Una Aproximación Orientada a Servicios Grid para el Análisis Estático y Dinámico de Estructuras de Edificación

J. M. Alonso, V. Hernández, R. López, G. Moltó
Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación,
Camino de Vera s/n, Valencia, Spain, 46022
{jmalonso,vhernand,rolopez,gmolto}@dsic.upv.es

Abstract

This paper exposes the implementation of a service oriented platform that performs an on demand 3D static and dynamic structural analysis of highrise buildings using a Grid Computing infrastructure. The Grid Service has been developed over the middleware Globus Toolkit 4, offering to the structural scientific community an on-line multi-user service. The requirements of high throughput and robustness needed, by such a system, have forced to integrate a highly reliable Grid meta-scheduler that enables to share appropriately the computational resources. The developed system offers a multilevel fault tolerance that guarantees that every simulation request received in the Grid Service will be satisfactorily attended. Another important aspect that has been taken into account has been the security, offering user authorization and authentication, and data privacy and integrity mechanisms, critical issues in a system available in the network. In addition, an advanced GUI client that interacts with the Grid Service has also been implemented, assisting the client in the pre-process and post-processing stages, and in the structural analysis parameter definition.

Keywords: Distributed Computing, Grid Technologies, Grid Services, Task Scheduling, 3D Structural Analysis.

Resumen

Este artículo presenta la implementación de una plataforma orientada a servicios que lleva a cabo un análisis 3D estático y dinámico de edificios de gran dimensión bajo demanda, empleando una infraestructura de computación Grid. El Servicio Grid ha sido desarrollado sobre el middleware Globus Toolkit 4, ofreciendo a la comunidad científica un servicio multiusuario on-line de cálculo estructural. Los requerimientos de alta productividad y fiabilidad de tal sistema han obligado a integrar un planificador Grid altamente robusto que permite compartir de manera equitativa los recursos computacionales. El sistema desarrollado presenta una tolerancia a fallos multinivel que garantiza que toda petición de cálculo recibida en el servicio será atendida satisfactoriamente. Otra característica importante que ha sido tenida en cuenta es la seguridad, ofreciéndose mecanismos de autorización y autenticación de usuarios, así como de privacidad e integridad de datos, aspectos todos ellos imprescindibles en un sistema accesible a través de la red. Adicionalmente también ha sido desarrollado un cliente gráfico avanzado que interactúa con el Servicio Grid, asistiendo al usuario en la etapa de entrada de datos, en la definición de los parámetros de cálculo y en la interpretación de los resultados de salida.

Palabras claves: Computación Distribuida, Tecnologías Grid, Servicios Grid, Planificación de Tareas, Análisis 3D de Estructuras.

1. INTRODUCCIÓN

El análisis dinámico de estructuras [7] de edificación de gran dimensión se ha llevado a cabo tradicionalmente introduciendo una serie de simplificaciones orientadas a reducir los tiempos de cómputo y los requerimientos de memoria. Aunque dichas simplificaciones resultan ser perfectamente válidas en estructuras sencillas y simétricas, no son adecuadas en estructuras que presenten una elevada complejidad en altura y planta. Sin embargo, un análisis dinámico 3D dinámico y realista de estructuras complejas y de gran dimensión puede requerir una serie de recursos computacionales que hagan inviable el que pueda abordarse en un PC tradicional.

Si bien es cierto que el uso de una aplicación paralela, capaz de ejecutarse sobre múltiples procesadores, podría proporcionar unos resultados realistas en tiempos razonables, también lo es el hecho de que los estudios de arquitectura o de ingeniería raramente poseen la infraestructura computacional necesaria para poder ejecutarla.

En este trabajo se ha implementado una Plataforma Grid de Cálculo de Estructuras bajo demanda, basada en el middleware Globus Toolkit 4 (GT4) [8], que a través de Internet hace accesible una serie de servicios que permiten a

cualquier usuario llevar a cabo un análisis estático y dinámico 3D de sus proyectos estructurales. Esta plataforma dispone de un planificador Grid que se encarga de seleccionar, de manera totalmente transparente para el usuario, el recurso computacional más adecuado para llevar a cabo la simulación de su estructura. En nuestro caso, se dispone de un Grid Computacional compuesto por varios clusters de PCs, donde finalmente se ejecutará la aplicación paralela. El servicio ofrece un sistema de tolerancia a fallos multinivel, que garantiza que toda petición de análisis de una estructura será atendida de manera satisfactoria. Además, se ha implementado una aplicación gráfica a través de la cual el cliente define las propiedades de su edificio, envía la petición de cálculo al Grid Computacional y evalúa posteriormente los resultados.

El artículo está estructurado del siguiente modo: En la sección 2, se explica brevemente en qué consiste una Arquitectura Orientada a Servicios. En las secciones 3, 4 y 5 se describe, respectivamente, la tecnología de los Servicios Web, los estándares OGSA y WSRF, y el middleware GT4, que implementa dichos estándares. La sección 6 se analiza el Servicio Grid de Cálculo de Estructuras desarrollado, así como los diferentes componentes que lo forman, mientras que la sección 7 presenta el interfaz gráfico del cliente implementado. Finalmente, en la sección 8, se recogen las conclusiones del trabajo desarrollado.

2. ARQUITECTURA ORIENTADA A SERVICIOS

La Arquitectura Orientada a Servicios (SOA) [15] es una metodología que proporciona un entorno de trabajo para el desarrollo de software que, a modo de servicio, está accesible a los usuarios a través de Internet. Dicha metodología define a su vez el modo de utilización de estos servicios.

En un entorno SOA, los agentes conectados a la red comparten sus recursos con otros usuarios como servicios independientes a los que se tiene un acceso estandarizado. La implementación de la tecnología SOA está relacionada directamente con los Servicios Web (empleando SOAP y WSDL), aunque también se puede emplear cualquier otra implementación basada en servicios.

En comparación con las arquitecturas orientadas a objetos, la arquitectura SOA está formada por servicios de aplicación débilmente acoplados y altamente interoperables. Debido a que dichos servicios funcionan sobre diferentes plataformas de desarrollo, tales como Java y .NET, los componentes son completamente reutilizables.

3. SERVICIOS WEB

Un Servicio Web es una colección de protocolos y estándares destinados a intercambiar datos entre aplicaciones [5]. Estos servicios representan una nueva apuesta en Computación Distribuida, frente a CORBA, RMI, EJB, etc., y han emergido recientemente como un estándar gracias a su flexibilidad, extensibilidad y total independencia de cualquier plataforma. Estas características dotan de cierta ventaja a esta tecnología, ya que gracias al uso de protocolos basados en XML (eXtensible Markup Language) podemos, por ejemplo, comunicar de manera trivial un cliente implementado en C++ con un servicio programado en Java. Además, el uso del protocolo HTTP (HyperText Transfer Protocol), empleado para transmitir los mensajes entre el cliente y el servicio, es plenamente tolerado por proxies y firewalls. Sin embargo, los Servicios Web conllevan una mayor sobrecarga en las comunicaciones, ya que obviamente la transmisión de los datos en XML es muchísimo más ineficiente que la basada en datos binarios.

3.1 Arquitectura

A grandes rasgos, la Figura 1 muestra un esquema de la arquitectura en la que se basan los Servicios Web:

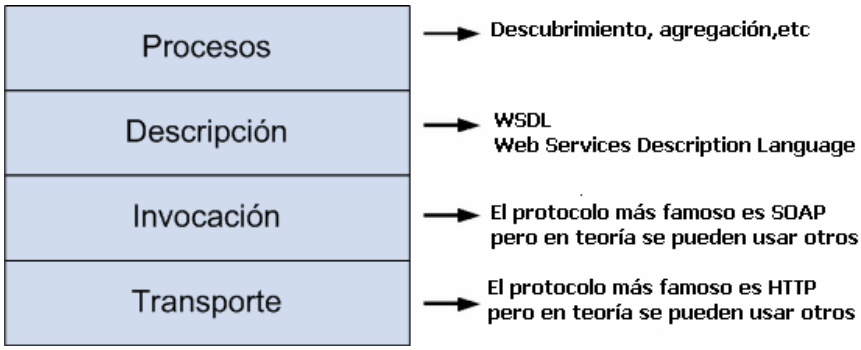

Figura 1. Arquitectura de los Servicios Web

- Procesos del Servicio: Esta capa de la arquitectura se concibe con una serie de herramientas que permiten la gestión de entornos compuestos por múltiples Servicios Web. A este nivel se contemplan, por ejemplo, los protocolos de

descubrimiento de recursos encargados de publicar todos los servicios existentes en el sistema.

- Descripción del Servicio: El propio Servicio Web es el encargado de describir y publicar su interfaz a los clientes. Esto significa que, una vez localizado el Servicio Web que buscamos, es posible pedirle que nos describa los métodos de los que consta y el modo de invocarlos. Esta información es manejada por el lenguaje WSDL (Web Services Description Language) [6].
- Invocación del Servicio: Invocar a un Servicio Web implica un intercambio de mensajes entre el cliente y el servidor. Estos mensajes están definidos por SOAP (Simple Object Access Protocol) [14], el cual especifica el formato de los datos de entrada y salida de un servicio.
- Transporte: Finalmente, todos los mensajes implicados deben ser transmitidos de algún modo entre el cliente y el servicio. El protocolo utilizado por los Servicios Web es HTTP, es decir, el mismo utilizado para acceder a los contenidos Web en Internet.

3.2 Descripción del Funcionamiento

El acceso a los Servicios Web es idéntico a las páginas Web convencionales, utilizando los tradicionales URIs (Uniform Resource Identifiers). La similitud con las direcciones de páginas Web puede ser tentadora a la hora de introducirla en un navegador, pero hay que tener en cuenta que los Servicios Web son invocados por programas y no por personas, con lo que aparecería un mensaje de error o algún código ininteligible.
Como se ha visto anteriormente, existen un buen número de protocolos y lenguajes alrededor de la tecnología de los Servicios Web. Sin embargo, un programador de servicios solamente necesita concentrarse en implementarlo en su lenguaje de programación preferido, además de en WSDL. El código SOAP encargado de invocar y recibir peticiones en el servicio es automáticamente generado e interpretado por una porción de software llamada *stub*, siendo numerosas las herramientas existentes que generan este código a partir de la descripción WSDL del servicio. De este modo, el programador no debe preocuparse del trabajo de bajo nivel que conlleva el generar e interpretar las peticiones SOAP, sino exclusivamente del propio código asociado al cliente o al servicio. Una vez detallado el funcionamiento básico, a continuación se describe la secuencia de pasos a seguir en caso de invocar a un servicio:

1. El cliente inicia la invocación del servicio a través de una herramienta *stub*, la cual convierte esta llamada local en una petición SOAP. Este proceso recibe el nombre de *serialización*.
2. La petición SOAP viaja a través de la red por medio del protocolo HTTP y es recibida por el servidor, delegando en el *stub* del servidor para su gestión. Mediante el proceso de *deserialización*, este *stub* interpreta la petición SOAP y la traduce en un lenguaje de programación concreto a fin de que pueda ser entendida por el propio servicio.
3. A continuación, el *stub* invoca a la implementación del servicio, que llevará a cabo la tarea que el cliente le ha solicitado.
4. Los resultados obtenidos de la invocación al servicio son recogidos también por el *stub* del servidor y empaquetados nuevamente en un mensaje SOAP.
5. La respuesta SOAP viaja a través de la red por medio del protocolo HTTP y es recibida por el *stub* del cliente, traduciéndola a la estructura de datos que éste entiende.
6. Finalmente, el cliente recibe la respuesta de la llamada al servicio en una estructura de datos que puede manejar y procesar.

4. OGSA Y WSRF

Una aplicación Grid está compuesta normalmente por varios componentes, o servicios, entre los cuales se suelen encontrar los siguientes:

- Gestión de Organizaciones Virtuales: Es el encargado de gestionar qué recursos y qué usuarios pertenecen a cada organización virtual.
- Descubrimiento y Gestión de Recursos: Permite que las aplicaciones lleven a cabo el descubrimiento y la gestión de los recursos que satisfagan sus necesidades.
- Gestión de Trabajos: Proporcionan el soporte necesario para que los usuarios puedan enviar tareas al sistema Grid.
- Sistemas de Seguridad: Permiten establecer diferentes niveles de seguridad a nivel de datos y de usuarios (autenticación, autorización, privacidad, integridad, etc.).
- Gestión de Datos: Ofrece servicios para el acceso a datos distribuidos.

Todos estos componentes están constantemente interactuando entre sí. Por ejemplo, el Gestor de Trabajos pedirá que el Descubridor de Recursos le proporcione el recurso computacional adecuado, de acuerdo a los requerimientos de una tarea que debe ser ejecutada. Esta interrelación entre los distintos componentes de un sistema Grid obliga a una estandarización de los mismos, definiendo un interfaz común para cada tipo de servicio.
El estándar OGSA (Open Grid Services Architecture) [11] está orientado a definir una arquitectura común y estándar para el desarrollo de aplicaciones Grid [9]. Su objetivo primordial consiste en estandarizar aquellos servicios que típicamente se encuentran en un sistema Grid, por medio de la especificación de un conjunto de interfaces estándar. En cuanto a la implementación de esta arquitectura, es necesaria la selección de un middleware de Computación Distribuida que actuará como base. Aunque en principio podrían haberse seleccionado cualquiera de los disponibles (CORBA, RMI, etc.), fueron los Servicios Web los elegidos.
Sin embargo, aunque la alternativa más apropiada sean los Servicios Web, éstos no cumplen con uno de los requisitos más importantes de OGSA, como es la de mantener el estado. Es decir, es necesario que el sistema desarrollado a partir de los Servicios Web posea un estado y que éste pueda ser modificado y accedido por los servicios que publica. Sin embargo, los Servicios Web fueron ideados como simples procesadores de mensajes y no poseen ningún estado. Como solución aparece la especificación WSRF (Web Services Resource Framework), la cual es una extensión de los Servicios Web tradicionales para dotarles de un estado.

5. GLOBUS TOOLKIT 4

El middleware Globus Toolkit (GT) incluye un conjunto de herramientas software que proporcionan una plataforma para el desarrollo de aplicaciones Grid [10]. En su versión 4, este software incluye varios servicios de alto nivel, que satisfacen los requerimientos del estándar OGSA, como la monitorización de recursos, el descubrimiento de servicios, la gestión del envío de trabajos, el soporte de los sistemas de seguridad y los servicios para la gestión de datos, entre otros [8]. Sin embargo, no se puede decir que GT4 sea una implementación exacta de OGSA, pero sí que es el estándar de facto empleado en la comunidad de usuarios Grid. Además, estos servicios están implementados sobre el estándar WSRF, ya que, de hecho, GT4 incluye una implementación completa del mismo. De este modo, se facilita enormemente la implementación de Servicios Web con estado usando GT4.
Se puede decir que el middleware GT ha experimentado, en su versión 4, una evolución natural hacia los Servicios Web, adoptando dicha tecnología como base para la definición tanto de su arquitectura como de sus interfaces. Como resultado aparecen los llamados Servicios Grid, es decir, Servicios Web que aplicados al dominio Grid establecen un estándar en el descubrimiento y acceso a recursos Grid.

6. EL SERVICIO GRID DE CÁLCULO DE ESTRUCTURAS

Partiendo del mencionado middleware GT4, en este trabajo se ha desarrollado un Servicio Grid de Cálculo de Estructuras que lleva a cabo, bajo demanda, un análisis tanto estático como dinámico de estructuras de gran dimensión.
Como la Figura 2 indica, el servicio está compuesto principalmente por 5 componentes: Service Manager, Task Notifier Daemon, Parallel Structural Simulator, Scheduler y Data Collector Daemon.
El *Service Manager* es el componente principal del sistema y se encarga de atender las peticiones de los clientes e interconectar todos los elementos que componen el servicio. El *Task Notifier Daemon* es el encargado de informar al usuario de los cambios de estado en la ejecución de sus simulaciones, de modo que el usuario sabe en cada momento en qué estado se encuentran sus tareas. Por otro lado, hemos desarrollado una aplicación basada en Computación de Altas Prestaciones [1] [2] (*Parallel Structural Simulator*), e implementada bajo la librería MPI [13], que se encarga de realizar un análisis realista estático o dinámico en 3D. Dicho cálculo dinámico puede llevarse a cabo a través de un análisis modal o mediante diferentes métodos de integración directa.
El *Scheduler* lleva a cabo la selección de recursos necesaria para ejecutar cada simulación, empleando una política definida. A partir de un Grid Computacional, compuesto en nuestro caso por varios clusters de PCs, este componente se encarga de seleccionar el recurso más adecuado, de llevar a cabo la transferencia de los ficheros de entrada, de ejecutar la tarea y de recoger los resultados. Finalmente, el *Data Collector Daemon* elimina todos aquellos ficheros de resultados que, una vez transcurrido un espacio de tiempo prudencial desde su generación, permanecen todavía en el servicio.

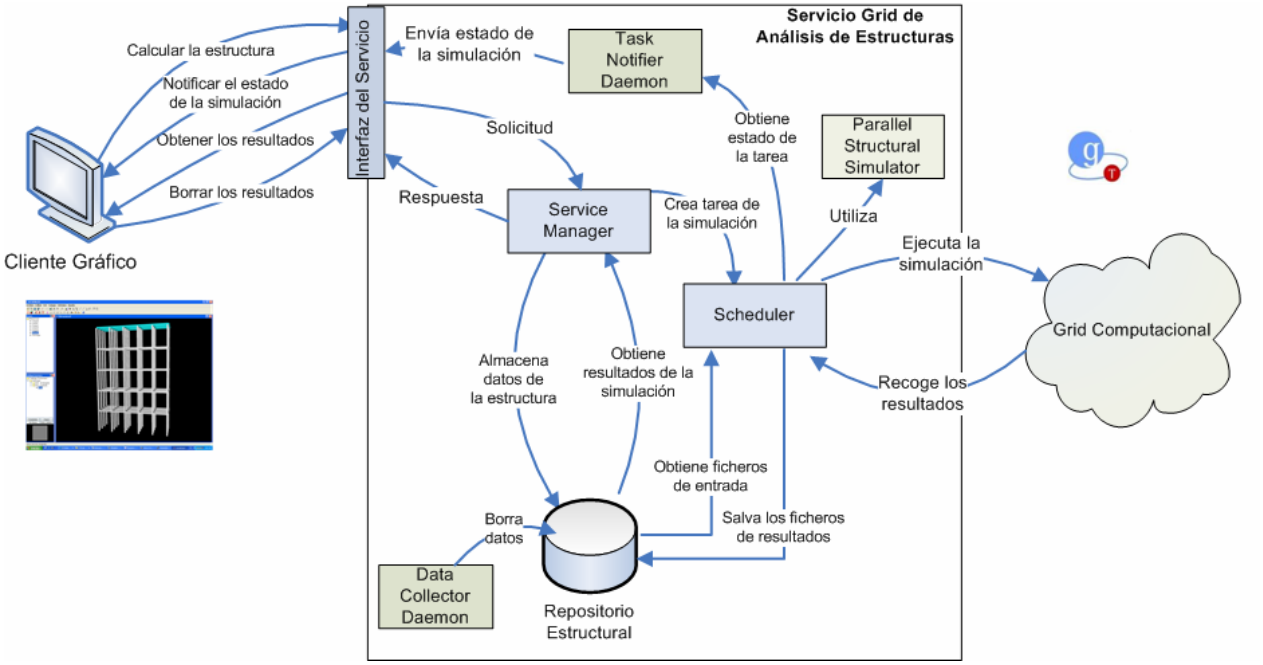

Figura 2. Esquema del Servicio Grid de Cálculo de Estructuras

6.1 Gestión de las Simulaciones

El proceso de análisis de una estructura conlleva una secuencia de pasos en los cuales intervienen e interactúan todos los agentes integrantes del servicio. Este proceso se inicia cuando el cliente lanza una petición de cálculo, enviando al servicio la geometría de la estructura a analizar, sus propiedades, las hipótesis de carga y los parámetros que definen el tipo de cálculo. Las peticiones son atendidas por el Service Manager, el cual procesa los datos, devuelve al cliente un identificador de la simulación y genera un fichero de entrada necesario para la aplicación de cálculo, almacenándolo a su vez en el Repositorio Estructural, que no es más que un directorio en disco contenedor de toda la información de la simulación. Una vez almacenado dicho fichero, el Service Manager crea la tarea a ejecutar, la cual consiste en un objeto que contempla toda la información necesaria para llevar a cabo su ejecución en el Grid Computacional.

En nuestro caso hemos escogido como planificador de las tareas al metaplanificador GMarte, que se explica con detalle mas adelante. Así, la tarea es añadida a dicho planificador, a fin de que lleve a cabo, de manera totalmente transparente para el usuario, la selección de recursos, la transferencia de los ficheros de entrada entre el Servicio Grid y el recurso computacional seleccionado, la ejecución de la tarea y la recogida de los resultados. Se ha definido una política de selección de recursos basada en un modelo de alta eficiencia y productividad, el cual tiene en cuenta las características de cada simulación, tales como el número de grados de libertad de la estructura, el tipo de cálculo, los privilegios asociados al tipo de cliente, etc. En función de dichos parámetros, la simulación es ejecutada con un número de procesadores determinado. A medida que la tarea progresa, el Task Notifier Daemon va informando al usuario del estado de la misma.

Los resultados de la ejecución son también almacenados en el Repositorio Estructural, y dependiendo del tipo de análisis, el Task Notifier Daemon enviará al usuario uno u otro tipo de notificación. Así, en el caso de un análisis estático, una vez finalizada la simulación se notifica al usuario que tiene todos los resultados disponibles y que puede proceder a recogerlos. Por otro lado, en el caso de un análisis dinámico se notifica al usuario cuando tenga un número adecuado de resultados parciales disponibles, independientemente de que la ejecución haya o no finalizado. En ambos casos, la petición de recogida de resultados implica procesar los ficheros de resultados, por parte del Service Manager, construyendo la estructura de datos diseñada expresamente para devolverlos apropiadamente al cliente. Por otro lado, se publica un método de borrado que elimina del servicio todos los datos referentes a la simulación y que debería ser empleado por los clientes una vez que han recogido sus resultados. Sin embargo, puesto que no es obligatoria su invocación, se ha implementado el demonio llamado Data Collector Daemon que periódicamente eliminará del repositorio los datos de simulaciones ya finalizadas.

Es importante destacar que cuando un cliente somete una petición de cálculo, éste no se queda bloqueado a la espera de su finalización. En realidad, se ha desarrollado un sistema de *tickets* donde a cada simulación se le asigna un identificador, el cual es enviado al cliente al someter una petición de cálculo, y que deberá ser utilizado en el resto de invocaciones para identificar a dicha simulación.

6.2 Comunicaciones Cliente Servicio

Uno de los principales problemas asociados a la utilización de la arquitectura orientada a servicios son las comunicaciones cliente-servicio, sobre todo cuando estas requieren el intercambio de un gran volumen de datos. Como es sabido, las comunicaciones se llevan a cabo mediante mensajes SOAP, que no son más que un tipo de ficheros XML especiales. Esto conlleva una notable sobrecarga en cuanto al tamaño de dichos mensajes, agravando el problema del enorme volumen de datos generado por un análisis 3D dinámico y realista de estructuras de gran dimensión.

Para tratar de mitigarlo en lo posible, sobretodo teniendo en cuenta que podemos encontrarnos con una alta variedad en el tipo de conexiones a Internet de los clientes, se ha implementado un esquema de codificación en hexadecimal. Es decir, en vez del desarrollo de un esquema completo XML que contenga todos los datos, se introducen datos binarios embebidos, de manera que se reduce notablemente el tamaño de los mensajes.

6.3 Gestión de la Transferencia de los Resultados

En función del tipo de análisis que es llevado a cabo, la transferencia de resultados es tratada de distinto modo tanto en las comunicaciones entre el cliente y el servicio como en las efectuadas entre el servicio y el recurso Grid.

En el caso de un análisis estático, el Scheduler recogerá automáticamente todos los resultados una vez finalizada la ejecución de la tarea. Estos resultados son almacenados en el Repositorio Estructural, notificándose a su vez la disponibilidad de los mismos al usuario. De este modo, el usuario puede someter una petición de recogida de datos, la cual empaqueta con una codificación hexadecimal todos los ficheros de salida en un mensaje SOAP. Este empaquetado en un único mensaje garantiza que la operación de recogida de los resultados se realice de forma atómica.

Por otro lado, en el caso de un análisis dinámico, debido principalmente al gran volumen de datos generado, se debe emplear un esfuerzo extra si queremos gestionar de manera óptima la recepción de los resultados. En cuanto a la transferencia de los resultados entre el recurso Grid y el servicio, se ha implementado un sistema de recogida de resultados parciales generados por la aplicación paralela para cada paso de tiempo de su proceso iterativo. Mediante el uso de un sistema de sufijos que distingue cada resultado parcial y conociendo el intervalo de tiempo que transcurre en la generación de resultados entre dos instantes de tiempo consecutivos (dato este último aportado por la herramienta de cálculo), el planificador se configura a fin de agilizar la recogida de resultados, solapándola de este modo con la ejecución de la simulación. De este modo, a medida que los resultados parciales del análisis dinámico son generados, éstos son transferidos al servicio sin necesidad de esperar a la finalización de la simulación, aproximación esta última que conllevaría un mayor tiempo de espera por parte del usuario.

En cuanto a la transferencia de resultados entre el cliente y el servicio, se ha seguido la misma política de solapar al máximo las comunicaciones con el tiempo de ejecución de la simulación. De este modo, el servicio envía una notificación al cliente cada vez que dispone de un conjunto determinado de resultados en el Repositorio Estructural. En consecuencia, este último puede comenzar a descargar y visualizar resultados parciales sin esperar a que su simulación haya finalizado, agilizándose al máximo la transferencia y reduciendo enormemente los tiempos de espera.

6.4 Planificación Grid

Como ya se ha explicado anteriormente, la ejecución de las simulaciones en los recursos computacionales disponibles es gestionada eficientemente por un planificador Grid. Actualmente, el servicio emplea como planificador a GMarte [3] [4], que ofrece servicios de ejecución remota de tareas y metaplanificación eficiente sobre máquinas basadas en GT. En concreto, el planificador GMarte se encarga de la selección de recursos, la transferencia de los ficheros de entrada, la ejecución de las simulaciones y la recogida de los resultados, todo ello de manera transparente para el usuario.

Otra característica a destacar de dicho planificador es el esquema de tolerancia a fallos multinivel ofrecido, dirigido a gestionar tanto los errores ocurridos durante la transferencia de ficheros como durante la ejecución de la tarea en la máquina remota. Esto garantiza que las simulaciones serán ejecutadas satisfactoriamente siempre y cuando existan recursos computacionales Grid disponibles.

Actualmente, GMarte soporta ejecuciones en GT 2.4 y 4.0.2. Su funcionalidad está disponible como un API en Java, lo que permite fácilmente integrar su funcionalidad como parte de diferentes servicios Grid, así como en XML, permitiendo de este modo utilizarlo sin necesidad de emplear el lenguaje Java.

Una de las principales ventajas adicionales que aporta GMarte es una capa de abstracción en el acceso a la información computacional ofrecida por los diferentes recursos heterogéneos. De esta manera, consultar el número de procesadores en máquinas con versiones diferentes de Globus se realiza, de la misma manera, mediante cómodos métodos de alto nivel para el usuario. Además, GMarte permite el descubrimiento de recursos computacionales, bien sea accediendo al componente GIIS (Grid Index Information Service) de Globus o bien accediendo a un BDII (Berkeley Database Information Index), que es un componente disponible en los despliegues del middleware LCG-2.

El planificador que ofrece GMarte está implementado mediante técnicas de multihilo lo que permite realizar las múltiples fases de ejecución remota de las tareas de manera simultánea. Esto es de vital importancia especialmente para la fase inicial de selección de recursos, donde, con esta aproximación, se obtienen importantes incrementos de velocidad para el proceso global de asignación de tareas. Además, este planificador considera los requerimientos computacionales

especificados por las tareas, así como la información dinámica del Grid acerca del estado de los recursos para realizar la asignación de las mismas.

6.5 Notificaciones

Las notificaciones son un patrón de diseño software que permite informar a los clientes de los cambios producidos en el servicio [12]. De este modo, ya no es el cliente el que consulta el estado de sus simulaciones, sino que es el servicio quien le informa por medio de mensajes de los cambios producidos en las mismas.

Por lo tanto, cuando una petición de cálculo es enviada por parte de un cliente, se crea en el servicio un ítem de notificación referente a dicha simulación. Este ítem es publicado, pudiendo el cliente suscribirse a él y comenzando en consecuencia a recibir las notificaciones referentes a los cambios de estado de su petición (en cola, en ejecución, finalizada, fallo, etc.). De este modo, el cliente es inmediatamente informado de la finalización de su tarea, en caso de un análisis estático, teniendo disponibles todos los resultados de cálculo, o cuando haya una cantidad considerable de los mismos que pueden ser accedidos, si se lleva a cabo un cálculo dinámico.

Debemos tener en cuenta que si no empleáramos esta aproximación basada en notificaciones, sería el cliente el que periódicamente pediría al servicio que le informara del estado de su simulación. Esta alternativa introducía una sobrecarga notable en el sistema, máxime si tenemos en cuenta que el servicio puede estar atendiendo simultáneamente peticiones de múltiples clientes.

6.6 Tolerancia a Fallos

La tolerancia a fallos es uno de los aspectos más relevantes en un sistema que trata de ofrecer un servicio bajo demanda, en el cual el cliente exigirá las máximas garantías. Existen varios niveles de tolerancia a fallos implementados en el sistema, tanto a nivel del servicio como a nivel de planificación y ejecución en Grid.

En cuanto al servicio, se ha implementado un esquema de persistencia que almacena una descripción de las simulaciones que están pendientes de ejecución, de aquellas que están ejecutándose, y de las que ya han finalizado pero aún tienen resultados pendientes de ser recogidos por parte del cliente. De este modo, ante una caída del servicio, se volverían a relanzar todas las tareas que estaban en cola o en ejecución, y se volverían a registrar los identificadores de las que aún tienen datos pendientes.

En cuanto a la ejecución de la tarea en el recurso Grid, el planificador GMarte es el que nos proporciona la tolerancia a fallos necesaria para que en caso de fallo en la ejecución de una tarea, ésta sea migrada automáticamente a otro recurso. En el caso de que se trate de un análisis dinámico, la ejecución se relanzaría desde el principio, y el usuario quedaría bloqueado a la espera de la generación de los resultados que le restan por descargarse.

Con estos esquemas de tolerancia a fallos implementados, se garantiza que toda petición que llegue satisfactoriamente al servicio será atendida, incluso ante caídas del propio servicio. Esta característica presenta una relevancia importante, siendo muy valorada por parte de los clientes.

6.7 Seguridad

La seguridad es un aspecto fundamental a tener en cuenta en un sistema on-line con el que van a interactuar multitud de usuarios, con los peligros asociados que ello conlleva. De este modo, y con el fin de ofrecer las suficientes garantías de seguridad y privacidad de datos, se plantean cuestiones como la autorización y la autenticación de los usuarios y la privacidad e integridad de los datos.

Por un lado, es necesaria la implantación de un sistema de autorización y autenticación que regule el acceso a los servicios publicados y permita saber qué usuario está realizando cada acción en cada momento. Para poner en marcha este esquema se emplea un fichero de configuración en el cuál aparece un listado de todos los usuarios que pueden interactuar con el servicio. La autenticación se lleva a cabo por medio de un certificado X.509, que identifica inequívocamente al usuario y que éste envía al servicio al iniciar la comunicación.

La privacidad e integridad de los datos se consigue utilizando sistemas de clave pública/privada, que se valen del mismo certificado X.509 del usuario para llevar a cabo la encriptación y firma de los datos que viajan entre el servicio y el cliente.

7. EL CLIENTE GRÁFICO

El Servicio Grid de Análisis de Estructuras no tendría sentido como un sistema independiente, ya que requiere de aplicaciones cliente que le sometan peticiones y recojan los resultados. Por lo tanto, se ha desarrollado una avanzada interfaz gráfica de usuario que le asiste de modo amigable en la etapa de entrada de datos, o preproceso, en la cual se configuran las distintas particularidades de los elementos estructurales del edificio (propiedades de las secciones, condiciones de sustentación, cargas, etc.), en la definición de los parámetros de cálculo, y en la interpretación de los resultados de salida, o postproceso.

Por medio de las librerías gráficas ofrecidas por el API Java 3D, esta aplicación multiplataforma (ver Figura 3) muestra una escena 3D en la cual el usuario puede interactuar con la estructura, rotarla, trasladarla, modificar su tamaño de visualización (zooms) y seleccionar elementos de la misma, todo ello dentro de los modos de visualización sólido y

alámbrico. Esta alta interactividad ofrecida facilita enormemente la ardua tarea del preproceso, convirtiéndose en un proceso muy intuitivo y rápido.

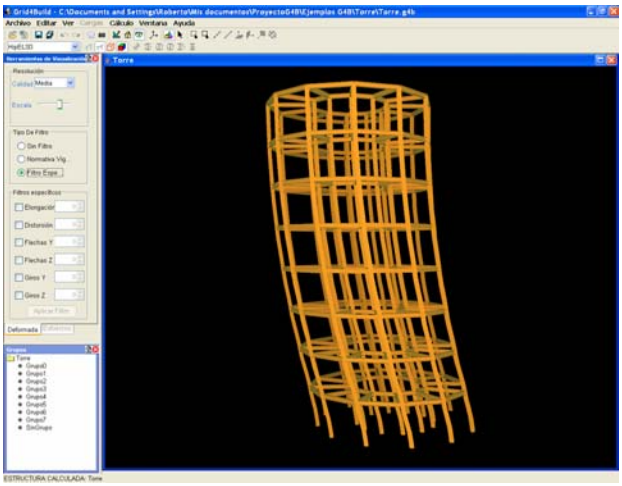

Figura 3. Aspecto del cliente gráfico del Servicio de Análisis de Estructuras

La aplicación gráfica analiza las estructuras por medio del Servicio Grid desarrollado, interactuando a través de su interfaz pública. De este modo, las propiedades del edificio a simular, así como los diferentes parámetros asociados al cálculo, son enviados mediante un mensaje SOAP. Seguidamente, el cliente se suscribe a las notificaciones de su simulación y comienza a ser informado del progreso de la misma. Finalmente, cuando es notificado de que hay resultados disponibles, éstos son recogidos por medio de otro mensaje SOAP y borrados del servicio a voluntad del cliente. A partir de estos datos comienza la etapa de postproceso, siendo a su vez representados gráficamente en el interfaz de modo que puedan ser interpretados fácilmente.
La tolerancia a fallos incluida en el cliente es otro aspecto a destacar, aprovechando el desacoplamiento existente con la etapa de análisis de la estructura. Esto garantiza que si el cliente cierra la aplicación, o ésta incluso falla, mientras se está llevando a cabo remotamente el análisis, no se va a producir ni la cancelación ni la pérdida de resultados, que podrán ser recuperados posteriormente gracias al uso del identificador de cálculo. En cuanto al análisis dinámico, debido a la existencia de diversos ficheros de resultados parciales, el cliente implementa un esquema de persistencia que va registrando automáticamente el identificador del último paso de tiempo descargado, permitiéndose fácilmente la reanudación de la descarga justo en el punto en el que se quedó. De este modo, se optimizan al máximo las transferencias de resultados entre el cliente y el servicio, garantizándose que todo resultado recibido y almacenado por el cliente no vuelve a ser reenviado.
En cuanto a los fallos que puedan existir en la red durante la transferencia de datos, el cliente llevará a cabo una serie de reintentos hasta que la conexión con el servicio se dé por perdida. Con posterioridad, el cliente volverá a solicitar, en cualquier momento, la comunicación con el servicio, a fin de recoger los resultados de su simulación.

8. CONCLUSIONES

Este artículo describe la implementación de un Servicio Grid de Análisis de Estructuras, basado en Globus Toolkit 4, que lleva a cabo un análisis 3D realista tanto estático como dinámico de estructuras de gran dimensión por medio de una aplicación basada en Computación de Altas Prestaciones. A fin de gestionar eficientemente cada una de las simulaciones recibidas en el servicio y ejecutarlas en los diferentes recursos computacionales que componen la infraestructura Grid, se ha empleado el planificador GMarte.
Todas las decisiones de diseño e implementación han estado dirigidas a satisfacer los requerimientos de alta productividad, robustez, fiabilidad y seguridad de un sistema multiusuario accesible a través de la red.
El principal inconveniente de sobrecarga en las comunicaciones introducida por la tecnología de los Servicios Web ha sido subsanado gracias a la utilización de esquemas de codificación que han permitido la inclusión de datos binarios en los mensajes XML. Por otro lado, se ha empleado un gran esfuerzo en reducir al mínimo los períodos de espera por parte de los usuarios, desarrollando un sistema que permite solapar el tiempo de ejecución de las simulaciones con la transferencia de resultados.
La alta fiabilidad del sistema desarrollado está sustentada en el desarrollo de un esquema de tolerancia a fallos multinivel, que garantiza que toda petición de análisis que llega al servicio será satisfactoriamente atendida. Este esquema de tolerancia a fallos se extiende por todos los componentes del sistema, bien sea el propio Servicio Grid de Análisis de Estructuras, el planificador de tareas GMarte y el cliente gráfico.

Finalmente remarcar que ha sido desplegada una política robusta de seguridad imprescindible en un sistema que va a estar disponible en Internet. Esta política contempla varios aspectos como la autenticación y autorización de usuario y la integridad y privacidad de los datos que viajan por la red.

Agradecimientos

Los autores desean agradecer el apoyo financiero recibido por parte del Ministerio de Ciencia y Tecnología y la Generalitat Valenciana para el desarrollo de los proyecto GRID-IT (TIC2002-0131) y GRID4BUILD (GV04B-424), respectivamente. La ayuda recibida en el proyecto GRID-IT está cofinanciada por el Fondo Europeo de Desarrollo Regional (FEDER) a través de los Fondos Estructurales.

Referencias

[1] Alonso, J.M., de Alfonso, C., García, G. and Hernández, V. Integrating HPC and Grid Computing for 3D Structural Analysis of Large Buildings. *Proceedings of the Fourth International Conference on Engineering Computational Technology (ECT 2004).* Paper 92.

[2] Alonso, J.M. and Hernández, V. Three-dimensional Structural Dynamic Analysis using Parallel Direct Time Integration Methods, *The Tenth International Conference on Civil, Structural and Environmental Engineering Computing (CC 2005).* Paper 232.

[3] Alonso, J.M., Hernández, V. and Moltó, G. An Object-Oriented View of Grid Computing Technologies to Abstract Remote Task Execution. *Proceedings of the Euromicro 2005 International Conference*, pp. 235-242.

[4] Alonso, J.M., Hernández, V. and Moltó, G. GMarte: Grid Middleware to Abstract Remote Task Execution. To appear in *Concurrency and Computation: Practice and Experience, 2006*.

[5] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. Web Services Architecture. *W3C, Working Draft*.

[6] Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. Web Services Description Language (WSDL) 1.1.W3C Note, March 2001.

[7] Clough, R.W. and Penzien, J. *Dynamics of Structures*. Second Edition, Singapore: McGraw-Hill International Editions. 1993.

[8] Foster, I. Globus Toolkit Version 4. Software for Service-Oriented Systems. *IFIP International Conference on Network and Parallel Computing*. Springer-Verlag LNCS. Vol. 3779. (2005), pp. 2-13.

[9] Foster, I. and Kesselman, C. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kauffmann, 2004.

[10] Foster, I. and Kesselman, C. Globus: A Metacomputing Infrastructure Toolkit. *Intl. J. Supercomputer Applications.* Vol. 11, No. 2, (1997), pp. 115-128.

[11] Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J. and Von Reich, J. The Open Grid Services Architecture. *OGSA-WG.*

[12] Graham, S. and Murray, B. Web Services Base Notification 1.2. *OASIS, Working Draft 03*.

[13] Gropp, W.D., and Lusk, E. *Users Guide for MPICH, a Portable Implementation of MPI*. Mathematics and Computer Science Division, Argonne National Laboratory, 1996.

[14] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J. and Frystyk Nielsen, H. SOAP Version 1.2 Part1: Messaging Framework. W3C Recommendation, June 2003.

[15] Mattew MacKenzie, C., Laskey, K., McCabe, F., Brown, P. and Metz, R. Reference Model for Service Oriented Architecture 1.0. *OASIS, Public Review Draft 1.0.*