Tesis de Máster en Computación Paralela y Distribuida
Departamento de Sistemas Informáticos y Computación
UNIVERSIDAD POLITÉCNICA DE VALENCIA

Aplicación de las Tecnologías Grid y de las Arquitecturas Orientadas a Servicios en el Análisis de Estructuras de Edificación

Roberto López Herrero
Director: Vicente Hernández García

Valencia, Octubre 2007

Índice de contenidos

Capítulo 1

Introducción

El análisis de estructuras de edificación de gran dimensión tradicionalmente se ha llevado a cabo introduciendo una serie de simplificaciones orientadas a reducir los tiempos de cómputo y los requerimientos de memoria, más aún cuando se trata de un análisis dinámico [1]. 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.

Actualmente, cada vez son más comunes edificaciones asimétricas, donde la aparición de fuerzas a torsión puede provocar en muchos casos el colapso del edificio, como ocurre ante seísmos. Por lo tanto, debido a los efectos dramáticos provocados por la eventualidad de un terremoto, se hace necesaria su simulación previamente a la construcción. Sin embargo, un análisis dinámico 3D y realista de estructuras complejas y de gran dimensión, puede requerir una serie de recursos computacionales que hacen inviable el que pueda abordarse en un PC tradicional. Gracias al uso de la Computación de Altas Prestaciones (HPC) es posible llevar a cabo este tipo de análisis realista reduciendo notablemente los tiempos de ejecución, incluso en análisis dinámicos. Si bien es cierto que, mediante el uso de estas tecnologías se puede desarrollar una aplicación capaz de ejecutarse sobre múltiples procesadores, proporcionando 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 la fase de diseño, un ingeniero estructural típicamente trabaja con diferentes configuraciones preliminares del edificio, considerando diferentes diseños o aplicando distintas secciones a los elementos estructurales que lo forman. Por lo tanto, el análisis de una estructura requiere llevar a cabo múltiples simulaciones tratando de encontrar la solución óptima, en cuanto a coste de los materiales y seguridad del edificio. Además, en el caso de un análisis dinámico, la normativa española (NCSE-02) requiere que el edificio sea evaluado como mínimo ante 5 terremotos representativos.

En este escenario, la Tesis de Master que nos ocupa está enmarcada dentro del Proyecto de Investigación Científica y Desarrollo Tecnológico titulado “Desarrollo de una Aplicación Grid de Altas Prestaciones para el Análisis Dinámico y Visualización en 3D de Estructuras de Edificación (Grid4Build)”, con referencia GV04B-424, financiado por la Conselleria de Cultura, Educación y Deporte de la Generalitat Valenciana, que fue desarrollado por el Grupo de Redes y Computación de Altas Prestaciones (GRyCAP) de la Universidad Politécnica de Valencia y cuyo investigador principal fue Vicente Hernández. Dicho proyecto de investigación planteó una solución novedosa a un problema relevante como es la simulación del comportamiento de estructuras ante acciones dinámicas (terremotos, viento, etc.). Este problema tradicionalmente se ha tratado de una forma simplificada debido, principalmente, a su alto coste computacional.

El proyecto Grid4Build tuvo como objetivo realizar una implementación optimizada basada en técnicas de Computación de Altas Prestaciones (Computación Paralela y Distribuida) y Tecnologías Grid, para resolver el problema dinámico estructural sin ningún tipo de simplificación, en cuanto a su tamaño y complejidad subyacente.

Mediante el uso de las Arquitecturas Orientadas a Servicios y las tecnologías Grid, el trabajo de la presente Tesis presenta el desarrollo de un Servicio Grid de Análisis de Estructuras, que ofrece un simulador basado en HPC y una plataforma hardware de análisis de estructuras de edificación bajo demanda, disponible a través de la red.

1.1. Objetivos de la Tesis

El trabajo a desarrollar parte de un simulador estructural HPC basado en MPI [2] que lleva a cabo el análisis dinámico y estático de estructuras en 3D [3,4]. Esta herramienta paralela ha sido desarrollada por el GRyCAP y es la herencia de una larga trayectoria del grupo de investigación relacionada con la aplicación de la Computación de Altas Prestaciones al análisis de estructuras.

El objetivo del presente trabajo es el desarrollo de una Arquitectura Orientada a Servicios, basada en las tecnologías Grid, que ofrezca a la comunidad de arquitectos e ingenieros estructurales una plataforma disponible a través de Internet, para llevar a cabo sus simulaciones empleando un esquema de computación bajo demanda. De este modo, los calculistas en lugar de invertir en hardware específico para ejecutar el simulador paralelo pueden emplear un conjunto de recursos distribuidos, de los cuales no tienen porqué ser propietarios, que les permiten satisfacer los altos requerimientos en cuanto a tiempos de cómputo y de almacenamiento del análisis de estructuras de gran dimensión. Esta Plataforma Grid puede acercar este tipo de tecnologías a pequeños y medianos estudios de arquitectura o ingeniería, que no poseen el potencial suficiente para invertir en este tipo de infraestructura, y a los que resultaría rentable la utilización de esquemas de pago por servicios.

Para el desarrollo de esta plataforma Grid se debe realizar un análisis del estado del arte en cuanto a las tecnologías distribuidas disponibles. El Servicio Grid de Análisis de Estructuras debe ser altamente ínteroperable y multiplataforma, no imponiendo ningún lenguaje de programación concreto ni ninguna plataforma específica para el desarrollo de las aplicaciones cliente. Estos aspectos permitirán una mayor difusión y utilización del sistema, ofreciendo a los usuarios una mayor libertad y flexibilidad para integrar el servicio dentro de aplicaciones concretas.

Por otro lado, nótese que la aplicación Grid a desarrollar consiste en un sistema multiusuario, donde la escalabilidad y la productividad serán aspectos a considerar. Este Servicio disponible a través de Internet deberá ser escalable con respecto al número de usuarios presentes en el sistema, ya que podrá haber varios calculistas interactuando con el mismo concurrentemente. Por otro lado, la habilidad del entorno a

la hora de llevar a cabo una distribución de tareas eficiente sobre la infraestructura Grid subyacente marcará las prestaciones del sistema global. De este modo, se hace necesario un estudio del estado del arte de las técnicas de metaplanificación Grid, y de los planificadores software disponibles, con el fin de optimizar la productividad del sistema, teniendo en cuenta que los recursos Grid disponibles son limitados.

Para que el Servicio desarrollado pueda ser utilizado con ciertas garantías, y permita ser explotado por una comunidad de ingenieros estructurales dentro de un sistema en producción, se hace imprescindible la inclusión de mecanismos de tolerancia a fallos y de seguridad. El entorno Grid desarrollado deberá garantizar que todas las peticiones sometidas sean satisfactoriamente atendidas, es decir, que no se pierdan simulaciones independientemente de la caída del Servicio Grid o de la infraestructura Grid que lo sustenta. Por otro lado, ya que la aplicación estará disponible online, con todos los riesgos de seguridad que ello entraña, deberá garantizarse que solamente aquellos usuarios autorizados puedan acceder a la misma. Además, será habitual la presencia de múltiples usuarios pertenecientes a diferentes organizaciones o corporaciones interactuando con el sistema, los cuales necesitarán preservar la privacidad de todos los datos referentes a sus proyectos, tanto durante el tránsito como en su almacenamiento en la infraestructura Grid.

Capítulo 2

Análisis de Estructuras

El proceso de diseño de una estructura de edificación lleva asociado una serie de análisis para evaluar el cumplimiento de un conjunto de condiciones de seguridad, funcionalidad y durabilidad previamente fijadas. Mediante este análisis, ingenieros y arquitectos tratan de determinar la respuesta de un edificio ante un conjunto conocido de cargas o acciones externas. Para ello se define un modelo estructural, cuya definición y propiedades variará en función de la tipología y características del edificio.

En el modelo estructural se incluyen las propiedades geométricas y estructurales del edificio, así como las cargas aplicadas. A partir de este conjunto de propiedades se obtienen las características del edificio a tener en cuenta en su análisis, que son su rigidez, en el caso de un análisis estático, y la masa y el amortiguamiento en el caso de un análisis dinámico.

El análisis completo de una estructura supone que, en teoría, hay que determinar su respuesta en un número infinito de puntos que la componen, así como en un número infinito de instantes de tiempo en el caso de un análisis dinámico. Esta aproximación en la práctica, es inviable por lo que se requiere introducir algún tipo de discretización en ambos dominios para obtener una solución numérica al problema. Por un lado, en el dominio espacial se define un conjunto finito de puntos de la estructura, predeterminados adecuadamente, en los cuales será obtenida la respuesta. Por otro lado, en el dominio temporal se define un conjunto de instantes de tiempo en los cuales será obtenida la respuesta estructural.

Una vez el modelo estructural es definido, la respuesta calculada será estática en el caso de que las cargas externas no varíen en el tiempo o dinámica en el caso de que si lo hagan, como en el caso de un análisis sísmico. Independientemente de si obtenemos una respuesta estática o en el tiempo, cada bloque de resultados de cálculo incluirá los desplazamientos y esfuerzos en cualquiera de los puntos pertenecientes a los elementos estructurales del edificio [5] y [6].

2.1. Análisis de Estructuras en el diseño de un Edificio

Para que un edificio sea finalmente ejecutado en obra previamente, en fase de diseño, habrán sido consideradas un conjunto de diferentes alternativas, Figura 2.1. Estos diferentes diseños del edificio tratarán de dar solución a las acciones que deberá soportar la estructura tanto en su vida útil como durante la fase de construcción. De este modo, de entre todas las alternativas diseñadas, el ingeniero buscará la solución que dé una mejor respuesta a las acciones externas consideradas. Puesto que esta fase de búsqueda puede ser muy costosa si todos los diseños con considerados en detalle, comúnmente se utilizan modelos simplificados. Sin embargo, aún con la inclusión de simplificaciones, esta fase previa es una de las que más tiempo consumen en el diseño de una estructura.

Una vez el diseño con un modelo geométrico más apropiado ha sido seleccionado, éste deberá ser estudiado mucho más en detalle en la etapa final. Para ello, un conjunto de configuraciones estructurales del mismo serán analizadas, teniendo en cuenta diferentes propiedades para cada uno de los elementos estructurales que la componen. Este abanico de diferentes configuraciones estructurales de un mismo modelo geométrico trata de facilitar la búsqueda de la solución más eficiente, de modo que se encuentre la aproximación más rentable a la vez que segura.

En esta última etapa el ingeniero lleva a cabo un proceso de prueba y error, llevando a cabo un análisis mucho más realista, a la vez que costoso, que le aproxima más a la realidad, con unos resultados más precisos. De este modo, a partir de los resultados obtenidos, a no ser que el modelo matemático no sea suficientemente correcto, solamente tendrá que efectuar pequeñas modificaciones en el modelo estructural. Por lo tanto, esta última fase sirve al ingeniero para, a partir del modelo geométrico, obtener el modelo estructural con las diferentes dimensiones y secciones de los elementos estructurales.

A continuación se describen con un poco más de detalle algunas de las etapas a seguir en el cálculo de una estructura:

- El establecimiento del esquema estructural suele ser una simplificación de la estructura real a efectos de cálculo, fijando su disposición general, sus propiedades físicas, su geometría, dimensiones, condiciones de apoyo, etc.
- La determinación de las hipótesis de carga, es decir las diferentes combinaciones de las acciones que debe soportar la estructura, deben elegirse de forma que se simulen los efectos más desfavorables posibles.
- El cálculo de esfuerzos y deformaciones se obtiene a partir de cada hipótesis de carga, considerando el principio de equilibrio y la compatibilidad de deformaciones. Las deformaciones muestran como reacciona la geometría de la estructura ante las cargas y los esfuerzos muestran las tensiones ejercidas en la sección en cuestión.
- Por último, en la comprobación de resultados, se comprueba que una sección conocida es capaz de resistir las cargas que se han configurado en las hipótesis. Mediante el diseño estructural se produce una realimentación que repercute directamente en las propiedades de la estructura, de forma que dichas secciones deberán ser las apropiadas para que soporten las cargas.

Por lo tanto, el diseño estructural determina los esfuerzos que actúan sobre la estructura a través de la Resistencia, Mecánica de los Materiales y de la Teoría de la Elasticidad, asegurando que en ningún lugar de la misma se presentan tensiones o deformaciones que excedan los límites permitidos.

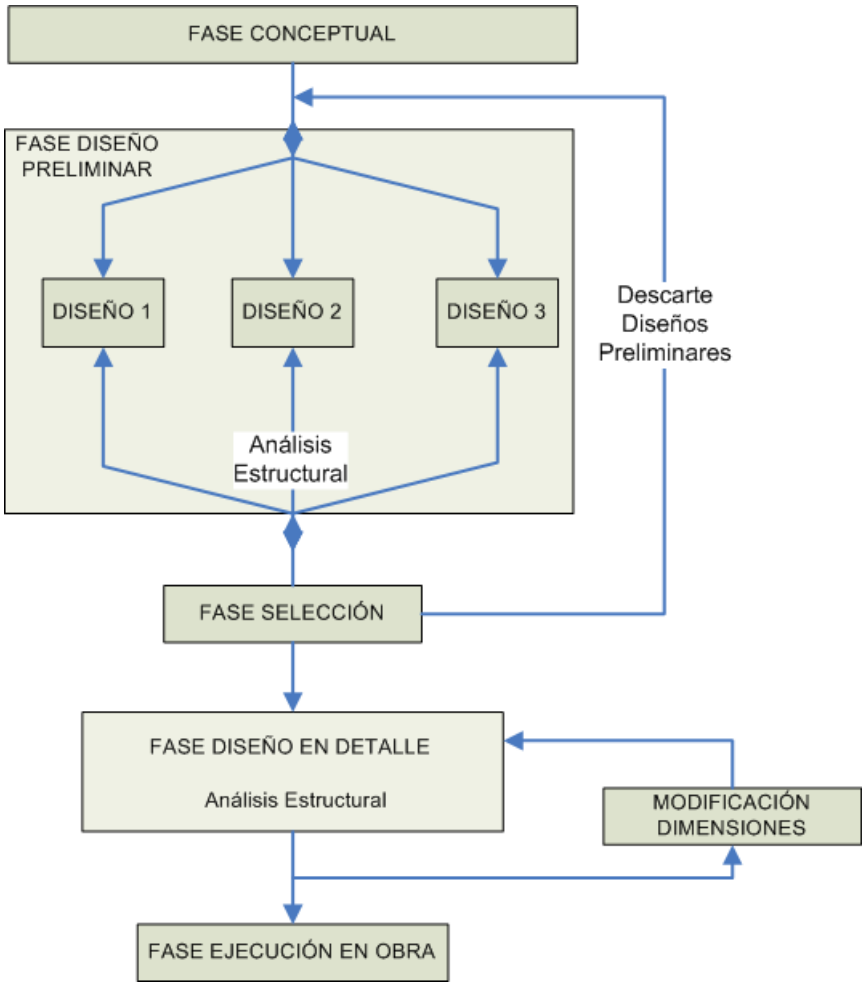

Figura 2.1. Diagrama de flujo describiendo el proceso de diseño de un edificio.

2.2. Definición de las cargas que actúan sobre la estructura

Las cargas son acciones capaces de producir tensiones en los distintos elementos de la estructura y se pueden clasificar en 2 grupos: acciones directas o acciones indirectas.

- Acciones directas (cargas permanentes y cargas variables).

Las cargas permanentes están constituidas por los pesos de los distintos elementos constructivos que forman el edificio y, por lo tanto, actúan constantemente con valor fijo tanto en posición como en magnitud. Los valores característicos de las cargas permanentes se determinarán de acuerdo con las dimensiones y pesos específicos de las distintas piezas.

Las cargas variables están constituidas por todas las fuerzas que son ajenas al edificio en sí, y pueden clasificarse en las siguientes tipologías:

- Cargas de explotación o de uso: son las propias del servicio que la obra debe rendir. Deben considerarse los pesos de los elementos que puedan actuar por razón de dicho uso: personas, muebles, elementos almacenados, vehículos, etc.

- Cargas climáticas: este tipo de cargas son las generadas por fenómenos meteorológicos como por ejemplo:
  - Acción del viento. Son el tipo de cargas producidas por las presiones y succiones que el viento origina sobre las superficies. Se admite que el viento en general actúa horizontalmente y en cualquier dirección. Debe considerarse en cada caso, la dirección o direcciones que produzcan las acciones más desfavorables sobre la estructura.
  - Acción de la nieve. Debe considerarse la máxima cantidad de nieve que podría acumularse sobre las cubiertas del edificio.
  - Cargas del terreno. Son las producidas por el empuje pasivo o activo del terreno sobre las partes del edificio en contacto con él. Debido a esto, se clasifican los terrenos en consideración a su comportamiento frente a las cargas de cimentación y a los efectos de determinar las presiones admisibles.

- Acciones indirectas o deformaciones impuestas (efectos térmicos, retracción, acciones sísmicas).

Las acciones indirectas son las originadas por fenómenos capaces de engendrar fuerzas de un modo indirecto. Pueden distinguirse los siguientes grupos:

- Acciones reológicas, producidas por deformaciones que experimentan los materiales a lo largo del tiempo dependiendo además del tipo de material del que consta la estructura.
- Acciones térmicas, producidas por las deformaciones debidas a las variaciones térmicas. Junto a las anteriores pueden obviarse en las estructuras formadas por pilares y vigas cuando se disponen juntas de dilatación a una distancia adecuada.
- Acciones por asiento, producidas por descensos diferenciales de los apoyos de la estructura, debidos al asiento del terreno de cimentación.
- Acciones sísmicas, producidas por los posibles movimientos sísmicos del terreno.

2.3. Introducción a los métodos matriciales

La evolución tecnológica de los computadores ha permitido su incorporación al mundo de la investigación, facilitando enormes avances en infinidad de campos de la ciencia, permitiendo a su vez el desarrollo de procedimientos numéricos apropiados para el uso de los mismos. En el campo del análisis de estructuras, la incorporación de la computación ha conducido al desarrollo de métodos basados en el álgebra matricial.

El empleo de la notación matricial presenta dos principales ventajas dentro del cálculo de estructuras:

- Permite, desde el punto de vista teórico, utilizar métodos de cálculo de una forma más compacta, precisa y al mismo tiempo completamente general. Los principios fundamentales del cálculo no se ven ocultados ni por las operaciones del mismo, ni por las distintas topologías geométricas de la estructura.
- Proporciona, desde el punto de vista práctico, un sistema adecuado de análisis y determina las bases idóneas para el desarrollo de un algoritmo de cálculo.

Sin embargo, los métodos matriciales no aportan sólo ventajas, ya que requieren de una gran cantidad de cálculo sistemático y su aplicación práctica necesita cierta adecuación del computador que va a realizar el esfuerzo numérico. Por lo tanto, su

principal campo de aplicación reside en estructuras grandes y complejas, donde los métodos manuales convencionales requieren un esfuerzo humano excesivo.

Para realizar cálculos complejos típicamente el ingeniero se encuentra con una amplia gama de posibles procedimientos alternativos de cálculo. La elección del más adecuado para el problema a resolver viene muchas veces condicionado por el nivel de aproximación requerido y por la experiencia y preferencia del ingeniero. Entre los distintos métodos que proporcionan una aproximación suficiente, el ingeniero debe tener en cuenta no sólo la complejidad computacional que cada uno implica, sino también la facilidad con que pueden detectarse y corregirse los errores cometidos.

En este sentido, las operaciones del álgebra matricial son fácilmente programables en un ordenador, requiriendo únicamente un conocimiento básico del cálculo matricial. Por otro lado, siempre existe la posibilidad de recurrir a rutinas desarrolladas por especialistas que están, generalmente, incorporadas en el mismo método de cálculo.

Por lo tanto, se puede afirmar que la introducción de los métodos matriciales en el cálculo de estructuras no implica la necesidad de grandes y difíciles conocimientos matemáticos. Tampoco exigen más principios estructurales que los elementales, tratados en infinidad de libros de texto referidos a esta área.

2.4. Métodos matriciales de análisis

Básicamente existen dos tipos diferentes de métodos matriciales para llevar a cabo el análisis de una estructura, llamados *Método de la Rigidez* (desplazamientos) y *Método de la Flexibilidad* (fuerzas o esfuerzos). Existe también un tercer método, llamado Método Combinado de Análisis, menos habitual que los dos anteriores, aunque presenta algunas ventajas cuando se aplica a ciertos tipos de estructuras.

Antes de entrar a describir cada método, el significado de la palabra análisis necesita ser correctamente definido. Algunos pueden interpretar el análisis como la determinación de los esfuerzos, y otros como la determinación de las deformaciones en varias partes de la estructura. Sin embargo, como hay una relación simple y única entre la forma deformada de la estructura y los esfuerzos, obtener un conjunto de resultados implica la determinación directa del otro. Para algunas estructuras es más fácil determinar primero los desplazamientos y después las fuerzas internas y para otras es más sencillo lo contrario.

Consideremos la estructura aporticada de esqueleto mostrada a continuación en la Figura 2.2.

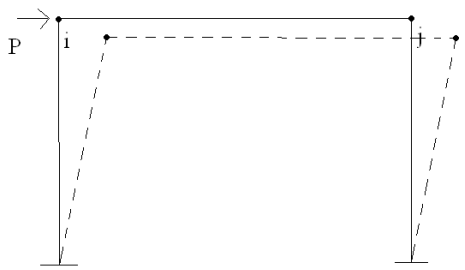

Figura 2.2. Estructura simple de ejemplo en alámbrico.

Supongamos que alguna de sus barras, la limitada por los nudos *i*, *j*, en este caso, se extrae del sistema en su forma deformada. Supongamos que además se han calculado, o se proporcionan, los desplazamientos de los nudos *i* y *j*. Entonces puede determinarse a partir de las relaciones fuerza-desplazamiento los esfuerzos en los nudos *i* y *j* o de cualquier punto intermedio situado entre ellos, así como la curva elástica (forma deformada) de esta barra. Por ejemplo, la siguiente formula nos da los esfuerzos que tienen lugar en el extremo *i* de esta barra en función de los desplazamientos de los nudos *i* y *j*.

$$\begin{pmatrix} P_i \\ P_j \end{pmatrix} = \begin{pmatrix} K_{ii}\ K_{ij} \\ K_{ji}\ K_{jj} \end{pmatrix} \begin{pmatrix} d_i \\ d_j \end{pmatrix}$$

Las matrices K_{ii} , K_{ij}, K_{ji} y K_{ij} forman la matriz de rigidez de la barra que une los nudos i,j y cuyos valores son funciones de la elasticidad del material, de la longitud y del área e inercia de su sección. Una vez conocidos los desplazamientos, el cálculo de los esfuerzos es bastante sencillo. La consideración anterior conduce al análisis por el Método Matricial de la Rigidez. Por el contrario, si el analista decide obtener primero los esfuerzos y a partir de ellos los desplazamientos, entonces sigue el Método de la Flexibilidad.

Los dos métodos comentados anteriormente satisfacen las ecuaciones de equilibrio de fuerzas y las condiciones de compatibilidad de los desplazamientos, pero no en el mismo orden. En el Método de la Rigidez primero se satisface el equilibrio de fuerzas y en el Método de la Flexibilidad lo hacen las compatibilidades de los desplazamientos. Por lo tanto, la selección de un método u otro depende de la estructura a analizar así como de la preferencia del analista. Para ciertas estructuras es fácil decidir qué método de análisis deberá seguirse, mientras que para otras puede ser preferible usar un método en ciertas partes de la estructura y otro en las otras. Este concepto sienta las bases para el Método Combinado de Análisis, que no se describirá en la presente Tesis de Master.

La herramienta de cálculo de la estructura que pondrá en marcha el interfaz gráfico a desarrollar utiliza el Método de la Rigidez o de las Deformaciones. Por ello dicho método se analiza de manera más detallada en el siguiente apartado.

2.5. Método de rigidez o de deformaciones

El Método de la Rigidez, también conocido como Método de las Deformaciones, no es más que un método para averiguar los movimientos de cada uno de los nudos que componen la estructura, al aplicar en ellos unas cargas exteriores conocidas. De este modo, al conocer los movimientos de los nudos extremos de cada una de las barras se puede estudiar cada uno de los elementos que configuran la estructura por separado y determinar las deformadas y los esfuerzos de todos los puntos intermedios que puedan interesarnos.

La secuencia a seguir para ello es siempre la misma y la detallamos a continuación:

- Primeramente se han de expresar las reacciones de los nudos que delimitan cada barra en función de los movimientos de las mismas. Dichas relaciones son lineales y se establecen a través de la llamada matriz de rigidez de una barra en ejes locales a la misma. Esta matriz se construye analizando los distintos modos de deformarse una barra.
- A continuación hemos de cambiar las relaciones anteriores del sistema de coordenadas local a la barra al sistema de coordenadas global. Por lo tanto tras

construir la matriz de transformación de cada barra pasaremos la matriz de rigidez la barra al sistema de coordenadas global. A partir de ahí, las reacciones sobre los nudos extremos de cada barra vendrán expresadas como funciones lineales de los movimientos globales de los mismos.

- Por hipótesis, la estructura está en equilibrio y, por lo tanto, también lo están sus nudos. Ese equilibrio no se perturba suprimiendo las sustentaciones o enlaces exteriores de la estructura, siempre que en los nudos correspondientes apliquemos las reacciones que ejercían dichas sustentaciones antes de suprimirlas. Al estar, por hipótesis, los nudos en equilibrio, en cada nudo debe ser nula la suma de la carga exterior que actúa sobre él y de las fuerzas que ejercen sobre él mismo las barras que allí confluyen. Puesto que estas últimas fuerzas las tendremos expresadas como funciones lineales de los movimientos de los nudos, esta consideración de equilibrio conducirá a un sistema de ecuaciones lineales de la forma Kd=p*, donde el vector incógnita d representa, como ya hemos comentado, los desplazamientos de los nudos de la estructura. La parte derecha del sistema de ecuaciones, p, almacena las cargas exteriores que actúan en los nudos, mientras que la matriz de coeficientes del sistema, K, constituye la llamada matriz de rigidez.
- Algunos elementos del vector de hipótesis no serán conocidos a priori, más exactamente los de aquellos nudos correspondientes a las reacciones de sustentación. Sin embargo, algunas componentes del movimiento de esos nudos, que sirven de sustentación, cumplen algunas condiciones y al expresarlas complementaremos todos los elementos.
- Finalmente bastará con resolver el sistema de ecuaciones deducido para obtener los movimientos de los nudos.

Capítulo 3

Tecnologías de las Aplicaciones Distribuidas

Previamente a la aparición de los ordenadores personales en los años 80, todas las aplicaciones informáticas se ejecutaban por medio de computadoras muy caras y aparatosas, cuyo acceso estaba compartido con el resto de usuarios de la empresa u organización. Ante este escenario, comenzaron a extenderse el uso de terminales muy sencillos que daban acceso a esas grandes computadoras, de manera que, las aplicaciones eran ejecutadas remotamente empleando comandos.

Con la aparición de los ordenadores personales y debido a las pobres prestaciones de las redes de comunicaciones, las arquitecturas monolíticas eran las únicas utilizadas para el desarrollo de aplicaciones informáticas, donde todo el cómputo era llevado a cabo en la misma máquina.

En la década de los 90, la popularización de las redes de interconexión y su evolución en cuanto a las prestaciones ofrecidas facilitó la aparición de nuevas tecnologías para el desarrollo de aplicaciones. Además, la irrupción de Internet como la gran red global produjo el auge de un novedoso paradigma de programación basado en Computación Distribuida. Por Computación Distribuida se entiende la fragmentación de un programa en partes independientes de modo que cada una resida en una computadora diferente, comunicándose por medio de una red de interconexión. La aparición de aplicaciones distribuidas es una evolución natural, resultado del uso de estas nuevas redes de comunicación, que permiten comunicaciones más eficientes con mayores anchos de banda y menores latencias.

Nótese que en un entorno distribuido es de vital importancia el establecimiento de mecanismos estándar que regulen la interacción entre las diferentes computadoras implicadas. Los protocolos de comunicación a emplear están por encima de las

particularidades de cada plataforma, de modo que independientemente de su naturaleza, cualquier computadora pueda comunicarse con el resto.

Otro aspecto interesante, en el cual aún se continúa trabajando, es la migración de código hacia un computador remoto de modo portable. El envío de software para que sea ejecutado remotamente resulta muy atractivo, pero aún a día de hoy no existe una solución clara, ya que los códigos compilados plantean muchos problemas de incompatibilidad. La utilización de códigos intermedios, que sean interpretados en el destino, es una de las soluciones más extendidas en la actualidad.

Puesto que el trabajo desarrollado dentro del marco de esta Tesis consiste en un Servicio Grid distribuido accesible a través de la Web, en el resto del capítulo se analiza el estado del arte en cuanto a arquitecturas y tecnologías de las aplicaciones distribuidas.

3.1. Arquitecturas Distribuidas

Por arquitectura distribuida se entiende tanto los componentes hardware como software que integran un sistema distribuido. Las arquitecturas distribuidas cubren toda una serie de niveles, desde el más bajo, que incluye la interconexión física de las CPUs, hasta el nivel más alto, donde se definen los mecanismos de comunicación entre los procesos.

El diseño de aplicaciones distribuidas puede adoptar diferentes arquitecturas o categorías que serán analizadas a continuación.

3.1.1. Cliente-Servidor

Una arquitectura cliente-servidor típicamente define dos tipos de roles en un sistema distribuido. Este tipo de arquitecturas presentan una asimetría muy evidente, ya que la carga se encuentra principalmente del lado del servidor. Aunque este paradigma podría utilizarse para aplicaciones residentes en un mismo computador, su verdadera potencia aflora en entornos distribuidos. Un ejemplo clásico de arquitectura Cliente-Servidor es el sistema de consulta de información a través de Internet, en este caso intervienen un navegador y un servidor Web.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay ningún tipo de distribución, ni a nivel físico ni a nivel lógico. Cada cliente podrá estar conectado a uno o varios servidores, a los cuales enviará los datos necesarios asociados a cada petición. La labor del servidor será la de atender estas peticiones, procesarlas y retornar la información requerida.

En cuanto a las ventajas ofrecidas por este tipo de sistemas nos encontramos con las siguientes:

- Las arquitecturas cliente-servidor permiten una distribución de roles y responsabilidades entre un conjunto de computadores independientes, las cuales solamente tienen conciencia unas de las otras gracias a la red. De este modo, la mantenibilidad del sistema global mejora sustancialmente, ya que es posible la substitución, actualización o reparación del servidor sin que los clientes sean afectados.
- El acceso a los recursos y la integridad de los datos son controlados por el servidor. Esto proporciona un control centralizado, de forma que el servidor garantizará el acceso solamente a clientes autorizados y un programa cliente defectuoso o no autorizado no podrá dañar el sistema.

- Debido a que el almacenamiento es centralizado, la actualización de la información es mucho más sencilla que en otro tipo de entornos como pueden ser los P2P [7]. En estos sistemas, las actualizaciones deberán ser propagadas por la red a todos los agentes, lo que requiere mucho más tiempo y puede provocar inundaciones en la red.

Por otro lado, no son todo ventajas lo que aporta el uso de estas arquitecturas:

- Se trata de sistemas cuya escalabilidad suele ser limitada. Conforme aumentamos el número de peticiones hacia un servidor concreto, el sistema puede verse seriamente afectado, tanto por la congestión de la red como por el saturamiento del propio servidor. Por ejemplo, este tipo de problema no existe en sistemas P2P.
- El paradigma cliente-servidor posee más puntos débiles que las arquitecturas más descentralizadas. El servidor se trata de un elemento crítico, ya que la eventualidad de una caída suya provocara el fallo de todo el sistema.

3.1.2. Arquitecturas multicapa

La arquitectura cliente-servidor, descrita en el apartado anterior está compuesta solamente por dos capas. Una arquitectura multicapa sería la compuesta por diversos agentes software. Por ejemplo, una aplicación que utilice algún middleware que interactúe entre los clientes y algún servicio empleará una arquitectura multicapa. Este tipo de sistema compuesto por 3 agentes es el más extendido después de la arquitectura cliente-servidor.

Un sistema compuesto por tres capas integra habitualmente un interfaz de usuario, un agente que lleva a cabo la lógica funcional y otro que realiza el almacenamiento y el acceso a los datos. Por lo tanto un sistema tricapa, además de una arquitectura también se considera un patrón de diseño.

Independientemente de las ventajas ofrecidas por la utilización de sistemas modulares, las arquitecturas multicapa permiten la actualización o substitución de cualquier componente sin que ello afecte al resto. Esta característica favorece la adaptabilidad de las aplicaciones a las nuevas tecnologías que vayan apareciendo en el mercado.

3.1.3. Arquitecturas P2P

Una arquitectura P2P [7] trata de explotar la alta conectividad de un conjunto de recursos, de modo que se aproveche el potencial de todos sus participantes. Al contrario que una arquitectura centralizada, este tipo de arquitectura será mucho más escalable y no dependerá de un reducido número de servidores. Las redes P2P son típicamente empleadas para la conexión de nodos ubicados en la periferia de Internet, empleando largos canales de comunicación. Aunque las redes P2P tradicionalmente han estado asociadas a la distribución de contenidos, actualmente se están empleando en muchas otras aplicaciones: sistemas de computación P2P, transmisión de datos en tiempo real, telefonía, etc.

Existen varias causas por las que los sistemas P2P cada vez tienen una mayor presencia. Por un lado, el hecho de que cada vez dispongamos de ordenadores personales más potentes y con mayor capacidad de almacenamiento, los cuales no aprovechamos. Por otro lado, la existencia de redes cada vez más rápidas que dan cierta ventaja para el desarrollo de aplicaciones descentralizadas.

En una red P2P desaparecen los conceptos de clientes y servidores, ya que en este caso solamente hablamos de pares, que simultáneamente desempeñan funciones de servidores y clientes dentro de la red. Este tipo de modelo difiere

enormemente de los sistemas centralizados, donde todo el flujo de información siempre iba dirigido a unos pocos nodos de la red.

Existe un amplio abanico de tipologías de sistemas P2P, desde sistemas parcialmente centralizados que requieren la existencia de algún tipo de nodo especial, sistemas descentralizados puros donde todos los nodos son exactamente idénticos, y sistemas híbridos. Por otro lado, también aparece otra división que habla de sistemas estructurados y desestructurados. Un sistema P2P será desestructurado cuando la ubicación de los recursos no tenga ninguna relación con la de la red lógica definida. En este tipo de sistemas es más complicada la localización de los recursos, ya que cada nodo solamente conoce a sus vecinos. Por otro lado, un sistema P2P será estructurado cuando la ubicación de cada nodo tenga una relación con cierta red lógica definida. Este tipo de sistema se suele emplear para la distribución de contenidos, y típicamente se establecen correspondencias entre los nodos de la red y el contenido que albergan.

3.2. Estado del Arte en las Tecnologías para el Desarrollo de Aplicaciones Distribuidas

En los primeros sistemas cliente-servidor desarrollados, la abstracción de las comunicaciones entre los procesos y el acceso a la red se superó empleando librerías y controladores específicos. Estos pequeños componentes software simplemente proporcionaban una capa de abstracción más alta para el acceso a cada sistema concreto, con lo que cada servidor proporcionaba su librería.

Con la aparición de los sistemas multicapa la situación se tornaba mucho más complicada, ya que las comunicaciones entre los clientes y los middlewares no podían llevarse a cabo del mismo modo que en los sistemas cliente-servidor tradicionales. En cambio, apareció la necesidad del uso de modelos estándar para la mejor integración en la plataforma de desarrollo de aplicaciones cliente. La primera tecnología que propició esta estandarización fue la aparición de los RPC (Remote Procedure Call) [8]. De este modo, las aplicaciones clientes podían hacer uso de llamadas a función que provocaban una invocación remota del mismo modo que lo harían si fuera local.

Con la aparición de la programación orientada a objetos la computación distribuida también trato de adaptar esta nueva metodología. En consecuencia, apareció el concepto de objetos distribuidos, que no son más que pequeñas piezas de software que pueden residir en diferentes computadores interconectados o bien en diferentes procesos dentro de una misma máquina. De este modo, un objeto requerirá el servicio de otro por medio del envío de un mensaje que describa su petición a la máquina remota donde resida el objeto servidor. Éste recibirá el mensaje, llevará a cabo la tarea solicitada, y devolverá el resultado en un mensaje de vuelta.

A continuación se incluye el estado del arte en cuanto a las tecnologías disponibles para el desarrollo de aplicaciones distribuidas.

3.2.1. COM y DCOM

A principio de los años 90 Microsoft centró sus esfuerzos en el desarrollo de un sistema de objetos distribuidos al que denominó OLE (Object Linking Package) [9]. Este sistema permitía el desarrollo de un sistema de publicación de escritorio para el intercambio de información entre aplicaciones diferentes. Además, dicho sistema permite embeber unas aplicaciones dentro de otras, de modo que aparezca incrustada en el interfaz. Por ejemplo, si una hoja de cálculo quisiera ser editada desde un procesador de texto, éste cargaría el procesador de hojas de cálculo y lo mostraría dentro de su ventana.

Microsoft rápidamente fue consciente que el uso de OLE no era suficiente para el desarrollo de sistemas de objetos distribuidos, ya que requería mecanismos estándar para el empaquetado de componentes. Además, OLE no solucionaba el problema de interoperabilidad entre componentes implementados en diferentes lenguajes. En respuesta a estas necesidades, Microsoft desarrolló las tecnologías conocidas como COM (Component Object Model) [10] para que proporcionaran la infraestructura necesaria sobre la que OLE se apoyaría.

COM consiste en la tecnología predominante en arquitecturas orientadas a componentes. Su primer objetivo fue asistir al desarrollo de nuevas interfaces gráficas de usuario. Un entorno optimizado para el desarrollo de este tipo de aplicaciones requiere el uso de componentes de alto nivel previamente definidos. Es por ello que COM se trata más de una arquitectura orientada a componentes que una arquitectura distribuida.

La interfaz de los componentes COM serán descritos empleando una especificación independiente de cualquier lenguaje de programación llamada IDL (Interface Description Language) [11]. Mediante este lenguaje son descritos las interfaces, los métodos y las propiedades asociadas a un componente. Las aplicaciones cliente interactuarán directamente por medio de la definición IDL de un objeto COM, en vez de tener que enfrentarse con los detalles específicos de la implementación del componente en cuestión, tales como su lenguaje de programación.

COM soporta los siguientes tres tipos de servidores para la implementación de componentes, Figura 3.1. :

- Servidores intraproceso. Un servidor intraproceso es normalmente implementado por medio de una DLL (Dynamic Linked Library) y es ejecutado dentro del mismo espacio de memoria que la aplicación. La sobrecarga de la invocación a un servidor de este tipo es mucho mas pequeña, ya que está ubicado dentro del mismo proceso cliente. Un servidor intraproceso es comúnmente conocido como ActiveX [12].
- Servidores locales. Un servidor local es ejecutado en un proceso separado dentro de la misma máquina cliente. La comunicación entre la aplicación y el servidor local es llevada a cabo mediante el uso del runtime de COM, por medio de protocolos de intercomunicación de procesos de alta velocidad. En este tipo de servidores la sobrecarga introducida es de un orden de magnitud superior a la de los servidores intraproceso.
- Servidores remotos. Un servidor remoto será ejecutado en una máquina distante. Para ello aparece la arquitectura DCOM [13], que no es más que una extensión de COM que ofrece una infraestructura basada en RPC para la comunicación entre procesos remotos. En este caso, el uso de servidores remotos introduce una sobrecarga de un orden de magnitud superior con respecto al uso de servidores locales.

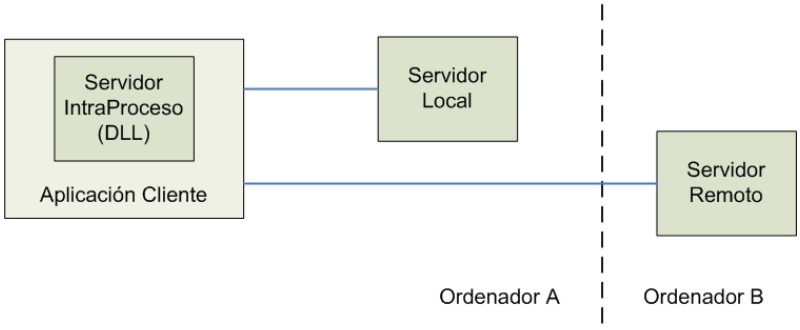

Figura 3.1. Esquema de interconexión COM

3.2.2. Java RMI

La gran difusión de Java en los últimos años ha provocado el desarrollo de múltiples arquitecturas distribuidas orientadas a objetos basadas en este lenguaje. La más famosa de todas, es la llamada JavaRMI (Java Remote Method Invocation) [14]. Este entorno ofrece mecanismos simples para el desarrollo de aplicaciones distribuidas compuestas por componentes desarrollados en Java.

Mediante RMI, un programa Java tendrá acceso fácilmente a la funcionalidad ofrecida por un objeto que resida en una aplicación remota. Para ello, la aplicación remota deberá emplear los mecanismos ofrecidos por el API para exportar este objeto. La exportación implicará que ese objeto estará disponible en la red, esperando conexiones a un puerto determinado. A partir de ahí, un cliente podrá conectarse a ese puerto teniendo acceso a los métodos y atributos del objeto publicado. El entorno RMI enmascara todos estos detalles de bajo nivel de acceso a la red y de comunicación, de modo que el cliente invocará a los métodos remotos del mismo modo que si se trataran de invocaciones locales.

La invocación de métodos remotos se basa en un mecanismo de empaquetado de parámetros llamado *marshaling*, basado en la serialización de objetos típica de Java. Una vez que el cliente realiza una invocación remota, queda a la espera de la que sea atendida su llamada. A partir de ahí, se llevará a cabo de forma remota la invocación de manera totalmente transparente al cliente. Primeramente, en la máquina cliente se realizará un empaquetado de los parámetros de entrada. A continuación, se produce la conexión y envío por la red de los parámetros previamente empaquetados. Una vez que los datos son recibidos en la máquina destino, se lleva a cabo un desempaquetado e invocación del método requerido. Finalmente, el resultado de la invocación también es empaquetado para ser enviado a la máquina cliente, la cual lo desempaquetará y servirá en su recepción.

La arquitectura del sistema RMI se estructura en cuatro niveles, como muestra la Figura 3.2., que son detallados seguidamente:

- Nivel de aplicación. Se trata de la capa de mayor nivel y hace referencia a las invocaciones de acceder y exportar objetos remotos. Cualquier aplicación que quiera que sus objetos sean exportados para que sean accedidos remotamente deberá marcarlos mediante el uso de la interfaz *java.rmi.Remote*.
- Nivel middleware. Es la capa compuesta por las piezas software llamadas stubs y skeleton. Las llamadas a objetos remotos y el empaquetado y desempaquetado de parámetros es resuelto en este nivel. Se trata de la capa con la que interactúa directamente el nivel de aplicación.
- Nivel de acceso remoto. Este nivel es el encargado del manejo y gestión de las referencias remotas de objetos, así como de la parte semántica de las invocaciones. Por otro lado, también es responsable de la gestión de la replicación de objetos y realización de tareas específicas, como el establecimiento de las persistencias semánticas y las estrategias adecuadas para la recuperación de conexiones perdidas.

Figura 3.2. Arquitectura Java RMI

- Nivel de transporte. Es el responsable de realizar las conexiones necesarias y la gestión del transporte de los datos de una máquina a otra. El protocolo de transporte subyacente en RMI es JRMP (Java Remote Method Protocol) [15], que solamente puede ser empleado en aplicaciones Java.

3.2.3. CORBA

Tradicionalmente CORBA [16] ha sido la tecnología dominante en el desarrollo de aplicaciones distribuidas. Esta tecnología nació de un consorcio llamado OMG (Object Management Group) que agrupó las empresas del sector para el desarrollo de una especificación estándar. Esta especificación se materializó en el estándar OMA (Object Management Architecture) [17], del que CORBA es una parte fundamental. CORBA es el primer intento serio de normalización de la semántica asociada a las invocaciones a métodos entre diferentes aplicaciones, sean remotas o no. El estándar definido permite llevar a cabo estas invocaciones independientemente de la plataforma y lenguajes de programación de las aplicaciones implicadas. Las llamadas a objetos mediante CORBA son totalmente transparentes, de modo que el objeto cliente no se tiene que preocupar de la ubicación del objeto servidor.

Aunque el estándar CORBA estaba orientado inicialmente a arquitecturas orientadas a objetos, en su versión 3.0 también integra modelos orientados a componentes. Sin embargo este tipo de sistemas orientados a componentes ha tenido poca presencia en las aplicaciones comerciales.

CORBA hace uso del lenguaje IDL (Interface Description Language) para especificar los interfaces de los objetos a publicar. A partir de este interfaz, CORBA especifica mapeos de IDL a implementaciones estándar, llamadas ORBs (Object Remote Broker), en C, C++ y Java. Además, existen mapeos no estándar a prácticamente cualquier lenguaje de programación, tales como Perl, Visual Basic, etc.

La especificación CORBA se basa en la utilización de componentes software llamados ORBs ("Object Request Broker"), mediante los cuales una aplicación puede interactuar con otros objetos. En la práctica, una aplicación solamente deberá inicializar el ORB y acceder a su adaptador de objetos, el cual mantiene el estado de las conexiones, referencias remotas, tiempos de vida, etc. Este adaptador de objetos es utilizado para instanciar objetos generados a partir de los interfaces IDL, donde se produce una traducción de estas interfaces de alto nivel a código específico para una plataforma y un lenguaje de programación determinado. Esta traducción es necesaria para proporcionar un acceso limpio a las interfaces definidas por una infraestructura CORBA.

Para emplear el estándar definido por CORBA, un desarrollador deberá implementar adicionalmente en IDL las interfaces que desea exportar de un objeto. A partir de estas representaciones, existen herramientas en el mercado que permiten su traducción a código específico en un leguaje y para una plataforma concreta. Estos códigos se utilizarán en los clientes como una representación del objeto servidor, y se encargarán de llevar a cabo transparentemente la invocación remota. De este modo, un cliente realizará dicha llamada como si se tratase de un objeto local, y estos objetos generados llevarán a cabo el acceso a la red y el transporte de la información automáticamente. Análogamente a RMI, la porción del software cliente se llama *stub* y la del servidor *skeleton*.

En resumen, las principales aportaciones que presenta CORBA para el desarrollo de sistemas distribuidos son las siguientes:

- Amplio soporte a las llamadas a métodos entre objetos de diferentes lenguajes de programación, tales como Java, C, C++, etc.

- Las aplicaciones desarrolladas mediante CORBA podrán coexistir en diferentes plataformas (UNIX, Windows, etc.).
- Ofrece una implementación IDL independiente de la plataforma para la descripción de las interfaces de objetos CORBA.
- Permite el descubrimiento e invocación de objetos CORBA de manera dinámica entre clientes y servidores.

3.2.4. Arquitecturas Orientadas a Servicios

Las arquitecturas SOA (Service-oriented Architecture) [18] son un patrón de diseño para la definición de interrelaciones débilmente conectadas entre productores y consumidores. Realmente se trata de un estándar que no tiene ninguna relación con ninguna implementación software, paradigma de programación o tecnología, aunque es comúnmente confundido con los Servicios Web [15].

En un entorno SOA, los servicios podrán ser accedidos sin conocer ni su implementación ni su plataforma subyacente. Estos conceptos pueden ser aplicados en comercio, administración y en otro tipos de sistemas productor/consumidor.

Una Arquitectura Orientada a Servicios supone una metodología de programación que proporciona un marco de trabajo común para el diseño, definición y uso de servicios vía Internet. En un entorno SOA, los agentes conectados compartirán sus recursos con el resto, presentándose como servicios independientes que podrán ser accedidos de modo estándar. La implementación de un sistema SOA está directamente relacionada con los Servicios Web, que son detallados en el siguiente capítulo. Sin embargo, además de los Servicios Web, cualquier otra implementación orientada a servicios, basada en el uso de diferentes protocolos, podría ser empleada.

Comparando SOA con las arquitecturas orientadas a servicios empleadas en las tecnologías previamente analizadas, este nuevo paradigma permite la integración de aplicaciones débilmente conectadas y altamente interoperables por medio de servicios.

3.2.5. Servicios Web

Los Servicios Web [19] se han convertido en el estándar más aceptado para el desarrollo de aplicaciones distribuidas. Los mecanismos definidos mediante los Servicios Web son, en la actualidad, los más utilizados para el intercambio de mensajes en la interacción entre diferentes aplicaciones remotas.

Estas tecnologías, basadas en el uso de estándares abiertos de Internet, permiten el desarrollo de aplicaciones distribuidas auto descriptivas, de modo que se puedan publicar, ubicar e invocar desde cualquier punto de la red, o desde el interior de una red local. Los Servicios Web combinan las mejores características de la programación orientada a componentes y la Web, permitiendo el desarrollo de elementos software de una alta versatilidad y capacidad de reutilización.

El acceso a los servicios Web se realiza a través de protocolos de Internet multiplataforma, ampliamente difundidos y utilizados, como son HTTP (HyperText Transfer Protocol) y SMTP (Simple Mail Transfer Protocol), además de otro tipo de estándares XML. Gracias a ello, tanto los servicios como los clientes de este tipo de sistemas podrán estar implementados en cualquier lenguaje de programación y bajo cualquier plataforma. De este modo, podemos, por ejemplo, comunicar de manera trivial un cliente implementado en C++ con un servicio programado en Java. Además, el uso del protocolo HTTP, empleado para transmitir los mensajes entre el cliente y el servicio, es plenamente tolerado por proxies y firewalls.

La utilización de arquitecturas basadas en los Servicios Web permite pasar de aplicaciones estrechamente cohesionadas a entornos totalmente desacoplados. Tradicionalmente, los componentes estrechamente ligados necesitaban vincularse en la fase de diseño o programación, mientras que esta nueva filosofía permite la vinculación directamente en tiempo de ejecución. Este tipo de aplicaciones totalmente desacopladas permiten una mayor escalabilidad, manejabilidad y capacidad de actualización.

Uno de los objetivos de los Servicios Web consiste en acercar a los usuarios al proceso de creación de aplicaciones que puedan descubrirse e interactuar dinámicamente, sin ningún tipo de restricción. 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.

3.2.5.1. Arquitectura de los Servicios Web

La arquitectura de los Servicios Web gira entorno a tres conceptos fundamentales, como son ubicación, descripción y llamada. Estos conceptos son agrupados en capas, como podemos ver en la Figura 3.3. A continuación se detallan las tecnologías y particularidades de cada una de estas capas.

Para que una aplicación cliente pueda acceder a un Servicio Web primero es necesario conocer su ubicación física, por medio de lo que se conoce como descubrimiento. Esto se lleva a cabo mediante el estándar conocido como UDDI (Universal Description Discovery and Integration) [20]. A partir de ahí, el cliente deberá poder conocer la funcionalidad y propiedades del Servicio Web con el que desea interactuar, lo cual se produce mediante la descripción del mismo y se lleva a cabo mediante el estándar WSDL (Web Services Description Language) [21]. Finalmente, el cliente deberá realizar la invocación remota oportuna, proporcionándole los parámetros de entrada necesarios y recibiendo la salida adecuada. Esto es en sí propiamente la llamada remota y es llevada a cabo utilizando las tecnologías SOAP (Simple Object Access Protocol) [22], las cuales se sitúan por encima del nivel de transporte, empleando los protocolos de Internet ampliamente soportados basados en HTTP para transferir los datos de las llamadas.

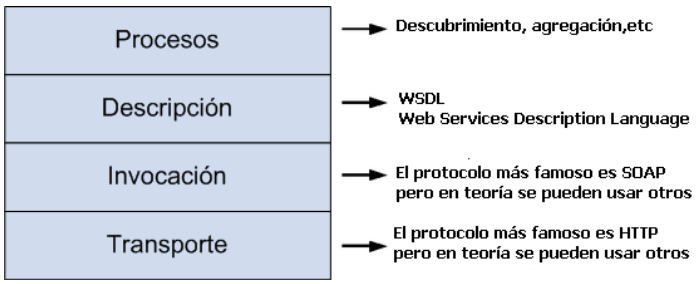

Figura 3.3. Arquitectura de los Servicios Web

3.2.5.2. SOAP

SOAP es el protocolo de comunicaciones basado en XML utilizado para el intercambio de información entre aplicaciones software distribuidas, de manera independiente de su sistema operativo subyacente, de los lenguajes de programación o de los modelos de orientación a objetos. En su descripción, SOAP se presenta como un protocolo ligero para el intercambio de información estructurada en un entorno distribuido y descentralizado. Por otro lado, SOAP no define ningún modelo de programación ni la semántica de una implementación específica, como podría ser una API, sino que define un mecanismo sencillo para el empaquetado de datos dentro de mensajes basados en esquemas XML.

Gracias a la interoperabilidad ofrecida por SOAP, los sitios Web podrán ofrecer servicios Web accesibles al resto de aplicaciones, sin necesidad de la intervención humana. De este modo, una aplicación podrá ensamblar o combinar varias de estas soluciones y obtener un valor añadido.

De este modo, solamente mediante un analizador XML y un servidor HTTP, es posible la generación de un objeto de negociación SOAP. Al igual que en CORBA, en los Servicios Web aparece el concepto de ORB, como sistema que asiste en la invocación remota, gestiona tiempos de vida, etc.

SOAP es más que un protocolo extensible. Ningún protocolo puede garantizar un funcionamiento tan uniforme en cualquier tipo de arquitectura, plataforma o infraestructura de red. De este modo, establece mecanismos para el envío de mensajes de tipo RPC.

Los mensajes SOAP son básicamente transmisiones unidireccionales con el contenido formateado a partir de esquemas XML con espacios de nombres adecuados para poder llevar a cabo llamadas remotas. A continuación se puede muestra un mensaje SOAP sencillo:

```
<SOAP-ENV:Envelope xlmns:SOAP-ENV=http://schemas.xmlsoap.org/soap/envelope
       SOAP-ENV: EncodingStyle=http://schemas.xmlsoap.org/soap/encoding>
       <SOAP-ENV:Body>
      <m:GetPrecioProducto xmlns:m=”Some-URI”
         <symbol>Leche</symbol>
      </m:GetPrecioProducto>
       </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Como se puede apreciar dentro del cuerpo del mensaje, especificado mediante la etiqueta *SOAP-ENV:Body*, se está invocando a un procedimiento remoto llamado *GetPrecioProducto*() al que se transmite un parámetro con valor Leche.

3.2.5.3. WSDL

A pesar de que SOAP proporciona un modo estándar de transportar mensajes para el uso de servicios Web, no proporciona ninguna manera de describir qué tipo de métodos publica ni qué parámetros necesita cada uno. Al comienzo de la implantación de SOAP, cuando los desarrolladores empezaron a implementar Servicios Web, se dieron cuenta de que el único modo de que un cliente conociera el modo de invocar un servicio era publicando una muestra de documento SOAP que ilustrara la llamada al servidor.

WSDL nace de los esfuerzos de Microsoft e IBM para solucionar el problema de descripción de Servicios Web. WSDL incluye la definición y descripción de los Servicios Web de igual modo que los ficheros de cabecera describen las bibliotecas binarias tradicionales.

WSDL proporciona una gramática para la descripción de servicios como un conjunto de puntos finales que intercambian mensajes. Por lo tanto, un documento WSDL está compuesto por definiciones que presentan un servicio como un conjunto de lo que se denomina puertos. Cada puerto tendrá un enlace específico al conjunto abstracto de operaciones y mensajes que el servicio implementará. Las operaciones representan el conjunto de servicios que se publican y que serán accesibles por la red, mientras que los mensajes incluyen los parámetros de entrada y salida a cada una de estas operaciones.

Aunque WSDL no está relacionado con ningún tipo de esquema específico para la definición de operaciones y mensajes, típicamente se emplean los esquemas XSD.

3.2.5.4. UDDI

Una vez determinada la forma de invocar a un Servicio Web y de como describirlo, se necesitan mecanismos estándar para que el resto de aplicaciones puedan localizarlo. Análogamente a los motores de búsqueda, la especificación UDDI describe como un proveedor puede publicar la existencia de un Servicio Web en un directorio.

Un usuario de un Servicio Web podrá buscar en el registro UDDI para tratar de localizar el servicio que necesita. Para ello, se establecen una serie de categorías jerarquizadas, las cuales estructuran las taxonomías semánticas empleadas en la descripción de los servicios. Las taxonomías no son más que el vocabulario empleados para asignar un significado concreto a un término, de modo que mediante la introducción de términos, sea fácil buscar en la estructura jerarquizada el Servicio Web que lleve a cabo la tarea solicitada.

De este modo, una vez encontrado el Servicio Web buscado, el siguiente paso consistirá en analizar sus detalles, su descripción y el contrato para orquestar la utilización del mismo.

3.2.5.5. Utilización de un Servicio Web

El acceso a los Servicios Web es idéntico a las páginas Web convencionales, utilizando los tradicionales URIs (Uniform Resource Identifiers). Estas direcciones pueden ser previamente conocidas u obtenerlas mediante UDDI. 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, por 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 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 su propio *stub* 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.

3.3. Comparativa de las Tecnologías Distribuidas

En este capítulo se ha analizado el estado del arte de las diferentes tecnologías existentes en el mercado para el desarrollo de aplicaciones distribuidas. Observando todo el espectro analizado, COM representa la arquitectura orientada a componentes dominante, mientras que CORBA, hasta la aparición de los Servicios Web, era la arquitectura claramente predominante en el desarrollo de sistemas distribuidos.

CORBA consistía en una arquitectura con una complejidad asociada que provocó la búsqueda de nuevas fórmulas, las cuales provocaron, a principios del año 2000, el desarrollo y implantación de los Servicios Web. Realmente, los Servicios Web suponen, a día de hoy, la tecnología más fácil de utilizar y más independiente de plataforma que existe. Mediante el uso de protocolos estándar, permite la interconexión de cualquier tipo de aplicación en cualquier tipo de entorno. Por todo ello, ésta ha sido la tecnología empleada en el desarrollo del middleware Grid de análisis de estructuras enmarcado en esta Tesis.

Capítulo 4

Computación en Grid

Tradicionalmente, las necesidades computacionales para la resolución de problemas científicos o de ingeniería han superado las prestaciones ofrecidas por los sistemas informáticos del momento. Estas necesidades han propiciado la aparición de tecnologías basadas en la Computación de Altas Prestaciones, empleando inicialmente sistemas multiprocesador, pasando por los sistemas multicomputador, los clusters de PCs y actualmente adoptando las tecnologías Grid [23].

Los retos cada vez más ambiciosos a los que se está enfrentando la ciencia en las últimas décadas, han puesto de manifiesto la necesidad de colaboración entre equipos multidisciplinares de científicos, perteneciendo a diferentes organizaciones e incluso países. En este escenario, fue donde se gestaron las bases de las tecnologías Grid actuales. Inicialmente el concepto del Grid estaba asociado al de las tecnologías distribuidas que permitían la colaboración entre grupos de científicos, por medio de la compartición de instrumental específico, como telescopios, microscopios, aceleradores de partículas, etc. Ante esa coyuntura, se fueron introduciendo paulatinamente aplicaciones computacionalmente intensivas y con una necesidad de almacenamiento que desbordaba las capacidades de las que una sola organización disponía. Por lo tanto, apareció la necesidad de compartición, no solamente de instrumental científico, sino también de recursos computacionales para resolver problemas de gran dimensión.

La mejora sustancial de las prestaciones ofrecidas por las redes de comunicaciones actuales ha sido un factor clave para la evolución y difusión de las tecnologías Grid. De este modo, apareció la idea de interconectar distintos recursos, geográficamente distribuidos, para crear una infraestructura global que ofreciera servicios de computación, almacenamiento y ancho de banda. La computación Grid permite la compartición, selección y agregación de un amplio conjunto de recursos heterogéneos distribuidos y pertenecientes a distintos dominios de administración. En la actualidad, el despliegue de infraestructuras Grid está claramente asociado a la

computación intensiva de problemas de gran dimensión en ciencia, ingeniería y comercio.

Se puede hacer una analogía entre las tecnologías Grid y la Web, ya que la Web consiste en un servicio para la compartición de información por medio de Internet, mientras que el Grid se puede ver como un servicio para la compartición de potencia computacional, capacidad de almacenamiento y ancho de banda a través de Internet. Uno de los principales objetivos de las tecnologías Grid es el establecimiento de una interfaz homogénea y estándar, que permita el acceso a recursos heterogéneos geográficamente distribuidos. Estos recursos pueden ser de diferente naturaleza, como supercomputadores, clusters de PC's, sistemas de almacenamiento, aplicaciones software, bases de datos, etc.

Uno de los escollos principales a los que se han enfrentado las tecnologías Grid es la coordinación de las diferentes políticas de seguridad internas a cada uno de los diferentes dominios de administración que pueden formar la infraestructura. De este modo, estas tecnologías tratan de definir un nivel superior de seguridad respetando cada una de las políticas locales.

Cabe señalar también que las tecnologías Grid se encuentran actualmente muy extendidas en el ámbito académico y de investigación, empezando levemente a aplicarse en el mundo empresarial.

4.1. Estado del Arte de la Computación en Grid

Durante la última década, la computación basada en Grid está siendo ampliamente utilizada tanto en áreas académicas como científicas. Los requerimientos de cómputo y almacenamiento asociados a los problemas de gran desafío, abordados por la ciencia actual, han propiciado la proliferación de acuerdos de colaboración entre diferentes equipos científicos. Estos acuerdos abarcan la cesión de recursos computacionales para un objetivo común, apareciendo el concepto de entornos colaborativos.

Actualmente, la mayoría de los grandes proyectos de investigación están adoptando las tecnologías Grid como la infraestructura más adecuada para utilizar, compartir e integrar todos los recursos pertenecientes a los diferentes centros de investigación implicados.

A nivel europeo, el proyecto LCG[1] (Large Hadron Collider Computing Grid Project) [24] fue lanzado en 2004 con el objetivo de desarrollar y desplegar una infraestructura Grid que diera soporte al análisis y procesado de las simulaciones que van a ser llevadas a cabo por el Large Hadron Collider (LHC), el acelerador de partículas más grande del mundo, cuya construcción está apunto de ser finalizada en el CERN. Se estima que cuando el LHC esté a pleno rendimiento, el análisis de los 15 PetaBytes por año de datos generados requerirá para su procesado alrededor de 100.000 CPUs de potencia similar a la de cualquier ordenador personal de hoy en día. Obviamente, esa gran cantidad de datos a analizar requiere de la colaboración de diversos centros y organizaciones, siendo la plataforma LCG la que trata de dar soporte a dicha infraestructura geográficamente distribuida. Partiendo de esa situación, el proyecto abarca el desarrollo de un middleware Grid orientado a simulaciones pertenecientes a física de altas energías. Sin embargo, el potencial de dicha infraestructura Grid rápidamente fue extrapolado a otras áreas, las cuales también van a verse beneficiadas de su utilización.

[1] http://lcg.web.cern.ch/LCG/

Un ejemplo paradigmático de despliegue y utilización de las tecnologías Grid en el campo científico es el proyecto EGEE[2] (Enabling Grids for E-sciencE) [25]. Este proyecto agrupa científicos e ingenieros de más de 32 países, con el interés común de construir y desplegar una infraestructura Grid para e-ciencia que esté disponible para los científicos 24 horas al día. Actualmente, el proyecto EGEE integra una infraestructura de más de 20.000 procesadores con más de 5 Petabytes de capacidad de almacenamiento, proporcionando a investigadores en ciencias de la tierra, física de altas energías, bioinformática y astrofísica el acceso a una infraestructura Grid en producción, independientemente de su ubicación geográfica.

Otro proyecto relevante dentro del panorama europeo es el proyecto InteliGrid[3] (Interoperability of Virtual Organisations on Complex Semantic Grid) [26], el cual trata de ofrecer una infraestructura Grid para la interoperabilidad e integración, a nivel semántico, de todo el conjunto de aplicaciones empleadas en áreas como construcción, automoción y aeronáutica. El principal objetivo es ofrecer un entorno extensible, seguro, robusto e interoperable dentro de un esquema de pago por servicios para acceder a una infraestructura de nivel superior que hará de interlocutora.

En cuanto a la utilización de las tecnologías Grid en el resto del mundo, existen infinidad de proyectos que tratan de aprovechar su aplicación en diversos campos científicos. Por citar algún ejemplo, en el campo de la ingeniería, el proyecto estadounidense NEESgrid[4] [27, 28] representa la integración e interconexión de todo un conjunto de herramientas software utilizadas en el proyecto NEES (Network for Earthquake Engineering Simulation). NEESgrid aparece como una capa de nivel superior al entorno OpenSees (Open System for Earthquake Engineering Simulation) [29, 30], como base del desarrollo de aplicaciones para la evaluación sísmica de todo tipo de sistemas estructurales. El principal objetivo de OpenSees es la mejora en ingeniería sísmica de las técnicas de modelado en la simulación de procesos naturales.

En el ámbito nacional, IRISGrid [5] se puso en marcha a comienzos de 2002 con el objetivo de coordinar a nivel académico y científico a los grupos de investigación nacionales interesados en el desarrollo, implantación y utilización de las tecnologías Grid. Una de las principales actuaciones estaba orientada a la creación de una infraestructura Grid nacional para el uso, desarrollo e investigación de estas tecnologías. Actualmente el proyecto integra 23 grupos, centros e institutos de investigación de España. El objetivo final de la iniciativa pretende la unión en una infraestructura común de todos los recursos distribuidos pertenecientes a las organizaciones integrantes, de modo que se dé soporte a la investigación en cualquier área de aplicación de las tecnologías Grid.

4.2. Globus Toolkit

4.2.1. Introducción

De entre todos los middleware Grid, Globus Toolkit [31] representa el estándar de facto para el despliegue de Grids computacionales a gran escala. Globus proporciona

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

[3] http://www.inteligrid.com

[4] http://neesgrid.ncsa.uiuc.edu/

[5] http://irisgrid.rediris.es

servicios y librerías para la utilización y despliegue de infraestructuras Grid. Este conjunto de herramientas permite a los usuarios el acceso a recursos remotos como si fueran locales, al mismo tiempo que se preserva un control local de acceso, estableciendo quién y cuándo pueden acceder a los mismos.

Globus Toolkit consiste en una herramienta de libre distribución que facilita la compartición de recursos computacionales de manera segura, formando entornos integrados por diferentes dominios de administración, los cuales no se ven forzados a sacrificar su control sobre sus recursos locales. El conjunto de herramientas ofrecido incluye tanto servicios de alto nivel como librerías para la monitorización, gestión y utilización de recursos computacionales y de almacenamiento. Los principales esfuerzos de Globus Toolkit se centran en solventar la heterogeneidad inherente a la integración de múltiples recursos pertenecientes a diferentes dominios de administración. Estos recursos pueden estar compuestos por diferentes hardwares con diferentes softwares instalados, por lo que es muy posible la aparición de problemas de incompatibilidad.

Inicialmente las versiones 1.0 (GT1) y 2.0 (GT2), se basaban en un uso intensivo de interfaces propietarias para la comunicación entre los diferentes servicios desplegados. Sin embargo, estas interfaces fueron implementadas utilizando protocolos estándar como LDAP (Lightweight Directory Access Protocol), FTP (File Transfer Protocol), etc. La versión GT2 sentó las bases de la computación Grid tal y como la conocemos en la actualidad. Su rápida aceptación e incorporación dentro de numerosos proyectos de investigación, relacionados con las tecnologías Grid, le convirtió en el estándar de facto para el despliegue de Grids computacionales.

El lanzamiento de la versión 3 (GT3) trató de ser un acercamiento al uso de protocolos abiertos que fracasó estrepitosamente, ya que la implementación de los Servicios Web ofrecida no respetó ningún estándar perteneciente a una arquitectura orientada a servicios. Este inconveniente produjo el rápido desarrollo e implantación de la versión 4 (GT4) [32], versión que está actualmente en vigor, la cual ha integrado satisfactoriamente las tecnologías orientadas a servicios por medio de los Servicios Web. Empleando estos protocolos estándar, GT4 ha definido su arquitectura e interfaces, dotando de una interacción mucho más flexible a todos sus servicios.

La aparición de GT4 produjo una escisión en cuanto a la evolución del desarrollo de las herramientas basadas en Globus, ya que la nueva versión es incompatible con las anteriores. De este modo, los servicios de versiones anteriores que no están basados en los Servicios Web cobraron la denominación de Pre-WS (pre-Web Services). Sin embargo, la gran difusión que tuvo GT2 junto con la confianza que infunde a los usuarios, hace que los servicios pre-WS aún se mantengan por compatibilidad.

4.2.2 OGSA y WSRF

Una aplicación Grid está compuesta normalmente por varios componentes y 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: Proporciona 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 solicitará al Descubridor de Recursos que 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) [33] está orientado a definir una arquitectura común y estándar para el desarrollo de aplicaciones Grid. Su objetivo primordial consiste en estandarizar aquellos servicios que típicamente se encuentran en un entorno Grid, por medio de la especificación de un conjunto de interfaces comunes. A la hora de la implementación de esta arquitectura, aparece la disyuntiva en cuanto a la tecnología de Computación Distribuida más adecuada que actuará como base. Aunque en principio podría haberse seleccionado cualquiera (CORBA, RMI, etc.), fueron los Servicios Web los elegidos.

Sin embargo, esta decantación hacia los Servicios Web hizo que afloraran problemas de incompatibilidad relacionados con el estándar OGSA, ya que requiere Servicios Web con estado persistente. 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 tradicionales fueron ideados como simples procesadores de mensajes los cuales no poseen ningún estado. Como solución aparece la especificación WSRF (Web Services Resource Framework) [34], la cual es una extensión de los Servicios Web tradicionales para dotarles de un estado.

El estándar WSRF define una serie de especificaciones para la creación de herramientas interoperables, entre las que podemos destacar:

- WS-Resource: Describe la relación entre un Servicio Web y la entidad que se encargará de salvaguardar su estado, también llamada *recurso*. La especificación describe los mecanismos necesarios para el acceso y modificación de dichos recursos por medio de interfaces basadas en Servicios Web.
- WS-ResourceProperties: Define los mecanismos mediante los cuales se establecerán y publicarán las propiedades de un recurso de modo semántico y cómo serán declaradas por parte del interfaz del servicio.
- WS-ResourceLifeTime: Describe los mecanismos necesarios para cubrir el ciclo de vida de un recurso, definiendo diferentes estrategias para la destrucción de recursos.
- WS-Notifications: Incluye especificaciones que permiten que determinados componentes, de una infraestructura Grid, sean informados de la ocurrencia de cualquier evento producido en el sistema global. Para ello, se establecen mecanismos de publicación y suscripción, de modo que en el sistema existirán componentes que publicarán listados de temas sujetos a notificaciones. A estos temas podrán suscribirse el resto de agentes de la infraestructura, recibiendo solamente información relacionada con los eventos asociados al mismo.

4.2.3. La Arquitectura de Globus Toolkit

En este apartado se incluye una pequeña descripción de la arquitectura de Globus Toolkit, analizando los diferentes componentes que integran la misma. Estos componentes están divididos en una serie de servicios básicos, cada uno con una funcionalidad específica: gestión de trabajos, servicio de información, gestión de datos, sistema de seguridad, etc. En la Figura 4.1 se puede observar este conjunto de servicios agrupados en bloques, apareciendo una clara distinción entre los componentes pre-WS y los de nueva generación incorporados en GT4. Cada despliegue GT4 tiene asociado un contenedor de servicios conocido como container, donde tanto los servicios básicos propios de Globus como los servicios de usuario serán instanciados en dicho despliegue.

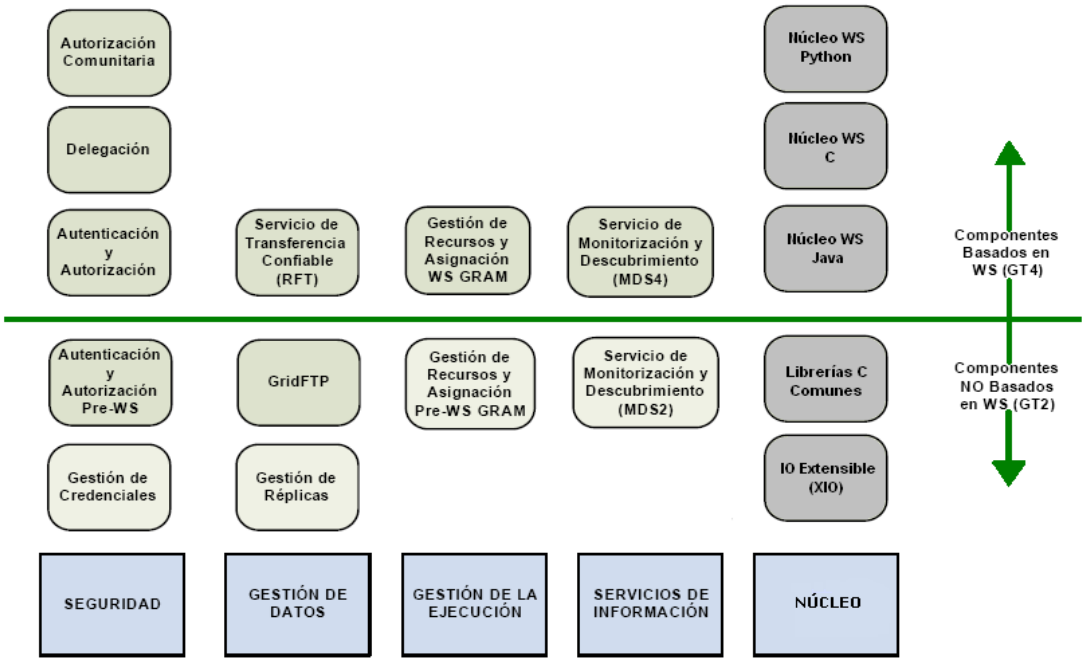

Figura 4.1. Arquitectura de Globus Toolkit

4.2.3.1. Gestión de trabajos

El servicio GRAM (Globus Resource Allocation Manager) [35], Figura 4.2, es el encargado de monitorizar y gestionar la ejecución de trabajos en recursos basados en Globus. GRAM ofrece una serie de mecanismos orientados a homogeneizar y estandarizar el lanzamiento remoto de trabajos, independientemente de los gestores locales, y cuya funcionalidad se puede agrupar en:

- Procesar las especificaciones de trabajos (RSL/XML). Estas especificaciones de trabajos establecen las características de las tareas: ficheros de entrada, ficheros de salida, tipo de trabajo, requerimientos específicos, etc. El procesado de las especificaciones puede provocar diferentes situaciones, desde una denegación de la ejecución por una especificación incorrecta, hasta el lanzamiento de varios procesos, en el caso de que se trate de un trabajo paralelo.
- Permitir una monitorización y gestión remota de los trabajos lanzados en respuesta a la petición recibida.

- Actualización automática de la información publicada por los Servicios de Información, atendiendo a las nuevas capacidades y disponibilidades de los recursos computacionales gestionados.

Por lo tanto, GRAM actuará como un servicio autónomo e independiente para la gestión de recursos Grid. La sintaxis definida, para la especificación de trabajos, trata de ser lo suficientemente concreta para que el mapeo de trabajos a recursos pueda llevarse a cabo sin ninguna interacción adicional entre GRAM y el cliente. Por otro lado, la definición de este interfaz común permite que usuarios y desarrolladores solamente deban conocer cómo interactuar con GRAM, sin tener que conocer los mecanismos concretos asociados a los gestores locales del recurso (PBS, LSF, SGE).

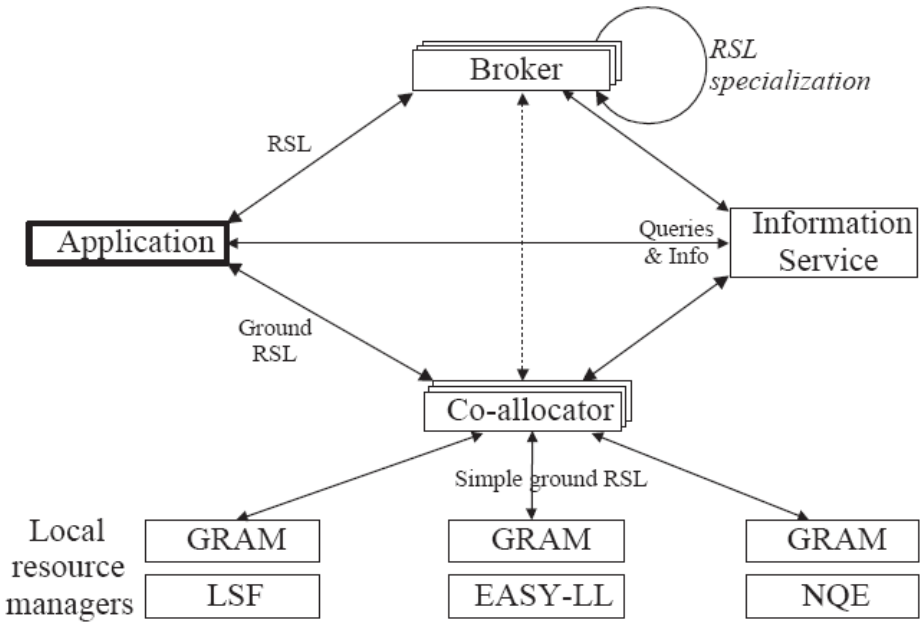

Figura 4.2. Esquema del funcionamiento de GRAM

Existe un soporte para el desarrollo de aplicaciones Grid en el marco de Globus Toolkit. Dentro del API ofrecido, existe un conjunto de funciones para el sometimiento y cancelación de tareas, así como para su monitorización. Cuando un trabajo es lanzado, automáticamente se devuelve un manejador de tarea, el cual será empleado para monitorizar su evolución. Adicionalmente, el sometimiento de un trabajo puede llevarse a cabo asíncronamente a la monitorización del mismo. De este modo, se sientan las bases para el desarrollo de sistemas de metaplanificación de más alto nivel, donde unos componentes se puedan encargar de llevar a cabo el envío de trabajos y otros de la monitorización de las ejecuciones.

4.2.3.2. Gestión de datos

Globus Toolkit integra una serie de mecanismos para la gestión y el acceso a datos almacenados dentro de una infraestructura Grid. El primer servicio que apareció para la gestión de datos fue GridFTP [36], orientado a realizar la transferencia de datos de modo seguro y ofreciendo altas prestaciones entre diferentes recursos de un Grid. GridFTP consiste en un protocolo basado en FTP que además incluye un conjunto de optimizaciones para llevar a cabo transferencia de grandes volúmenes de datos sobre redes de área extensa.

Posteriormente, en GT4 apareció RFT (Reliable File Transfer) [37], Figura 4.3., que consiste en una extensión para transferencias altamente fiables, empleando GridFTP. RFT ofrece un interfaz de Servicio Web para la transferencia y borrado de

ficheros en un recurso Grid. Las peticiones son recibidas vía mensajes SOAP por HTTP y las transferencias son llevadas a cabo por medio de GridFTP. Por otro lado, este protocolo utiliza esquemas de persistencia almacenando en bases de datos el listado de ficheros que está siendo transferido así como el estado de las trasferencias. Esto supone que, ante la aparición de un fallo que interrumpa alguna transferencia, aquella se retomaría a partir del punto el que fue interrumpida.

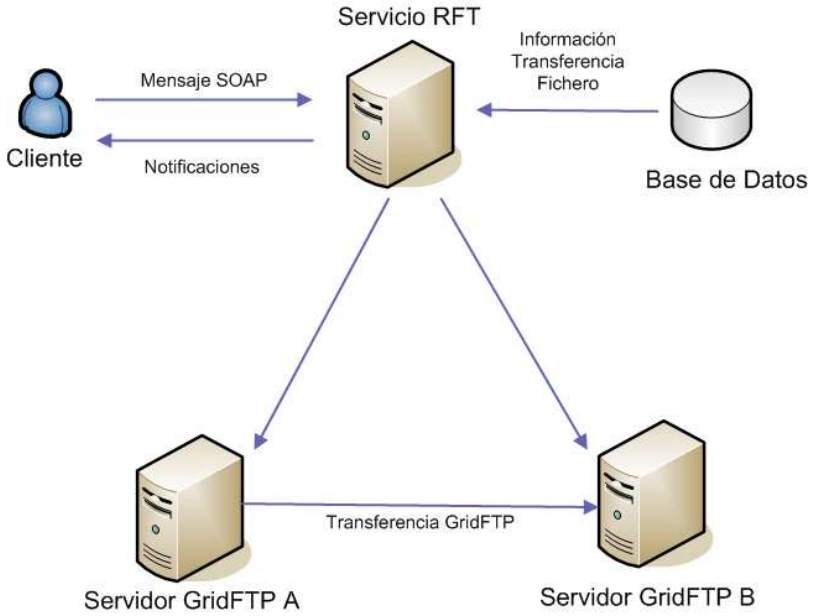

Figura 4.3. Ejemplo transferencia por RFT

Por otro lado, GT4 también incluye un conjunto de servicios de alto nivel que permiten integrar sistemas de réplicas de datos en el despliegue de infraestructuras Grid de almacenamiento. El servicio RLS (Replica Location Service) es el encargado de mantener y proporcionar información acerca de la ubicación física de las réplicas asociadas a cada fichero lógico. Este servicio permite, a partir de un nombre lógico, el acceso a múltiples réplicas físicas, dotando de redundancia de datos a una infraestructura Grid de almacenamiento. Por otro lado, el servicio DRS (Data Replication Service) ofrece mecanismos para la creación de réplicas físicas en un entorno Grid y su registro en el servicio RLS. DRS emplea RFT para la transferencia de los ficheros y utiliza RLS para encontrar y registrar las réplicas. Actualmente DRS en un componente en fase de desarrollo, no estando todavía integrado en GT4.

4.2.3.3. Servicios de información

El servicio MDS (Monitoring and Discovery Services) [38] se encarga del registro, distribución e indexado de cualquier información acerca del estado de los recursos, servicios y configuraciones de un Grid. La información recogida acerca de estas entidades es utilizada para el descubrimiento de nuevos recursos y servicios, y también para la monitorización del estado del sistema. Una de las cualidades de MDS es que ofrece mecanismos estándar y comunes para la publicación y acceso a la información publicada por recursos heterogéneos.

La implementación de MDS pre-WS (MDS2) está basada en una estructura jerárquica de servicios compuestos por dos elementos principales: GRIS (Grid Resource Information Service) y GIIS (Grid Index Information Service). El servicio GRIS es el encargado de almacenar una recopilación de la información publicada por un recurso Grid. Por otro lado, el servicio GIIS supone un servicio de nivel superior en la jerarquía definida, encargado de agregar en un directorio la información relativa a múltiples servicios GRIS. El protocolo y formato de datos empleado en MDS2 es

LDAP, para el acceso a la información publicada tanto por GIIS como por el servidor GRIS de un recurso.

En GT4 aparece un nuevo servicio de información basado en WSRF y WS-Notifications llamado MDS4. El entorno de desarrollo de Servicios Grid basado en WSRF, definido en GT4, permite el registro y publicación de modo transparente de las propiedades de los recursos definidos en un Servicio Grid. Por ejemplo, las propiedades de los recursos asociados a los servicios GRAM y RFT son ya publicadas por medio de MDS4. El nuevo Sistema de Información integra dos servicios de alto nivel, llamados Index Service y Trigger Service, para la publicación y registro de información.

El Index Service representa el núcleo de la implementación del sistema MDS en GT4. Cada despliegue del container GT4 contiene una instancia del Index Service expuesta como un servicio WSRF (DefaultIndexService). Este servicio es el encargado de interactuar con las fuentes de información, empleando mecanismos estándar para la consulta de las propiedades WSRF de los recursos, así como esquemas de suscripción y notificación (WS-ResourceProperties y WS-BaseNotification). Por lo tanto, un Servicio Grid publicará información automáticamente a través de las propiedades de sus recursos. Como se puede observar, el Index Service agrupará en un único punto toda la información relativa a las múltiples fuentes de datos de un recurso. La información publicada podrá ser consultada a través de las interfaces proporcionadas empleando consultas XPath. Al igual que en MDS2, los Index Service pueden ser organizados de manera jerárquica, sin embargo esto no implica la existencia de un único Servicio de Información central que publique la información relativa a todos los recursos del Grid. Cada registro de información tendrá un tiempo de vida asociado y requerirá una renovación periódica indicando que el recurso o servicio sigue vivo.

Por otro lado, el Sistema de Información de GT4 incluye un servicio adicional llamado Trigger Service. La tarea de este servicio consiste en el análisis de la información publicada, por medio de unas condiciones definidas por el administrador del recurso, y fijadas en un fichero de configuración. Cundo alguna de estas condiciones es satisfecha, se desencadena la ejecución de una acción determinada, como podría ser el envío de un mail, la ejecución de un script, etc. De este modo, se dota al administrador de una herramienta muy potente para la detección automática de problemas o condiciones anómalas en un Grid.

4.2.3.4. Sistema de Seguridad GSI

En un entorno Grid los requerimientos de seguridad cobran mucha importancia, ya que es habitual la presencia de recursos pertenecientes a diferentes organizaciones. Además, el hecho de integrar recursos pertenecientes a diferentes dominios de administración requerirá políticas de seguridad más estrictas, ya que la seguridad global estará comprometida por la seguridad del recurso o servicio más débil. En consecuencia, deben ser considerados aspectos de autenticación y autorización, así como la privacidad e integridad de los datos.

La autenticación es el proceso mediante el cual se verifica la identidad de todas las entidades participantes en el Grid, como pueden ser usuarios, recursos, Servicios Grid, etc. Para la verificación de identidades, se utilizan sistemas de certificación digital basados en el despliegue de Autoridades de Certificación en las que todos confían. De este modo, empleando sistemas de clave pública/privada se fijan mecanismos para evitar la suplantación de identidades. Por otro lado, para garantizar qué recursos y servicios solamente sean accedidos por usuarios que posean esos privilegios, se emplean técnicas de autorización. El establecimiento de estos mecanismos de control de acceso está directamente relacionado con la autenticación, ya que es indispensable conocer la identidad de los usuarios previamente a aplicar estas políticas.

Por otro lado, la privacidad de los datos contenidos en los mensajes a intercambiar entre servicios y clientes es un requisito indispensable en muchos entornos Grid. En consecuencia, debe garantizarse que estos datos intercambiados solamente puedan ser accedidos por los actores implicados y nunca por terceras personas. Adicionalmente, igual de peligroso es un mensaje leído por terceras personas como uno con su contenido alterado. Para evitar estas situaciones, se emplean mecanismos de integridad de datos, garantizando que si un mensaje es alterado en el tránsito, el destinatario será capaz de reconocer este hecho. Tanto la privacidad como la integridad se consiguen empleando mecanismos de cifrado y firma digital, por medio de sistemas de clave pública/privada.

Otro requerimiento muy importante en los sistemas Grid es la inclusión de mecanismos de delegación. En un entorno donde múltiples servicios y recursos pueden llegar a intervenir para llevar a cabo una tarea, es indispensable que los clientes puedan ceder los derechos a actuar en su nombre a estos agentes.

Por lo tanto, para satisfacer todos estos requerimientos de seguridad en un Grid, Globus Toolkit incluye una capa de seguridad llamada GSI (Grid Secuirity Infraestructure) [39]. De la funcionalidad ofrecida por GSI cabe destacar que:

- Ofrece canales de comunicación confidenciales e íntegros entre los diferentes componentes de un entorno Grid.
- Incluye sistemas de autenticación única, donde un usuario no tiene que presentar sus credenciales en cada acción.
- Obliga a que todas las partes implicadas en una acción estén autenticadas entre sí.
- Incorpora la delegación para que los diferentes Servicios Grid ofrecidos puedan actuar en el nombre de los clientes.
- Define varios niveles de seguridad, tanto a nivel de container, como de recurso y de servicio, de modo que para cada uno de ellos permite configurar diferentes políticas de autorización en función del usuario que trate de acceder.

4.3. El Middleware LCG

4.3.1. Introducción

Como ya hemos comentado con anterioridad, fruto del amplio apoyo a las tecnologías Grid ofrecido por diversos proyectos de investigación a nivel europeo (LCG, EGEE), se ha gestado el despliegue de una infraestructura Grid en producción a gran escala, Figura 4.4.

El nacimiento de esta infraestructura Grid trataba inicialmente de dar soporte al gran volumen de información que se generará en los experimentos del nuevo acelerador de partículas del CERN. Sin embargo, gracias al proyecto EGEE, se definieron una serie de áreas de interés, en diversos campos de ciencia e ingeniería, a las que también se les ha dado cobertura dentro de esta potente infraestructura.

El acelerador de partículas LHC (Large Hadron Collider), que actualmente está siendo construido en el CERN, representa el instrumento científico más grande del planeta. Cuando a mediados del 2008 entre en funcionamiento, generará alrededor de 15 Petabytes de datos al año, los cuales deben estar accesibles para ser analizados por más de 5000 científicos de 500 institutos de investigación y universidades distribuidos alrededor del mundo. Además, todos los datos generados deberán estar disponibles durante todo el tiempo de vida del LHC, que se estima que será de 15 años. Por otro lado, el análisis de los datos requerirá alrededor de 100.000 CPUs de

potencia similar a un PC convencional. Una aproximación tradicional ubicaría toda esa capacidad de manera centralizada, próxima a donde van a ser llevados a cabo los experimentos. Sin embargo, en el caso del proyecto LHC se ha optado por emplear un modelo distribuido basado en las tecnologías Grid.

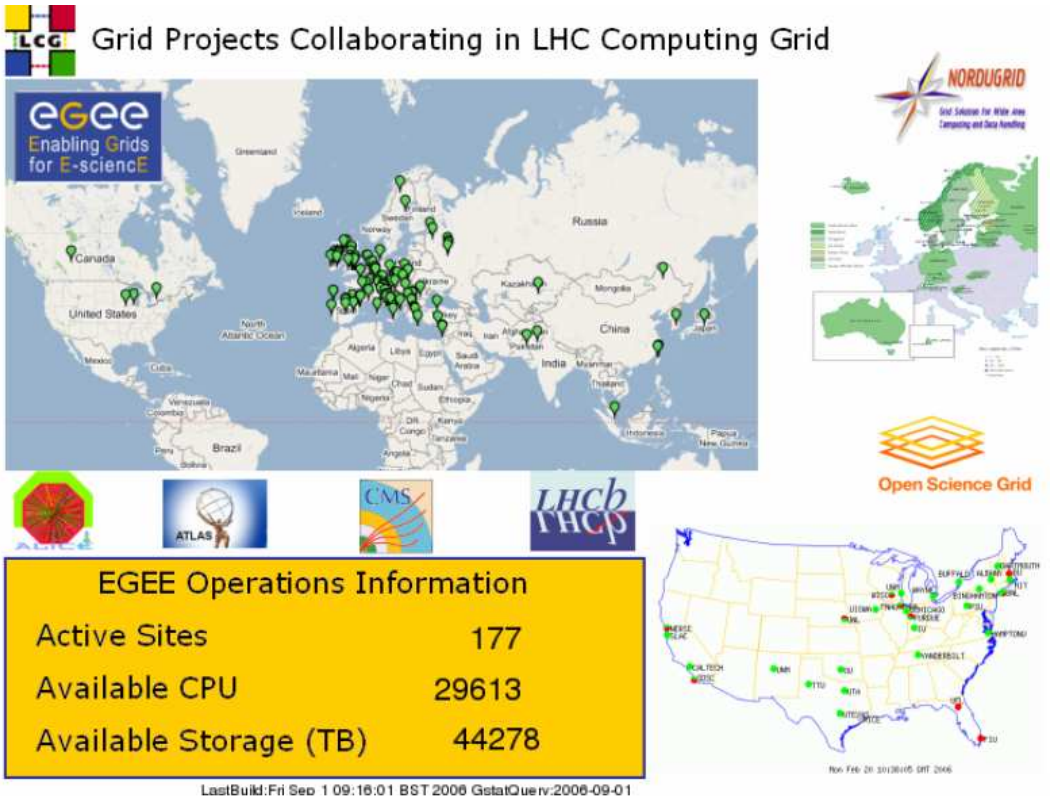

Figura 4.4. Estado del Despliegue de la infraestructura Grid en producción LCG en Septiembre 2006

La apuesta por las tecnologías Grid en el proyecto LHC se ha basado en los siguientes puntos:

- El coste del mantenimiento y actualización de una infraestructura de tales características será más fácil de sobrellevar en un entorno distribuido, ya que cada organización individualmente tendrá estas competencias, de modo que se implica a todos en la contribución por el bien común.
- En un sistema distribuido existen menos puntos débiles que en una arquitectura centralizada. El hecho de utilizar redundancia y replicación de datos, además de una distribución de la carga equitativa, facilitará el acceso a los datos por parte de los científicos, independientemente de su ubicación geográfica.

Los datos generados por los experimentos del LHC serán distribuidos en la infraestructura construida de acuerdo a un modelo de 4 niveles. El primer nivel llamado *tier-0* dará soporte al sistema primario de copia de seguridad y estará en el CERN. Después de un procesado inicial, estos datos serán distribuidos en grandes centros de computación que componen la siguiente capa *tier-1*. Los siguientes niveles se alimentarán del nivel *tier-1*, y se encargarán, en momentos puntuales, de almacenar información para que los científicos lleven a cabo análisis más específicos.

En consecuencia, uno de los principales objetivos del proyecto LCG es el desarrollo de un middleware que dé soporte al despliegue y mantenimiento de una infraestructura Grid de cómputo y almacenamiento. Por lo tanto, el proyecto se centra

en ofrecer herramientas y servicios para asistir a los científicos del proyecto LHC en el análisis de los experimentos. Entre sus objetivos cabe destacar:

- El desarrollo de componentes software que den soporte al almacenamiento y acceso a los datos generados en aplicaciones científicas, así como a su análisis dentro de infraestructuras Grid.
- El desarrollo de servicios basados en la computación Grid que integren recursos distribuidos pertenecientes a diversos centros de investigación.
- La gestión de los usuarios y de sus privilegios de manera descentralizada en un entorno Grid heterogéneo.

4.3.2. Arquitectura del middleware LCG

El middleware desarrollado dentro del proyecto LCG posee una arquitectura distribuida compuesta por una serie de componentes y servicios. Estos elementos típicamente residirán en computadoras distintas, aunque pueden residir en una misma máquina:

- Interfaz de Usuario (UI). Se trata del punto de acceso a la infraestructura Grid LCG donde el usuario contará con una cuenta personal con su certificado instalado. Este Interfaz de Usuario es la puerta de entrada a los Servicios Grid disponibles y para su utilización el usuario deberá ser autenticado y autorizado. Una vez que el usuario sea reconocido por el sistema, desde el UI podrá realizar las siguientes acciones típicas de un Grid: lanzamiento de trabajos, cancelar trabajos, listar todos los recursos disponibles para el lanzamiento, obtener la salida de los trabajos, obtener el estado de las tareas, etc.
- Computing Element (CE). Es el encargado de recibir y gestionar la petición de ejecución de trabajos en la infraestructura. En cierto modo, este servicio se puede ver como un gestor de colas batch de tipo Grid. Por lo tanto, en un mismo recurso, podrán existir múltiples colas instaladas, siendo cada una de ellas identificada como un CE diferente. Este sistema es empleado para definir diferentes colas, en función del tamaño de los trabajos, de la organización virtual a la que pertenezcan, etc. En sistemas con gestores locales de recursos instalados (Local Resource Management System), LCG permite su integración por medio de la instalación de un CE en el front-end. Normalmente, en ese tipo de sistemas, el CE actuará como punto de entrada a un subsistema compuesto por nodos computacionales llamados Worker Nodes (WN). En LCG los gestores soportados son PBS, LSF, Torque y Condor.
- Worker Node (WN). Se trata de los nodos computacionales que llevarán a cabo la ejecución de las tareas enviadas por un CE. Un WN tendrá acceso a la red externa y dispondrá, instalados, de todos los comandos y librerías necesarios para el acceso a otros recursos Grid.
- Resource Broker (RB). Este componente se encarga de llevar a cabo el reparto de la carga dentro de una infraestructura LCG. Para ello, es el responsable de la gestión de los trabajos sometidos desde los UI y de su distribución a los CE más adecuados, en función de los requerimientos de las tareas y de la disponibilidad de los recursos. Tendrá acceso a los sistemas de información y al servicio de catálogo para poder llevar a cabo su cometido.
- File Catalogs (LFC). Este servicio proporciona mecanismos para la búsqueda y creación de réplicas de ficheros en el Grid. A partir de un sistema de nombres lógicos, Logical File Names (LFN), el catálogo contendrá referencias hacia la réplicas físicas que haya creadas en la infraestructura. De este modo un

usuario conocerá la ubicación exacta de cada réplica asociada al nombre lógico original del fichero.

- Storate Element (SE). Este elemento proporciona una capa de acceso homogénea a los recursos de almacenamiento. Un SE puede gestionar servidores de disco tradicionales, grandes arrays de disco e incluso sistemas de almacenamiento masivo. Normalmente cada RB que se integre en la infraestructura proporcionará al menos un SE por defecto en el que almacenar los ficheros.
- Information Service (IS). Es el responsable de almacenar y gestionar la información relativa al estado y características de los recursos Grid de una infraestructura LCG. Este servicio es la base sobre la que se articula todo el resto de componentes, ya que registrará todos los servicios y recursos disponibles, permitiendo su consulta para la gestión de réplicas, el balanceo de la carga, etc. La información publicada permitirá la monitorización del Grid, para llevar a cabo análisis de utilización, detección de fallos y cualquier evento que resulte interesante. En LCG, inicialmente fue adoptado el sistema MDS de Globus como el principal servicio de información. Recientemente, un nuevo tipo de sistema de información está comenzando a ser desplegado. Este sistema más avanzado basado en modelos relacionales es denominado R-GMA (Relational Grid Monitoring Architecture) [40]. Por otro lado, al igual que Globus, LCG utiliza una estructura jerárquica compuesta por los servicios GRIS y GIIS, aunque, por encima de ambos, integra el servicio BDII (Berkeley Databaste Information Index), el cual contendrá toda la información relativa a la infraestructura Grid de una organización. Por lo tanto, usuarios y servicios consultarán los BDII de cada organización.

4.4. Metaplanificación Grid

Como ya ha sido detallado en los apartados anteriores, el middleware Globus Toolkit representa el estándar de facto para el despliegue de infraestructuras Grid. Sin embargo, Globus Toolkit únicamente ofrece un conjunto de servicios de bajo nivel y herramientas básicas para el despliegue y utilización de entornos Grid. Por lo tanto, sin la existencia de herramientas de nivel superior como un planificador, una adecuada utilización de la infraestructura Grid subyacente requiere la utilización de varios de estos servicios básicos de manera conjunta. De este modo, por ejemplo, la ejecución remota de tareas por medio de Globus Toolkit requiere la utilización coordinada de varios servicios básicos como son el sistema de información (MDS), el sistema de transferencia de ficheros (GridFTP) y el sistema de ejecución de tareas (GRAM).

Por otro lado, LCG consiste en un middleware Grid en producción que incluye una serie de servicios y mecanismos de más alto nivel. Como ya se ha detallado en el apartado anterior, LCG ofrece una serie de componentes de alto nivel con roles ya prefijados para el almacenamiento (Storage Elements), la computación intensiva (Computing y Working Nodes), selección de recursos (Resource Broker), etc. Sin embargo, no se trata de una infraestructura generalista, siendo un sistema cerrado muy poco portable y que requiere ciertas habilidades para su instalación, mantenimiento y utilización.

Como se puede observar, el uso de las tecnologías Grid implica lidiar con un conjunto diferente de protocolos, paradigmas de programación y tecnologías que dificultan enormemente su uso y despliegue a gran escala. Es por ello que su mayor campo de aplicación en la actualidad se centra en el ámbito académico y de investigación, y en menos medida en el mundo empresarial.

Por otro lado, para un uso eficiente y adecuado de una infraestructura Grid se hace necesaria la introducción de mecanismos de metaplanificación. La metaplanificación consiste en la gestión eficiente de la ejecución de un conjunto de tareas sobre un conjunto de recursos computacionales disponibles en las diferentes organizaciones integradas en una infraestructura Grid. Tradicionalmente, el concepto de planificación de tareas es asociado a la gestión de la ejecución dentro de los límites de una organización. Debido a que una infraestructura Grid abarca múltiples organizaciones, se utiliza el concepto de metaplanificación refiriéndose a la gestión de tareas sobre múltiples dominios de administración.

4.4.1. Funcionalidad de un Metaplanificador

Típicamente un metaplanificador Grid requerirá una especificación de las características del trabajo por parte del usuario. Esta especificación incluirá una descripción de la tarea, con toda la información necesaria para poder llevar a cabo su ejecución. Por ejemplo, podrá incluir requisitos de ejecución como la memoria RAM requerida o el número de procesadores a emplear en el caso de una ejecución paralela, los ficheros de entrada necesarios y cualquier otra información adicional. A partir de dicha información, en el caso que el planificador no disponga de una lista de recursos predefinida, deberá llevar a cabo un descubrimiento de recursos, proceso mediante el cual tendrá consciencia de la infraestructura en la que se encuentra ubicado. Por ejemplo, este descubrimiento de recursos se conseguirá mediante el acceso a servicios proveedores de información del middleware Grid, como puede ser el de Globus, GIIS (Grid Index Information Service) o el de LCG, BDII (Berckeley Database Information Index).

Una vez el metaplanificador dispone de la lista de recursos, partiendo de los requerimientos de cada tarea y empleando las políticas de planificación definidas, se lleva a cabo un filtrado y selección de recursos. La selección de recursos es la tarea más importante, ya que del conjunto de recursos que podrían ejecutar un conjunto de trabajos, el sistema debe realizar las asociaciones adecuadas acorde a las políticas definidas y este problema está catalogado como NP-Completo. En este punto, aparece una gran variedad de heurísticas, donde cada una otorga más prioridad a unos parámetros o a otros, en función de sus objetivos y siempre buscando ejecutar el mayor número de trabajos por unidad de tiempo y un reparto óptimo de la carga. Sin embargo, como no se conoce a priori la duración de cada trabajo y estos pueden entrar en el sistema en cualquier momento, las aproximaciones voraces son las más utilizadas. Estas estrategias están basadas en un trato particularizado de cada tarea, de modo que de manera independiente es obtenido para cada trabajo el recurso más apropiado.

Finalmente, el metaplanificador llevará a cabo todas las tareas necesarias para la ejecución remota de los trabajos. En el caso de que el recurso no disponga de los datos de entrada o del ejecutable, estos deberán ser transferidos. En este punto se lleva a cabo la ejecución remota y monitorización del trabajo, pudiendo ofrecer al usuario información permanente de la evolución del mismo. Finalmente, dependiendo del tipo de metaplanificador, los datos de salida generados podrán ser automáticamente recogidos a la máquina cliente o ubicados en algún repositorio de almacenamiento.

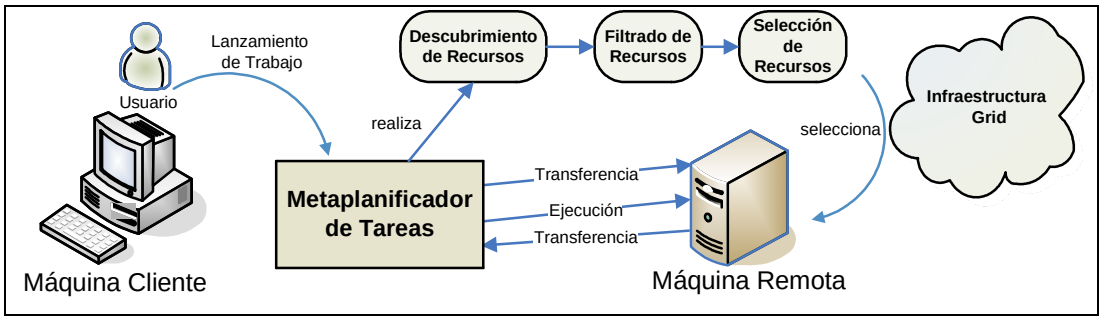

Figura 4.5: Ejemplo de las etapas implicadas en un proceso de metaplanificación Grid

4.4.2. Estado del Arte de la Metaplanificación en Grid

La utilización de sencillas herramientas de Metaplanificación Grid permitirá un acercamiento de las tecnologías Grid a usuarios no profanos en el área, avanzando enormemente en la usabilidad de las mismas. Además, como ya se ha explicado en el capítulo 4, la utilización de Arquitecturas Orientadas a Servicios dotarán a un sistema de una enorme flexibilidad en su utilización e integración dentro de sistemas distribuidos de un mayor nivel.

Es importante destacar que el desarrollo de un Servicio Grid de Análisis de Estructuras, como es objetivo de esta Tesis de Máster, ofrecerá una serie de servicios de alto nivel orientados a la ejecución de simulaciones estructurales de modo eficiente en una infraestructura Grid. En consecuencia, el despliegue Grid aparecerá como una caja negra, en la cual la ejecución de las simulaciones se llevará a cabo por medio de servicios de alto nivel de metaplanificación. Estos servicios se encargarán de realizar de una forma totalmente transparente la selección de recursos, la transferencia de ejecutables y datos de entrada, la monitorización de las ejecuciones y la recogida de los datos de salida. La interfaz de conexión con el servicio se llevará a cabo empleando los estándares de Servicios Web SOAP y WSDL, lo cual dotará al sistema de una alta interoperabilidad.

Nótese que, previamente al desarrollo de un Servicio Grid de Análisis de Estructuras, se hace necesario un estudio del estado del arte en el área de la metaplanificación Grid. Por lo tanto, en el resto del capítulo se analizan diferentes middlewares de metaplanificación Grid existentes, tratando de buscar el más apropiado. Cabe destacar que la selección del middleware Grid es un aspecto clave, ya que el sistema global quedará determinado por sus características y prestaciones.

4.4.2.1. GMarte

GMarte [41] es un middleware Grid orientado a objetos que permite la ejecución transparente de cualquier tipo de aplicación en infraestructuras Grid basadas en Globus Toolkit o LCG. Este middleware ofrece, por medio de una API (Application Programing Interface) de alto nivel, mecanismos de metaplanificación tolerantes a fallos para la distribución de tareas en recursos remotos, ofreciendo una capa de mayor nivel de abstracción para el uso de Grids computacionales. GMarte puede interactuar tanto con las versiones Pre-Servicios Web como con las versiones WSRF de Globus Toolkit.

La funcionalidad ofrecida por el API de GMarte permite al usuario centrar su esfuerzo en la especificación de los trabajos a ejecutar, en vez de en la utilización de la infraestructura Grid subyacente con la complejidad que ello conlleva. Para lograr la ejecución remota de trabajos, GMarte primeramente introduce una capa de abstracción homogénea para el acceso a los Sistemas de Información de los recursos

Grid. Debido a la posible existencia de diferentes Sistemas de Gestión Local de Recursos en las máquinas remotas, los cuales ofrecen la información de diferente modo, GMarte proporciona un acceso homogéneo y sencillo a la información publicada, como puede ser la memoria RAM disponible, el número total de procesadores disponibles, etc. Sobre esta capa de abstracción, GMarte ofrece mecanismos para el descubrimiento de recursos permitiendo encontrar los recursos disponibles en un momento dado. Para ello, GMarte integra sistemas de consulta tanto a GIIS (Grid Index Information Service) para los sistemas Globus, así como a BDII (Berkeley Database Information Index) para LCG.

Finalmente, como nivel más alto, GMarte integra un potente metaplanificador multi-hilo que permite llevar a cabo de manera concurrente las diferentes fases involucradas en la metaplanificación y la ejecución remota de tareas. Esta capa, a partir de las tareas definidas por los usuarios y de sus requerimientos, se encarga de manera transparente de llevar a cabo el descubrimiento y la selección de los recursos, la transferencia de los datos, el lanzamiento y la monitorización de la tarea, así como la recogida final de los datos de salida.

Por otro lado, GMarte integra un conjunto multi-nivel de mecanismos tolerantes a fallos, permitiendo fallos en la ejecución remota de tareas tanto durante la transferencia de datos como en la ejecución. De este modo, el metaplanificador gestiona automáticamente dichos fallos, migrando las tareas fallidas a otros recursos disponibles en la infraestructura Grid.

Cabe señalar que, una de las características más destacables del middleware GMarte, es el API Java de alto nivel ofrecido. Este API, por un lado, permite la utilización de la herramienta en cualquier plataforma que disponga de una Máquina Virtual Java instalada. Además, el API ofrecido es altamente actualizable y mantenible, ya que permite integrar nuevos middlewares subyacentes sin forzar un cambio en la interfaz empleada por el usuario, facilitando de este modo la extensibilidad y actualización del código. Por otro lado, permite su fácil integración dentro de cualquier tipo de aplicación Grid, bien para el acceso al sistema de información, la transferencia de ficheros o la ejecución eficiente de tareas en una infraestructura Grid computacional.

4.4.2.2. Condor

Condor consiste en un planificador batch para la ejecución de trabajos intensivos en una infraestructura compuesta por recursos heterogéneos distribuidos. Incluye un sistema de colas, políticas de planificación, esquemas de prioridad, monitorización y gestión de recursos. A diferencia de GMarte, que requiere un middleware de bajo nivel para llevar a cabo la ejecución remota de tareas, Condor ofrece un sistema independiente y autónomo. Los usuarios pueden someter al sistema tanto trabajos secuenciales como paralelos, los cuales son automáticamente encolados por Condor. Seguidamente, acorde a la política definida, es seleccionado el recurso más apropiado, llevándose a cabo tanto la ejecución como la monitorización de la misma. Finalmente, el usuario es informado de la finalización de sus trabajos.

Aparentemente, Condor parecería un sistema gestor de colas batch, sin embargo, su arquitectura está diseñada para aprovechar los ciclos de CPU ociosos, incluso de estaciones de trabajo que formen parte de la infraestructura distribuida. Por lo tanto, la infraestructura podría combinar clusters dedicados de nodos computacionales con estaciones de trabajo, que en un momento dado se encuentren ociosas.

El sistema ofrecido incluye técnicas de detección de disponibilidad de recursos, permitiendo la migración y checkpoint transparente de trabajos desde recursos, que dejan de estar disponibles, hacia máquinas ociosas. De este modo, se da un soporte

más adecuado a sistemas dinámicos, donde la disponibilidad de recursos tiene cierta volatilidad, como ocurre en una infraestructura compuesta por estaciones de trabajo.

Por otro lado, un despliegue Condor puede integrar recursos de diferentes dominios de administración gobernados por diferentes organizaciones. Típicamente, usuarios que pertenezcan a diferentes dominios de administración encuentran cierta dificultad para integrar todos los recursos disponibles dentro de una misma infraestructura. Para dar soporte a ello, dos mecanismos llamados Condor flocking y Condor-G permiten el despliegue mediante Condor de entornos Grid.

Condor flocking es un protocolo que permite a múltiples sistemas Condor independientes trabajar conjuntamente. Para ello, se ofrece la migración transparente de tareas de un dominio de administración a otro, dependiendo de la disponibilidad de recursos. De este modo, cuando la infraestructura local no dispone de recursos para la ejecución de una tarea, está es migrada automáticamente a otra infraestructura perteneciente a un dominio de administración diferente. Una de las principales ventajas del flocking es su transparencia, ya que un usuario no tiene que añadir ninguna especificación adicional a la descripción de los trabajos.

Condor-G [42] es el mecanismo empleado cuando se desea, a partir de la cola local de Condor, emplear recursos que dispongan de Globus como middleware gestor de la ejecución de trabajos. El sistema ofrece las mismas características para trabajos Globus como para el resto de trabajos Condor. En consecuencia, un usuario puede someter una petición de ejecución de trabajos Globus a través de una cola de Condor, con lo que potencialmente tiene acceso a un conjunto mucho más amplio de recursos. Sin embargo, el administrador del sistema debe añadir y configurar manualmente los recursos Globus al sistema Condor. Por otro lado, Condor-G emplea mecanismos muy básicos para la selección de recursos Globus, ya que no son considerados parámetros dinámicos del estado de los mismos, algo esencial en un entorno Grid. El algoritmo matchmaking empleado se basa en la utilización de una lista de recursos que es secuencialmente analizada, asociando las tareas al primer recurso que se encuentre disponible.

4.4.2.3. GridWay

El entorno GridWay [43] consiste en un middleware Grid que, a partir de los servicios básicos de Globus, define una capa de mayor nivel orientada a compartir recursos distribuidos de modo seguro y fiable. De este modo, un usuario podrá integrar y utilizar recursos distribuidos pertenecientes a una única organización o a diferentes dominios de administración.

GridWay proporciona una API de alto nivel para la ejecución fiable y desatendida de tareas en una infraestructura Grid basada en Globus. El metaplanificador integrado es capaz de gestionar y manejar las diferentes situaciones de fallo, que pueden acontecer en las diversas fases implicadas en la ejecución remota de tareas.

El núcleo de la herramienta está compuesto por un agente encargado de atender el envío de trabajos, el cual lleva a cabo todas las fases de la metaplanificación y monitorización de la ejecución de tareas. Este agente dá soporte a un amplio abanico de especificaciones de trabajos, desde tareas simples e independientes entre sí, hasta conjuntos de tareas con dependencias entre ellas. Adicionalmente, las condiciones variables de una infraestructura Grid, incluyendo las posibles caídas o fallos, son soportados incluyendo mecanismos adaptativos de metaplanificación dinámica. Una vez que una tarea ha sido asignada y ubicada en un recurso, el sistema es capaz de detectar cualquier tipo de fallo o cambio en el estado del mismo, produciendo una migración automática a otro recurso disponible si fuera necesario.

Por otro lado, GridWay ofrece tanto un API de alto nivel con dos implementaciones DRMAA (Distributed Resource Management Application API [44]) en Java y en C,

como una serie de herramientas de línea de comandos. Estos dos conjuntos de herramientas actúan de interfaz con el agente gestor de trabajos, antes mencionado, y tratan de ofrecer la máxima flexibilidad a los usuarios a la hora especificar sus problemas computacionales empleando un entorno Grid.

4.4.2.4. GridBus Broker

GridBus Broker [45] representa una herramienta Grid que dá soporte al despliegue y utilización de aplicaciones Grid computacionales. Al igual que GMarte, su arquitectura está implementada en Java, proporcionando acceso transparente a diversos middleware Grids como Globus, Achemi, Condor, etc. Por lo tanto, GridBus permite el acceso a sistemas distribuidos integrando técnicas para el descubrimiento y la monitorización de los recursos, la gestión de trabajos y la transferencia de datos.

El API en Java que ofrece permite su fácil incorporación en el desarrollo de aplicaciones Grid, permitiendo la gestión fácil y transparente de una infraestructura Grid heterogénea.

El mecanismo de planificación incluido no emplea ningún parámetro asociado a la disponibilidad de los recursos, simplemente gestiona la infraestructura desde un punto de vista estadístico. Por lo tanto, el mecanismo de planificación emplea conjuntamente por cada recurso un ratio de trabajos finalizados para estimar sus prestaciones y un parámetro configurable de límite de tareas. A diferencia de el resto de metaplanificadores analizados en este capítulo, para cada tarea el planificador elegirá el recurso que se espera que ejecute el trabajo en el menor tiempo, siempre que no sobrepase su límite de trabajos y sin tener en cuenta ningún otro requerimiento de la tarea.

GridBus Broker incluye un gestor multinivel de fallos que garantiza el correcto funcionamiento del planificador, independientemente del fallo de alguno de los recursos considerados. Además, la integración de técnicas de replanificación de trabajos permite un fácil manejo de las situaciones de fallo, llevando a cabo una migración eficiente de los trabajos. Por otro lado, una caída del propio metaplanificador es soportada incluyendo un módulo adicional de recuperación basado en la utilización efectiva de esquemas de persistencia.

4.4.3. Metaplanificador Seleccionado

Las secciones anteriores han descrito y analizado el estado del arte en cuanto a metaplanificadores Grid de cara a su integración dentro de una infraestructura Grid. Existen varios criterios a tener en cuenta a la hora de decantarse por la utilización de un software u otro. Por un lado, es importante estudiar la facilidad de uso y la existencia o no de un API con ciertas garantías. Por otro lado, teniendo en cuenta que la planificación de tareas multiprocesador a partir de unidades de tiempo es un problema NP-Completo, el análisis a llevar a cabo debe buscar el middleware que más se aproxime a la solución óptima.

Condor representa un planificador muy fiable que permite la gestión de trabajos y recursos dentro de un entorno distribuido independiente gobernado por él. Sin embargo, integra mecanismos de selección muy poco elaborados cuando tratamos de interactuar con recursos Globus, requiriendo en este caso un uso extensivo de toda la plataforma Condor para poder aprovechar toda su potencia. Por otro lado, el dinamismo habitualmente presente en entornos distribuidos no es tenido en cuenta en el proceso de selección de recursos, haciendo inadecuado su uso en un entorno Grid.

En cuanto a los middlewares restantes, tanto GMarte como GridWay y GridBus ofrecen un API de alto nivel en Java que permite la ejecución en recursos Globus ofreciendo mecanismos tolerantes a fallos multinivel. Por lo tanto, todos ellos pueden

ser considerados en el mismo peldaño, a priori, requiriendo un análisis más detallado para poder tomar la decisión correcta.

Aunque GridBus supone un entorno muy potente en cuanto a los algoritmos de metaplanificación incluidos, no tiene en cuenta la información dinámica del estado de los recursos de la infraestructura. Este detalle no tendría importancia en sistemas estables donde las condiciones de carga, disponibilidad, etc., estuviesen más o menos acotadas, pero desgraciadamente eso es algo excepcional dentro de una infraestructura Grid.

Una de las situaciones habituales en simulación científica y de procesos de ingeniería, y con la que también nos encontramos en el análisis dinámico de estructuras, es la existencia de procesos que a lo largo del tiempo generan un conjunto de resultados parciales, para cada paso de tiempo. GMarte ofrece la posibilidad de la descarga automática de estos resultados parciales a la vez que se está llevando a cabo la ejecución la tarea. Esta característica, que no se encuentra presente en el resto de middlewares, permite un solapamiento entre ejecución y el transporte de los datos, acelerando notablemente el proceso. Además, la alternativa multihilo incluida en GMarte permite la planificación y ejecución remota de múltiples tareas concurrentemente, aumentando la productividad del sistema, máxime cuando múltiples tareas son sometidas al sistema.

Por lo tanto, aunque GridWay y GridBus Broker podrían perfectamente pasar a formar parte de la infraestructura Grid, entendemos que GMarte es el middleware más adecuado.

Capítulo 5

Computación Grid aplicada al Análisis de Estructuras de Edificación

Tradicionalmente, el análisis de estructuras de edificación de modo realista ha sido un problema con unos altos requerimientos de cómputo y almacenamiento. Debido a ello, los programas comerciales de cálculo de estructuras incluyen una serie de simplificaciones que han permitido reducir notablemente los tiempos de cómputo. Sin embargo estas simplificaciones, que son razonables en edificios regulares convencionales, tienen dudosa justificación en el caso de edificios de gran dimensión asimétricos o con alguna singularidad.

Por ello, y a fin de llevar a cabo un análisis de estructuras de gran dimensión de modo realista, se hace necesario recurrir a técnicas de Computación de Altas Prestaciones (HPC), que nos permitan abordar el problema en tiempos razonables. En consecuencia, dentro del GRyCAP se ha desarrollado un simulador MPI muy potente que permite abordar estructuras de gran dimensión de manera muy eficiente incluso ante cálculos dinámicos.

La utilización de este tipo de herramienta paralela requiere la disposición de hardware específico. Este hardware permitiría llevar a cabo simulaciones de estructuras de manera muy eficiente, en secuencial o en paralelo, aprovechando las buenas prestaciones del simulador estructural. Sin embargo, los estudios de arquitectura e ingeniería raramente poseen este tipo de infraestructura, y supondría un esfuerzo económico adicional la inversión en hardware específico exclusivamente para utilizar dicho simulador.

Por este motivo, en el marco de esta tesis se ha implementado un Servicio Grid de Análisis de Estructuras, SAGS (Structural Analysis Grid Service), basado en el middleware Globus Toolkit 4 y en las tecnologías SOA, que ofrece, a través de Internet, a la comunidad de ingenieros y arquitectos un análisis estático y dinámico 3D de sus proyectos estructurales bajo demanda.

5.1. Estado del Arte del uso de Computación Distribuida y Grid en el Análisis de Estructuras

Gracias a la evolución de las redes de comunicaciones, y al avance de las tecnologías distribuidas y Grid, son nuevos los escenarios que surgen para el uso de aplicaciones informáticas relativas a la simulación de estructuras de edificación [46]. De este modo, gracias al uso de las tecnologías Grid, los ingenieros pueden tener acceso y adoptar como herramienta de trabajo cualquier tipo de recurso remoto, computadoras de altas prestaciones, sistemas de almacenamiento, aplicaciones software, etc.

Peng y Law [47, 48] presentan la implementación de una plataforma prototipo que permite la utilización y el desarrollo colaborativo de aplicaciones para el cálculo de estructuras, sacando partido de la alta conectividad ofrecida por Internet y de las tecnologías Web. Un ejemplo concreto de utilización de ese prototipo es el desarrollo de una aplicación Web que permite a los ingenieros tener acceso directo, vía Internet, al programa de simulación OpenSees y al análisis de los resultados empleando un navegador Web u otro tipo de cliente. En esta aplicación concreta, el software necesario para llevar a cabo el análisis de estructuras es ejecutado en un entorno distribuido y paralelo conocido. Posteriormente, la plataforma ha sido mejorada [49] para permitir el desarrollo de programas basados en elementos finitos a partir de Servicios Web. De este modo, se flexibilizaba el acceso a la plataforma por parte de usuarios y desarrolladores, permitiendo la incorporación de nuevas tecnologías, algoritmos y estrategias para mejorar el proceso de análisis de estructuras. En esta línea, Dolenc [50] describe una plataforma SOA, que proporciona un conjunto de especificaciones y servicios que asisten a investigadores en el desarrollo de aplicaciones específicas, orientadas a servicios, para el análisis estático de estructuras usando elementos finitos. El prototipo expuesto trata de establecer una herramienta común para el análisis de estructuras mediante la cual los investigadores pueden construir, testear e incorporar fácilmente nuevos desarrollos.

Raghunath [51] presenta el desarrollo de un entorno de computación de elementos finitos orientado a servicios, empleando un modelo cliente-servidor distribuido basado en CORBA. El código paralelo de elementos finitos empleado se ejecuta en un cluster de estaciones de trabajo. Por otro lado, Chen y Lin [52] exponen la arquitectura de un sistema prototipo, accesible mediante Internet, que proporciona un servicio para el análisis de estructuras mediante elementos finitos. A partir de una aplicación gráfica Java, el usuario podrá llevar a cabo las fases de pre-proceso y post-proceso y acceder vía Internet a la aplicación MPI de elementos finitos. En la misma línea, Chen y Fen [53] han desarrollado un entorno Web distribuido para dar soporte al problema concreto de optimización topológica de una estructura. En este entorno, la comunicaciones entre los componentes integrantes del sistema se llevan a cabo empleando la tecnología de Java RMI. En [54] se describe un prototipo de programa basado en elementos finitos para la simulación numérica del colapso de estructuras reforzadas accesible desde un portal Web, donde los usuarios tienen acceso a los módulos de pre-proceso, proceso y post-proceso.

Como prueba de concepto del potencial de la aplicación de las tecnologías Grid al análisis de estructuras, dentro del GRyCAP se desarrolló un prototipo Grid que por

medio de shell scripts permitía llevar a cabo el análisis estático de estructuras en 3D en un entorno Grid basado en GT2 [55]. Posteriormente, se desarrolló un sistema de alta productividad [56], que utilizaba un planificador Grid para llevar a cabo el análisis de estructuras también por medio de GT2. En la línea de alta productividad, Dolenc [57] describe el entorno ofrecido por los servicios y recursos Grid que componen la infraestructura del proyecto InteliGrid [58]. En este caso, el sistema de colas de Condor es el utilizado como planificador para las simulaciones paramétricas.

Wei, Zheng y Zhang [59] presentan una plataforma Grid, basada en Globus Toolkit que ofrece un entorno integrado para el desarrollo de aplicaciones de ingeniería. En un principio, la plataforma ha sido testeada para ejecutar aplicaciones de mecánica de sólidos. También utilizando las tecnologías Grid, Goodyer [60] expone el desarrollo de un sistema software para la resolución de problemas en ingeniería, capaz de ejecutar trabajos en recursos paralelos distribuidos pertenecientes a un Grid computacional. Por otro lado, Petrinja y Turk [61] presentan una aplicación Web basada en GT4 que da soporte a un entorno distribuido para la detección de fallos o defectos en la fabricación de productos.

Aziz [62] propone el uso de Grids semánticos para el desarrollo de un sistema para la gestión de emergencias por medio de la computación Grid y los Servicios Web. Este sistema hace un uso extensivo de la modelización, simulación, minería de datos y visualización para dar soporte a una red distribuida compuesta por especialistas en diferentes áreas implicadas, ingeniería estructural, analistas, etc. Finalmente, en [63] se presenta una aplicación de las tecnologías Grid para el diseño de entornos colaborativos para aplicaciones de diseño.

5.2. El Servicio Grid de Análisis de Estructuras de Edificación (SAGS)

Como ya se ha mencionado anteriormente, uno de los principales objetivos del Servicio Grid de Análisis de Estructuras es el de facilitar el uso del simulador MPI por parte de la comunidad de ingenieros estructurales, de modo que mediante clientes ligeros instalados en PCs convencionales se puedan calcular estructuras de cualquier dimensión y tipología. SAGS se presenta como una puerta de acceso a una infraestructura Grid compuesta por un conjunto de recursos de cómputo y almacenamiento. De este modo, por un lado permite explotar la potencia del simulador estructural mediante su ejecución en recursos paralelos y por el otro, dá soporte a la ejecución concurrente de múltiples simulaciones estructurales.

La infraestructura Grid subyacente es gestionada en SAGS por medio del planificador GMarte, el cual se encarga, de manera totalmente transparente al usuario, de seleccionar el recurso computacional más adecuado para llevar a cabo cada simulación, así como de la monitorización y la recogida de resultados. En nuestro caso, se dispone de un Grid Computacional compuesto por varios clusters de PCs, donde finalmente se ejecutará la aplicación paralela.

Empleando la flexibilidad ofrecida por los Servicios Web, las diferentes funcionalidades de SAGS han sido encapsuladas dentro de diferentes métodos: envío de la simulación, recogida de resultados, borrado de los datos, etc. Todos estos métodos son publicados mediante WSDL y son invocados mediante peticiones SOAP. El uso de estos protocolos, totalmente independientes de la plataforma, establece un interfaz estándar permitiendo su utilización por cualquier tipo de cliente.

Además, el servicio desarrollado incluye un sistema de tolerancia a fallos multinivel, garantizando que toda petición de análisis sea atendida satisfactoriamente. Los mecanismos de gestión de fallos se extienden a toda la aplicación, abarcando el

servicio y el planificador Grid, incluyendo la gestión de las peticiones, el transporte de los datos y la ejecución remota de las simulaciones. De este modo, diferentes tipos de fallos son tolerados, por ejemplo, la caída de un recurso Grid se gestiona provocando la migración automática de sus tareas hacia otro recurso que esté disponible.

Un sistema que esté accesible online deberá ofrecer ciertas garantías de seguridad, preservando la integridad de los datos y el correcto funcionamiento del sistema. Por ello, SAGS ofrece una robusta capa de seguridad, basada en Globus GSI [39], que garantiza la confidencialidad de los datos intercambiados y que lleva a cabo un control de acceso al servicio y a los recursos. Empleando mecanismos de autorización y autenticación se impide a usuarios no autorizados la utilización del sistema. Adicionalmente, la inclusión de algoritmos de cifrado de clave asimétrica permite a SAGS preservar la privacidad e integridad de los datos, evitando que sean accedidos o alterados por usuarios no autorizados para ello.

Para la utilización de SAGS se dispone de una potente aplicación gráfica 3D desarrollada también dentro del GRyCAP, la cual asiste a los ingenieros en las fases de pre-proceso y post-proceso del análisis de estructuras. Mediante esta aplicación, el cliente definirá las propiedades de su edificio de manera muy intuitiva, permitiendo llevar a cabo el cálculo por medio de SAGS. Para ello, se ha incluido un interfaz de conexión con SAGS que gestiona el envío de las peticiones y la recogida de los resultados por medio de Internet. Esto permite el acceso al servicio mediante una herramienta gráfica que puede estar instalada en un PC convencional, dotando de más usabilidad al sistema. Una vez que los resultados son recogidos, estos son automáticamente representados gráficamente en el interfaz, asistiendo al usuario en la fase de post-proceso y análisis de resultados.

5.2.1. Arquitectura de SAGS

Como la Figura 5.1 indica, el servicio está compuesto principalmente por 5 componentes: Service Manager, Task Notifier Daemon, Parallel Structural Simulator, Scheduler y Data Collector Daemon.

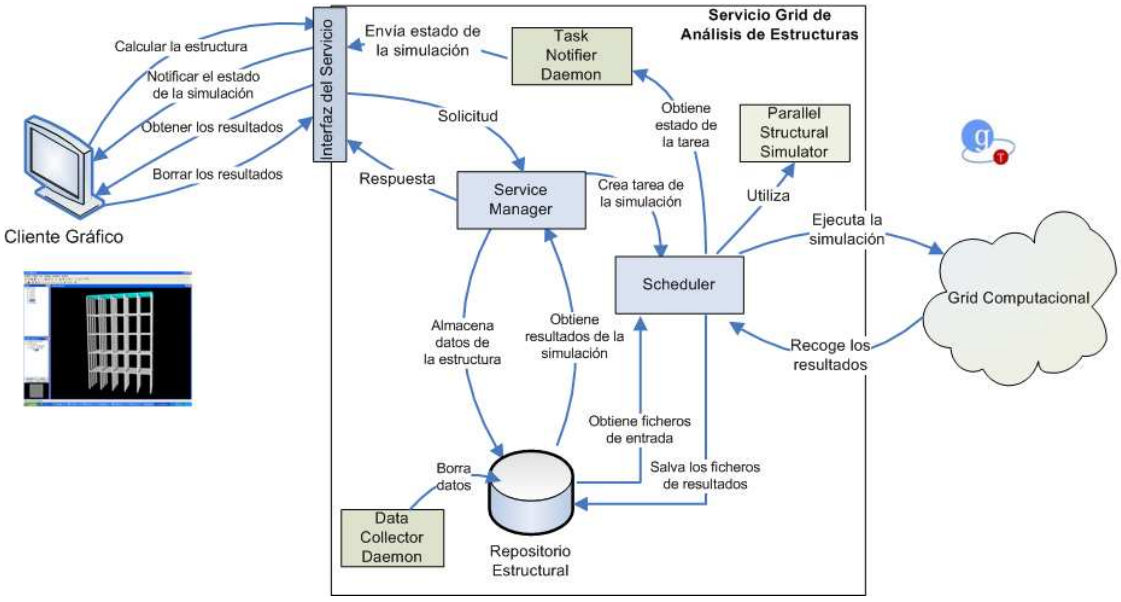

Figura 5.1. Arquitectura Servicio Grid de Análisis de Estructuras.

El *Service Manager* es el componente principal del sistema encargado de atender las peticiones de los clientes. Puede decirse que se trata del front-end del sistema, con

el que los usuarios interactúan, y que interconecta todos los elementos que componen el servicio.

El *Scheduler* realiza la ejecución remota de las simulaciones en la infraestructura Grid disponible. A partir de un Grid Computacional, compuesto en nuestro caso por varios clusters de PCs, este componente lleva a cabo un descubrimiento y selección de recursos, tratando de seleccionar el recurso más adecuado a partir de una política de planificación definida. Adicionalmente, para implementar la ejecución remota, el metaplanificador realiza de modo transparente la transferencia de los ficheros de entrada y la recogida de los resultados, así como la monitorización de la tarea para la detección de fallos. Dentro de SAGS, se ha incluido una instancia de GMarte para llevar a cabo todas estas tareas de metaplanificación. De este modo, SAGS proporcionará a GMarte una descripción de cada tarea para que éste gestione la ejecución remota de la misma.

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 sepa en cada momento en qué estado se encuentran sus tareas, bien sea en cola, ejecutándose, finalizadas, etc. Por otro lado, contamos con el simulador paralelo que lleva a cabo el análisis 3D estático y dinámico de las estructuras. El cálculo dinámico puede implementarse a través de un análisis modal o mediante diferentes métodos de integración directa.

Todos los datos de entrada y de salida correspondientes a cada simulación son almacenados en el Repositorio Estructural. El almacenamiento temporal de todos estos datos permite al sistema la re-ejecución de las simulaciones en el caso de que se produzca un fallo. Además, como puede observarse, estos esquemas de persistencia permiten a SAGS actuar no solamente como proveedor de capacidad de cómputo para el análisis de estructuras, sino también como un Servicio de Almacenamiento. Este valor añadido permitirá a los usuarios autorizados tener disponible online toda la información relativa a los datos y resultados de sus simulaciones, permitiendo su consulta desde cualquier punto de la red.

Finalmente, el *Data Collector Daemon* elimina periódicamente todos aquellos ficheros de resultados que pertenezcan a usuarios sin privilegios de almacenamiento. De este modo, dependiendo del nivel del usuario, los ficheros permanecerán disponibles en el repositorio o serán eliminados una vez que hayan sido recogidos.

5.2.1.1. Proceso de análisis de una estructura en SAGS

A continuación se expone una pequeña descripción de la secuencia de pasos que sigue el sistema, y de los componentes implicados para llevar a cabo el análisis de una estructura.

El cliente gráfico parte con una representación del interfaz llamado stub de cliente, previamente generada a partir de la descripción WSDL, la cual le permitirá interactuar con SAGS empleando una metodología orientada a objetos de alto nivel. Mediante este stub de cliente, el proceso se inicia con una petición de cálculo, la cual envía al servicio la geometría de la estructura a analizar, sus propiedades, las hipótesis de carga y los parámetros que definen el tipo de cálculo. Todos estos datos viajarán en XML dado que empleamos el estándar de los Servicios Web.

Una vez que la petición es recibida en la máquina que aloja SAGS, ésta será atendida por el Service Manager, el cual procesará los datos, los almacenará en el Repositorio Estructural y devolverá al cliente un identificador de la simulación. A partir de los datos de entrada enviados en XML por el cliente, se generará un fichero binario propietario con el formato impuesto por el simulador paralelo. Una vez almacenado dicho fichero, el Service Manager creará la tarea a ejecutar, la cual contendrá todas las propiedades necesarias para que pueda ejecutarse en el Grid Computacional.

Una vez definida la tarea, ésta es añadida al *Scheduler*, el cual está basado en el planificador GMarte a fin de que lleve a cabo, de manera transparente, la selección de recursos y la gestión de la ejecución de la simulación. Para llevar a cabo estas etapas, el planificador realizará la transferencia de los ficheros de entrada entre SAGS 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 mediante un esquema basado en notificaciones. De este modo, el cliente es perfectamente consciente en todo momento del estado de sus simulaciones: enviada, en cola, en ejecución, finalizada, fallida. Esta alternativa reduce notablemente la sobrecarga del sistema frente a una aproximación de consulta periódica por parte de los clientes.

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, debido a que se trata de un proceso iterativo con generación de resultados a cada paso de tiempo, se notifica al usuario cuando existe un número adecuado de resultados parciales disponibles, independientemente de que la ejecución haya o no finalizado. Esta aproximación permite una notable reducción en los tiempos de espera, ya que los resultados pueden empezar a ser transferidos aunque la simulación no haya finalizado en el recurso Grid, solapándose los tiempos de transferencia con los tiempos de ejecución. De este modo, el cliente podrá empezar a analizar los resultados sin tener que esperar a que la simulación haya finalizado.

La recogida de resultados puede ser llevada a cabo empleando dos estrategias diferentes. La primera es la basada en los Servicios Web empleando el estándar SOAP, donde la petición de recogida de resultados implica procesar los ficheros de resultados por parte del Service Manager, y construir la estructura de datos diseñada expresamente para devolverlos apropiadamente al cliente en XML. Por otro lado, se ofrece una alternativa para la descarga más eficiente de grandes volúmenes de datos mediante GridFTP. En esta aproximación, el cliente tiene acceso directo al espacio en disco dentro del Repositorio Estructural donde se han almacenado los resultados proporcionados por el simulador. En consecuencia, por medio de GridFTP, el cliente podrá descargarse directamente los ficheros binarios de resultados de sus simulaciones. Para ello, se ha desarrollado un componente de alto nivel fácil de integrar en cualquier cliente gráfico. A partir de este componente, serán transparentes para el cliente los detalles de bajo nivel del acceso y transporte de los datos que desee intercambiar con SAGS mediante GridFTP.

Por otro lado, SAGS 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 para todos aquellos clientes que no tengan permiso de almacenamiento de resultados.

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.

5.2.2. Diseño de SAGS

En este apartado pasamos a describir la fase de diseño que ha sido llevada a cabo previamente a la implementación de SAGS. Para ello, en este punto nos adentramos en la estructura de clases que compone SAGS, presentando los diagramas UML (Universal Modeling Language) diseñados, así como una descripción de alto nivel de la funcionalidad de cada una de ellas.

Como se puede observar en la Figura 5.2., que muestra una visión general de las clases que componen el sistema, los componentes de alto nivel especificados en la arquitectura son traducidos a este nivel en a clases.

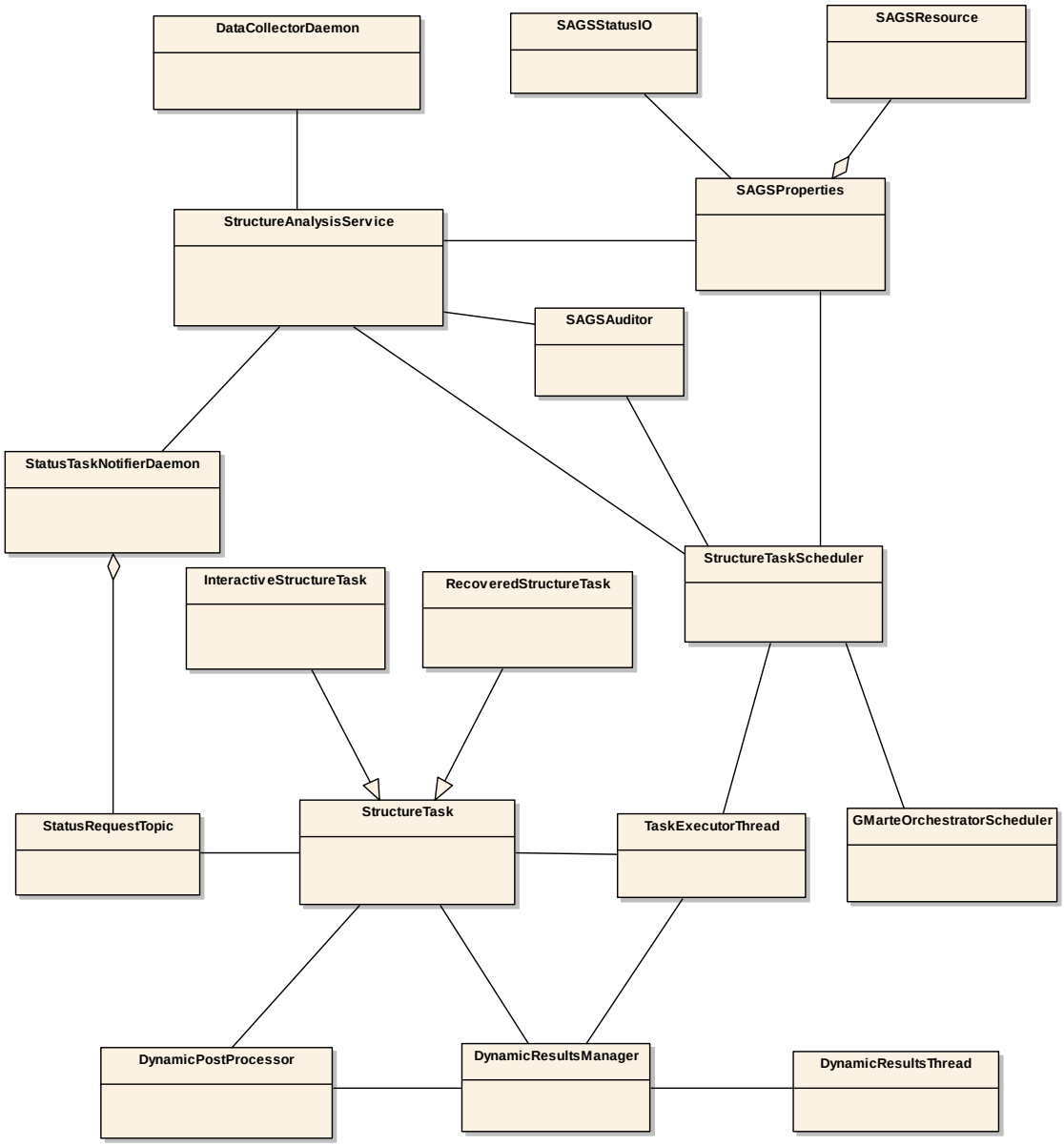

Figura 5.2. Diagrama de clases de SAGS.

La clase *StructureAnalysisService* es la que encapsula todos los servicios ofrecidos a la comunidad de arquitectos e ingenieros estructurales. Esta clase ofrecerá

los métodos publicados en la especificación WSDL de SAGS, mediando entre los usuarios SAGS y el resto del sistema. Como se puede observar en la Figura 5.3, esta clase se corresponde con el componente de alto nivel llamado ServiceManager, ofreciendo los métodos para el envío de las simulaciones y la recogida de los resultados.

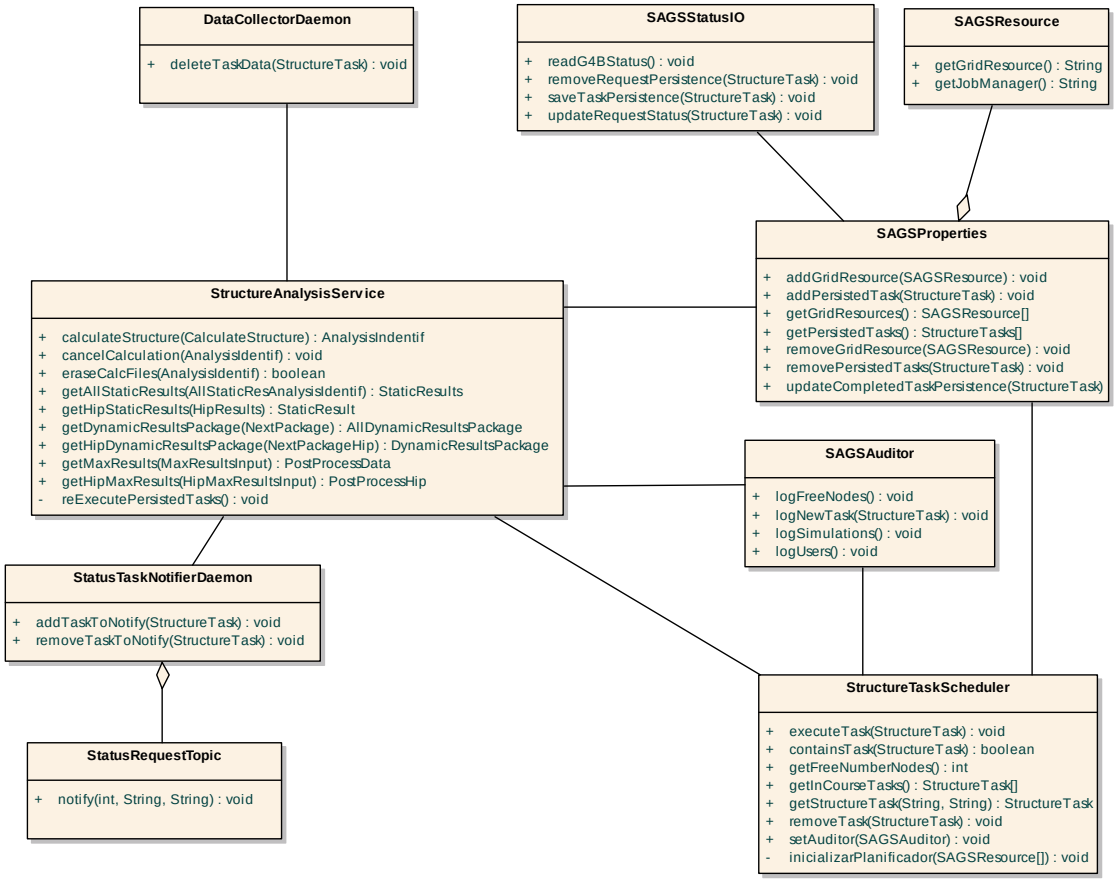

Figura 5.3. Clase StructureAnalysisService y sus relaciones.

Por otro lado, la clase *DataCollectorDaemon* actuará como un thread encargado de inspeccionar el repositorio estructural y de limpiar los datos de las tareas que hayan acabado pasado un tiempo prudencial. Como una primera aproximación, los datos pertenecientes a las tareas que pertenezcan a usuarios sin privilegios de almacenamiento serán eliminados a los 3 días desde que la simulación haya finalizado.

La clase *StatusTaskNotifierDaemon* es concebida como un thread independiente diseñado para llevar a cabo la monitorización las tareas que están siendo ejecutadas en la infraestructura Grid. De este modo, cuando una tarea es sometida a SAGS, ésta será añadida a la clase *StatusTaskNotifierDaemon*, la cual se encargará de mantener informado al cliente acerca de su evolución en el sistema. Para ello, se creará una entrada específica en la lista de notificaciones, *StatusRequestTopic*, para cada simulación, de modo que el cliente deberá suscribirse a él para que la clase StatusTaskNotifierDaemon le mantenga informado de cualquier cambio en el estado de la misma.

SAGS ha sido diseñado para ofrecer una máxima flexibilidad y adaptabilidad a cualquier tipo de entorno Grid. Es por ello que, dinámicamente, pueden ser añadidos o eliminados recursos a la infraestructura Grid subyacente, siendo SAGS capaz de adaptarse a estos cambios. Para ello se define la clase *SAGSProperties*, que a partir

de una serie de ficheros de configuración y teniendo en cuenta la evolución del Grid tendrá registrados el conjunto de recursos disponibles, así como sus características necesarias para poder llevar a cabo una ejecución remota en los mismos. Adicionalmente, esta clase es la encargada del mantenimiento persistente del estado de SAGS, de modo que ante un eventual fallo permita restaurar el sistema en el mismo punto en el que se encontraba. El estado salvado persistentemente englobará una descripción precisa de las tareas que estén pendientes de ser ejecutadas en el Grid, que estén en ejecución, así como de las que hayan finalizado pero tengan resultados aún disponibles por recoger. Para dar soporte a esta gestión y salvado del estado, se define la clase *SAGSStatusIO* que será la encargada del acceso a disco para la escritura y lectura de esta información.

En un sistema en producción, es necesaria la inclusión de técnicas de accounting para su explotación bajo un modelo basado en pago por servicios. Es por ello que aparece la clase *SAGSAuditor*, que será la encargada de llevar a cabo este tipo de tareas dentro de SAGS, posibilitando un control y registro del uso que cada usuario haga del sistema. Mediante estos mecanismos se dota al sistema de la capacidad de efectuar auditorias automáticas, cuando sea requerido. En una primera aproximación se registrarán las tareas que han sido ejecutadas, los tiempos de uso y el espacio de almacenamiento empleado por cada usuario del sistema, así como el número de nodos libres de la infraestructura Grid. Este primer paso está orientado a sentar las bases para el futuro desarrollo de interfaces de usuario que, de manera remota, permitan tele administrar el sistema.

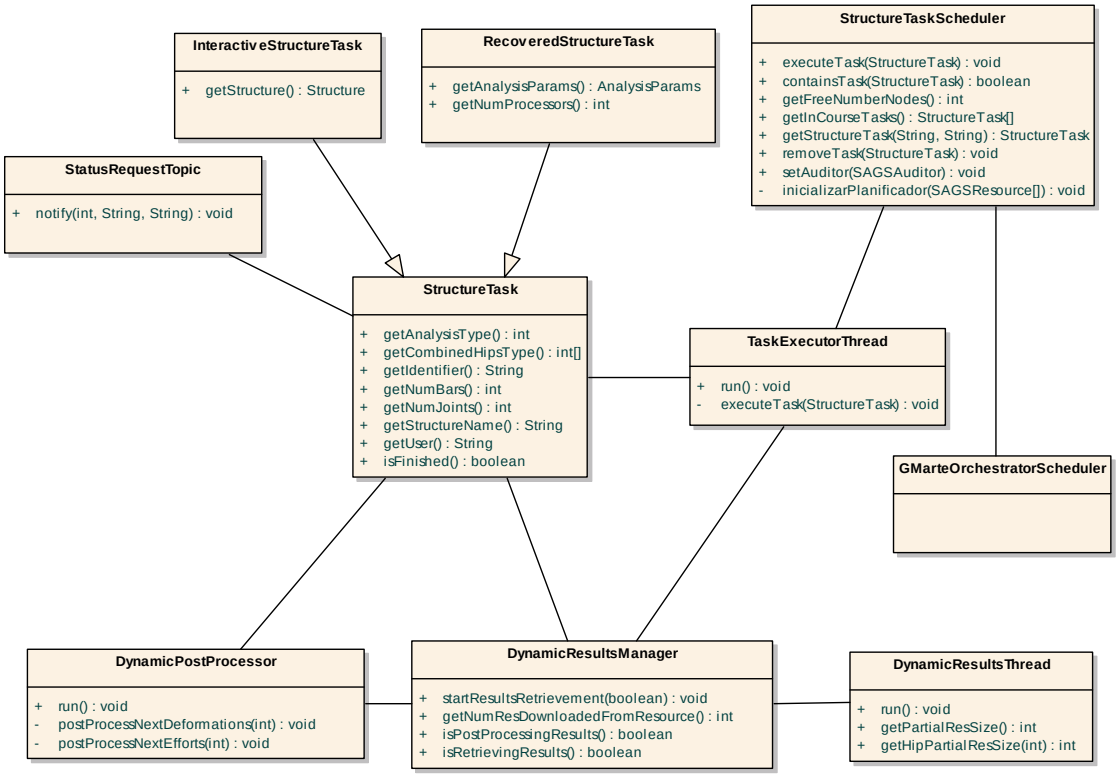

Figura 5.4. Arquitectura de las clases encargadas de la gestión de la ejecución de las simulaciones estructurales vía Grid.

Todas las simulaciones estructurales en SAGS son modeladas mediante clases hijas de la clase abstracta denominada *StructureTask*, Figura 5.4. Esta clase define una serie de métodos que permiten el acceso a las características de cada simulación. De este modo, se encapsulan todos sus datos de entrada teniéndolos centralizados y facilitando su ejecución en Grid. Por otro lado, como se puede observar en la Figura

5.4, la clase *StuctureTask* estará asociada a un conjunto de clases, cada una encargada de una tarea específica: gestión de sus notificaciones, gestión de su ejecución y gestión de la recogida de sus resultados. De este modo, cada tarea tendrá asociada una instancia diferente, las cuales en muchos casos se convierten en hilos independientes, llegando a una aproximación multihilo que se muestra muy eficiente en entornos con múltiples usuarios interactuando concurrentemente con SAGS.

A partir de la clase *StructureTask*, se definen dos tipos de tareas en el sistema, las que han sido lanzadas por un usuario siguiendo el proceso normal de ejecución y las que han sido recuperadas debido a una caída de SAGS. Esta distinción atiende al hecho de que se debe realizar un tratamiento especial para la gestión de las tareas que han sido recuperadas tras un fallo. Por ejemplo, en el caso de que la tarea ya haya sido planificada con unos requerimientos específicos, estos deberán volver a ser establecidos.

La clase *StructureTaskScheduler* representa el elemento principal, sobre el que se sustenta la gestión de la ejecución de las simulaciones estructurales en la infraestructura Grid disponible. Para ello, esta clase ofrece métodos para añadir y eliminar tareas, así como para consultar las tareas que están siendo ejecutadas y el número de nodos Grid disponibles. En el apartado 5.1.3.1. se detallará la política de planificación de tareas y repartición de la carga definida para finalmente llevar a cabo una distribución eficiente de las tareas, empleando las clases ofrecidas por el API de GMarte.

A partir de la política de planificación definida, la clase *StructureTaskScheduler* asignará unos requerimientos computacionales a cada tarea, que serán empleados por el planificador GMarte para llevar a cabo la selección de recursos. Estos requisitos, especificarán, dependiendo de las características y tipología de cada simulación, el número de procesadores con el que será ejecutada. Estos mecanismos están dirigidos a ofrecer un eficiente reparto de carga y aprovechamiento de los recursos Grid. Una vez que la tarea esté completamente definida, se creará un thread llamado *TaskExecutorThread*, exclusivamente dedicado a interactuar con GMarte para llevar a cabo la ejecución remota de la simulación. La estrategia multihilo seguida permite que cada tarea tenga asociado un hilo independiente para gestionar su ejecución.

Por otro lado, en el caso de realizar un análisis dinámico, se definen un conjunto de clases orientadas a la gestión de la recogida de los resultados parciales y del post-proceso de los mismos. Como ya se ha explicado anteriormente, el análisis dinámico de estructuras se basa en un proceso iterativo donde en cada paso de tiempo se generan resultados parciales que deben ser analizados. Esta característica posibilita la descarga de resultados del recurso Grid a la vez que la simulación está siendo ejecutada, a diferencia de un análisis estático donde los resultados son obtenidos una vez que la simulación ha finalizado. La recogida y post-proceso de resultados dinámicos de manera eficiente se implementa por medio de la clase *DynamicResultsManager.* Se trata de la clase principal encargada de la gestión de la ejecución de simulaciones estructurales dinámicas. Adicionalmente, SAGS integra mecanismos de post-proceso que permiten un filtrado de los datos, evitando, en el caso que el cliente lo desee, la descarga de todos los datos generados por su simulación y devolviendo por ejemplo valores máximos de los elementos estructurales.

Para la descarga de los resultados parciales de cada simulación, la clase *DynamicResultsManager* instanciará un thread llamado *DynamicResultsThread*. Este hilo se encargará de manera independiente de la descarga de cada conjunto de resultados parciales generado por unidad de tiempo. Por otro lado, el post-procesado de los datos dinámicos será llevado a cabo por la clase *DynamicPostProcessor*, que a modo de hilo analizará cada resultado parcial descargado, actualizando el registro de los máximos valores que comentábamos anteriormente. Este registro de máximo

valores será almacenado dentro de un fichero en el Repositorio Estructural, siendo posible su descarga a través de SAGS.

5.2.2.1. Ejecución en el Grid

SAGS ejecuta el Simulador Estructural Paralelo empleando la funcionalidad ofrecida por GMarte. Como ya se ha comentado anteriormente, GMarte es un software Grid de alto nivel que ofrece un API orientada a objetos en Java, tanto para la descripción de tareas y recursos computacionales como para la ejecución de la tarea en el recurso computacional y la monitorización de su estado. De este modo, GMarte ofrece la funcionalidad necesaria para llevar a cabo la ejecución de tareas sobre entornos Grid y LCG-2.

Como ya ha sido comentado anteriormente, la clase *StructureTaskScheduler* es la que se ocupa de interactuar con GMarte para proporcionar una descripción de las simulaciones estructurales y llevar a cabo una ejecución eficiente de las mismas. En este sentido, puesto que el Simulador Estructural Paralelo es una herramienta basada en MPI, SAGS aprovecha sus características para definir, en tiempo de ejecución, el número de procesadores que en paralelo resolverán el problema. A partir de un análisis de prestaciones de la herramienta paralela y de la disponibilidad de los recursos Grid, se ha implementado una heurística que trata de repartir la carga eficientemente. Mediante el análisis de prestaciones de la herramienta paralela y basándonos en los porcentajes de eficiencia ofrecidos, llegamos a la conclusión de que la mejora ofrecida por una ejecución en más de 4 procesadores no compensaba la ocupación de tal número de nodos, a no ser que se demandará una cantidad de memoria RAM que los hiciera necesario, ya que una ejecución paralela en un elevado número de procesadores suponía un pobre aprovechamiento de la infraestructura Grid, puesto que otras tareas de SAGS tendrían que estar en cola.

Por lo tanto, se ha diseñado una heurística de reparto de carga basada en un modelo de alta productividad, donde en función del tipo de análisis y de la dimensión del problema (número de grados de libertad de la estructura), se ejecutan las simulaciones en 1, 2 o 4 procesadores. Esta aproximación saca partido tanto al paralelismo ofrecido por el Simulador Estructural como a la distribución de tareas ofrecida por una infraestructura Grid. Como se puede observar se trata de buscar una mejora en las prestaciones del sistema global, y no en una mejora en la ejecución de tareas individuales, a fin de reducir los tiempos de espera de los usuarios.

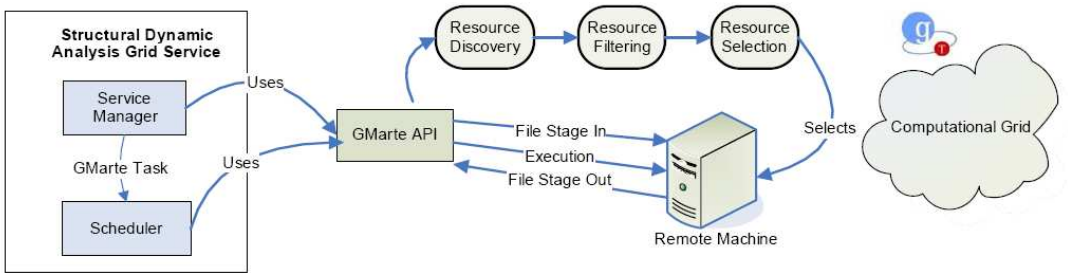

Figura 5.5. Interacción de SAGS con GMarte.

Adicionalmente a los requerimientos de cómputo de la simulación, la clase *StructureTaskScheduler* definirá, para cada tarea, los ficheros de entrada y de salida que requiere, y que GMarte deberá transferir automáticamente entre la máquina que aloja SAGS y el recurso Grid. Una vez que han sido especificadas todas las características y requisitos que son necesarios para poder llevar a cabo la ejecución remota de la simulación, el planificador de GMarte se encargará de realizar este

proceso, por medio de una secuencia de pasos, tal y como muestra la Figura 5.5. En primer lugar, tiene lugar una selección de recursos orientada a seleccionar el mejor recurso computacional disponible para ejecutar la simulación. A partir de ahí, todos los ficheros de entrada necesarios serán automáticamente transferidos desde el Repositorio Estructural hacia la máquina remota, antes de que sea lanzada la ejecución. Una vez que la simulación haya finalizado, en el caso de un análisis estático, o que haya comenzado a generar resultados, en el caso del análisis dinámico, estos serán automáticamente transferidos desde el recurso remoto al Repositorio Estructural.

Una de las características ventajosas de SAGS al emplear GMarte es la aproximación de metaplanificación multihilo que este último ofrece, y que permite de manera concurrente llevar a cabo la fase de selección de recursos y la fase de ejecución remota. Esta alternativa reduce notablemente los tiempos de espera asociados al proceso de metaplanificación, y en consecuencia aumenta la productividad del sistema. De este modo, en el caso en el que coincidan en el sistema múltiples simulaciones, ninguna de ellas se verá bloqueada esperando que la tarea que le precede promocione de estado.

5.2.3. Detalles de implementación

5.2.3.1. Utilización de SAGS

Pasamos a continuación a describir los diferentes métodos que son publicados por SAGS en WSDL y como deben ser invocados y empleados por la comunidad estructural.

El método *CalculateStructure*, será el primer método que un usuario deberá emplear para someter una petición de análisis a SAGS. Este método recibe en un parámetro encapsulada toda la geometría y las propiedades estructurales del edificio, así como las cargas externas aplicadas, y devuelve un identificador de la tarea. Por un lado, las propiedades geométricas son definidas por medio de las coordenadas 3D de los nudos de la estructura y las conexiones de los mismos definidos por las barras. Por otro lado, las propiedades estructurales son definidas por medio de las secciones de cada barra y por los impedimentos en el movimiento de los nudos. Las deformaciones o los esfuerzos serán obtenidos en función de las cargas que también serán suministradas. Además de los datos relativos a la estructura y sus propiedades, los parámetros de análisis también deberán ser suministrados ya que son requeridos por el Simulador Paralelo. Estos parámetros determinan el tipo de análisis, estático o dinámico, con métodos de integración directa o análisis modal. Adicionalmente al tipo de análisis, si se trata de un análisis dinámico, se deberá especificar la duración de la simulación, el intervalo de paso de tiempo a emplear, cada cuantos instantes de tiempo se desean salvar los resultados a disco, etc. En el caso que se trate de un análisis por métodos de integración directa, se deberá proporcionar el método que se desea emplear junto con sus parámetros asociados, mientras que si se desea aplicar un análisis modal se deberá proporcionar, entre otros parámetros, el número de frecuencias naturales a obtener.

El método *CancelCalculation*, recibirá como entrada, el identificador de la tarea que habrá sido obtenido a partir del método *CalculateStructure*. A partir de ese identificador, SAGS eliminará la simulación del sistema, borrando todos sus datos del Repositorio Estructural, y en el caso de que esté siendo ejecutada en el Grid será cancelada empleando el API de GMarte.

En cualquier momento, el estado de la simulación puede ser obtenido por los clientes empleando el método *GetCalculationStatus*. Los posibles estados de las simulaciones son los siguientes: enviada, en cola, en ejecución, completada, fallida,

cancelada. Sin embargo, a partir de la inclusión del esquema de notificaciones, la utilización de este método queda desaconsejado y solamente se mantiene por propósitos de depuración. Esto es así puesto que, como ya se ha mencionado anteriormente, la utilización de un esquema basado en notificaciones es la más adecuada en este tipo de sistema multiusuario, a fin de no colapsar el servicio sometiendo peticiones periódicas para conocer el estado de sus tareas. Para ello se empleará el método *Subscribe*, al cual se le deberá pasar el identificador de la tarea a la que nos queremos suscribir.

Dependiendo del tipo de análisis, SAGS ofrece diferentes métodos para la recogida de los resultados. Por un lado, los métodos *GetHipStaticResults* y *GetAllStaticResults* permiten al usuario la recogida de los resultados de cálculo para una hipótesis de carga concreta o para todas. En el caso de que se desee recoger solo los resultados de una hipótesis, deberá ser pasado como parámetro el identificador de la misma. Por otro lado, los resultados dinámicos son enviados dentro de paquetes que agruparán resultados de varios pasos de tiempo. El número de pasos de tiempo a enviar, en un mismo paquete, dependerá del tamaño de los ficheros asociados a cada uno. Teniendo en cuenta los problemas que presenta el transporte de grandes mensajes SOAP, los expertos recomiendan que su tamaño no exceda de los 4 Mbytes. Por lo tanto, empleando esta recomendación, al igual que para la recogida de resultados estáticos, aparecen dos métodos *GetHipDynamicResultsPackage* y *GetDynamicResultsPackage*, los cuales devolverán un paquete de resultados dinámicos a partir del paso de tiempo que se le pase como parámetro.

Finalmente, en el caso que se haya llevado a cabo un análisis dinámico, se pueden emplear los métodos *getHipMaxResults* y *getMaxResults* para recoger los resultados filtrados. Estos mecanismos permiten a los usuarios descargarse únicamente los valores máximos y mínimos de esfuerzos y deformadas alcanzados en la simulación. De este modo, se suministra a los clientes solamente la información que necesitan para llevar a cabo la interpretación de la normativa de diseño, reduciendo en varios órdenes de magnitud el tamaño de los datos a descargar desde SAGS.

5.2.3.2. Gestión de la transferencia de resultados

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. Como ya ha sido comentado anteriormente, en un análisis dinámico la transferencia de los resultados entre el recurso Grid y SAGS se ha implementado por medio un sistema de recogida de resultados parciales. Para ello se ha desarrollado un mecanismo configurable, basado en la definición de diferentes sufijos para los ficheros de salida. GMarte ofrece la posibilidad de la descarga y subida de ficheros a partir de una descripción mediante sufijos o prefijos. De este modo, el Simulador Estructural Paralelo genera sus ficheros de salida con un sufijo asociado a cada paso de tiempo (t1, t2, t3, etc.). A partir de estos sufijos, la clase *DynamicResultsManager* del servicio irá dinámicamente añadiéndolos a la configuración de la simulación para irlos descargando. Sin embargo, es necesario llevar a cabo un proceso de sincronización en la recogida de los resultados parciales entre la configuración de GMarte y la generación de los mismos en el Simulador Paralelo. Esta sincronización se logra empleando un fichero adicional en el cual el Simulador Paralelo registrará el tiempo real de ejecución que transcurre entre la simulación de dos pasos de tiempo consecutivos en los que se almacenan resultados.

Por otro lado, uno de los principales problemas relacionados con el uso de las Arquitecturas Orientadas a Servicios está relacionado con la sobrecarga introducida en las comunicaciones entre los clientes y los servicios, más aún cuando existe un gran volumen de datos a transferir como habitualmente sucede en nuestro caso. Como ya

ha sido explicado en secciones anteriores, las comunicaciones con SAGS son llevadas a cabo mediante mensajes SOAP basados en XML, lo cual introduce una sobrecarga importante. Como alternativa más eficiente para transferir grandes volúmenes de datos, se ofrece el uso de GridFTP.

La alternativa basada en SOAP fue la aproximación natural, apoyada en el uso del estándar de los Servicios Web, que se empleó para la transferencia de los datos entre SAGS y sus clientes. En un primer prototipo, se emplearon esquemas completos de especificación que contemplaban en detalle los datos de entrada y salida asociados al análisis de estructuras mediante SAGS. Sin embargo, esta aproximación fue rápidamente descartada por ser inviable su utilización con grandes volúmenes de datos. A partir de ahí, se desarrolló un esquema de codificación que daba soporte a la inclusión de datos binarios embebidos en mensajes de texto SOAP.

Existen varias alternativas de codificación, evaluando en nuestro caso las más extendidas: codificación Hexadecimal y codificación Base64. Un esquema de codificación hexadecimal convierte cada byte binario en dos caracteres, incrementando en un 100% el tamaño de los datos originales. Sin embargo, el proceso de codificación y decodificación es muy ligero y simple. Por otro lado, una estrategia de codificación Base64 emplea un proceso de mapeo a caracteres ASCII. Este proceso fragmenta cada grupo de tres bytes binarios en cuatro grupos de seis bits, y por medio de una tabla los traduce a cuatro caracteres ASCII. Mediante este sistema solamente se incrementa en un 33% el tamaño de los datos, sin embargo el proceso de codificación y decodificación es más complejo y costoso. Finalmente, el esquema de codificación Base64 ha sido el incluido para dar soporte de manera eficiente a las comunicaciones mediante SOAP. Esta decisión se apoya en el hecho de que, a este nivel, el cuello del botella de SAGS está mas del lado de las comunicaciones que del cómputo, y una codificación Base64 introduce una mejor sobrecarga que la Hexadecimal, aportando una mejora sustancial en los tiempos de descarga de los resultados de análisis. Por lo tanto, las comunicaciones SOAP que actualmente integra SAGS, se basan en el uso de esquemas híbridos, con texto y datos binarios codificados en Base64. En las comunicaciones que requieren grandes volúmenes de datos a transferir, como es el caso de la recogida de resultados, se emplea este esquema híbrido. Por otro lado, para los mensajes cortos se emplea el mecanismo tradicional basado en texto XML.

En cuanto a la aproximación basada en GridFTP, los datos binarios son directamente recogidos empleando mecanismos altamente eficientes de transferencia. Como ya se ha mencionado en el capítulo 4, el protocolo GridFTP extiende el estándar FTP para incluir mecanismos, altamente seguros y eficientes, para la transferencia de grandes volúmenes de datos en un sistema Grid. De este modo, se ofrece a los clientes acceso directo al espacio reservado para sus simulaciones dentro del Repositorio Estructural. Sin embargo, esto requeriría por parte de los usuarios de SAGS el adquirir ciertas habilidades para poder interactuar con el servicio GridFTP. Como solución, se ha desarrollado una herramienta cliente GridFTP, empleando Java CoG Kit [64], que facilita el acceso al Repositorio Estructural a los usuarios. Esta aproximación es mucho más eficiente que la alternativa SOAP, ya que los datos son directamente transportados en formato binario, sin ninguna información extra y de manera altamente eficiente. Sin embargo, requiere la instalación de algunas librerías de GT4 en la parte cliente, lo cual plantea serios problemas de portabilidad a diferentes plataformas. Por lo tanto, la alternativa basada en GridFTP es reservada para clientes que necesiten una máxima eficiencia en la transferencia de los datos desde SAGS, y donde la portabilidad del cliente no represente un problema.

5.2.4. Tolerancia a fallos

La inclusión de un esquema de tolerancia a fallos robusto es un aspecto crítico en un sistema que trata de ofrecer un servicio bajo demanda, en el cual el cliente exigirá las máximas garantías. En SAGS se ha desarrollado un esquema tolerante a fallos multinivel que manejará tanto los fallos hardware como software de todos los componentes del sistema, tanto a nivel del servicio como a nivel de planificación y ejecución en el 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 es muy importante siendo muy valorada por los clientes.

5.2.5. Aspectos de seguridad

La inclusión de una capa de seguridad en SAGS es esencial, ya que va a ser un sistema accesible a diferentes tipos de usuarios y permanentemente disponible online. Por lo tanto, es importante analizar los requerimientos de seguridad de tal tipo de sistema, considerando aspectos como la autorización y la autenticación o la privacidad e integridad de los datos. Todos estos mecanismos han sido incluidos en SAGS, empleando la capa de seguridad que ofrece el sistema GSI de Globus Toolkit.

Por un lado, la inclusión de mecanismos de autenticación y autorización es necesaria para establecer un control de acceso en los servicios publicados y poder registrar las acciones llevadas a cabo por cada cliente. Estos mecanismos han sido implementados empleando el modelo de autorización basado en ficheros Gridmap ofrecido por Globus, que no es más que una lista de usuarios autorizados similar a una lista de control de acceso. De este modo, solamente los clientes que estén incluidos en ella podrán invocar el servicio, siendo rechazadas las peticiones del resto. Este esquema de autorización está apoyado en la utilización de certificados X.509. Cada cliente poseerá un certificado propio el cual será enviado a SAGS al principio de la comunicación. Dentro de este certificado viaja lo que se conoce como “*distinguished name*” del usuario, que no es más que un identificador único de usuario. A partir de este identificador y empleando la lista definida por el fichero Gridmap, el usuario será validado en el sistema. SAGS ha sido implementado definiendo diferentes niveles de usuarios con diferentes privilegios asociados, determinando qué acciones podrá llevar a cabo cada uno de ellos.

Por otro lado, SAGS contiene un fichero propio de configuración el cual centraliza y define el tipo de acceso requerido para la invocación de cada uno de sus métodos. Por lo tanto, para cambiar algunas configuraciones de seguridad solamente hace falta

modificar dicho fichero y no el código fuente. Para cada método publicado, este descriptor de seguridad especifica cómo se debe invocar al mismo desde el punto de vista de la seguridad (si los datos van cifrados, firmados, etc.). Además, también se ha definido el esquema de protección que se utilizará para el cifrado, es decir, si se va a emplear a nivel de transporte o a nivel de mensaje. Tanto el cifrado a nivel de transporte como a nivel de mensaje está basado en los esquemas de clave pública empleados en GSI, con lo que pueden garantizar la privacidad e integridad de los datos. Existen ciertas diferencias entre ambas aproximaciones a la hora del cifrado de los datos, ya que a nivel de transporte se cifra el mensaje SOAP completo, mientras que a nivel de mensaje solamente se cifra el cuerpo del mensaje SOAP.

Cabe destacar que existen unas altas medidas de seguridad involucradas en la ejecución remota de tareas sobre recursos basados en Globus. Por ejemplo, también son necesarias credenciales de usuario válidas para poder llevar a cabo la ejecución en un recurso computacional. En el sistema desarrollado, se han generado unas credenciales únicas con suficientes privilegios para ejecutar todas las simulaciones en la infraestructura Grid. SAGS es el poseedor de esas credenciales, las cuales son suministradas a GMarte cuando gestiona la ejecución de una tarea. Esta decisión ha sido tomada para flexibilizar al máximo la inclusión de nuevos usuarios en SAGS, evitando tener que modificar todos los esquemas de seguridad de los recursos Grid. Se entiende que un usuario, que ya posea privilegios para ejecutar en el Grid, habrá pasado el filtro previo fijado por SAGS definido anteriormente, con lo que se evita la aparición de intrusiones indeseadas. De este modo, todos los usuarios autorizados y autenticados por SAGS podrán ejecutar sus simulaciones en la infraestructura Grid sin tener permisos explícitos para ello.

5.3. 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, dentro del GRyCAP, y fuera del marco de esta tesis de Máster se ha desarrollado una avanzada interfaz gráfica de usuario muy realista que le asiste de modo amigable en la etapa de entrada de datos, o pre-proceso, en la cual se configuran las distintas particularidades de los elementos estructurales del edificio (propiedades de las secciones, condiciones de sustentación, cargas, etc.), la definición de los parámetros de cálculo y la interpretación de los resultados de salida, o post-proceso.

Por medio de las librerías gráficas ofrecidas por el API Java 3D [65], esta aplicación multiplataforma (ver Figura 5.6.) 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 pre-proceso, convirtiéndose en un proceso muy intuitivo y rápido.

La funcionalidad y características de esta aplicación gráfica se resumen en los siguientes bloques:

- Asignación de secciones a las barras de la estructura. Con esta herramienta el usuario podrá cambiar el tipo de sección y el material de las barras que forman el edificio. Esta asignación modificará tanto las propiedades físicas estructurales de la barra como su representación gráfica en la escena. Existen prontuarios de secciones estándar, que ayudan a la definición de un catálogo propio para cada estructura. Con dicho catálogo, el usuario puede seleccionar secciones concretas y asignarlas a las barras.

- Creación de cargas estáticas y dinámicas en la estructura. El usuario dispone de herramientas para la definición de las diferentes acciones o cargas externas que son aplicables a la estructura, tanto en su fase de construcción como en la de su vida útil. Estas cargas se generarán dentro de una hipótesis de carga concreta, las cuales son utilizadas por el módulo de cálculo para obtener la respuesta de la estructura ante la acción de las mismas.
- Creación de coacciones en la estructura. Con esta herramienta el usuario puede limitar la libertad de movimiento (ante las tensiones producidas por las cargas) de distintos elementos de la estructura. Por ejemplo, se pueden empotrar nudos, con lo cual se les obligará a permanecer inmóviles con respecto a su posición original, cosa que puede ocurrir, por ejemplo, en aquellos nudos correspondientes a la cimentación de la estructura.
- Creación de hipótesis de carga básicas y combinadas. La aplicación soporta la creación de hipótesis de carga básicas y gestionará su combinación mediante las llamadas hipótesis combinadas.
- Visualización realista de los resultados de cálculo. La aplicación gráfica integra herramientas gráficas altamente realistas para la visualización de esfuerzos y deformadas. De este modo, el usuario podrá detectar fácilmente cualquier problema que afectaría a la estructura en las condiciones que haya establecido para la simulación.
- Filtrado de resultados. Se ofrece mecanismos para el análisis y filtrado de resultados que permite al usuario la identificación de manera muy rápida de los elementos que superen algún umbral establecido.

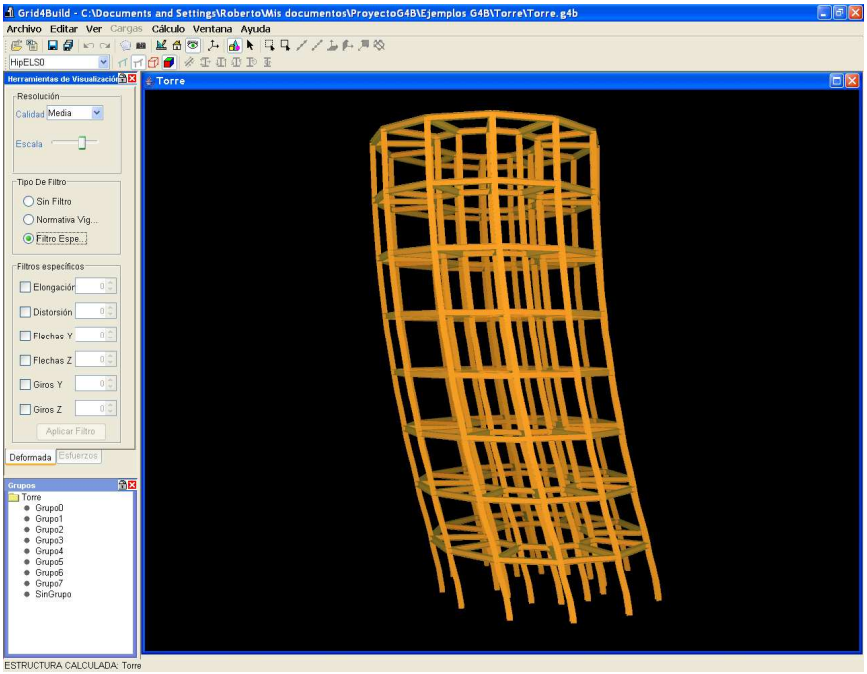

Figura 5.6. Aspecto del cliente gráfico del Servicio de Análisis de Estructuras.

El análisis de las estructuras es llevado a cabo por medio de SAGS, 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 gráfico se suscribe a las notificaciones de su

simulación y comienza a ser informado del progreso de la misma. Finalmente, cuando se notifica que hay resultados disponibles, estos 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 post-proceso, 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 incluso si ésta 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 SAGS, 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.

5.4. Caso de Estudio Estructural

Para evaluar las prestaciones de SAGS en un entorno multiusuario, se ha analizado un caso de estudio compuesto por múltiples simulaciones empleando una infraestructura Grid.

El caso de estudio propuesto está dirigido a reproducir las condiciones de un entorno en producción con múltiples usuarios interactuando con SAGS, compuesto en nuestro caso por 30 simulaciones. Cada una de ellas representa el análisis dinámico de una estructura de hormigón armado compuesta por 68.800 nudos y 137.390 barras.

En las simulaciones han sido aplicados diferentes terremotos representativos acordes a la situación geográfica del edificio. Los acelerogramas empleados tienen una duración de entre 5 y 10 segundos, incluyendo la aceleración experimentada en el terreno cada 0.01 segundos. Debido a la variabilidad en la duración de los acelerogramas y tratando de definir un caso de estudio lo más homogéneo posible, las simulaciones han sido fijadas con una duración de 5 segundos usando pasos de tiempo de 0.01 segundos. El método Newmark [66] ha sido el elegido como método de integración directa, y el Simulador Estructural Paralelo ha sido configurado para emplear la librería WSMP [67]. Los datos de salida contienen información acerca de esfuerzos y deformadas en múltiples puntos intermedios de todos los elementos estructurales que componen el edificio. Estos datos fueron configurados para ser salvados cada 0.5 segundos. Como resultado, los datos de salida ascendieron a 646MBytes de espacio por simulación, conformando un total de 19GBytes de espacio en disco en el caso de estudio completo.

La ejecución del caso de estudio fue llevada a cabo en una infraestructura Grid compuesta por recursos computacionales pertenecientes al GRyCAP, ya que se trataba de recursos con una mayor disponibilidad para nosotros. Esta infraestructura incluye 2 clusters de PCs, cuyas principales características son descritas en la Tabla 5.1. Ambos clusters están interconectados con la máquina que alberga SAGS por medio de una red local con un ancho de banda de 100 Mbits/sec. La versión 2.4 de

Globus Toolkit fue previamente instalada en los clusters que integran esta infraestructura, además de la versión 4 en la máquina que alberga el servicio.

Ordenador	Procesadores	Memoria	Tareas Asignadas
Seker	4 Intel Pentium Xeon@2.8Ghz	4Gb	Servidor SAGS
Kefren	20 dual Intel Pentium Xeon@2.0 Ghz	1 Gb	16
Odin	55 dual Intel Pentium Xeon@2.0 Ghz	2Gb	14

Tabla 5.1. Detalle de las máquinas que componen la infraestructura Grid.

Siguiendo la política de selección del número de procesadores, dependiendo de las características de cada simulación, SAGS estimó en dos los procesadores involucrados en cada ejecución paralela. Esta decisión permitió una compartición eficiente de los recursos computacionales disponibles, ya que pudieron ejecutarse simultáneamente múltiples simulaciones.

5.4.1. Análisis de las prestaciones

La Tabla 5.1 muestra que un número similar de simulaciones fueron ubicadas en cada recurso Grid. De hecho, el componente de GMarte encargado de la selección de recursos implementa una política que distribuye la carga entre los diferentes recursos del Grid, tratando de minimizar el impacto de la caída de algún recurso determinado. Obviamente, la selección de recursos es una tarea clave en el procedimiento de asignación de tareas. Quizás, una mejora en esta fase que resultara en una mayor asignación de tareas a Odín habría supuesto una mejora en las prestaciones.

La ejecución del caso de estudio estructural en la infraestructura propuesta requirió un total de 38 minutos, incluyendo los tiempos consumidos en la descarga de resultados desde los recursos Grid a la máquina que alberga SAGS. Por un lado, ejecutando todas las simulaciones empleando una plataforma secuencial, simulando una después de otra, y empleando 1 PC de Odín, el cluster más rápido, requirió 566 minutos, lo que se traduce en unas 10 horas. Por otro lado, empleando una aproximación basada en la Computación de Altas Prestaciones, asumiendo un cluster típico de 8 CPUs y llevando a cabo ejecuciones paralelas con 2 procesadores en Odin (4 simulaciones concurrentes), se necesitaron un total de 105 minutos.

Por lo tanto, la aproximación empleando SAGS obtuvo un speedup de 14.89 con respecto a la ejecución secuencial y de 2.76 comparado con la aproximación HPC. Obviamente, la mejora de la velocidad de ejecución depende directamente de la cantidad de recursos computacionales Grid disponibles en la infraestructura. De todos modos, es importante señalar que la aproximación empleada mediante SAGS introduce una sobrecarga no despreciable tanto a nivel de la metaplanificación (selección de recursos) como a nivel de toda la transferencia de datos de entrada y de salida.

Capítulo 6

Conclusiones y Trabajos Futuros

La presente Tesis de Master expone la utilización de las tecnologías Grid junto con las Arquitecturas Orientadas a Servicios para dar cobertura a un problema complejo como es el análisis dinámico de estructuras de edificación. Para ello, se describe el análisis, diseño e implementación de un Servicio Grid de Análisis de Estructuras (SAGS) basado en GT4. Esta herramienta Grid permitirá a ingenieros y arquitectos llevar a cabo un análisis 3D realista tanto dinámico como estático de estructuras de gran dimensión a través de Internet, sin necesidad de ningún software ni hardware específico. SAGS hará uso de un simulador basado en Computación de Altas Prestaciones, el cual podrá ejecutarse en uno o varios procesadores, obteniendo unas notables prestaciones en cuanto a tiempos de cómputo y precisión en los resultados.

Se ha empleado GMarte 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. De entre todos los planificadores estudiados, es GMarte el que mejor se adapta a los requerimientos de nuestra aplicación.

Como se puede observar, tanto el diseño como la implementación han estado dirigidos a satisfacer los requerimientos de alta productividad, robustez y fiabilidad de un sistema multiusuario accesible a través de la red. De este modo, será posible, de manera muy sencilla, explotar el sistema de un entorno en producción, tanto en un ámbito académico como comercial.

Uno de los principales inconvenientes a los que nos enfrentábamos al incorporar una Arquitectura Orientada a Servicios es la inherente sobrecarga en las comunicaciones introducida por las tecnologías de los Servicios Web. Este escollo ha sido solventado ofreciendo a los usuarios dos alternativas. La primera de ellas es

totalmente compatible con los estándares SOA y está basado en la utilización de esquemas de codificación que han permitido la inclusión de datos binarios en los mensajes XML que deben intercambiar SAGS y los clientes. Por otro lado, se ofrece una alternativa altamente eficiente basado en el estándar de facto GridFTP, empleado para el transporte masivo de datos.

En la línea de ofrecer un servicio altamente eficiente a la comunidad de ingenieros estructurales, se ha realizado un gran esfuerzo en reducir los periodos de espera involucrados en el envío de datos, planificación y ejecución de la tarea y recogida de resultados. De este modo, se han empleado en SAGS mecanismos multihilo en todas las fases de ejecución de la simulación de la estructura, permitiendo que las tareas sean gestionadas concurrentemente. Además, se ha desarrollado un sistema avanzado de recogida de resultados que permite solapar el tiempo de ejecución de las simulaciones dinámicas con la transferencia de sus resultados.

La alta fiabilidad del sistema desarrollado está sustentada en la inclusión de un esquema de tolerancia a fallos multi-nivel, que garantiza que toda petición de análisis que llega al servicio sea 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 o incluso el cliente Gráfico.

Es importante remarcar que ha sido desplegada una robusta política de seguridad en SAGS, imprescindible en un sistema accesible a través de Internet. Esta política contempla varios aspectos como la autenticación y la autorización de usuarios y la integridad y privacidad de los datos que viajan por la red. Además, SAGS define una serie de niveles de usuarios, permitiendo desplegar diferentes esquemas de utilización del sistema en función de los privilegios de los mismos. De este modo, se puede ofrecer una mejor calidad de servicio a un grupo de usuarios dependiendo de su posición o necesidades.

Adicionalmente, abriendo una nueva línea de trabajo, se han incluido en el sistema mecanismos de publicación de información que permitirán de manera remota conocer el estado de SAGS. Mediante estos mecanismos de publicación SAGS podrá publicar información de interés administrativo y de planificación. A nivel administrativo, se podrá publicar el uso que cada usuario hará del sistema, en cuanto a tiempo de cómputo, almacenamiento, etc. Por otro lado, SAGS podrá publicar su carga, número de usuarios activos, número de nodos libres, etc., de cara a poder integrarse en un sistema con múltiples servicios disponibles, donde el cliente o un servicio de nivel superior seleccionarán en cada caso el servicio más apropiado.

El comportamiento de SAGS ha sido evaluado definiendo un caso de estudio que emula múltiples clientes tratando de analizar al mismo tiempo diferentes estructuras. A partir de este caso de estudio, se ha puesto de manifiesto la mejora aportada por la aproximación Grid con respecto a otras alternativas. De este modo, se han comparado los tiempos de simulación de la aproximación ofrecida por SAGS, con respecto a una solución secuencial y a otra basada en una aplicación tradicional de la Computación de Altas Prestaciones.

Por lo tanto, las tecnologías Grid se han mostrado como un marco de trabajo perfecto para el desarrollo y despliegue de sistemas orientados a servicios, ofreciendo suficiente capacidad de cómputo y almacenamiento, a la vez de una alta productividad, fiabilidad y robustez, para el análisis 3D realista de estructuras de gran dimensión.

6.1. Trabajos futuros

Actualmente, la infraestructura ofrecida para el análisis de estructuras de edificación se centra en un despliegue individual de SAGS, como único punto de entrada al Grid. Sin embargo, la aparición de Grids de nueva generación, donde será habitual la existencia de un entorno Grid global disponible, permitirá la evolución del sistema hacia una infraestructura SAGS ubicua. Esta infraestructura estará compuesta por múltiples SAGS independientes, permitiendo desarrollar un sistema altamente disponible y fiable, siendo accesible desde cualquier punto del planeta.

En una infraestructura SAGS global, los datos relativos a las simulaciones deberán residir en servidores especiales distribuidos, llamados SRS (Structural Repository Service), Figura 6.1., siendo directamente descargados y guardados por medio de servicios específicos. El cambio en la ubicación del Repositorio Estructural atiende a la necesidad de tener los datos de las simulaciones disponibles, incluso ante caídas de SAGS. De este modo, incluyendo esquemas de réplicas de datos, estos estarán disponibles incluso ante caídas de los propios servidores SRS.

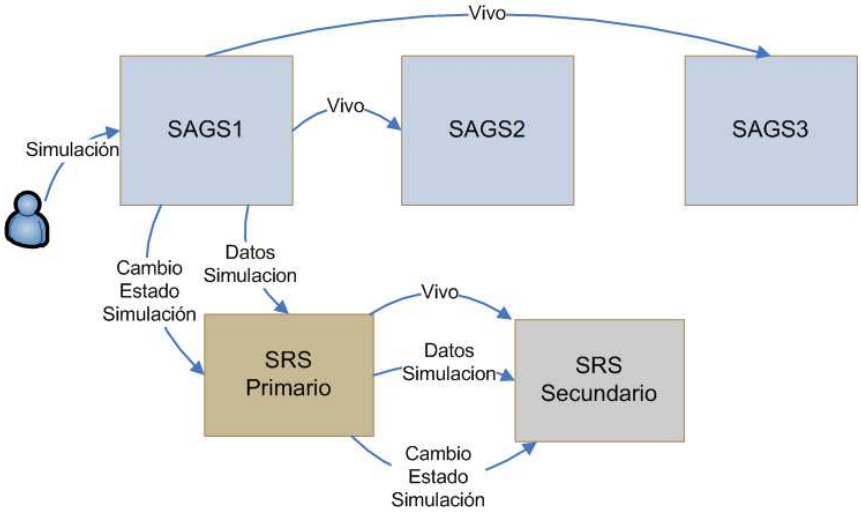

Figura 6.1. Infraestructura SAGS Global.

La inclusión de los componentes SRS implicará el desarrollo o la inclusión de sistemas basados en catálogos, de modo que cada fichero será registrado por medio de un nombre lógico y podrá tener asociadas múltiples réplicas físicas. Este esquema es el empleado en la infraestructura LCG y sería interesante analizar su posible inclusión en la infraestructura SAGS.

En cuanto a la distribución de las tareas entre los diferentes SAGS disponibles, serán dos las alternativas a considerar, dependiendo de quien lleve a cabo el proceso de planificación, el cliente mismo o un componente de alto nivel que también debería ser incluido en la infraestructura. La primera aproximación delega la tarea de la planificación en el cliente, siendo una alternativa poco escalable y flexible, ya que los clientes deberán conocer en todo momento los servicios SAGS disponibles, pudiendo degenerar en un sistema completamente estático y cerrado donde los recursos SAGS disponibles siempre serán los mismos y los nuevos no serán tenidos en cuenta.

La inclusión de agentes de alto nivel de planificación, SAGSS (Structural Análisis Grid Service Scheduler), será una pieza clave en el despliegue global de una infraestructura para el análisis de estructuras de edificación. Estos componentes serán el punto de entrada al sistema, siendo capaces de monitorizar y descubrir los servicios

SAGS y SRS disponibles en la red, así como de llevar a cabo la distribución de las simulaciones.

A partir del desarrollo de esta infraestructura global para el análisis de estructuras, se plantea el desarrollo de una plataforma de propósito general. Para ello, continuando el trabajo dentro del proyecto GMarte, el cual ya define un Servicio Grid de Metaplanificación GMarteGS [68], y a partir de la experiencia obtenida en el desarrollo y despliegue una infraestructura global SAGS, sería viable el desarrollo de una infraestructura ubicua de metaplanificación genérica. Esta plataforma, permitiría a usuarios de cualquier tipo de aplicación, cuyos requerimientos computacionales no son cubiertos por su organización, ejecutar sus aplicaciones remotamente de manera muy sencilla y eficiente accediendo a un Servicio Web de alto nivel o mediante clientes gráficos muy sencillos, Figura 6.2.

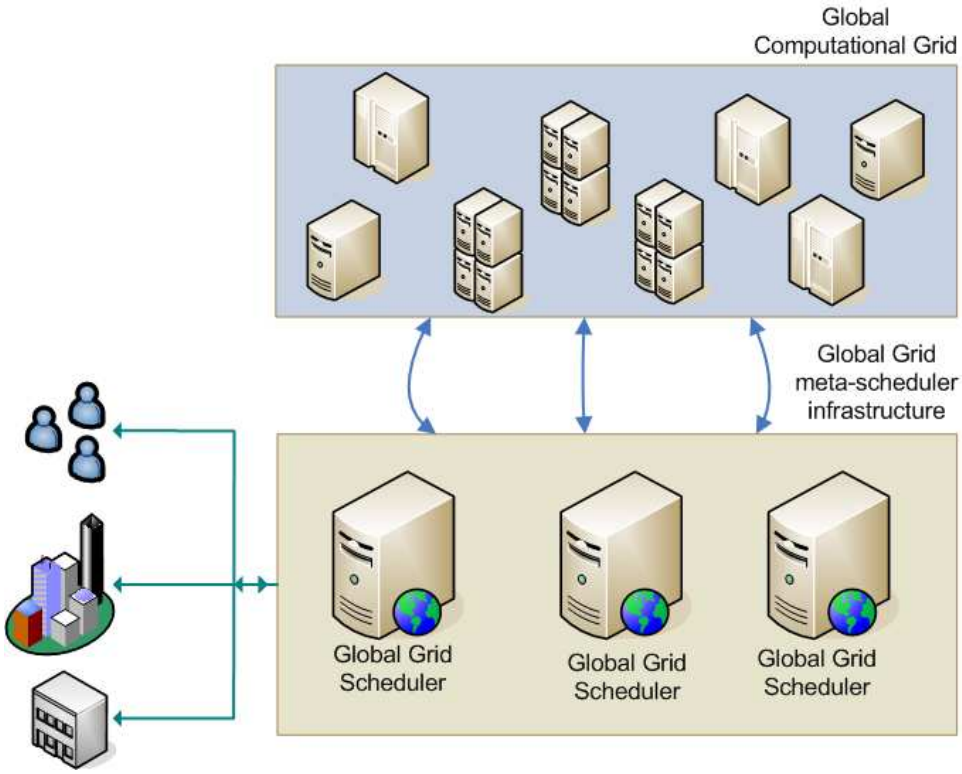

Figura 6.2. Esquema de infraestructura global de metaplanificación.

Publicaciones Realizadas

La presente Tesis de Máster ha dado lugar a las siguientes publicaciones de investigación:

- J. M. Alonso, V. Hernández, R. López, y G.Moltó. A Grid Service Development for 3D Structural Analysis. CST2006: The Eighth International Conference on Computational Structures Technology, Paper 122, 2006.
- J. M. Alonso, V. Hernández, R. López, y G.Moltó. Experiences using Grid Services in Structural Dynamics. Proceedings of ECCPM 2006: e-Business and e-Work in Architecture, Engineering and Construction, pp. 359–366, 2006.
- J. M. Alonso, V. Hernández, R. López, y G.Moltó. Una Aproximación Orientada a Servicios Grid para el Análisis Estático y Dinámico de Estructuras de Edificación. Proceedings of the XXIII Conferencia Latinoamericana de Informática (CLEI 2006), 2006.
- J. M. Alonso, V. Hernández, R. López, y G.Moltó. A Service Oriented System for on Demand Dynamic Structural Analysis over Computational Grids. VECPAR'06: Seventh International Meeting on High Performance Computing for Computational Science Lecture Notes in Computer Science, 4395:13–26, 2007.
- J. M. Alonso, V. Hernández, R. López, y G. Molto. Grid4Build: A High Performance Grid Framework for the 3D Analysis and Visualisation. IBERGRID: 1st Iberian Grid Infrastructure. Conference Proceedings, pp. 353–364, 2007.

Bibliografía

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

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

[3] Alonso J.M., Hernández V., "*A parallel implementation of 3D modal analysis of building structures*". Proceedings of the Tenth International Conference on Civil, Structural and Environmental Engineering Computing: Paper 232, 2005.

[4] Alonso J.M., Hernández V., "*Three-dimensional structural dynamic analysis using parallel direct time integration methods*". Proceedings of the Fifth International Conference on Engineering Computational Technology, B.H.V. Topping, G.Montero and R. Montenegro (Editors), Civil-Comp Press, Stirlingshire, United Kingdom, paper 120, 2006.

[5] Kardestuncer H., "*Introducción al análisis estructural con matrices*". University of Connecuit U.S.A..McGRAW-HILL DE MEXICO, SA. , 1975.

[6] Martínez J., "*Métodos matriciales para cálculo de estructuras*". H. Blume Ediciones. 1970.

[7] Androutsellis-Theotokis S., Spinellis D., "*A Survey of Peer-to-Peer Content Distribution Technologies*". ACM Computing Surveys, Vol. 36, No. 4, December 2004.

[8] RFC 1057 - RPC: Remote Procedure Call Protocol specification: Version 2

[9] Microsoft Corporation, "*OLE Automation Programmer's Reference, Programmer's Reference Library*". Microsoft Press 1996. ISBN 1-55615-851-3.

[10] Box D., "*Essential COM, Addison-Wesley Object Technology Series*". ISBN 0-201-63446-5, 1998.

[11] Object Management Group, "*CORBA 3.0 – OMG IDL Syntax and Semantics chapter*", June 2002. http://www.omg.org/cgi-bin/apps/doc?formal/02-06-39.pdf.

[12] Chappell D., "*Understanding ActiveX and OLE*". Microsoft Press, 1996. ISBN: 1-57231-216-5

[13] Horstmann M., Kirtland M., "*DCOM architecture*". http://msdn2.microsoft.com/en-us/library/ms809311.aspx, 1997.

[14] Downing T.B., "*Java RMI: Remote Method Invocation*". IDG Books Worldwide, Inc., Foster City, CA, 1998.

[15] Krishnaswamy V., Walter D., Bhola S., Bommaiah E., Riley G., Topol B., Ahamad.M., "*Efficient implementations of Java Remote Method Invocation (RMI)*". Proc. of the 4th USENIX Conference on Object- Oriented Technologies and Systems (C00TS'98), 1998.

[16] Object Management Group, "*Common Object Request Broker Architecture: Core Specification*". Marzo 2004. http://www.omg.org/docs/formal/04-03-12.pdf.

[17] Soley R., "*Object Management Architecture Guide, revision 3.0*". Object Management Group, 1997. http://www.omg.org/docs/ab/97-05-05.pdf.

[18] MacKenzie CM, Laskey K, McCabe F, Brown P, Metz R., "*OASIS, referente model for service oriented architecture 1.0*". Public Review Draft 1.0, OASIS Open, 2006.

[19] Booth D, Haas H, McCabe F, Newcomer E, Champion M, Ferris C, Orchard D., "*Web services architecture*". W3C, Working Draft, 2003.

[20] UDDI Spec TC, "*UDDI Version 3.0.2, UDDI Spec Technical Committee Draft*" 2004. http://uddi.org/pubs/uddi_v3.htm.

[21] Christensen E., Curbera F., Meredith G., Weerawarana, S., "*Web Services Description Language (WSDL) 1.1.*" W3C Note 2001.

[22] Gudgin M., Hadley M., Mendelsohn N., Moreau J.J., Frystyk Nielsen H.,"*SOAP version 1.2 part1: Messaging framework*". W3C Recommendation 2003.

[23] Foster I., Kesselman C., "*The GRID 2: blueprint for a new computing Infrastructure*". 2nd Edition. Morgan-Kaufmann, 2004.

[24] "LCG: LHC Computing Grid Project", http://lcg.web.cern.ch/LCG/.

[25] Gagliardi F., "*The EGEE european Grid infrastructure Project*". Springer-Verlag LNCS 2402/2005, 2005.

[26] Dolenc M., Turk Z., Katranuschkov P., Kurowski K., Hannus M., "*Towards a Grid enabled engineering collaboration environment*". Proceedings of the Tenth International Conference on Civil, Structural and Environmental Engineering Computing (CC 2005), 2005, paper 69.

[27] Spencer B., Finholt T.A., Foster I., Kesselman C., Beldica C., Futrelle J., Gullapalli S., Hubbard P., Liming L., Marcusiu D., Pearlman L., Severance C., Yang G., "*NEESgrid: A distributed collaboratory for advanced earthquake engineering experiment and simulation*". Proceedings of the 13th World Conference on Earthquake Engineering, Canada, 2004, paper 1764.

[28] Pearlman L., Kesselman C., Gullapalli S., Spencer B.F. Jr., Futrelle J., Ricker K., Foster I., Hubbard P., Severance C., "*Distributed hybrid earthquake engineering experiments: Experiences with a ground-shaking Grid application*". Proceedings of the 13th IEEE Symposium on High Performance Distributed Computing (HPDC-12), 2004.

[29] McKenna F., "*Object-oriented _nite element programming: frameworks for analysis, algorithm and parallel computing*". Ph.D. Thesis, University of California, Berkeley, CA, 1997.

[30] Peng J., Law K.H., "*Software framework for collaborative development of nonlinear dynamic analysis program*". PEER Report 2003/02, Pacific Earthquake Engineering Research Center, University of California, Berkeley, 2003.

[31] 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.

[32] Foster I., Kesselman C., "*Globus: a metacomputing infrastructure toolkit*". International Journal of Supercomputer Applications 1997; 11:115-128.

[33] Foster I., "*Globus Toolkit Version 4. Software for service-oriented systems*". Journal of Computer Science and Technology, 2006; 21(4):513-520.

[34] Snelling D., Robinson I., Banks T., "*Oasis web services resource framework (WSRF) TC*". April 2006.

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.

[35] Feller M., Foster I., Martin S., "*GT4 GRAM: A Functionality and Performance Study*". TeraGrid 2007 Conference, Madison, USA, 2007.

[36] Allcock B., Bester J., Bresnahan J., Chervenak A.L., Foster I., Kesselman C., Meder S., Nefedova V., Quesnel D., Tuecke S., "*Data Management and Transfer in*

High-Performance Computational Grid Environments". Parallel Computing Journal (2002) 28(5):749-771.

[37] Allcock W.E., Foster I., Madduri R., "*Reliable Data Transport: A Critical Service for the Grid"*. Building Service Based Grids Workshop, Global Grid Forum 11, June 2004.

[38] Czajkowski K., Fitzgerald S., Foster I., Kesselman C., "Grid information services for distributed resource sharing". Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10) (I. Press, ed.), 2001.

[39] Welch V., Siebenlist F., Foster I., Bresnahan J., Cajkowski K., Gawor J., Kesselman C., Meder S., Pearlman L., Tuecke S., "Security for grid services". Twelfth IEEE Internacional Symposium on High-Performance Distributed Computing (HPDC-12), 2003.

[40] Coghlan B., Cooke A.W., Datta A., Djaoui A., Field L., Fisher S., Magowan J., Nutt W., Oevers M., Soni M., Podhorszki N., Ryan J., Wilson A.J., Zhu X., "*R-GMA: A Grid Information and Monitoring System*". WP3. UK e-Science all hands conference, Sheffield, 2-4 September 2002.

[41] Alonso J.M., Hernández V., Moltó G., *"GMarte: Grid Middleware to Abstract Remote Task Execution"*. Concurrency and Computation: Practice and Experience, 18(15):2021-2036, 2006.

[42] Frey J., Tannenbaum T., Foster I., Livny M., Tuecke S., *"Condor-G: A Computation Management Agent for Multi-Institutional Grids"*. Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10), 55-63, 2001.

[43] Montero R.S., Huedo E., Llorente I.M., "*The GridWay approach for job submission and management on Grids*". In: iAstro Workshop on Distributed Processing, Transfer, Retrieval, Fusion and Display of Images and Signals: High Resolution and Low Resolution in Data and Information Grids, Spain, 2003.

[44] Rajic H. et al, "*Distributed resource management application API specification 1.0*". Technical Report, The Global Grid Forum, DRMAA Working Group, 2003.

[45] Venugopal S., Buyya R., Winton L., *"A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids"*. Proceedings of the 2nd International Workshop on Middleware for Grid Computing, ACM Press, 2004.

[46] Augenbroe G., "*Building simulation trends going into the new millenium*". Proceedings of the Seventh International IBPSA Conference, Brazil, 2001.

[47] Peng J., Law K.H., "*Software framework for collaborative development of nonlinear dynamic analysis program*". PEER Report 2003/02, Pacific Earthquake Engineering Research Center, University of California, Berkeley, 2003.

[48] Peng J., Law K.H., "*A prototype software framework for internet-enabled collaborative development of a structural analysis program*". Engineering with Computers 2002; 18:38-49.

[49] Peng J., Law K.H., "*Building fnite element analysis programs in distributed services environment*". Computers and Structures 2004; 82:1813-1833.

[50] Dolenc M., "*Web services for finite element analysis*". Proceedings of the Fourth International Conference on Engineering Computational Technology (ECT 2004), 2004, paper 524.

[51] Raghunath M.S., Ramakrishnan C.V., Ahlawat A., Shoma Shekar B.P.B., "*Networked client-server environment with CORBA interface for parallel FE analyis*". Advances in Engineering Software 2004; 35:757-769.

[52] Chen H.M., Lin Y.C., "*Internet based analytical services using a Java based GUI*". Proceedings of the Tenth International Conference on Civil, Structural and Environmental Engineering Computing (CC 2005), 2005, paper 14.

[53] Chen H.C., Fen C.S., "*A web-based distributed problem-solving environment for engineering applications*". Advances in Engineering Software 2006; 37:112-128.

[54] Marante M.E., Suárez L., Quero A., Redondo J., Vera B., Uzcategui M., Delgado S., León L.R., Núñez L., Flórez-López J., "*Portal of damage: a web-based finite element program for the analysis of framed structures subjected to overloads*." Advances in Engineering Software 2005; 36:346-358.

[55] Alonso J.M., de Alfonso C., García G., Hernández V., "*Integrating HPC and Grid computing for 3D structural analysis of large buildings*". Advances in Engineering Software, 38(11-12):738-749, 2007

[56] Alonso J.M., Hernández V., Moltó G., "*A Grid-based application of the threedimensional dynamic analysis of high-rise buildings*". Advances in Engineering Software. Aceptado y pendiente de publicación en 2008. Doi:10.1016/j.advengsoft.2007.05.005

[57] Dolenc M., Zevnik J., Kurowski K., "*Parametric studies on the InteliGrid platform*". Proceedings of the Fifth International Conference on Engineering Computational Technology (ECT 2006), 2006, paper 124.

[58] Turk Z., Dolenc M., Nabrzyski J., Katranuschkov P., Balaton E., Balder R., Hannus M., "*Towards Engineering on the Grid*". Dikbas A. and Scherer R. (eds.) Proceedings. ECPPM Conference, Estambul, 2004.

[59] Wei G.Y., Zheng Y., Zhang J.F., "*Grid Service-based parallel finite element analysis*". Grid and Cooperative Computing, Springer-Verlag LNCS 3032/2004, 2004, p. 123-130.

[60] Goodyer C.E,. Berzins M., Jimack P.K., Scales L.E., "*A Grid-enabled problem solving environment for parallel computational engineering design*". Advances in Engineering Software 2006; 37:439-449.

[61] Petrinja E., Turk Z., "*Grid service oriented provenance support for product modelling*". Martínez & Schere (eds), eWork and eBusiness in Architecture, Engineering and Construction, 367-373, 2006.

[62] Aziz Z., Anumba J., Bouchlaghem N.M., Ruikar D., Carrillo P.M., "*Towards a semantic Grid computing platform for disaster management in built environment*." Proceedings of the Xth International Conference on Computing in Civil and Building Engineering (ICCCB3-X), 2004.

[63] Li Z., Jin X., Cao Y., Zhang X., Li Y., "*Architecture of collaborative design grid and its application based on LAN*". Advances in Engineering Software 2007; 38:121-132.

[64] Von Laszewski G., Foster I., Gawor J., Lane P., "*A Java Commodity Grid Kit*".Concurrency and Computation-Practice & Experience 2001; 13(8-9):645-662.

[65] Sowizral H., Rushforth K., Deerin M., "*The Java 3D API specification*". Adison-Wesley. Boston, 2000.

[66] Wilson E.L., "*A Computer Program for the Dynamic Stress Analysis of Underground Structures*". Technical Report SESM Report 68-1, Division of Structural Engineering and Structural Mechanics, University of California, Berkeley (1968).

[67] Gupta A., "*WSMP: Watson Sparse Matrix Package Part I - Direct Solution of Symmetric Sparse Systems*". Technical Report Technical Report IBM Research Report RC 21886(98462), IBM (2000).

[68] Moltó G., Hernández V., Alonso J.M., "*A Service-Oriented WSRF-based Architecture for Metascheduling on Computational Grids* Future Generation Computer Systems, 24(4):317-328, 2008.