, S. Hand, T. Ahrris, R. A. Ho, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Symposium on Operating Systems Principles*, pages 164–177, October 2003.

[7] D. Breuer, P. Wieder, S. van den Berghe, G. von Laszewski, J. MacLaren, D. Nicole, and H. C. Hoppe. A UNICORE-Globus Interoperability Layer. *Computing and Informatics*, 21:399–411, 2002.

[8] R. Buyya, J. Broberg, and A. M. Goscinski. *Cloud Computing Principles and Paradigms*. Wiley Publishing, 2011.

[9] B. Carpenter. Architectural Principles of the Internet. RFC 1958 (Informational), June 1996. URL `http://www.ietf.org/rfc/rfc1958.txt`. Updated by RFC 3439.

[10] N. G. Carr. The end of corporate computing. *MIT Sloan Management Review*, 46(3):67–73, Spring 2005.

[11] N. G. Carr. *The big switch: rewiring the world, from Edison to Google* . W. W. Norton Co., New York, 2008.

[12] J. Chase, D. Irwin, L. Grit, J. Moore, and S. Sprenkle. Dynamic virtual clusters in a grid site manager. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, pages 90 – 100, June 2003.

[13] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Herne, and J. Matthews. Xen and the art of repeated search. In *USENIX Annual Technical Conference*, pages 47–47, 2004.

[14] C. Cotelo, A. Gomez, J. Lopez, D. Mera, J. Cotos, J. Perez Marreroc, and C. Vazquez. Retelab: A geospatial grid web laboratory for the oceanographic research community. *Future Generation Computer Systems*, 26(8):1157–1164, October 2010.

[15] S. Deering. Watching the waist of the protocol hourglass. In *6th IEEE International Conference on Network Protocols*, 1998.

[16] K. Dowd. *High Performance Computing*. O'Reilly and Associates, Inc., Sebastopol, CA, USA, 1993.

[17] D. Erwin. Unicore - a grid computing environment. In *Lecture Notes in Computer Science*, pages 825–834. Springer-Verlag, 2001.

[18] L. Field. Getting Grids to work together. *CERN Computer Newsletter*, 41(5):8–9, 2006.

[19] I. Foster. What is the Grid? A Three Point Checklist. *GRIDToday*, 1: 32–26, July 2002. URL `ttp://www.gridtoday.com/02/0722/100136.html`.

[20] I. Foster and C. Kesselman. Globus: A toolkit-based grid architecture. In *The Grid: Blueprint for a Future Computing Infrastructure*, pages 259–278. Morgan-Kaufmann, 1998.

[21] I. Foster, T. Freeman, K. Keahy, D. Scheftner, B. Sotomayor, and X. Zhang. Virtual clusters for grid communities. In *Proceedings of the Sixth IEEE International Symposium on Cluster Computer and the Grid (CCGRID 06)*, pages 513–520. IEEE Computer Society, 2006.

[22] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer. OGSA Basic Execution Service. *Document GFD-R.108, Open Grid Forum*, November 2008. URL `http://www.gridforum.org/documents/GFD.108.pdf`.

[23] B. Hamemeier, R. Menday, B. Schuller, and A. Streir. A Universal API for Grids. *Cracow Grid Workshop (CGW'06)*, pages 312–219, August 2007.

[24] J. Herrera, E. Huedo, R. S. Montero, and I. M. Llorente. Gridway DRMAA 1.0 Implementation Experience Report. *Document GFD.E-104, DRMAA Working Group, The Open Grid Forum*, 2007.

[25] R. Hockney and C. Jesshope. *Parallel Computers 2:Architecture Programming and Algorithms*. Adam Hilgee Ltd., 1998.

[26] E. Huedo, R. Montero, and I. Llorente. A framework for adaptive execution in grids. *Software: Practice and Experience*, 34(7):631–651, 2004.

[27] E. Huedo, R. Montero, and I. M. Llorente. An Evaluation Methodology for Computational Grids. *International Conference on High Performance Computing and Communications*, 3726:409–504, 2005.

[28] P. Kacsuk, T. Kiss, and G. Sipos. Solving the Grid Interoperability Problem by P-GRADE Portal at Workflow Level. *Future Generation Computer Systems*, 24(7):744–751, July 2008.

[29] A. Kertész and P. Kacsuk. Grid meta-broker architecture: Towards an interoperable grid resource brokering service. In *Euro-Par 2006: Parallel Processing*, volume 4375 of *Lecture Notes in Computer Science*, pages 112–115. Springer Berlin / Heidelberg, 2007.

[30] C. Kesselman and I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Nov. 1998. ISBN 1558604758. URL `http://www.worldcat.org/isbn/1558604758`.

[31] K. Leal, E. Huedo, and I. Llorente. A decentralized model for scheduling independent tasks in federated grids. *Future Generation Computer Systems*, 25(8):840–852, 2009.

[32] K. Leal, E. Huedo, R. Montero, and I. Llorente. Performance-based scheduling strategies for htc applications in complex federated grids. *Concurrency and Computation: Practice and Experience*, 22(11):1416–1432, 2010.

[33] W. Li and A. Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, July 2006. URL `http://dx.doi.org/10.1093/bioinformatics/btl158`.

[34] M. Livny, J. Basney, R. Raman, and T. Tannenbaum. Mechanisms for High Throughput Computing. *SPEEDUP Journal*, 11(1), June 1997.

[35] I. Llorente, R. Montero, E. Huedo, C. Vázquez, J. Fontán, J. Herrera, and J. Vázquez-Poletti. Gridway interfaces for on-demand access to egee. *EGEE'07*, October 2007.

[36] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 2009.

[37] R. M. Metcalfe and D. R. Boggs. Ethernet: distributed packet switching for local computer networks. *Commun. ACM*, 19(7):395–404, July 1976. URL `http://dx.doi.org/10.1145/360248.360253`.

[38] R. Montero, E. Huedo, and I. Llorente. Benchmarking of high throughput computing applications on Grids. *Parallel Computing*, 32(4):267 – 279, 2006. URL `http://www.sciencedirect.com/science/article/pii/S0167819105001651`.

[39] R. Murch. *Introduction to utility computing: How it can improve TCO*. Prentice Hall, 2005.

[40] A. H. Peter Troger, Hrabri Rajic and P. Domagalski. Standardization of an API for Distributed Resource Management Systems. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, pages 619–626, 2007.

[41] G. J. Popek and R. P. Goldberg. Formal requirements for virtualizable third generation architectures. *Commun. ACM*, 17(7):412–421, July 1974. URL `http://dx.doi.org/10.1145/361011.361073`.

[42] R. Ranjan, A. Harwood, and R. Buyya. Grid Federation: An Economy Based, Scalable Distributed Resource Management System for Large-Scale Resource Coupling. *Technical Report, GRIDS-TR-2004-10, Grid Computing and Distributed Systems Laboratory*, 2004.

[43] M. Riedel, E. Laure, T. Soddemann, L. Field, J. P. Navarro, J. Casey, M. Litmaath, J.-P. Baud, B. Koblitz, et al. Interoperation of world-wide production e-science infrastructures. *Concurrency - Practice and Experience*, 21(8):961–990, 2009.

[44] M. Rodriguez, D. Tapiador, J. Fontan, E. Huedo, R. S. Montero, and I. M. Llorente. Dynamic Provisioning of Virtual Clusters for Grid Computing. In *Proceedings of the 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 08), in conjuction with EuroPar08*, 2008.

[45] A. Rubio-Montero, E. Huedo, R. Montero, and I. Llorente. Management of virtual machines on globus grids using grideway. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007)*, pages 1–7, March 2007.

[46] C. Smith, T. Kielmann, S. Newhouse, and M. Humphrey. The HPC Basic Profile and SAGA: Standardizing Compute Grid Access in the Open Grid Forum. *Concurrency and Computation: Practice and Experience*, 21(8), 2009.

[47] J. C. Vaquero, Luis M. and J. J. Hierro. *Open Source Cloud Computing Systems: Practices and Paradigms*. IGI Global, Feb. 2012.

[48] C. Vazquez, E. Huedo, R. Montero, and I. Llorente. Evaluation of a utility computing model based on the federation of grid infrastructures. In *Proceedings of the 13th International Conference on Parallel Processing (Euro-Par 2007)*, volume 4641 of *Lecture Notes in Computer Science*, pages 372–381. Springer, 2007.

[49] C. Vazquez, J. Fontan, E. Huedo, R. Montero, and I. Llorente. Transparent access to grid-based compute utilities. In *Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics (PPAM 2007)*, volume 4967 of *Lecture Notes in Computer Science*, pages 817–824. Springer, 2008.

[50] C. Vazquez, E. Huedo, R. Montero, and I. Llorente. A performance model for federated grid infrastructures. In *16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, pages 188–192, February 2008.

[51] C. Vazquez, E. Huedo, R. Montero, and I. Llorente. Federation of TeraGrid, EGEE and OSG infrastructures through a metascheduler. *Future Generation Computer Systems*, 26(7):979–985, 2010.

[52] C. Vazquez, E. Huedo, R. Montero, and I. Llorente. On the use of clouds for grid resource provisioning. *Future Generation Computer Systems*, 27 (5):600–605, May 2011.

[53] C. Vázquez. Gridway: A Metascheduler for Globus based Grids. *TeraGrid'07 Conference*, June 2007.

[54] C. Vázquez. Scaling drmaa codes to the grid: A demonstration on egee, tg and osg. *22nd edition of the Open Grid Forum (OGF'22)*, October 2008.

[55] C. Vázquez, E. Huedo, R. Montero, and I. Llorente. Dynamic provision of computing resources from grid infrastructures and cloud providers. In *Workshop on Grids, Clouds and Virtualization (WGCV). Grid and Pervasive Computing Conference (GPC 2009)*, pages 113–120. IEEE, May 2009.

[56] R. Walker, M. Vetterli, R. Impey, G. Mateescu, B. Caron, A. Agarwal, A. Dimopoulos, L. Klektau, C. Lindsay, R. Sobie, and D. Vanderster. Federating Grids: LCG Meets Canadian HEPGrid. In *Proceedings of Computing in High Energy and Nuclear Physics (CHEP 2004)*, page 982, 2004.

[57] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. Paravirtualization for HPC Systems. In *Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC)*, volume 4331, page 474. Springer, 2006.

[58] L. Youseff, K. Seymour, H. You, J. Dongarra, and R. Wolski. The Impact of Paravirtualized Memory Hierarchy on Linear Algebra Computational Kernels and Software. *High Performance Distributed Computing (HPDC)*, pages 141–152, June 2008.

[59] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3-4):171–200, 2005.