Proyecto Fin de Máster en Sistemas Inteligentes

Adaptación al entorno Grid de las simulaciones Montecarlo para el telescopio MAGIC

Adolfo Vázquez Rodríguez

Facultad de Informática
Universidad Complutense de Madrid

Septiembre 2008

Agradecer la guía y consejo de mis tutores, el Prof. Juan Pavón Mestras y el Prof. José Luis Contreras por parte de la Universidad Complutense de Madrid, y Aitor Ibarra Ibaibarriaga e Ignacio De la Calle por parte de Ingeniería y Servicios Aeroespaciales S.A (INSA), sin los cuales este proyecto no habría sido posible. También agradecer la ayuda que para distintos aspectos de este proyecto me han prestado desde la sección de INSA en el European Space Astronomy Center (ESA/ESAC), el proyecto EGEE y la colaboración MAGIC. Vaya un agradecimiento especial a Daniel Tapiador de Pedro y a los miembros del Grupo de Altas Energías de la Universidad Complutense de Madrid en cuyo seno se realizó este trabajo. Por último agradecer a INSA y a la colaboración MAGIC la oportunidad de participar en este proyecto.

Aprovecho también para dedicar este trabajo a mi familia, sin cuyo apoyo mantenido a lo largo de los años no habría sido posible. Paula, José y Laura, ¡va por vosotros!

Índice

Capítulo 1. Introducción

Este trabajo, de nombre MAGICGrid, se ha llevado a cabo dentro del proyecto MAGIC. Su principal objetivo es la adaptación al entorno Grid de las simulaciones Montecarlo para el telescopio MAGIC, así como la simplificación, de cara al usuario de estas, del proceso de generación de las mismas.

Computación en Grid

¿Qué es el Grid?

El Grid surge como la evolución de los sistemas distribuidos que sustituyeron al modelo de computación centralizado a mediados de los 90. Estos sistemas aprovechaban el bajo coste de los componentes y la creciente velocidad de las redes de comunicaciones para ofrecer una alternativa válida para ciertos tipos de problemas. Frente a estos primeros sistemas distribuidos, el Grid surge como un tipo particular de sistema distribuido, cuya principal característica consiste en permitir aunar recursos de distintos dominios de administración.

Según Ian Foster en su artículo *What is the Grid? A three point checklist* [1], el Grid es un sistema de computación distribuida que no dispone de un control centralizado, respetando así las políticas de seguridad internas de sus usuarios, así como las herramientas de gestión de éstos, que utiliza una serie de protocolos e interfaces para gestionar aspectos como la autentificación, la autorización y la gestión de recursos, y que permite ofrecer calidad de servicios con respecto al tiempo de respuesta, la disponibilidad o la seguridad.

La ventaja del Grid en cuanto a coste y flexibilidad es clara, pero a su vez esta flexibilidad juega en su contra en especial en lo referente al rendimiento en la ejecución de trabajos, con nodos que pueden o bien tener un rendimiento menor al deseado, convirtiéndose en un cuello de botella, o incluso dejar un trabajo sin terminar. Siendo este un escenario relativamente común en los Grids actualmente en producción, muchas de las iniciativas puestas en marcha en este área, apuestan por ofrecer herramientas que, en una u otra media, den solución a este problema.

Simulaciones Monte Carlo en el Grid

El nombre de Monte Carlo fue tomado por un grupo de científicos en los años cuarenta para designar a un tipo de métodos numéricos basados en el uso de números aleatorios. Actualmente, los métodos de Monte Carlo son ampliamente usados como una clase de algoritmos computacionales para simular el comportamiento de sistemas físicos y matemáticos complejos. Se distinguen de otros métodos de simulación por su índole estocástica, usando en la práctica números pseudo-aleatorios.

Para muchos tipos de problemas, la eficiencia relativa de este tipo de métodos con

respecto a otros métodos numéricos aumenta al aumentar la dimensión del problema. La gran expansión de estos métodos en las últimas cinco décadas se ha podido producir gracias a los avances en la capacidad de cálculo.

Los problemas relacionados con la interacción de partículas con la materia, se prestan de manera intrínseca a la simulación Monte Carlo, ya que los procesos que las gobiernan son en sí mismos aleatorios. En nuestro caso el proceso a simular será la llegada de una partícula de alta energía a la alta atmósfera. El resultado será la señal registrada por un telescopio para cada partícula incidente. Estudiar estas señales nos permitirá estimar la respuesta media del telescopio a flujos de partículas y descubrir diferencias entre las señales pertenecientes a distintos tipos de partículas.

Como la llegada e interacción de cada partícula es totalmente independiente de las de las demás, las simulaciones Montecarlo, que serán la base de este proyecto, son por definición fácilmente paralelizables. Al contrario que en otras aplicaciones distribuidas, la comunicación entre distintas subtareas es nula, lo que hace de este tipo de simulaciones uno de los mejores candidatos a utilizar la infraestructura Grid. La tarea original es fácilmente divisible en subtareas, cada una de las cuales se puede asignar a un nodo del Grid. Esta asignación se puede hacer en base a las características y la carga de trabajo de los distintos nodos. Otro aspecto importante a tener en cuenta es la capacidad de almacenamiento local de los nodos para que se puedan ir almacenando los datos parciales que se vayan generando. En un estadio superior, se pueden unificar los datos provenientes de las distintas subtareas pertenecientes al trabajo original.

Por su naturaleza, las simulaciones Montecarlo se adaptan bien al entorno dinámico del Grid, en donde una tarea puede fallar o un nodo puede caerse. Esto implica que no todas las subtareas se podrán recuperar y que será importante y necesario establecer un número suficiente para garantizar la validez de la simulación. Esto permite evitar cuellos de botella no deseados debido a que es sólo necesario que un número suficiente de simulaciones lleguen a su fin. Este tipo de computación encaja en lo que se denomina entorno HTC (High Throughput Computing), en donde el objetivo pasa a ser aumentar el número de ejecuciones por unidad de tiempo, desapareciendo los problemas derivados del paralelismo entre simulaciones.

Arquitectura Grid

El uso de una infraestructura Grid, incluso a nivel de usuario, requiere conocer cuáles son los servicios del Grid y qué información puede facilitar cada uno de ellos, a fin de proporcionar robustez a la aplicación. Existen varias aproximaciones al concepto de Grid, desde Globus Toolkit, considerado como el estándar de facto, hasta Glite, un middleware o software que ofrece un conjunto de servicios homogéneo sobre una plataforma heterogénea como es el Grid, en este caso EGEE. A su vez, estas aproximaciones han ido sufriendo cambios dentro de su diseño adoptando finalmente una arquitectura SOA (Open Grid Services Architecture), como fruto de la colaboración entre las comunidades de Grid y de servicios web. Foster, Kesselman y Tuecke en [2] ofrecen una visión en capas de la arquitectura Grid que recoje aquellas características que han ido consolidándose en cada uno de los niveles de dicha arquitectura. Como se puede ver en la figura 0, distinguen

cuatro capas, de mayor a menor nivel: *application, collective, resource, connectivity* y *fabric*.

Estas capas contienen varias de las funcionalidades que caracterizan a un Grid, como se detalla a continuación:

- Fabric: gestión remota de recursos.
- Connectivity: single sign-on (punto único de acceso al sistema para el usuario) y delegación por parte del usuario de su identidad de forma que pueda acceder a recursos remotos.
- Resource: servicio de información sobre el estado y características de los recursos, de forma que pueda elegirse aquellos que se adapten a las necesidades concretas del usuario que ofrece la capa.
- Collective: sistemas de directorio, monitorización y autorización.
- Application: aplicaciones disponibles de cara al usuario final, como los metaplanificadores.

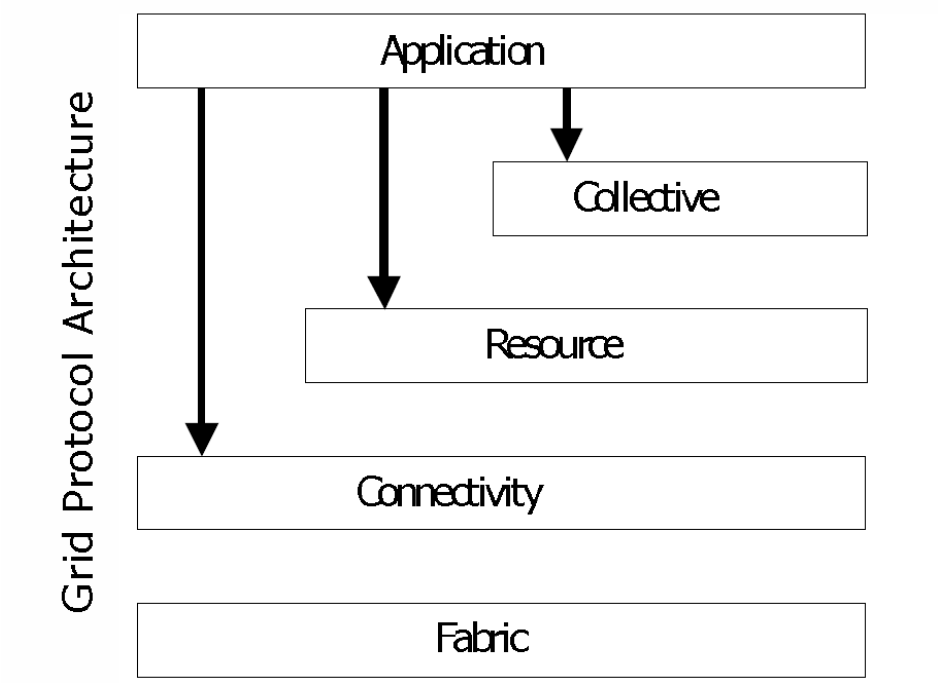

Figura 0. Arquitectura Grid propuesta por Foster, Kesselman y Tuecke.

Aparte de esta arquitectura por capas, otro de los conceptos claves para entender el uso del Grid en proyectos como el que se aborda, es el de las *organizaciones virtuales*. Estas organizaciones trasladan al mundo virtual las colaboraciones existentes en el mundo real, de forma que instituciones de distintos países colaborando en un mismo proyecto puedan compartir recursos Grid para ese fin. La organización virtual no sólo se limita a agrupar los recursos, sino que también identifica a aquellos usuarios y máquinas que están

autorizados a utilizar los recursos asociados a éstas, otorgándoles una serie de roles que indicarán su nivel de acceso a la hora de utilizar los recursos de la organización virtual, desde lanzar trabajos a instalar software dentro de la misma. En los Grid actuales las organizaciones virtuales son relativamente estáticas, aunque la evolución que se plantea para estas organizaciones es que en un futuro sean dinámicas, pudiendo formarse y desaparecer en periodos relativamente cortos para dar cobertura a problemas puntuales. Es en este entorno donde se enmarcan los siguientes servicios, pertenecientes al middleware Glite, que como proyecto usuario del Grid vinculado al proyecto EGEE, ha utilizado MAGICGrid en su despliegue.

- Virtual Organization Membership Service (VOMS)

  La necesidad de que cada organización virtual mantenga una base de datos central donde se guarden los usuarios autorizados por sus respectivos administradores a utilizar los recursos de dicha organización, ha hecho que los sistemas básicos de autentificación de la Globus Toolkit (Grid Security Infrastructure) hayan sido ampliados a un sistema de autentificación basado en una base de datos relacional. Como se detalla en [3], VOMS utiliza una pareja cliente-servidor para que, una vez enviado el certificado del usuario por parte del cliente, el servidor devuelva a éste las credenciales de éste dentro de la organización. Otra pareja cliente-servidor existe para que el o los administradores de la organización virtual puedan gestionar los roles, grupos y usuarios de esta.

- User Interface (UI)

  Como su nombre indica, es el servicio que utiliza el usuario para lanzar los trabajos al Grid y recibir sus resultados. En la UI el usuario también almacenará su certificado, con el que podrá generar proxies que serán enviados a través del Grid. Por lo general las UI incluyen además de los comandos glite-* vinculados al middleware Glite, otros heredados de proyectos predecesores, como por ejemplo lcg-infosites, por lo que es frecuente tener varias opciones a la hora de crear proxies y gestionar trabajos.

- Computing Element y Storage Elements (CE y SE)

  Son los recursos del Grid propiamente dichos, referidos a un recurso de computación en el caso del computing element y a un recurso de almacenamiento en el caso del storage element. Los computing elements en realidad son una interfaz a un conjunto de CPUs o Worker Nodes. El usuario puede dirigir los trabajos a un CE en concreto.

- Resource Broker

  Es el servicio encargado de distribuir y gestionar los trabajos en el Grid. También se encarga de localizar el computing element con menor carga de trabajo, distribuyendo así la carga de trabajo de forma uniforme en el Grid. El proyecto LCG aportó a EGEE el resource broker lcg-rb, que después ha sido sustituido por el Workload Management System (WMS).

- Sistemas de información

Por lo general cada computing o storage element dispone de un servicio de información que devuelve al usuario las características de éste, aunque este sistema de información puede estar estructurado de forma jerárquica de forma que la información de un conjunto de CE y SE se guarde en un sistema de información externo a los mismos. Una u otra configuración depende de la política de la institución que administra estos recursos. En el caso del proyecto que se aborda, esta decisión influye a la hora de recabar información acerca de los CE/SE que serán utilizados para ejecutar las simulaciones.

- Metaplanificadores

  Estos se sitúan a alto nivel dentro de la colección de software del Grid, y evitan al desarrollador de aplicaciones tener que lidiar con los problemas habituales de los trabajos (fallo al enviar/ejecutar, el CE al que el trabajo ha sido enviado no responde en un tiempo razonable, ...). Estos metaplanificadores ofrecen además otros servicios, como el envío de colecciones de trabajos, parametrizando alguna de las variables de estos, así como APIs que facilitan su uso. Un ejemplo de estos metaplanificadores es Gridway, que se incluye en la distribución de Globus Toolkit.

- MyProxy

  Es un servicio que permite guardar los certificados del usuario de forma centralizada, de manera que si un servicio del Grid necesita hacer uso de las credenciales, puede consultar directamente al servicio MyProxy, en vez de tener que acceder a la user interface donde estas se encuentren almacenadas.

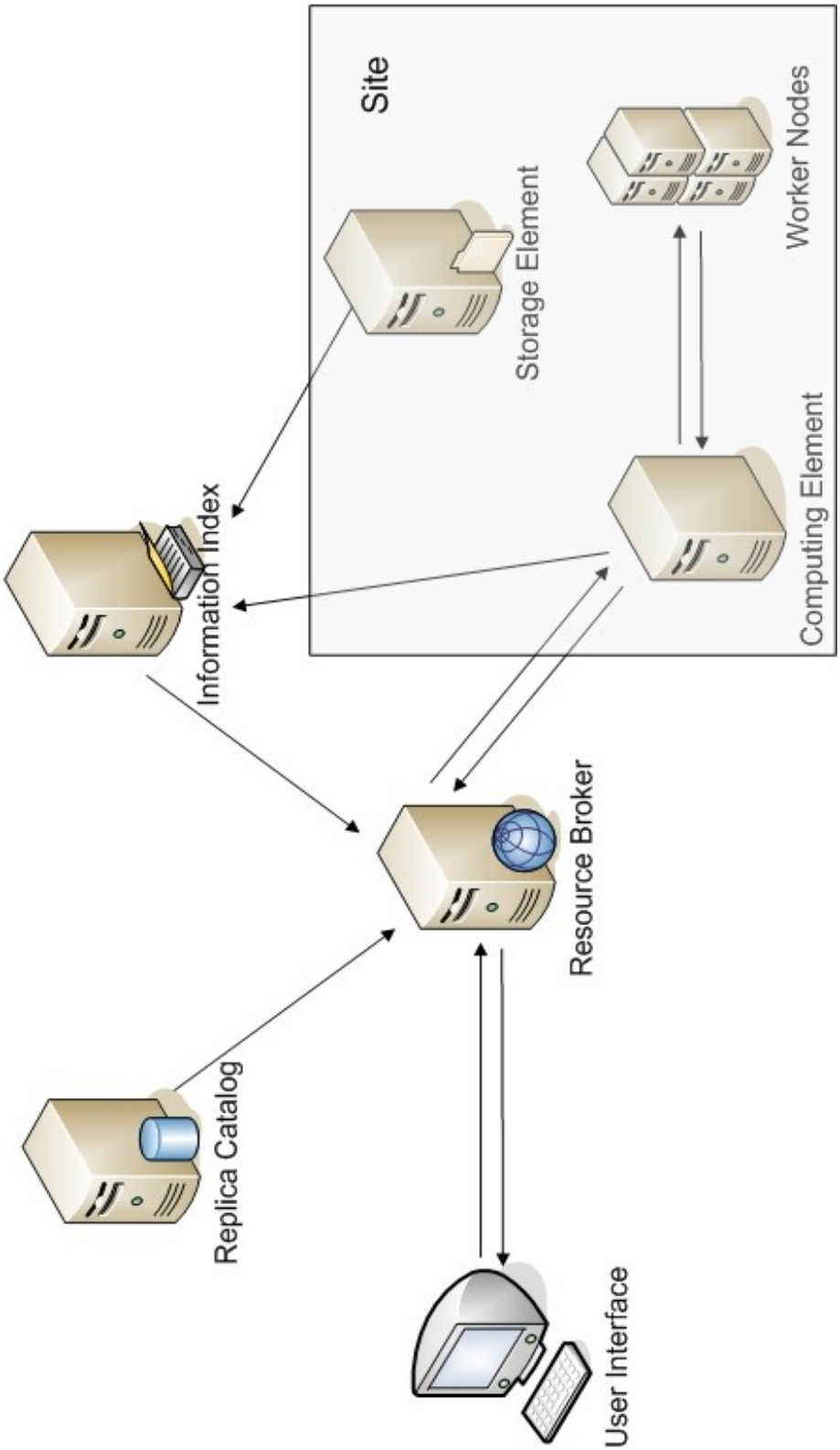

Figura 1. Elementos básicos de un Grid basado en el middleware Glite

Enabling Grids for E-sciencE (EGEE)

El Grid que se ha utilizado para alojar la organización virtual de MAGIC es el vinculado al proyecto EGEE [4]. Este proyecto surgió en el 2004 como respuesta a la necesidad de disponer en Europa de un Grid en producción, como evolución del propuesto por el proyecto DataGrid a principios de siglo que aglutinaba a veinte centros de investigación. EGEE auna la experiencia conseguida en DataGrid con las de iniciativas Grid nacionales como UK e-science, INFN Grid y NorduGrid, ofreciendo compatibilidad con Grid de terceros, como el promovido por la National Science Foundation estadounidense, con el fin de crear una infraestructura Grid global. En el momento de escribir esta memoria, EGEE se encuentra en su tercera fase (EGEE-III).

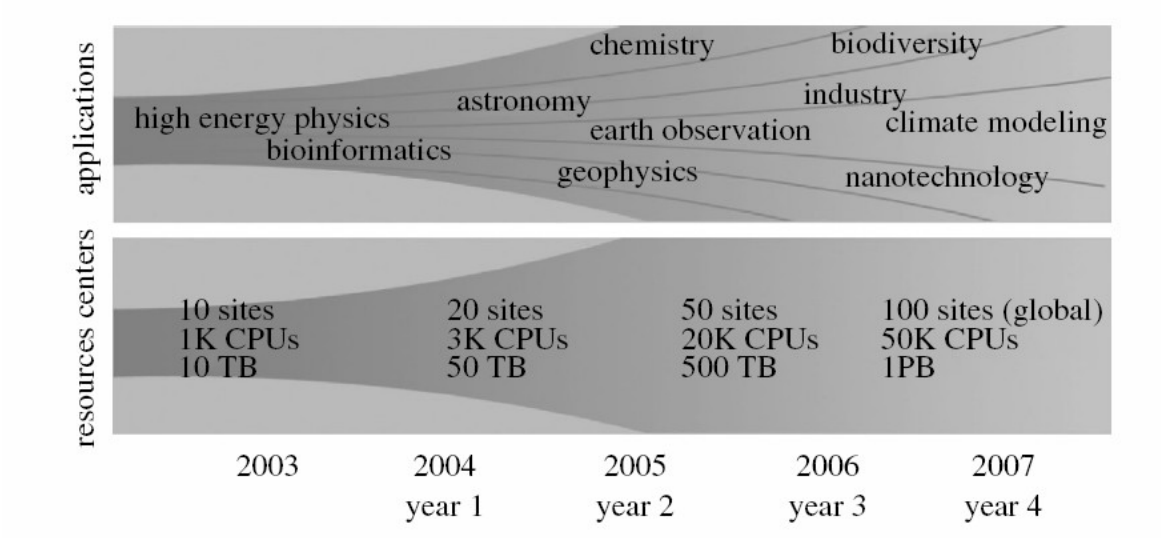

Figura 2. Evolución del proyecto EGEE

Desde su nacimiento en 2004, como se puede ver en la figura 2, EGEE ha ido creciendo tanto en recursos como en aplicaciones, contando en Abril de 2008 con más de 200 organizaciones virtuales que aglutinan a unos 7500 usuarios [5], que tienen a su disposición más de 68000 CPUs y una capacidad sólo en disco de 20PB (a esto habría que añadir la capacidad en cintas), pasando de las veinte instituciones que aglutinaba DataGrid a unas ciento cuarenta instituciones.

Uno de los principales objetivos de EGEE fue el desarrollo de un middleware robusto, que pudiera servir como base de un Grid en producción. Con la experiencia ganada anteriormente con EDG, el middleware del proyecto DataGrid, y utilizando como punto de partida el middleware LCG realizado por el CERN para su uso en el Large Hadron Collider (LHC), se puso en marcha lo que hoy se conoce como Glite, middleware realizado bajo el proyecto EGEE.

El proyecto MAGIC

MAGIC, el telescopio

El telescopio MAGIC (Major Atmospheric Gamma Immaging Cherenkov Telescope) [7] es un telescopio Cherenkov de 17 metros de diámetro para el estudio del Universo en rayos gamma de muy alta energía. MAGIC está situado en la isla canaria de La Palma a 2200 metros sobre el nivel del mar, dentro de las instalaciones del observatorio del Roque de los Muchachos del Instituto Astrofísico de Canarias. Las primeras ideas y estudios para la construcción de este telescopio Cherenkov se remontan al año 1995 [Lorenz 1995], completándose el dossier técnico de MAGIC tres años después [Barrio et al. 1998]. Los fondos para su construcción fueron concedidos en el año 2000, y la construcción del telescopio comenzó en Septiembre de 2001. La inauguración oficial se celebró en Octubre de 2003, si bien, la instalación completa de todos los espejos no se acabó hasta agosto de 2004. A partir de la inauguración comenzó la fase de calibración y pruebas, con la observación de un pequeño número de fuentes, como la Nebulosa del Cangrejo y Mrk 421. En la figura 3 se pueden observar los principales componentes del telescopio: la montura, el reflector y la cámara. La montura es de fibra de carbono para lograr un rápido reposicionamiento (de menos de 30 segundos) en la observación de Gamma Ray Bursts (GRB). El reflector lo componen 964 espejos cuadrados, que envían la luz a la cámara hexagonal situada en el plano focal del reflector, y formada por 577 fotomultiplicadores.

Figura 3. El telescopio MAGIC

MAGIC, como se ha señalado anteriormente, pertenece a la familia de telescopios denominados Cherenkov, que se distinguen por observar flashes de radiación Cherenkov (espectro de luz visible/ultravioleta) con un intervalo de duración muy corto, del orden de nanosegundos, producidos cuando radiación gamma de alta energía interacciona con la atmósfera. Esto quiere decir que la astronomía gamma desde tierra se basa en la detección indirecta de rayos gamma de muy alta energía emitidos por objetos de origen galáctico y extragaláctico, tales como pulsares o núcleos de galaxia activos.

La figura 4 ilustra el proceso de creación de la luz Cherenkov. Una partícula primaria, por ejemplo un gamma, llega a los límites de la atmósfera terrestre donde interaccionará con las moléculas de las que se compone el aire (por lo general O_2 y N_2), de forma que con procesos similares a los que ocurren en los aceleradores de partículas, esta partícula se irá descomponiendo en otras de menor energía, generando en el proceso una cascada de partículas. La componente cargada de estas cascadas emite lo que se conoce como luz Cherenkov, que aunque es visible al ojo humano, por su brevedad es casi imperceptible. Dicha luz, es recogida en el suelo por los telescopios Cherenkov, los espejos de los cuales reflejan dicha luz sobre la cámara, que dará lugar a una imagen dentro del área de la cámara como el que se puede observar en la figura 4.

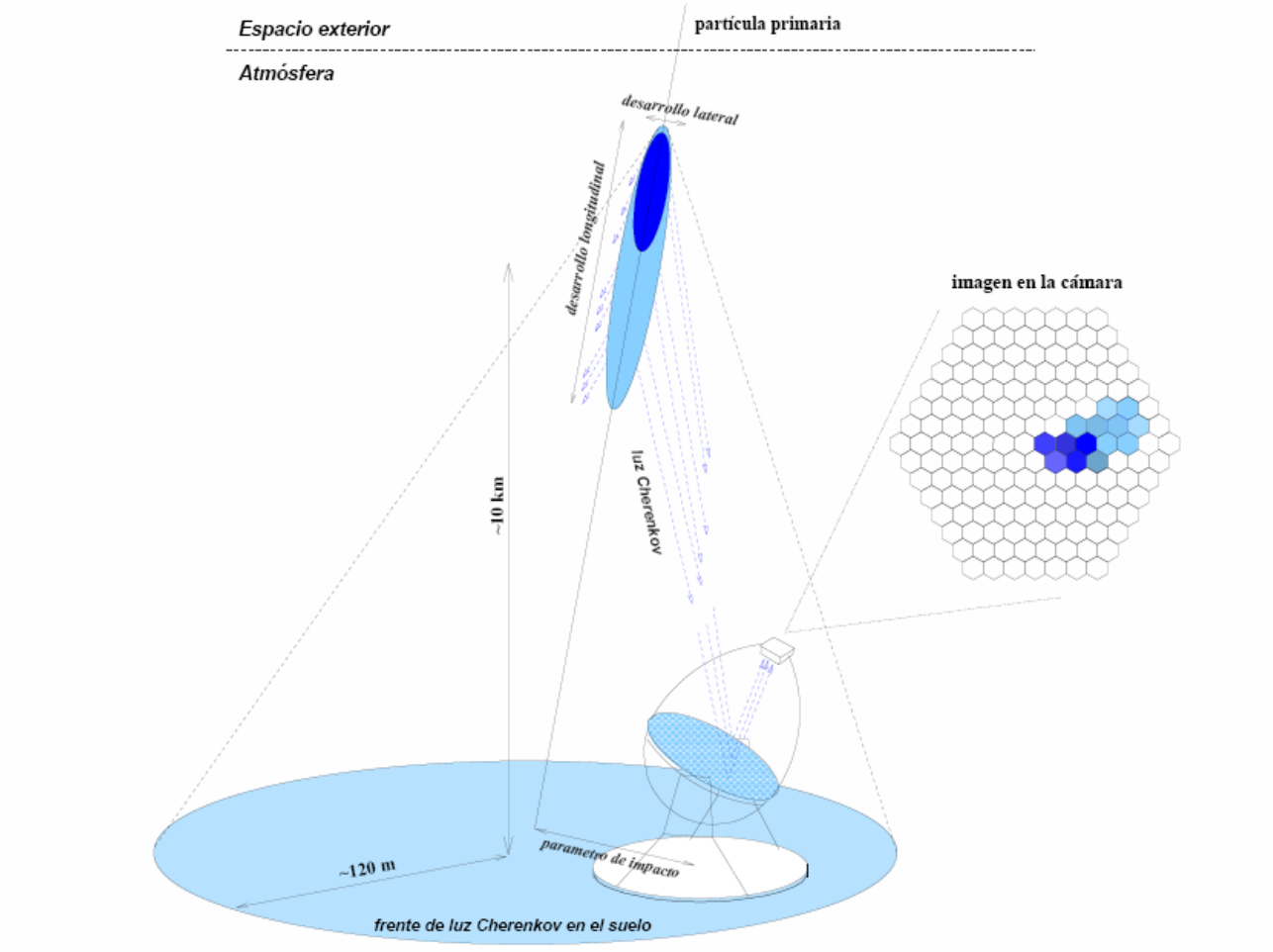

Figura 4. Proceso de detección de una cascada generada por partículas gamma

El mayor problema a la hora de detectar los rayos gamma mediante este tipo de telescopios [8], es el gran número de partículas de otros tipos que llegan a la atmósfera creando cascadas similares a las citadas. Para poder distinguir las cascadas generadas por las partículas gamma de aquellas generadas por otros tipos de partículas hay que analizar el patrón visual generado por dicha cascada en la cámara. Simulaciones Montecarlo de cascadas muestran que en el caso de las partículas gamma éstas son elípticas y con orientaciones hacia el centro de la cámara, debido a que provienen de fuentes a las que el telescopio apunta y a la forma en que las cascadas se desarrollan en la atmósfera. En el caso de imágenes producidas por cascadas iniciadas por otras partículas, las formas y orientaciones de éstas en la cámara son más aleatorias. Es en este punto donde las simulaciones Montecarlo cobran protagonismo, creándose, a partir de los resultados de las mismas, una biblioteca con patrones de aquellas creadas por partículas gamma, a fin de poder comparar estos patrones con aquellos que se producen en el telescopio. Esto permite determinar cuáles de las cascadas observadas son más probables de haber sido originadas por partículas gamma y cuáles provienen del fondo de partículas no deseadas.

MAGIC, la colaboración

El telescopio MAGIC ha sido desarrollado dentro de una colaboración internacional creada para tal fin, que cuenta con más de 150 científicos e ingenieros de 19 instituciones repartidas por Europa, América y Oceanía. El trabajo aquí presentado, ha sido realizado dentro de uno de los grupos miembro de dicha colaboración, el Grupo de Altas Energías de la Facultad de Físicas de la Universidad Complutense de Madrid.

Flujo de datos dentro de MAGIC

Al contrario de lo que ocurre con los detectores de partículas, en la astronomía gamma no es posible testear el telescopio para saber cual va a ser su respuesta, es por esto que se hace necesario la ejecución de simulaciones Monte carlo. Para la simulación de las cascadas generadas por las partículas que interaccionan con la atmósfera, la colaboración MAGIC utiliza Corsika (desarrollado por la colaboración KASKADE), un programa que permite además de simular todo el proceso físico anterior, tener en cuenta otros aspectos como la atmósfera o el campo geomagnético terrestre.

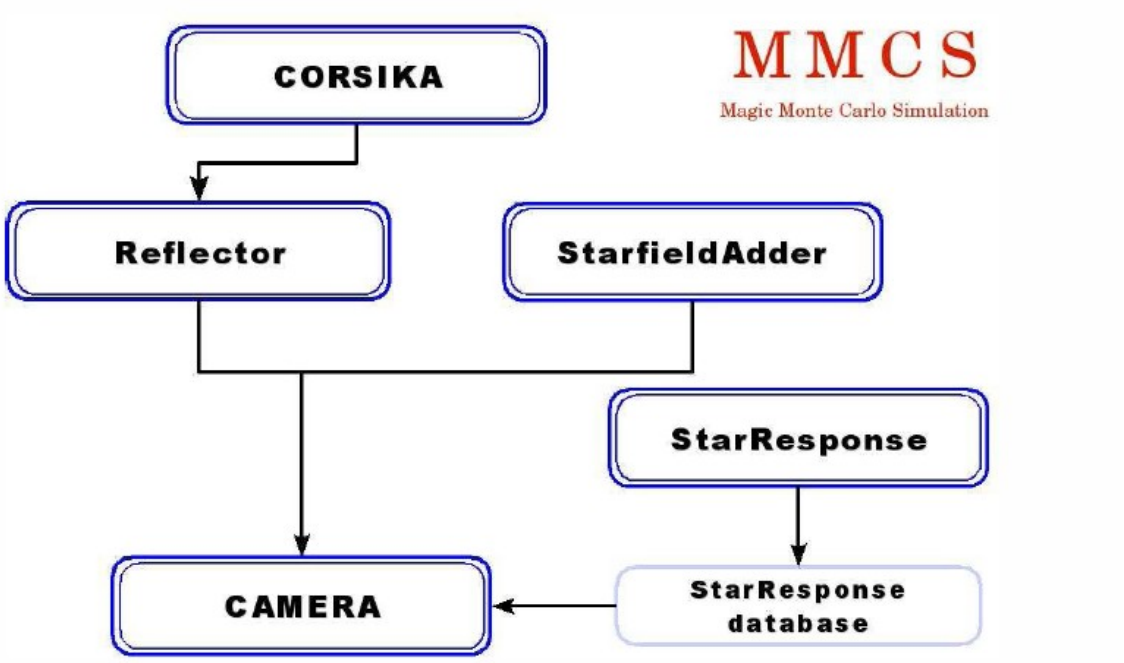

Figura 6. Flujo de trabajo del MAGIC Montecarlo Simulation

La funcionalidad de Corsika se divide en cuatro áreas: una primera que realiza el trazado de las partículas, una segunda que simula los procesos físicos que tienen lugar dentro de las cascadas, una tercera área que aborda las interacciones electromagnéticas de las partículas implicadas en esa cascada, y una cuarta que se encarga de la generación de la luz Cherenkov. Corsika dispone de varios modelos opcionalmente activables dependiendo de la precisión deseada por el usuario. Los ficheros de salida que Corsika genera contienen todos los fotones Cherenkov que llegan al suelo en un área alrededor del telescopio. Una vez simuladas las cascadas, el siguiente paso es simular como éstas serían observadas por el telescopio, para luego ver cual sería el resultado sobre el plano de la cámara. Para llevar esta simulación además de Corsika se utilizan el siguiente conjunto de programas.

▪Reflector. Simulación de la propagación de los fotones Cherenkov en la atmósfera, teniendo en cuenta la atenuación de la luz en la atmósfera, y su reflexión en los espejos del telescopio. Recibe como entrada los resultados de Corsika junto con información sobre la geometría de los espejos. Su salida incluye información sobre todos los fotones que vayan a alcanzar la cámara situada en el plano focal del telescopio.

▪StarfieldAdder. Simulación del campo de visión, de forma que añade la luz no trivial procedente de las estrellas a las imágenes tomadas por el telescopio, dada una posición en el cielo.

- StarResponse. Simulación del fondo de cielo nocturno.

- Camera. Simulación de la electrónica de la cámara del telescopio MAGIC, incluyendo los fotomultiplicadores.

La mayor parte de los datos que se obtienen con telescopios Cherenkov, son debidos a eventos de fondo producidos por partículas cargadas. El objetivo del análisis de datos es eliminar esta información de fondo no útil, y extraer la información de la señal debida exclusivamente a los rayos gamma para poder realizar un estudio sobre su punto de origen y su distribución en energía. Para ello se emplea el método desarrollado por M. Hillas [9], que caracteriza las imágenes producidas por las cascadas atmosféricas para, en función de ciertos parámetros de las imágenes, extraer las imágenes producidas por rayos gamma sobre las del fondo. Esta caracterización requiere de un número considerable de recursos computacionales en simulaciones Montecarlo.

Objetivos

Los objetivos que se persigue lograr con el proyecto MAGICGrid son los siguientes:

- Puesta en marcha de la organización virtual de MAGIC en EGEE, proporcionando soporte a los usuarios y centros proveedores de recursos de la misma. En colaboración con varios institutos, se han configurado los distintos servicios Grid para la organización virtual MAGIC, desarrollando scripts con los que llevar a cabo producciones Monte Carlo a bajo nivel para probar dicho entorno.

- Adaptación del software de simulación utilizado por MAGIC al entorno Grid y automatización en la instalación de éste en dicho entorno. Se ha adaptado tanto el software principal utilizado en la simulación, Corsika, como varias de las librerias que este utiliza, para su correcto funcionamiento en el entorno Grid.

- Desarrollo de una interfaz gráfica accesible vía web para la configuración y ejecución de simulaciones Montecarlo, que facilite un único punto de acceso a la ejecución de simulaciones para el usuario. Se ha desarrollado una interfaz Java que permite al usuario acceder a todos los parámetros de configuración de la simulación mediante un navegador web, sin necesidad de la instalación de software adicional.

- Instalación y configuración de un nodo Grid local, basado en Globus y Gridway, que se encargue de enviar los trabajos recibidos desde la interfaz gráfica vía servicios web a los recursos asociados a la organización virtual de MAGIC en EGEE. Se ha instalado y configurado un nodo Grid de las características anteriormente citadas, configurando un grupo de computing elements para su correcto funcionamiento con este nodo.

Estructura del documento

Este trabajo se divide en seis capítulos, en los que tras una introducción al entorno Grid y al proyecto MAGIC dentro del cual se realiza este trabajo, se lleva a cabo una introducción al problema a abordar y las iniciativas previas que a este respecto se han llevado a cabo dentro de la colaboración. A continuación se detalla el trabajo llevado a cabo tanto en lo referente a la puesta en marcha de la organización virtual, como en el diseño, implementación y configuración de la interfaz gráfica para la configuración y envío de simulaciones al Grid como del nodo Grid basado en Globus y Gridway encargado de enviar dichos trabajos. Por último se ofrecen varias propuesta de ampliación para la herramienta, que serán abordadas en la segunda fase del proyecto, de Septiembre a Diciembre de 2008. Más en detalle, el contenido de cada capítulo es el siguiente.

Capítulo 1. Introduce el entorno Grid, y de qué manera las características de éste pueden ayudar a escalar la infraestructura de computación y almacenamiento con la que cuenta la colaboración MAGIC. A lo largo del capítulo se detalla la arquitectura y funcionamiento de EGEE, mostrando el funcionamiento de aquellos elementos con los que MAGICGrid interaccionará. En un segundo apartado, se introduce el proyecto sobre el que se desarrolla este trabajo, el telescopio MAGIC, sus características y funcionamiento, así como el proceso físico que se simulará. También se aborda el flujo de datos dentro de MAGIC, detallando los programas que forman parte del proceso de simulación.

Capítulo 2. Realiza un repaso a la literatura que recoge las propuestas que previamente se han realizado dentro de la colaboración MAGIC en lo referente a la adopción del entorno Grid como infraestructura de computación y almacenamiento. Por otro lado se analizan las soluciones implementadas en terceros proyectos en lo referente a las simulaciones Montecarlo masivas. Detallando las soluciones propuestas dentro de los experimentos ZEUS, ATLAS y CMS.

Capítulo 3. Aborda el trabajo realizado en la puesta en marcha de la organización virtual de MAGIC, así como el material desarrollado para la gestión y uso de la misma. Dentro del mismo se detalla un esquema de la arquitectura de la plataforma de prueba que se ha utilizado para llevar a cabo el desarrollo y pruebas asociadas a MAGICGrid. También se incluye la documentación y scripts desarrollados para la instalación remota y el envío de simulaciones, estos últimos haciendo uso de las user interfaces que fueron configuradas y utilizadas en la configuración de la organización virtual.

Capítulo 4. Detalla la arquitectura, diseño y los detalles de la implementación de la herramienta MAGICGrid. Asimismo se abordan los detalles de la configuración del nodo Grid basado en el middleware Grid Globus y el metaplanificador Gridway.

Capítulo 5. Contiene el estudio de las tres ampliaciones que se llevarán a cabo en la segunda fase del proyecto, realizando un estudio del arte disponible en cada una de las mismas. En primer lugar se detalla el uso de una base de datos que permita la gestión de los trabajos enviados al Grid mediante el uso de metadatos asociados a los mismos. Por otro lado se aborda el uso de agentes software para el procesamiento y gestión de los trabajos, y por último se analiza el almacenamiento de los datos en un formato compatible

con la iniciativa Virtual Observatory, de forma que estos sean accesibles por las herramientas asociadas a esta iniciativa.

Capítulo 6. Incluye las conclusiones y resultados del trabajo realizado.

Capítulo 2. Situación previa del proyecto

Estado del proyecto MAGICGrid

Hasta ahora, el proyecto MAGIC llevaba a cabo la generación de las simulaciones Montecarlo de forma local, utilizando los recursos de uno de los institutos asociados a la colaboración, sin posibilidad de compartir toda la potencia computacional disponible a través de los distintos grupos de la colaboración. Hay que tener en cuenta que a la carga de las simulaciones se une la de los datos reales, así el sistema de adquisición de datos del telescopio MAGIC recibe en media de 500 a 600 GB de información por noche, lo que se duplicará con la puesta en marcha del segundo telescopio en Septiembre de 2008. A esto hay que añadir los datos que provee el sistema de control del telescopio y la información que proporciona la estación.

Estas simulaciones constan de varios paquetes. Por un lado, el programa CORSIKA [10] se emplea para la simulación de cascadas atmosféricas, simulando las interacciones que tienen lugar desde la entrada de la partícula en la atmósfera terrestre, hasta que la cascada de partículas secundarias llega al suelo. La velocidad computacional depende de varios factores, como por ejemplo, la energía y naturaleza de la partícula primaria que se quiera simular. Por ejemplo, según los resultados recogidos en [11] indican que en torno a 6,5 segundos en un Pentium 2Ghz se emplean para generar un evento. Otras mediciones según el mismo autor, sugieren que un cluster basado en Condor, produce en torno a $3 \cdot 10^5$ eventos al día, de los cuales sólo el 7% de los gamma y el 0,1% de los protones sirven para realizar un análisis posterior. De esta forma para obtener los datos simulados correspondientes a lo que sería una noche de observación, en torno a tres millones y medio de sucesos, se necesitarían entre 288 días (sólo rayos gamma) y 19200 días (sólo fondo).

Por otro lado, está la simulación del detector, que incluye el reflector, la cámara y la electrónica encargada de la toma de datos. Sin embargo, esta segunda fase corresponde a una pequeña fracción del tiempo total necesario para simular un evento completo, desde su primera interacción en la atmósfera hasta su detección y caracterización. A la capacidad de computación necesaria para realizar las simulaciones Montecarlo, hay que añadir la necesidad de llevar a cabo una gestión óptima de los datos para llevar a cabo el análisis de los mismos. Para optimizar la generación de simulaciones Montecarlo de cara al lanzamiento del segundo telescopio de la colaboración, MAGIC-II, situado a pocos metros del telescopio MAGIC, en 2004 se puso en marcha un grupo de trabajo dentro de la colaboración para portar las simulaciones Montecarlo a un entorno Grid. De entre los beneficios que se señalaban en la propuesta realizada por Harald Kornmayer para la colaboración titulada *Production of MAGIC Monte Carlo on the Grid*, en las áreas de intercambio de datos, producción de Monte Carlo y el análisis de datos, destacaban los siguientes:

- Ofrece a los usuarios de la colaboración un punto de acceso único, que facilite el acceso tanto a los datos como a la ejecución de las simulaciones que dan lugar a estos
- Desvincular de los usuarios la obligación de instalar y configurar el software, y traer los datos a su máquina en local, pudiendo gestionar todo esto de forma automática el Grid, mejorando tanto el tiempo de acceso y la tolerancia a fallos, gracias a las réplicas
- Ofrecer un entorno de computación unificado para todos los miembros de la colaboración, haciendo posible el uso de recursos geográficamente distribuidos y no accesibles hasta ahora
- Instalar las herramientas de análisis de datos de MAGIC dentro del Grid, pudiendo analizar los datos allí donde se encuentren los resultados de las simulaciones Montecarlo, con la mejora en tiempos y de experiencia de usuario que esto supone

Para la puesta en marcha de este proyecto se analizaron los requisitos que la aplicación para la generación de simulaciones Montecarlo y el análisis de datos para MAGIC debía de tener, dando como resultado la siguiente lista:

- Seguridad. El acceso y uso de los recursos asociados al Grid proporcionado por las distintas instituciones participantes en la colaboración MAGIC debe estar limitado a los miembros de dicha colaboración.
- Acceso a los datos. El acceso a los datos en sus distintas fases de producción (raw, preprocesados, analizados y resultado final) debe ser sencillo para los usuarios.
- Transferencia de datos. La transferencia de datos, debido al tamaño de los mismo debe ser limitada al máximo, debiendo ser el sistema el que se encargue automáticamente de situar los datos en aquella localización que minimice el gasto de ancho de banda.
- Accesibilidad. El acceso al sistema debe ser de fácil manejo, de forma que haya un único punto de entrada para un usuario a la hora de hacer uso de las capacidades que le ofrece la organización virtual MAGIC.
- Disponibilidad. El sistema debe ser tolerante a fallos con el objetivo de ofrecer una disponibilidad 24x7.
- Escalabilidad. El sistema será capaz de añadir al mismo de forma dinámica otros recursos pertenecientes a la colaboración MAGIC, tanto de computación como de almacenamiento, de forma que se reduzca el tiempo necesario tanto para las simulaciones como para el almacenamiento de datos.
- Distribución del software. El software asociado a las simulaciones Montecarlo y al análisis de datos debe poder ser instalado de forma automática en todos los nodos Grid pertenecientes a la organización virtual MAGIC.

Para alcanzar estos objetivos se distinguieron los siguientes casos de uso dentro de la aplicación [12], recogidos en la figura 5, que se detallan más adelante.

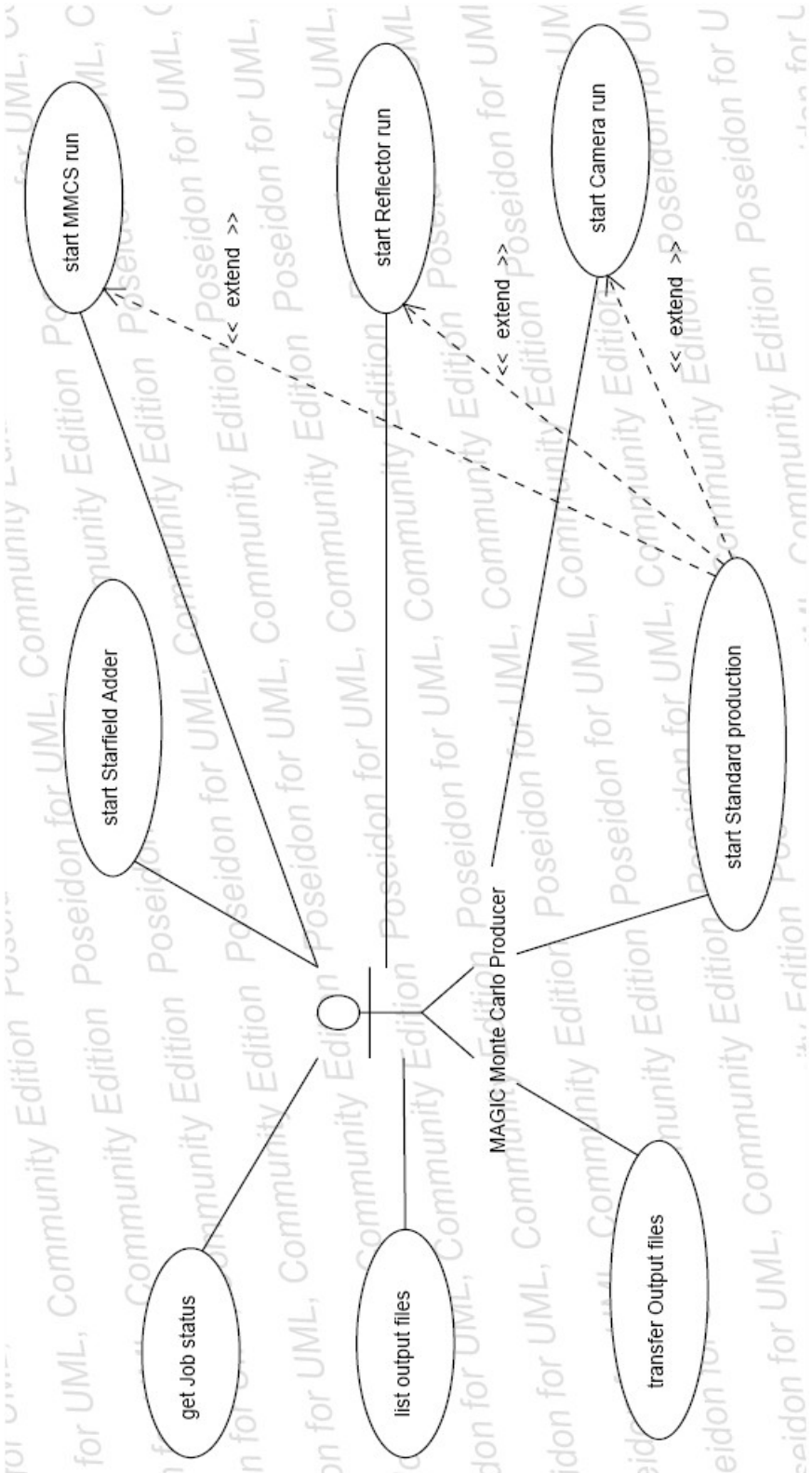

Figura 5. Casos de uso para la producción de Montecarlo

Caso de Uso: start MMCS run	
General	El productor de simulaciones Montecarlo comienza con la generación de simulaciones de cascadas mediante el programa Corsika.
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema
Secuencia de Actividades	P-1: El usuario especifica los parámetros de entrada para MMCS. P-2: Los usuarios obtienen el runnumber de la base de datos, y guardan allí los parámetros de entrada. P-3: El usuario envía los trabajos al Grid. La distribución de los trabajos y la organización de los datos es transparente para el usuario. P-4: Los parámetros del trabajo Grid se guardan en la base de datos.

Caso de Uso: list output files	
General	El productor de simulaciones Montecarlo obtiene una lista de sus trabajos y su estado.
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema
Secuencia de Actividades	P-1: Se obtiene de la base de datos el estado de todos los trabajos. P-2: Se consulta el estado de los trabajos sin terminar, actualizando esta información en la base de datos. P-3: Se le muestra al usuario la información generada sobre el estado de los trabajos.

Caso de Uso: transfer Output files	
General	El productor de simulaciones Montecarlo obtiene una lista de todos los trabajos terminados cuyos datos no hayan sido aún transferidos, y selecciona que datos deben ser transferidos a su cuenta local.
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema
Secuencia de Actividades	P-1: Todos los trabajos para los que los datos aún no han sido transferidos, están listos. P-2: El usuario selecciona los datos a transferir. P-3: Los datos son transferidos.

Caso de Uso: start Reflector run	
General	El productor de simulaciones Montecarlo simula la reflexión de los fotones Cherenkov en el espejo del telescopio MAGIC.
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema El actor ya ha simulado las cascadas mediante Corsika
Secuencia de Actividades	P-1: El usuario especifica los parámetros de entrada para Reflector. P-2: El usuario obtiene la localización de los datos de entrada de Reflector de la base de datos (estos habrán sido generados previamente por Corsika). P-3: El usuario envía los trabajos al Grid. La distribución de los trabajos y la organización de los datos es transparente para el usuario. P-4: Los parámetros del trabajo Grid se guardan en la base de datos.

Caso de Uso: start Camera run	
General	El productor de simulaciones Montecarlo simula la imagen que generan en la cámara del telescopio MAGIC los fotones reflejados en su espejo.
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema El actor ya ha simulado la reflexión de los fotones Cherenkov en el espejo del telescopio MAGIC mediante Reflector
Secuencia de Actividades	P-1: El usuario especifica los parámetros de entrada para Camera. P-2: El usuario obtiene la localización de los datos de entrada de Camera de la base de datos (estos habrán sido generados previamente por Reflector). P-3: El usuario envía los trabajos al Grid. La distribución de los trabajos y la organización de los datos es transparente para el usuario. P-4: Los parámetros del trabajo Grid se guardan en la base de datos.

Caso de Uso: start Starfield Adder	
General	El productor de simulaciones Montecarlo simula la llegada a la cámara del telescopio MAGIC de los fotones producidos por las estrellas.
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema El actor ya ha simulado la reflexión de los fotones Cherenkov en el espejo del telescopio MAGIC mediante Reflector
Secuencia de Actividades	P-1: El usuario especifica los parámetros de entrada para Starfield Adder. P-2: El usuario obtiene la localización de los datos de entrada de Starfield Adder de la base de datos (estos habrán sido generados previamente por Reflector). P-3: El usuario envía los trabajos al Grid. La distribución de los trabajos y la organización de los datos es transparente para el usuario. P-4: Los parámetros del trabajo Grid se guardan en la base de datos.

Caso de Uso: get Job status	
General	El productor de simulaciones Montecarlo obtiene el estado de un trabajo
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema
Secuencia de Actividades	P-1: Se obtiene de la base de datos el estado del trabajo. P-2: Se le muestra al usuario la información generada sobre el estado del trabajo.

Caso de Uso: start Standard production	
General	El productor de simulaciones Montecarlo simula el proceso completo de observación del telescopio MAGIC. Generación de cascadas, reflexión en su espejo y captación de los fotones por la cámara.
Actor	El productor de simulaciones Montecarlo
Precondiciones	El actor ya está conectado al sistema
Secuencia de Actividades	P-1: El usuario simula las cascadas utilizando Corsika. P-2: El usuario simula la reflexión de los fotones en el espejo utilizando Reflector. P-3: El usuario simula la imagen que generan en la cámara del telescopio MAGIC los fotones reflejados en su espejo utilizando Camera.

Arquitectura del sistema de producción

Para la puesta en marcha del sistema se escogió a tres centros con experiencia en Grid, el PIC (España), CNAF (Italia) y GRIDKA (Alemania), siendo el PIC el encargado de recibir los datos directamente desde el telescopio en La Palma. Para implementar el sistema Grid se utilizó el middleware EDG sobre el Globus toolkit. Sobre esta infraestructura se pusieron en marcha, en todos los centros, las aplicaciones de simulación de cascadas (MMCS) y de análisis de datos (MARS).

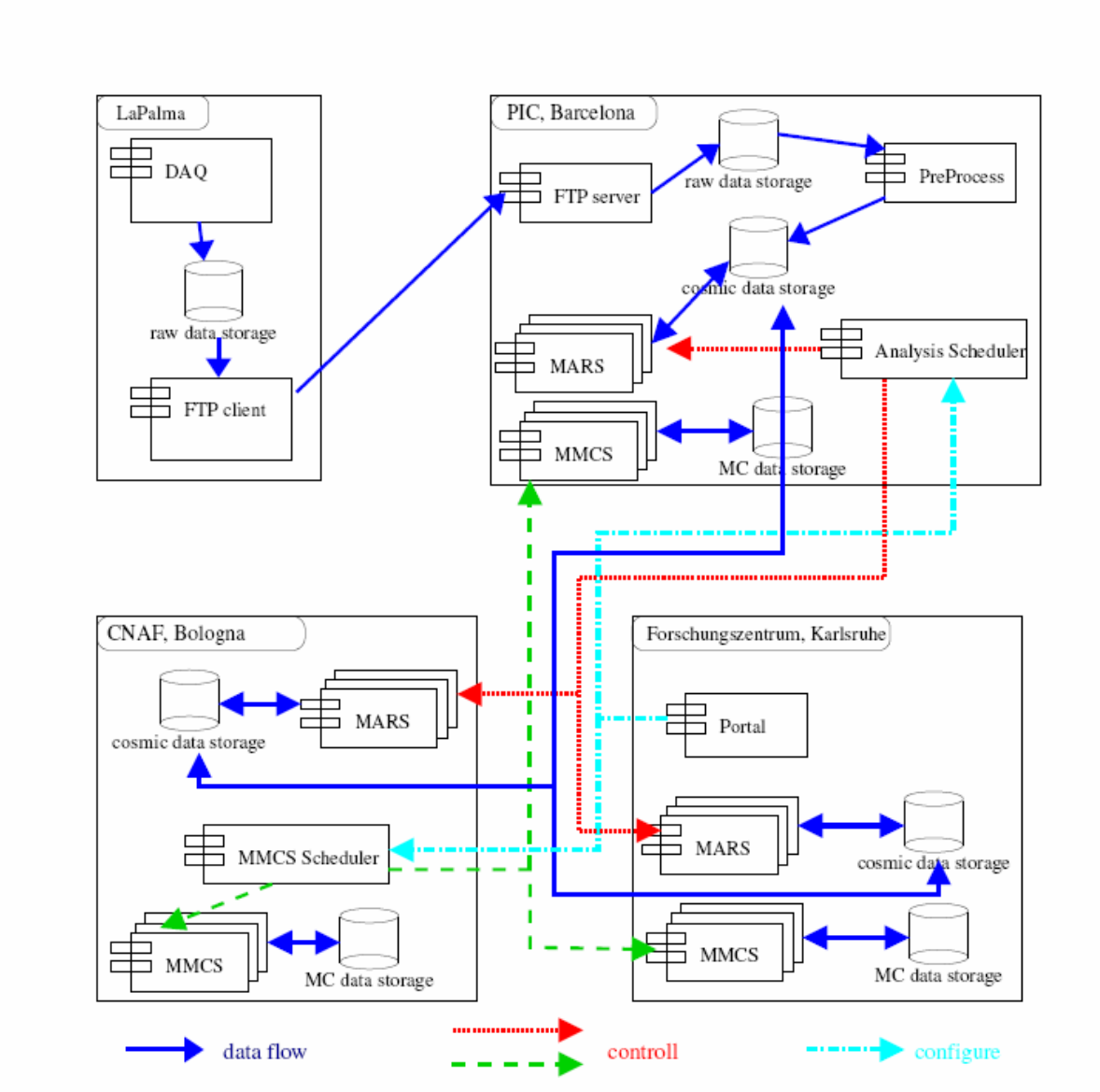

Figura 7. Arquitectura de la solución propuesta en la primera versión de MAGICGrid.

Los datos reales de las cascadas recibidas por el telescopio son enviados a todos los centros participantes, donde también se corren MARS, para el análisis de datos, y MMCS, para la simulación de cascadas.

Tanto los datos reales de las cascadas recibidas por el telescopio, como aquellos correspondientes a las simulaciones, se distribuían entre todos los centros. El CNAF era el encargado de gestionar las peticiones correspondientes al MMCS a través de su resource broker, de la misma forma que el PIC era el encargado de gestionar el asociado al análisis de los datos. El punto único de acceso al sistema se llevaba a cabo a través de un portal situado en el GRIDKA. Más en detalle, la distribución de servicios por centro era la siguiente.

▪Telescopio MAGIC, La Palma. Aquí se encuentra el Data Adquisition System (DAQ) encargado de controlar el telescopio y crear un fichero único con toda la información y datos que recibe de los distintos subsistemas del telescopio. La transferencia de datos desde LaPalma al PIC se llevaría a cabo mediante GridFTP, que añade seguridad a la transmisión de los datos.

▪PIC, Barcelona. Una vez transmitidos los datos desde el telescopio, y reducidos mediante distintos métodos para limpiar las imágenes, los datos se almacenan en una base de datos distribuida en todos los centros, de nombre cosmic data storage, que utiliza la funcionalidad (replica catalogs) que a este respecto ofrece Globus. La salida de las simulaciones del MMCS se guardan igualmente de forma distribuida en la base de datos MC data storage.

▪CNAF, Bolonia. Acoge el resource broker del sistema MMCS, basado en el resource broker que ofrece el Globus Toolkit.

▪GRIDKA, Karlsruhe. Ofrece a través de un portal de acceso vía web un punto de acceso único a todo el sistema. A través de este portal se pueden enviar peticiones tanto los schedulers encargados de gestionar las simulaciones Montecarlo como el análisis de datos.

Los centros asociados a MAGIC podían conectar sus equipos al más cercano de estos centros a fin de compartir sus recursos computacionales. En estos equipos se instalarían tanto el MMCS como MARS, registrándose previamente en los schedulers de ambas aplicaciones, que se encargarían de enviar los trabajos a estos nuevos recursos. El desarrollo e implementación del sistema comenzó en el verano del 2004, completando las siguientes fases:

-Adaptación de MMCS para el entorno Grid. Adaptación de la aplicación MMCS para el entorno Grid, e instalación del mismo en los tres centros principales (PIC, CNAF y GRIDKA). Junto a esto se puso en marcha una primera versión del portal que permitiera gestionar las simulaciones.

-Producción de Montecarlo y puesta en marcha del análisis. En la segunda fase se comenzó con la producción de simulaciones Montecarlo, en las cuales participaron sitios adscritos a la colaboración distintos de los tres centros principales. Se puso en

marcha también la transmisión de datos entre el telescopio y su centro del control en La Palma y el PIC, y se adaptó el portal para permitir la gestión del análisis de los datos, el programa asociado al cual, MARS, había sido instalado en PIC, CNAF y GRIDKA.

-Producción del análisis. En esta última fase el análisis de los datos se extendió al resto de centros participantes, pudiendo adaptar el análisis cada uno de estos a sus propias necesidades. Se comenzó con las pruebas para utilizar sistemas de punto de acceso único para el usuario, como el Migration Desktop, perteneciente al proyecto CrossGrid, así como para probar la escalabilidad del sistema con más centros participantes e incluso proveedores comerciales.

El desarrollo de MMCS se centró en un primer momento en tres casos de uso: job submission, job monitoring y data management, para los que se crearon las correspondientes interfaces de usuario. Para gestionar el estado de las simulaciones Montecarlo se creo una base de datos dedicada a tal fin. Las primeras pruebas con esta primera versión del sistema se llevaron a cabo sobre CrossGrid, el cual contaba en aquel momento con 16 centros participantes y utilizaba LCG-2 como middleware. Se enviaron 250 trabajos, de los cuales el 10% fallaron debido al sistema de réplica de datos aunque pudieron reenviarse gracias a la gestión vía base de datos, produciendo aproximadamente 250000 rayos gamma. Con la llegada del proyecto EGEE, la colaboración MAGIC fue aceptada como generic application por los gestores de dicho proyecto, comenzando la colaboración con el mismo en Septiembre de 2004

Estado del arte en simulaciones Montecarlo en entornos Grid

Existen un gran número de aplicaciones para las simulaciones Montecarlo, no sólo en física de altas energías, sino en otros campos tan dispares como la medicina o las finanzas. La llegada del Grid ha hecho que muchos de los proyectos que antes dependían de infraestructuras de computación distribuida, más o menos ad hoc, hayan portado estas soluciones a Grid, bien adaptando sus scripts al nuevo entorno o utilizando alguna de las aplicaciones puente entre el software de cluster y el de Grid, como por ejemplo el Condor glide-in mechanism, que permite añadir al gestor de clusters Condor, worker nodes de Grid de forma transparente al usuario. En EGEE los proyectos implicados generalmente han optado bien por utilizar la aplicación de user interface que ofrecen middleware como LCG o Glite, dejando abierta la puerta al desarrollo de conectores para el envío de trabajos a Grids no basados en alguno de los distintos middleware proporcionados por el proyecto EGEE, o bien por desarrollar soluciones propias, generalmente basadas en agentes software, que se encargan de gestionar los trabajos y enviarlos al Grid, sea cual sea la naturaleza de éste. Para ilustrar el trabajo que se está llevando a cabo en esta área, se han elegido cuatro experimentos que están o recientemente han estado implicados en la producción masiva de simulaciones Montecarlo, en concreto dos de los experimentos del Large Hadron Collider (LHC), ATLAS y CMS, el detector ZEUS del proyecto HERA y el

experimento BaBar del Stanford Linear Accelerator Center (SLAC). De este último se tratara la producción de Montecarlo dentro del entorno EGEE, uno de los muchos de los que ha hecho uso este experimento para llevar a cabo la producción de Montecarlo.

Arquitectura de producción en ZEUS

En el caso de ZEUS, la introducción del Grid en el proceso de producción de Montecarlo se ha hecho sin eliminar el antiguo método, permitiendo así utilizar bien la infraestructura Grid o la anteriormente utilizada. La arquitectura del sistema de producción de ZEUS incluye muchos de los elementos que se encuentran en este tipo de aplicaciones. En la figura 8 se pueden distinguir los dos elementos principales: el sistema de producción y el user interface de LCG utilizado como puerta de acceso al Grid, además de la posibilidad de enviar trabajos al antiguo sistema de producción (classical site) que ya no recibe actualizaciones.

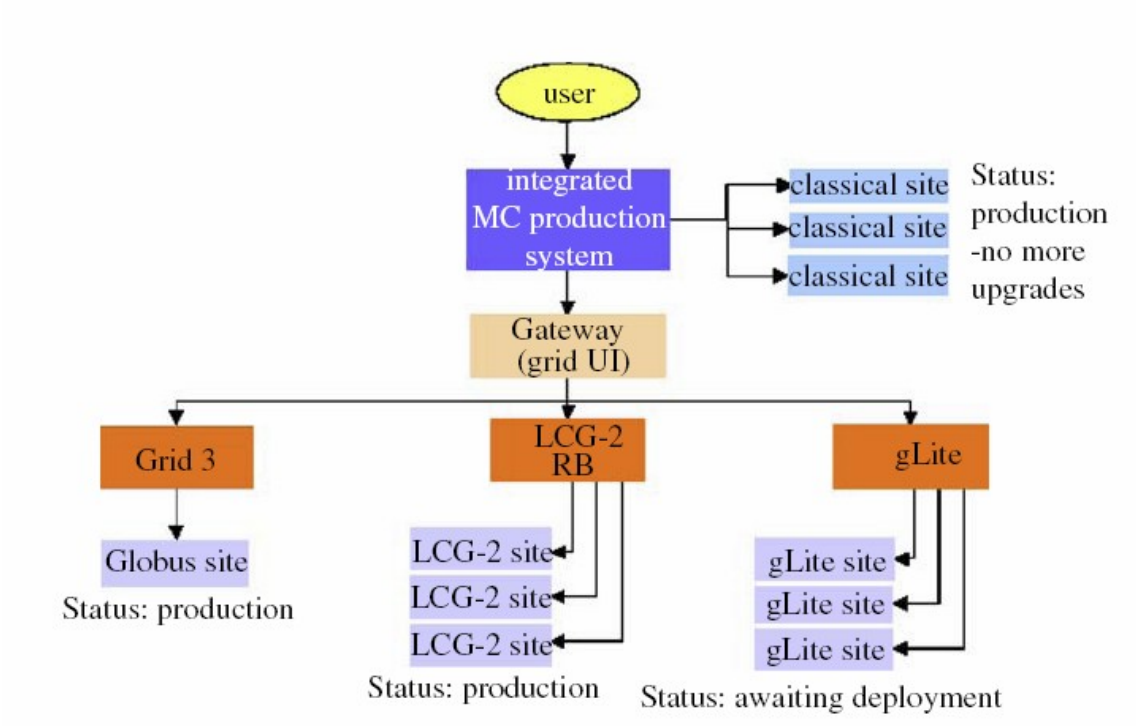

Figura 8. Arquitectura del sistema de simulación ZEUS

Junto con la posibilidad de lanzar trabajos a entornos Grid que utilicen distinto middleware, otro de los puntos más característicos de este tipo de aplicaciones es el uso de una base de datos para gestionar los trabajos, sobre todo en lo referente al reenvío de estos en el caso de fallo, que suele producirse de manera habitual en los Grids en producción. A la hora de enviar simulaciones de este tipo al Grid se plantea la duda de cual debe ser el tamaño de estos trabajos. En el caso concreto del sistema de producción de ZEUS, como se detalla en [13], se ha optado por crear a partir de cada uno de los

trabajos que se envían, pequeños trabajos Grid que generen entre 1000 y 2000 eventos cada uno, lo que supone un tiempo de espera de tres horas, un tiempo de espera razonable en el caso de tener que reenviar el trabajo debido a un fallo.

Arquitectura de producción en ATLAS

Como el resto de experimentos del LHC, ATLAS ha hecho uso de la infraestructura Grid puesta a punto dentro del proyecto LHC Computing Project (LCG) para llevar a cabo la producción de simulaciones. En este caso (Figura 9) [14] y [15], se introducen algunos elementos nuevos respecto de la solución propuesta en ZEUS. Aquí los trabajos no son enviados directamente al Grid, sino que son almacenados temporalmente, junto con toda la información relativa a los mismos, por ejemplo la localización de los datos necesarios para su ejecución, en una base de datos (prodDB). El resto del entorno de ejecución se basa en un sistema multiagente, donde unos agentes supervisores se encargan de seleccionar trabajos de esta base de datos y enviárselos a los ejecutores. Para esto el supervisor hace uso de una descripción del trabajo en XML que los ejecutores transformarán posteriormente a uno de los distintos lenguajes de definición de trabajos para el Grid, por ejemplo JDL. Estos ejecutores se encargarán del trabajo de bajo nivel, enviándolos a la infraestructura Grid o similar, para las que hayan sido preparados.

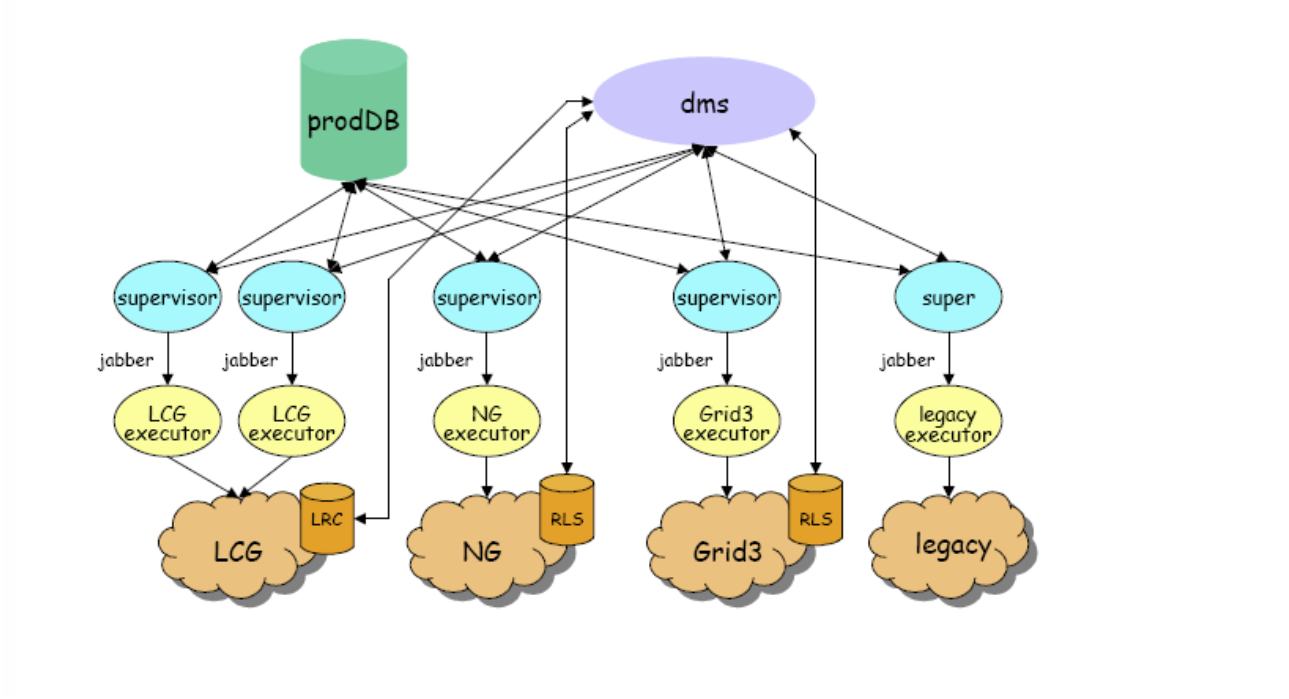

Figura 9. Arquitectura del sistema de producción de simulaciones para ATLAS

Así ahora mismo existen versiones de estos ejecutores para LCG, NorduGrid y Grid3, y para sistema de gestión de colas como LSF, PBS y BQS. El trabajo del supervisor no termina en el traspaso de la responsabilidad del mismo al ejecutor, sino que será el mismo

el que se encargue de monitorizar si el trabajo ha terminado, gestionando y almacenando los resultados y la información asociada a dicho trabajo, o en el caso de no haber terminado, reubicando el trabajo en la base de datos para su posterior reenvío. A la hora de enviar un trabajo el supervisor indicará al ejecutor algunas necesidades del mismo, como que en el worker node esté instalado el software de simulación para ATLAS o que la conexión tenga la suficiente calidad y ancho de banda como para transferir los datos del resultado. Otro de los aspectos nuevos en ATLAS respecto a ZEUS es el uso por parte de los ejecutores, de un metaplanificador, en este caso WMS que permite la gestión avanzada del trabajo eliminando la lógica necesaria para esta tarea de los ejecutores. La gestión de los datos se lleva a cabo de forma centralizada, utilizando los sistemas que para este fin ofrecen los distintos middleware Grid (Globus RLS en el caso de NorduGrid y Grid3 y LRC en el caso de LCG). De esta forma las consultas de datos realizadas al sistema de gestión de datos general, de nombre Don Quijote, se envían de forma transparente para el usuario a los distintos sistemas de gestión.

Arquitectura de producción en CMS

El sistema de simulación del experimento CMS también basa su funcionamiento en un sistema multiagente [16]. Una vez el usuario envía el trabajo, el ProdManager se encarga de enviar dicho trabajo a aquel agente (ProdAgent) cuyos recursos asociados están libres, existiendo ProdAgent para las distintas implementaciones del middleware Grid. En este caso es interesante ver la forma de llevar a cabo dentro de este sistema la recolección de datos. Estos datos en vez de ser gestionados de manera centralizada, se guardan localmente en aquellos centros que los han procesado, de forma que una vez existen un número suficiente de datos en dicho centro, el sistema se encarga de enviar un trabajo que aúna todos esos datos. Al igual que en los dos casos anteriores, se utiliza una base de datos para gestionar los trabajos.

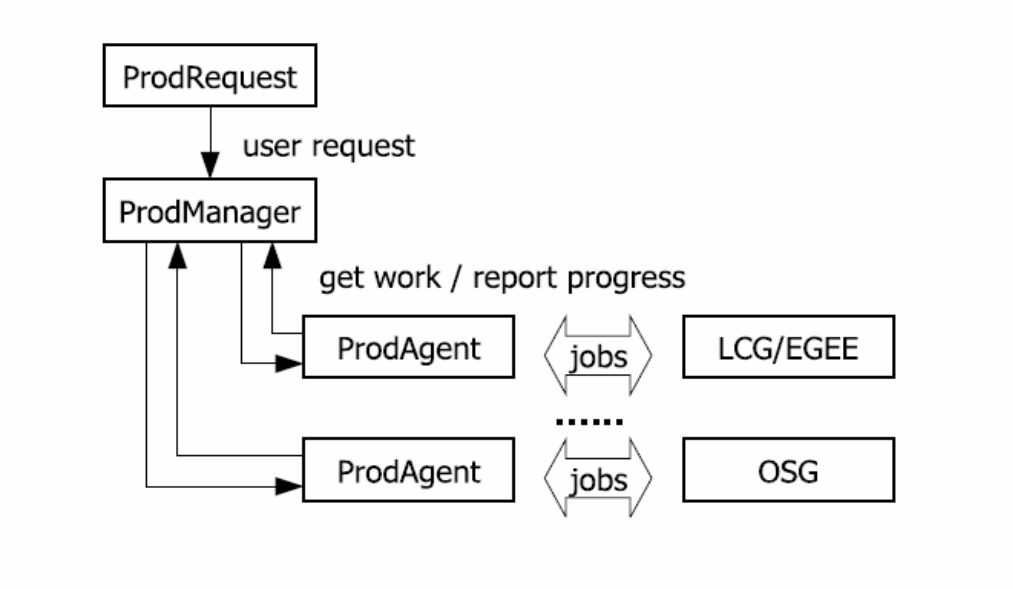

Figura 10. Sistema de producción de simulaciones Montecarlo para CMS

Por último señalar brevemente el caso del experimento BaBar del SLAC, donde su uso dentro del proyecto EGEE, se limitó al desarrollo de una serie de scripts que hacían uso del conjunto de comandos para la gestión de trabajos incluidos dentro del middleware LCG.

Aplicación a MAGIC

Como se ha visto, el número de opciones en lo que al desarrollo de la arquitectura en un sistema de producción de simulaciones Montecarlo basado en Grid, es grande, si bien los elementos comunes suelen aparecer en prácticamente todos los sistemas: un sistema ad hoc para la gestión de trabajos, generalmente basado en la información contenida en una base de datos, y una serie de ejecutores que en base a dicha información envían los trabajos a aquellas infraestructuras Grid para las que han sido diseñados. En MAGICGrid se ha optado por hacer uso de una base de datos similar a las propuestas anteriores, pero utilizando, al igual que ATLAS, un metaplanificador (Gridway) para evitar tener que volver a implementar toda la lógica de gestión de trabajos dentro del Grid.

El problema del envío de simulaciones Montecarlo al Grid no termina con la definición de la arquitectura del sistema. Uno de los problemas a la hora de crear este tipo de sistemas distribuidos, es ofrecer al usuario, por lo general no familiarizado con dichos entornos, un punto de entrada único y sencillo de utilizar, a través del cual pueda configurar sus trabajos y recibir sus datos sin necesidad de realizar ninguna interacción con los sistemas del Grid. Existen varias opciones para este fin. Las más conocidas son GridPort [17], que evita al usuario tener que programar a bajo nivel funciones básicas como el envío de trabajos, añadiendo además la posibilidad de que la autentificación del usuario vía certificados pueda ser llevada a cabo directamente desde la aplicación desarrollada por el usuario. Esta herramienta ha sido desarrollada pensando en la creación de portales web (Gridport es una aplicación web J2EE) y puede ser accedida directamente vía API de Java o bien mediante servicios web. Esta misma filosofía de ofrecer acceso a los servicios del Grid vía servicios web es la que implementa el proyecto PROGRESS [18], que apuesta por la creación de un módulo (Grid Service Provider) que se sitúe entre los servicios del Grid y la aplicación GUI a la que tiene acceso el usuario, eliminando de ese modo la necesidad de duplicar de forma innecesaria la lógica de acceso a los servicios Grid. Migrating Desktop [19] por su parte da un paso más allá eliminando la necesidad de construir esa GUI adaptada a las necesidades concretas del usuario, ofreciendo un entorno gráfico, que incluye toda la funcionalidad que se puede esperar de una interfaz de usuario de estas características, configurable según las necesidades del usuario, que interacciona con el Grid a través de un Roaming Access Server (RAS) que se encarga además de guardar las preferencias de éste.

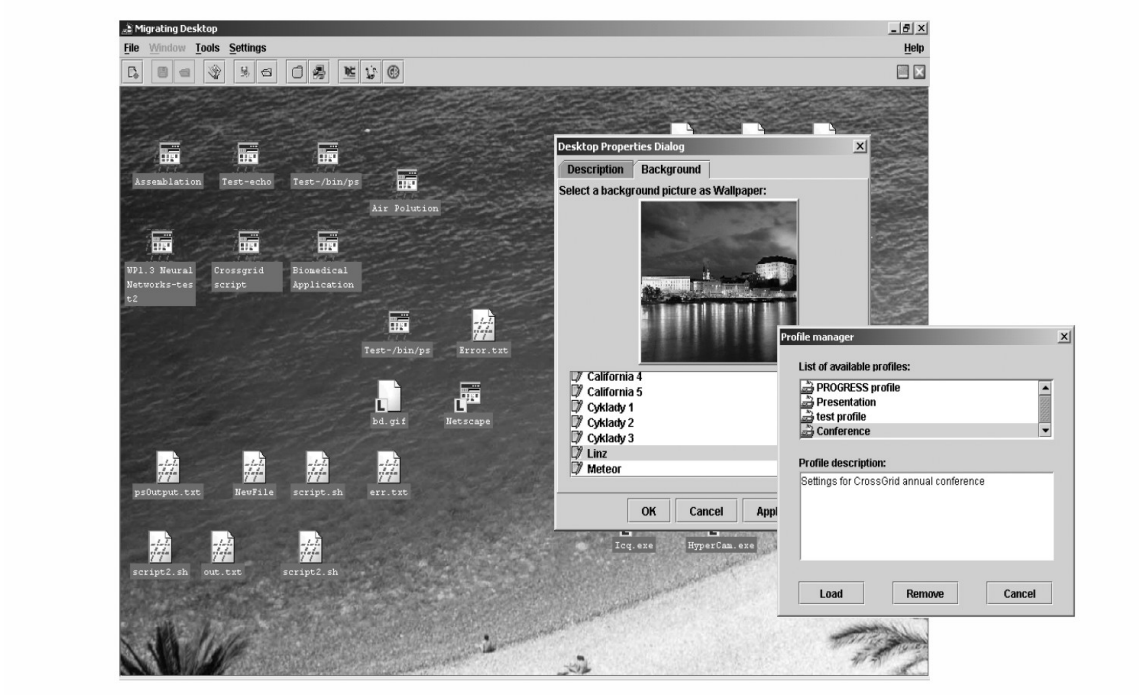

Figura 11. Ejemplo de uso del entorno Migrating Desktop

Aunque no tan cercana al desarrollo de interfaces para usuario, otra alternativa a citar en lo que al desarrollo de aplicaciones Grid se refiere es el Java Commodity Grid Kit [20], que implementa en Java la funcionalidad ofrecida por Globus, entre otros un sistema de información compatible con Globus MDS o la posibilidad de gestionar certificados a través de un servidor MyProxy.

Capítulo 3. La organización virtual: puesta en marcha

Parte del objetivo de este trabajo ha sido la reactivación de la organización virtual de MAGIC dentro del proyecto EGEE, en este capítulo se detallan las características de este tipo de organizaciones, así como las acciones llevadas a cabo.

Organizaciones virtuales (VO) en el proyecto EGEE

Como se comentó en el primer capítulo, las colaboraciones científicas a fin de utilizar el Grid suelen agruparse en lo que se conoce como organizaciones virtuales. Estas organizaciones por lo general engloban tanto a usuarios como recursos, proporcionando a los primeros un entorno distribuido de computación adaptado a necesidades concretas dentro de su proyecto o colaboración. Por ejemplo, la instalación del software necesario para llevar a cabo análisis de datos, o la transferencia directa de datos desde el instrumento científico a los elementos de almacenamiento asociados a la organización virtual dentro del Grid. Uno de los mayores problemas a los que se enfrenta una colaboración cuando pone en marcha una organización virtual es mantener una monitorización correcta de los recursos asociados a la misma, para que no sólo la disponibilidad, sino la correcta configuración y el adecuado funcionamiento de los elementos que forman parte de esta no interfieran con la experiencia del usuario. Por poner un ejemplo de esta situación, tras la experiencia poniendo en marcha de nuevo la organización virtual de MAGIC, ha sido habitual encontrarse con configuraciones que debido a la continua evolución del middleware Glite, han hecho que determinados servicios cambiasen de puerto, teniendo que modificar las herramientas desarrolladas convenientemente. Aún queda por alcanzar cierta homogeneidad de configuraciones que no de recursos dentro del Grid.

Autentificación dentro de las VO

Una organización virtual cuenta con un gran número de elementos asociados a ella, los más conocidos son los computing elements (CEs) y los storage elements (SEs). A estos se añade un buen número de servicios intermedios que hacen posible el envío de trabajos. Uno de los servicios más importantes de cara a poner en marcha toda la parte administrativa de la organización virtual es la autentificación de los usuarios y nodos. Esta se lleva a cabo mediante la gestión de certificados. EGEE utiliza certificados X.509 para llevar a cabo la autentificación de sus usuarios dentro de las organizaciones virtuales. En Europa estos certificados se obtienen a través de los organismos nacionales incluidos dentro de la iniciativa EUGridPMA [21]. En España estos certificados los emite la entidad pkIRISGrid a través de sus múltiples entidades de registro en universidades y centros de investigación. Estos certificados no sólo se limitan a personas físicas sino también a máquinas. Para realizar MAGICGrid se ha utilizado el certificado personal del autor de este

trabajo y uno expedido para la máquina desde la que se lanzan al Grid los trabajos asociados al proyecto. A partir de estos certificados se crean los proxies, una versión exportable del certificado del usuario, que como se verá a continuación, serán los que autentifiquen al usuario a través de los distintos elementos del Grid, todo esto en base a los permisos que este usuario (y su certificado asociado) tengan en el VOMS (Virtual Organization Membership Service). El VOMS [22] es un sistema de gestión de permisos que se describe a continuación.

El VOMS permite por un lado la consulta por parte de los distintos elementos de la organización virtual, de cuales son los permisos de cada usuario, a que grupos pertenece dentro de la organización y que roles desempeña dentro de ella, y por otro la modificación de los mismos a mano de los administradores de la organización (VO managers). Estos certificados, en el caso de los VO managers, no sólo servirán para operar dentro de la organización virtual, ya sea por ejemplo enviando trabajos o instalando software, sino que también le permitirá el acceso a determinados servicios cerrados al público en general. Estos servicios, están generalmente basados en una interfaz web que facilita su uso, como pueden ser el CIC Portal, donde se almacena toda la información necesaria para la gestión diaria de la organización virtual, o la interfaz de VO manager del servidor VOMS, donde es posible gestionar a los usuarios del VOMS, así como el perfil de estos.

Gestión de las VOs y roles

Por lo general, la configuración típica de las organizaciones virtuales que utilizan el middleware EGEE suele habilitar una user interface como punto de entrada a toda la organización virtual, de forma que cada institución que haga un uso extensivo del mismo, suele poner en marcha su propia user interface. Como se ha visto en el capítulo 2, esto no tiene porque ser algo a tener en cuenta a la hora de crear aplicaciones tan específicas como la producción de simulaciones Montecarlo, donde la configuración basada en una user interface es sustituida muchas veces por otras alternativas, que permiten interaccionar directamente con el Grid.

Respecto de la gestión diaria de la organización virtual, mucha de esta gestión se ha ido simplificando con los sucesivos proyectos EGEE, recayendo todo lo referente a operaciones a los propios administradores de los centros que proporcionan recursos a EGEE. La mayor parte del trabajo realizado por el VO manager es administrativo, debiendo por un lado gestionar las altas y bajas de la organización virtual, y por otro lado manteniendo actualizada la información disponible en la VO Card de la organización virtual. En dicha VO Card, disponible en el caso de EGEE en el anteriormente citado CIC Portal, se incluye información para la implementación por parte de los correspondientes administradores en los distintos recursos asociados a la organización virtual. Aunque no suele ser la regla general, estos administradores pueden excluir a un usuario autorizado dentro de la organización virtual a utilizar un conjunto de recursos. Dentro de la anterior VO Card, además de la información de contacto de los VO manager e información sobre el servidor VOMS, también se incluyen los roles y grupos que esta organización soporta. Estos roles

permiten fijar una serie de permisos en base al perfil del usuario, existiendo por lo general en el caso de los sistemas de producción de simulaciones Montecarlo dos roles, uno dedicado a la instalación de software y otro a la producción propiamente dicha. De cualquier modo, aparte de estos usuarios también se crea un rol llamado lcgadmin que sirve de alias al rol dedicado a la instalación de software. Estos roles son asumidos a través del sistema VOMS por usuarios al crear sus correspondientes proxies pudiendo así operar en base a estos permisos. Al llegar las peticiones de estos usuarios a los recursos concretos, el rol se mapea a un usuario local en la máquina, por ejemplo un usuario UNIX, que tendrá los permisos adecuados a la función que debe desempeñar ese rol. Así por ejemplo para el rol lcgadmin, dicho rol se mapeará a un usuario que tenga permisos para instalar software en un directorio determinado, a donde apuntará la variable de entorno *VO_NOMBREDELAVO_SW_DIR* , que los usuarios del rol de producción podrán utilizar con el fin de llamar al programa para correr las simulaciones Monte Carlo. De forma muy similar funcionan los grupos, pensados para virtualizar la distribución en grupos de trabajo dentro de las colaboraciones. Dentro de un grupo pueden existir distintos roles.

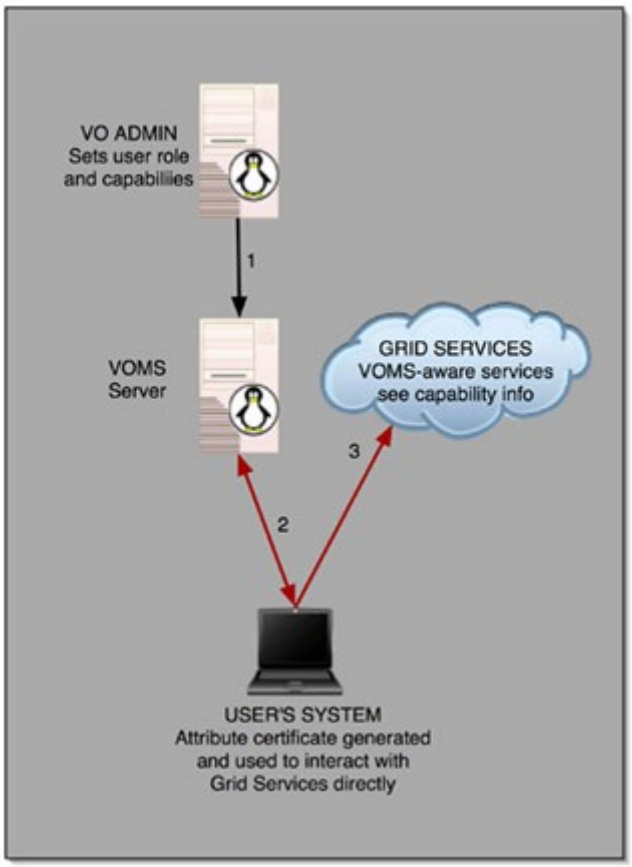

Figura 12. Funcionamiento del servicio VOMS

Aunque la gestión que debe realizar el VO manager se limita más a gestionar los usuarios asociados a su organización virtual, existen algunas herramientas útiles para comprobar que los recursos asociados a la misma están funcionando correctamente, es el caso del comando *lcg-infosites*, el cual permite monitorizar cuales son los recursos asociados a la

organización dentro de EGEE y cual es el estado de cada uno de ellos. Por ejemplo en el caso de los CEs, el número de trabajos activos y en espera, o las CPUs asociadas a los mismos.

La organización virtual MAGIC

La colaboración MAGIC solicitó en Septiembre de 2004 su reconocimiento como aplicación EGEE, siéndole reconocido tal estatus y poniendo en marcha su organización virtual. Los primeros pasos que se dieron fueron orientados a poner en marcha la producción de simulaciones Montecarlo dentro de la colaboración, para lo cual se llevaron a cabo los pasos detallados en la sección Estado del proyecto MAGICGrid del segundo capítulo. En Diciembre de 2007 se ha retomado el trabajo de administración y puesta a punto de esta organización, que a día de hoy (verano 2008) cuenta con aproximadamente unos 45 CEs y los correspondientes SEs, asociados a catorce instituciones, son las que se detallan a continuación.

- Port d'Informació Científica (España)
- Istituto Nazionale di Fisica Nucleare (Italia)
- Rutherford Appleton Laboratory (Reino Unido)
- HellasGrid (Grecia)
- Lancaster University (Reino Unido)
- Poznaskie Centrum Superkomputerowo-Sieciowe (Polonia)
- TU Dortmund (Alemania)
- Consiglio Nazionale delle Ricerche (Italia)
- GRNET (Grecia)
- Liverpool University (Reino Unido)
- Scuola Normale Superiore di Pisa (Italia)
- Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (España)
- Rechenzentrum Garching – Max Planck (Alemania)
- Institute for Parallel Processing (Bulgaria)

Debido a la falta de uso de esta organización virtual a la hora de asumir la tarea de relanzar el funcionamiento de dicha organización, muchos de los servicios asociados a la misma no estaban al día. Es por ello que para llevar a cabo el presente proyecto, se optó por poner en marcha parte de la organización virtual, de forma que se pudiera garantizar el funcionamiento de los servicios implicados. Para ello se optó por una solución similar a la planteada en la primera fase de este proyecto en 2004, fijando tres centros (PIC, CIEMAT y TU Dortmund) como los proveedores de recursos, tanto computacionales como de almacenamiento, y poniendo en marcha sendas user interfaces en el PIC y CIEMAT, tal como muestra la figura 13. La complejidad del entorno Grid, supuso la necesidad de un trabajo de configuración intensivo en los distintos elementos de la organización virtual, y en aquellos que dentro de los tres centros citados daban soporte a la organización, por parte tanto de los administradores de los centros como del autor. Para toda la administración de la organización virtual que se detalla en los siguientes capítulos se ha

utilizado como resource broker (rb01.pic.es) y como WMS (https://prod-wms-01.pd.infn.it:7443/glite_wms_wmproxy_server).

Envío de trabajos en EGEE

Antes de comenzar a detallar el contenido de los trabajos enviados al Grid para realizar la instalación del software de MAGIC y la ejecución de las simulaciones Montecarlo sobre ese software, se va a abordar el formato de envío de los trabajos al Grid. En las dos user interfaces con las que se ha trabajado en este proyecto, se instalaron varias versiones de middleware, utilizando para las pruebas principalmente Glite y EDG. Estas distribuciones proporcionan al usuario un conjunto de comandos mediante los cuales puede gestionar los trabajos y los proxies. Para utilizar una user interface es necesaria la creación de una cuenta en el equipo donde el software de esta está instalada. En principio los comandos son similares a los de cualquier sistema de gestión de trabajos que corriera sobre un cluster. Existen comandos para:

- envío de trabajos
- cancelación de trabajos
- consulta del estado de trabajos utilizando su identificador global
- descarga a la user interface de los resultados del trabajo

Según se use una u otra distribución, también se incluyen comandos para enviar datos a SEs mediante GridFTP, o para gestionar los proxies.

Al ser enviado un trabajo al Grid mediante el comando correspondiente haciendo uso del WMS, a éste le es asignado un identificador global, en lo que se refiere al conjunto del Grid. Dicho identificador es luego utilizado para conocer cual es el estado del trabajo o de los datos asociados al mismo. En el siguiente ejemplo se hace uso del comando grid-job-submit para enviar un trabajo a uno de los CEs de la TU Dortmund, seleccionado previamente de entre los CEs devueltos por el comando lcg-infosites. Tras conectarse al WMS correspondiente, definido o bien globalmente por el administrador del user interface o bien por el usuario en un fichero de configuración glite_wms.conf, para enviar el trabajo, éste devolverá un identificador global, que en este caso concreto es `https://prod-lb-01.pd.infn.it:9000/4dLvoJOd3RCsTN-vca0zQw`

Esquema

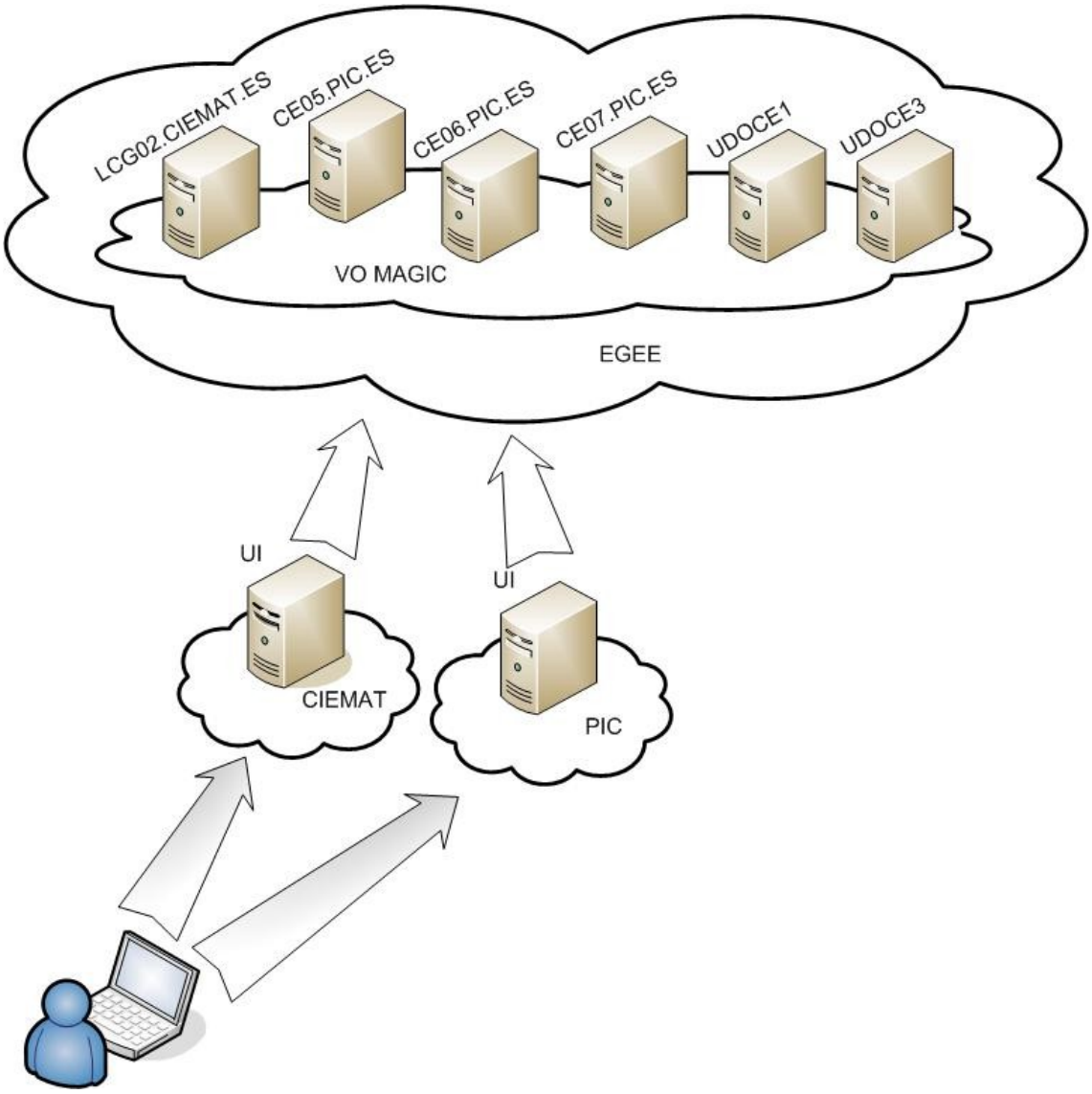

Figura 13. Estado actual de la organización virtual MAGIC. Se detallan los CEs utilizados para el desarrollo y la realización de pruebas del proyecto MAGICGrid

```
[avazquez@egeeui01 ~]$ glite-wms-job-submit -a --config glite_wms.conf -r udo-
ce03.grid.tu-dortmund.de:2119/jobmanager-lcgpbs-magic corsikaLS.jdl

Connecting to the service https://prod-wms-01.pd.infn.it:
7443/glite_wms_wmproxy_server


====================== glite-wms-job-submit Success ==================

The job has been successfully submitted to the WMProxy
Your job identifier is:

https://prod-lb-01.pd.infn.it:9000/4dLvoJOd3RCsTN-vca0zQw
```

Una vez enviado el trabajo sólo queda esperar su finalización. El estado del trabajo puede ser comprobado con la ayuda del comando glite-wms-job-status, como se muestra en el siguiente ejemplo. Este comando indica el estado del trabajo que ya se encuentra en fase de ejecución en el CE correspondiente. El comando devuelve un código de error en el caso de que lo hubiera habido o cero en el caso contrario.

```
[avazquez@egeeui01 ~]$ glite-wms-job-status https://prod-lb-01.pd.infn.it:
9000/4dLvoJOd3RCsTN-vca0zQw

**************************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://prod-lb-01.pd.infn.it:9000/4dLvoJOd3RCsTN-
vca0zQw
Current Status:     Running
Status Reason:      Job successfully submitted to Globus
Destination:        udo-ce03.grid.tu-dortmund.de:2119/jobmanager-lcgpbs-magic
Submitted:          Sun Jul 20 20:26:32 2008 CEST
**************************************************************
```

Identificación

Para poder enviar estos trabajos primero es necesario previamente llevar a cabo todo el proceso de autentificación dentro del Grid. Existen esencialmente dos formas de llevar esto, aquellos casos donde se necesite que el usuario asuma un rol y aquellos en los que no. Para este último caso un proxy generado con cualquiera de los comandos de proxy de Glite, EDG o incluso el propio Globus, mediante el comando grid-proxy-init, servirá. En el primer caso sin embargo, será necesario hacer uso de los comandos correspondientes al sistema VOMS, voms-proxy-*, a los cuales se les puede especificar tanto el grupo como el rol que asume el usuario que está generando el proxy. Sea cual sea el tipo de proxy necesario, primero habrá que colocar en el directorio home del usuario su certificado, en concreto la versión .PEM (se puede convertir de p12 a pem utilizando un script disponible

en las webs de la mayoría de autoridades de registro nacionales) de su certificado .P12 que será el formato en el que se le proporcione a través de la interfaz web dicha autoridad de certificación. Así para crear el proxy correspondiente se procedería de con el comando voms-proxy-init de la forma que se indica más abajo, en el caso de querer generar un proxy dentro de la organización virtual MAGIC asumiendo el rol de lcgadmin. Toda organización virtual dentro de EGEE tiene un fichero vomses que la identifica, el cual por lo general se mantiene por los administradores de aquellos recursos que soportan dicha organización.

```
[avazquez@egeeui01 ~]$ voms-proxy-init -voms magic:/Role=lcgadmin
Cannot find file or dir: /home/avazquez/.glite/vomses
Enter GRID pass phrase:
Your identity: /DC=es/DC=irisgrid/O=ucm/CN=Adolfo-Vazquez
Creating temporary proxy ..................................... Done
Contacting  voms.grid.sara.nl:30015
[/O=dutchgrid/O=hosts/OU=sara.nl/CN=voms.grid.sara.nl] "magic" Done
Creating proxy ................................................ Done
Your proxy is valid until Mon Jul 21 08:49:46 2008
```

El tiempo de validez del proxy, que se muestra al final del anterior ejemplo será uno de los puntos a tener en cuenta en MAGICGrid, ya que el sistema se supone 24x7 por lo que será necesario fijar explícitamente al crear el proxy un tiempo de duración del mismo adecuadamente largo.

Envío de trabajos

A la hora de enviar trabajos, se veía en el ejemplo anterior como el fichero de entrada correspondía a un fichero con extensión JDL. Actualmente existen varios formatos soportados para el envío de trabajos como JDL o JSDL, este último un lenguaje XML de definición de trabajos, a los que hay que añadir aquellos proporcionados por metaplanificadores como Gridway. El Job Definition Language (JDL) es el más utilizado, siendo además por el que se ha optado para definir los trabajos de MAGICGrid. A pesar de que incluye opciones más avanzadas, como por ejemplo la definición de características que tienen que tener los recursos a utilizar, en el caso de un CE que tenga un software y un sistema operativo determinados, o que su hardware reúna unas características dadas, los trabajos que van a ser definidos utilizarán una plantilla similar a la siguiente.

```
Executable="scriptBUILDINSTALL6500";
StdError="stderror.log";
StdOutput="stdoutput.log";
InputSandbox = {"scriptBUILDINSTALL6500"};
OutputSandbox={"stderror.log", "stdoutput.log"};
```

Para hacer más flexible el envío de comandos a través de JDL se ha optado por enviar un script con los comandos a ejecutar, en vez de varios ficheros JDL con las distintas órdenes a ejecutar (los argumentos se podrían añadir mediante la sección Arguments). El trabajo además cuenta con dos secciones para definir los ficheros a los que se redireccionará la salida estándar y la salida estándar de error. Dichos ficheros se obtendrán de vuelta una vez terminado el trabajo, por ejemplo en el caso del middleware Glite mediante el

comando glite-wms-job-output. Para ello es necesario situarlos dentro de la sección OutputSandbox, de la misma forma que para enviar el fichero que contiene el script a ejecutar se sitúa este en la sección InputSandbox de la definición JDL.

Despliegue del software de MAGIC

El anterior ejemplo con el que se ilustraba el uso de JDL corresponde en realidad al JDL empleado para la instalación del software de simulación Montecarlo (Corsika) en los distintos CEs para MAGICGrid. Existen dos caminos a la hora de instalar software, bien enviar el software dentro del propio trabajo, o bien lanzar una serie de trabajos que instalen y comprueben la correcta instalación de dicho software. El primer caso es más adecuado si el software no tiene un uso habitual, de forma que sólo esté disponible para esa sesión. Esto evita tener que mandar trabajos adicionales para instalar y actualizar el software de simulación. Debido a la naturaleza del proyecto se ha optado por la segunda solución. Para ello se ha descargado el software de simulación Monte Carlo directamente desde la página web del Grupo de Altas Energías, compilándolo directamente desde los ficheros fuente e instalándolo en el directorio habilitado para la organización virtual MAGIC para tal fin en los CEs. Estos pasos se pueden ver en el siguiente script.

```
#!/bin/bash
rm -rf $VO_MAGIC_SW_DIR/Mmcs6500
rm -f MMcs
pwd
wget http://www.gae.ucm.es/~avazquez/Mmcs6500.tar.gz
/bin/tar zxvf Mmcs6500.tar.gz -C $VO_MAGIC_SW_DIR
cd $VO_MAGIC_SW_DIR/Mmcs6500/
export FLUPRO=$VO_MAGIC_SW_DIR/Mmcs6500/fluka2006_3
make all SYSTEM=LINUX
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/ $VO_MAGIC_SW_DIR/MMcs
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/cc6501p-linux
$VO_MAGIC_SW_DIR/Mmcs6500/corsika
ls $VO_MAGIC_SW_DIR
ls $VO_MAGIC_SW_DIR/Mmcs6500
```

Se ha definido una variable de entorno de nombre FLUPRO para indicar la localización de la librería FLUKA necesaria para esta versión de Corsika para el proyecto MAGIC. Aunque estos métodos de instalación, junto con el ejercido por los administradores de manera local, son los más comunes, en realidad el proyecto LCG puso en marcha un sistema de instalación automática de software basado en dos comandos lcg-ManageSoftware y lcg-ManageVOTag. El primero permite enviar directamente al CE el software comprimido en un formato determinado de forma que este se instale siguiendo los parámetros indicados, añadiendo a su vez una serie de etiquetas que pasaran a formar parte de la información que proporcione el CE sobre si mismo. Estas etiquetas pueden ser a su vez gestionadas directamente a través del segundo comando. Este sistema fue descartado para el proyecto MAGICGrid debido a la falta de soporte en los CEs participantes a estas herramientas.

Prototipo para el envío de simulaciones Montecarlo

En una primera aproximación al envío de simulaciones Montecarlo al Grid, se optó por enviar estas del mismo modo que se instaló el software en remoto, haciendo uso de un script (detallado más adelante). Para ello se definió el siguiente JDL, que incluía con respecto al visto anteriormente, los ficheros de salida de datos de Corsika en la OutputSandbox y el fichero de configuración de Corsika en la InputSandbox.

```
Executable="scriptSIM";
MyProxyServer = "rb01.pic.es:7772";
StdInput="inputcard.card";
StdError="stderror.log";
StdOutput="stdoutput.log";
InputSandbox = {"inputcard.card", "scriptSIM"};
OutputSandbox={"stderror.log", "stdoutput.log", "dat000001",
"cer000001"};
```

Este JDL ejecuta un script que se encarga de enlazar en el directorio de trabajo de la sesión en la que el usuario se encuentre, aquellos ficheros que Corsika necesita para llevar a cabo las simulaciones. Una vez definidas localmente estas librerías se llama a Corsika, ubicado en el directorio de software para la organización virtual definido dentro del CE, introduciendo por la entrada estándar el fichero de configuración que define las simulaciones.

```
#!/bin/bash
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/NUCNUCCS $PWD/NUCNUCCS
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/NUCLEAR.BIN $PWD/NUCLEAR.BIN
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/QGSDAT01 $PWD/QGSDAT01
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/SECTNU $PWD/SECTNU
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof1.dat $PWD/atmprof1.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof2.dat $PWD/atmprof2.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof3.dat $PWD/atmprof3.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof4.dat $PWD/atmprof4.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof5.dat $PWD/atmprof5.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof6.dat $PWD/atmprof6.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof9.dat $PWD/atmprof9.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof11.dat $PWD/atmprof11.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/atmprof12.dat $PWD/atmprof12.dat
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/EGSDAT5_.05 $PWD/EGSDAT5_.05
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/EGSDAT5_1. $PWD/EGSDAT5_1.
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/EGSDAT5_.15 $PWD/EGSDAT5_.15
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/EGSDAT5_.25 $PWD/EGSDAT5_.25
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/EGSDAT5_3. $PWD/EGSDAT5_3.
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/EGSDAT5_.4 $PWD/EGSDAT5_.4
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/qgsdat-II-03 $PWD/qgsdat-II-03
ln -s $VO_MAGIC_SW_DIR/Mmcs6500/sectnu-II-03 $PWD/sectnu-II-03
export FLUPRO=$VO_MAGIC_SW_DIR/Mmcs6500/fluka2006_3
$VO_MAGIC_SW_DIR/Mmcs6500/corsika < inputcard.card
```

Corsika dispone de un gran número de parámetros para la simulación de cascadas,

pudiendo definir desde el número y tipo de partículas que indicen en la misma, hasta la energía de las mismas. A continuación se muestra un ejemplo de un fichero de configuración (inputcard) de Corsika para correr simulaciones Montecarlo donde se definen valores concretos correspondientes a los que se utilizan para el telescopio MAGIC. Más información sobre Corsika y las opciones de configuración del mismo en [23]. La introducción de estos parámetros se ha automatizado con MAGICGrid, ofreciendo la posibilidad de introducirlos a través de una interfaz gráfica.

```
PRMPAR   1       particle type
ERANGE  200 600         energy range
EVTNR   1       number of first shower event
RUNNR       44
NSHOW   1       number of showers to generate
ESLOPE  -2      slope of primary energy spectrum
THETAP  0. 0.   range of zenith angle (degree)
PHIP    90. 90.         range of azimuth angle (degree)
SEED     4401 2 0       Changed by AWK
SEED     4402 2 0       Changed by AWK
SEED     4403 2 0       Changed by AWK
DIRECT  ./
OBSLEV  2200.E2         observation level (in cm)
ELMFLG  F T     em. interaction flags (NKG,EGS)
RADNKG  200.E2  outer radius for NKG lat.dens.determ.
ARRANG  0.      rotation of array to north
FIXHEI  0. 0    first interaction height target
FIXCHI  0.      starting altitude (g/cm**2)
MAGNET  29.5 23.0       magnetic field at LaPalma
HADFLG  0 0 0 0 0 0     flags for hadr. interaction
ECUTS   0.3 0.3 0.02 0.02       e.cuts: had, mu, elec y fot
MUADDI  F       additional info for muons
MUMULT  T       muon multiple scattering angle
LONGI   T 10. T T       longit.distr. step size fit
MAXPRT  0       max. number of printed events
ECTMAP  1.E4    cut on gamma factor for printout
STEPFC  0.1     mult. scattering step length fact.
DEBUG   F 6 F 1000000   debug flag and log.unit for out
CWAVLG  290. 600.       Cherenkov wavelength band
CSCAT   1 0. 20000.     scatter Cherenkov events
CERSIZ  1.      bunch size Cherenkov photons
CERFIL  T       Cherenkov output to extra file
CERTEL  1
        0. 0. 0. 0. 0. 2000. 1700.      Location and size of each CT
EXIT
```

Con el fin de desarrollar un primer prototipo que implementara el envío de varios trabajos al Grid se han desarrollado un par de scripts, basados en awk y en los comandos del middleware Glite, a modo de prueba para el envío y recepción de los trabajos asociados a las simulaciones Montecarlo.

Instalación de nodo Grid basado en Globus/Gridway

La mayoría de los proyectos de producción de simulaciones Grid, como se ha visto en el segundo capítulo, basan su arquitectura en una interfaz de usuario o en agentes software que se encargan de gestionar los trabajos adaptando su gestión a la infraestructura Grid para la que han sido diseñados. En el caso de MAGICGrid se ha optado por enviar los trabajos al Grid a través de un metaplanificador incluido dentro de la distribución de la Globus Toolkit, Gridway[24]. Este metaplanificador incluye soporte tanto para servicios Grid basados en servicios web como para la generación anterior de estos. Asimismo incluye una API tanto para C como para Java, esta última utilizada dentro de MAGICGrid. Para lanzar trabajos se ha realizado, en el mismo equipo desde donde se corre la parte del servidor de la aplicación MAGICGrid, una instalación de Globus básica con soporte para servicios Grid pre-servicios web. Sobre esta instalación se ha instalado el metaplanificador Gridway, configurado igualmente para dar soporte a servicios Grid pre-servicios web. Esta configuración, cuya arquitectura se muestra en la figura 14, permite enviar los trabajos de forma directa a los CEs, reduciendo el tiempo de espera debido a la gestión de los mismos en elementos como el WMS. Para adaptar Gridway a esta configuración se han tenido que modificar el script gw_im_mad_mds2_glue, incluido dentro del directorio bin de la distribución de Gridway, modificando éste de forma que la variable BDII_PORTNUMBER tomará en vez del valor por defecto (el puerto 2135), el nuevo puerto por el que Glite escucha al servicio de información de los CEs, el 2170. Para ello se ha introducido la siguiente modificación en el código de dicho script.

```
set_filter_env()
{
    if [ "$INFSERVER" == "lcg02.ciemat.es" ] || [ "$INFSERVER" == "udo-
ce03.grid.tu-dortmund.de" ]
   then
      BDII_PORTNUMBER=2170
      PORTNUMBER=$BDII_PORTNUMBER
      BASE_BDII="mds-vo-name=local,o=grid"
      BASE_CE="mds-vo-name=resource,o=grid"
   else
      BDII_PORTNUMBER=2135
      PORTNUMBER=$BDII_PORTNUMBER
      BASE_BDII="mds-vo-name=local,o=grid"
      BASE_CE=$BASE_BDII
   fi
}
```

Esta solución permite definir el nuevo puerto para un número determinado de CEs, aunque en un futuro esta configuración se podría parsear directamente desde la salida de un servidor BDII, por ejemplo bdii.pic.es. Junto con esta modificación se ha incluido la siguiente configuración pre-servicios web en los tres MADs que incluye Gridway.

```
IM_MAD = mds2_glue:gw_im_mad_mds2_glue:-l etc/host.list -q
(GlueCEAccessControlBaseRule=*magic):dummy:prews_nsh
EM_MAD = prews_nsh:gw_em_mad_prews:-p 2170:rsl_nsh
```

```
TM_MAD = dummy:gw_tm_mad_dummy:-g
```

De esta forma se envían los trabajos únicamente a aquellos CEs incluidos en la lista de CEs definida (etc/host.list) que pertenezcan a la organización virtual MAGIC. Para enviar trabajos a través de esta instalación de Gridway a la organización virtual MAGIC, al estar esta basada en VOMS, es necesario crear un proxy que contenga las extensiones necesarias para contener información sobre el rol al que el usuario está autorizado. Para ello hay que crear el proxy en una de las user interfaces habilitadas, copiándolo al servidor con la instalación Globus/Gridway con el nombre /tmp/x509_x, siendo x509_x el nombre del certificado que el usuario tiene asignado. Este paso es necesario llevarlo a cabo en las user interfaces, debido a que la distribución Globus instalada no dispone de los comandos VOMS necesarios. Esta configuración es sencilla de crear y de portar a terceras máquinas. Al igual que en el caso de las user interfaces, se realizaron pruebas para el envío de trabajos utilizando Gridway a través de la línea de comandos, para posteriormente enviar estos haciendo uso de la API Java que proporciona dicha herramienta. Para ello se utilizaron los comandos habilitados por Gridway para la gestión de trabajos.

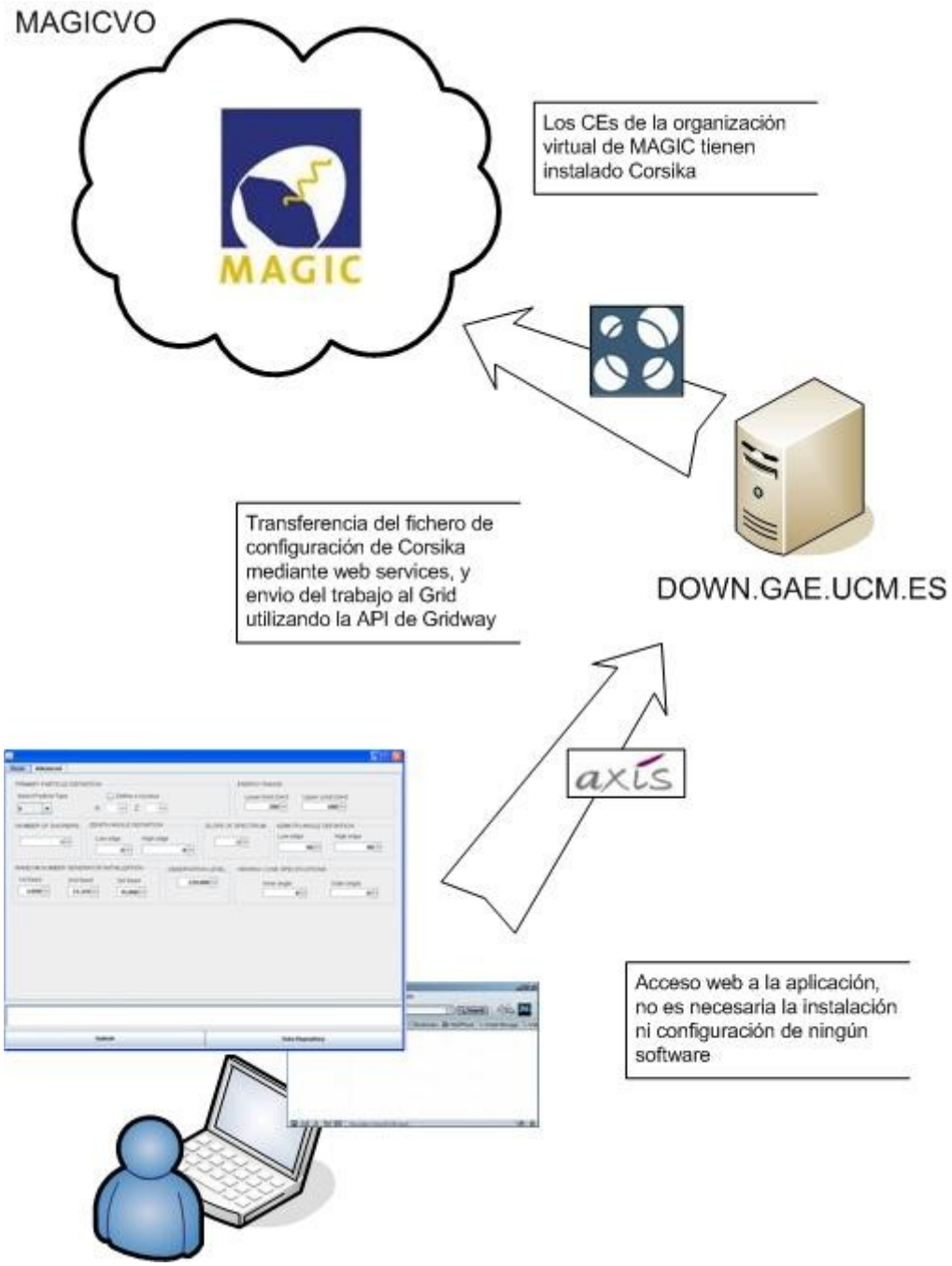

Figura 14. Configuración del nodo Grid para el envío de trabajos en MAGICGrid, basado en una instalación de Globus/Gridway

Capítulo 4. Proyecto MAGICGrid: diseño y desarrollo

Los objetivos que se han perseguido con MAGICGrid, han sido por un lado la gridificación de las simulaciones Montecarlo, portando estas al entorno Grid europeo, EGEE, y por otro ofrecer a los usuarios una interfaz sencilla de manejar sin tener que realizar ningún tipo de instalación de software en sus equipos. Para cubrir el primer objetivo se ha configurado, como se comentaba en anteriores capítulos, un nodo Grid basado en la Globus Toolkit, que hace uso del metaplanificador Gridway para lanzar dichos trabajos. Esto permite enviar los trabajos directamente a los CEs que se configuren dentro del metaplanificador, limitando el uso a aquellos que den un servicio aceptable. Para cubrir el segundo objetivo, se ha desarrollado una GUI en Java que mediante servicios web basados en Apache Axis, corriendo sobre un servidor Tomcat, se conecta con dicho servidor, haciendo uso de la tecnología Java Web Start, que viene incluida por defecto en el Java Runtime Environment. A continuación se va a detallar por un lado el diseño de la aplicación y por otro lado ofrecer algunas notas sobre la implementación de la misma, en especial en lo referente a las tecnologías utilizadas y su uso en el proyecto.

Arquitectura

Tal como se muestra en la Figura 15, el componente MGRID Server se ejecuta en un servidor y es accedido utilizando el componente MGridClient que los clientes tienen que instalar y que les ofrece una interfaz para utilizar el sistema. La estructura de clases de estos componentes se describe a continuación. Los clientes se comunican con el componente MGrid Server utilizando protocolos de Servicios Web, lo cual se describe posteriormente.

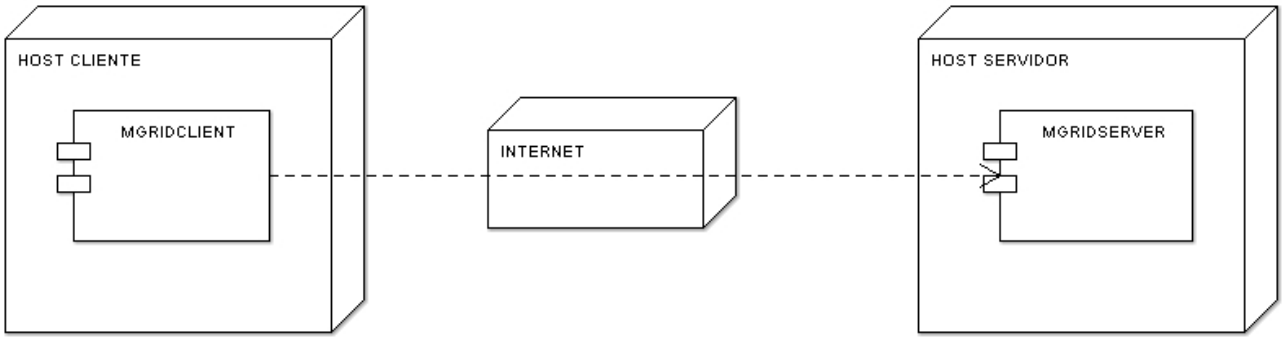

Figura 15. Diagrama UML de la Arquitectura de MAGICGrid

Estructura de los componentes

Estructura del cliente

Como se muestra en la figura 16, el cliente contenido en GUI utiliza los métodos de buildingXML para construir el fichero XML que contendrá los parámetros de configuración de Corsika que posteriormente serán enviados haciendo uso de las clases contenidas en el paquete WS, generado utilizando las utilidades de Apache Axis.

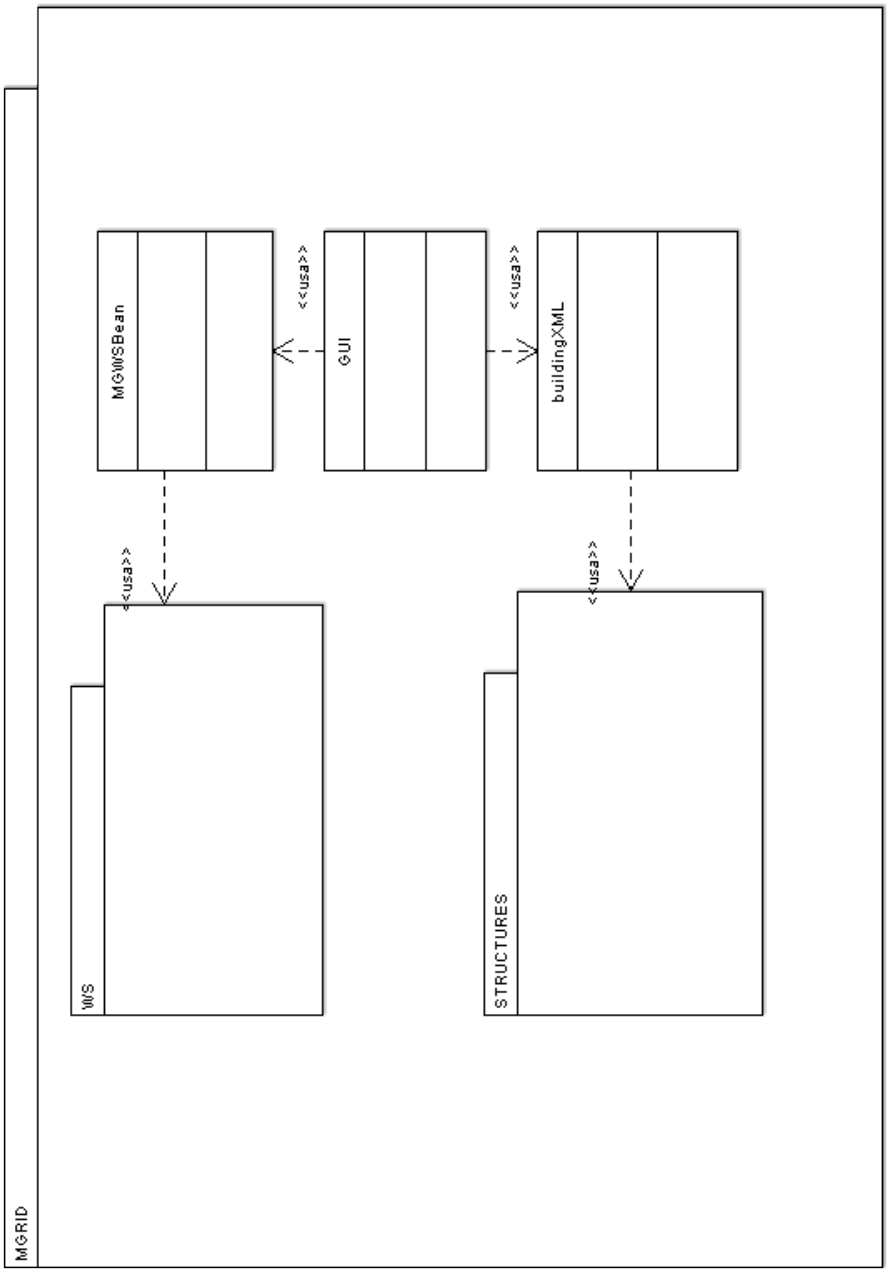

Figura 16. Diagrama de paquetes UML del cliente

Estructura del Servidor

Como se muestra en la figura 17, el servidor recibe el fichero XML con los parámetros de Corsika utilizando el paquete WS, que contiene aquellas clases e interfaces necesarias por Apache Axis para este fin. Una vez recibido este fichero, éste es parseado mediante los métodos contenidos en parseXML, haciendo uso de JDOM, siendo utilizado el resultado como entrada para el trabajo (JDL) que se configura en MagicGridImpl para enviarse mediante Gridway a la organización virtual MAGIC, haciendo uso de la Java API de dicha aplicación.

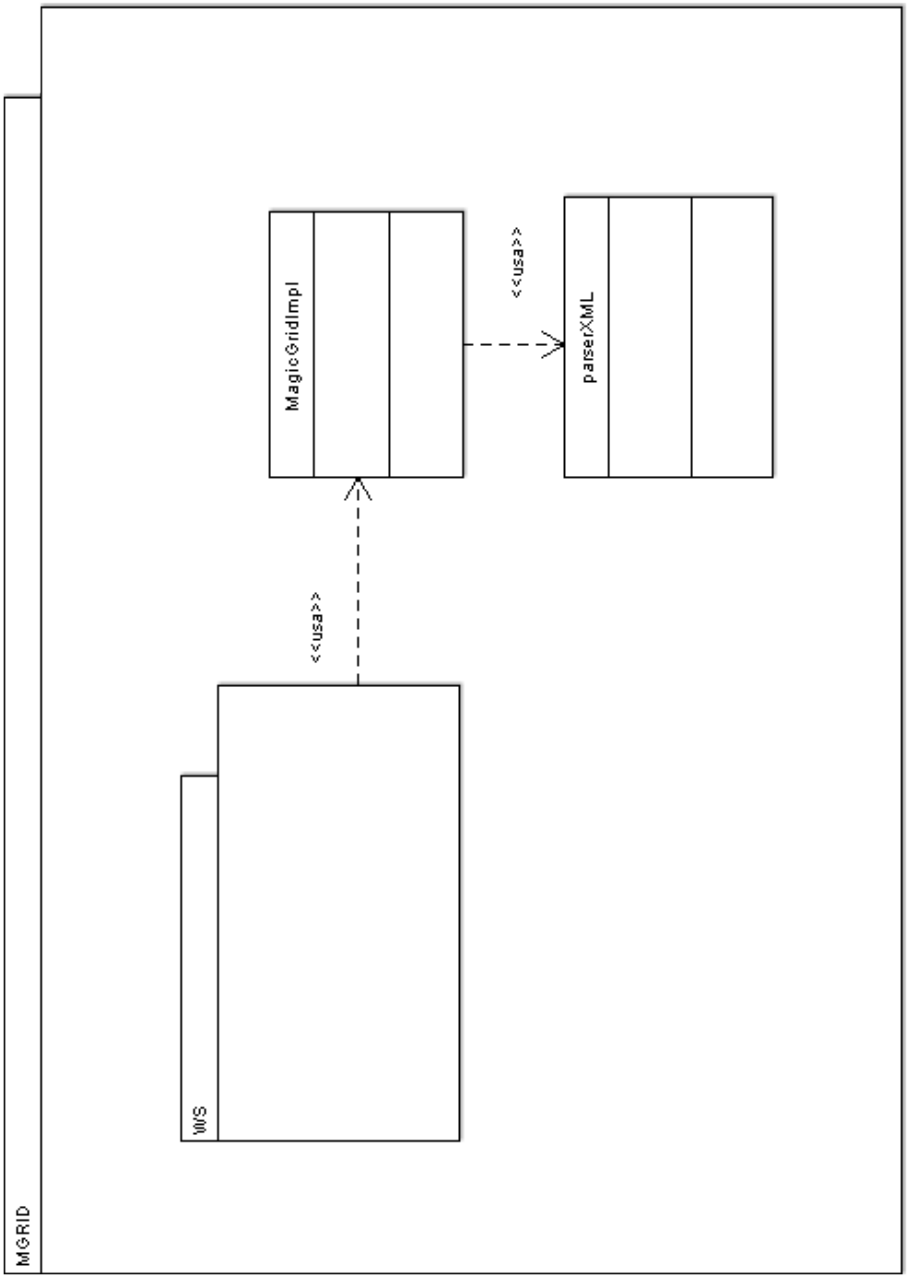

Figura 17. Diagrama de paquetes UML del servidor

Servicios web

Para llevar a cabo la comunicación entre el cliente y el servidor se ha utilizado la plataforma de servicios web que proporciona Apache Axis, ejecutado como una aplicación web dentro de Apache Tomcat. En un primer paso, para realizar las primeras pruebas de este software, el despliegue de los servicios web se realizó mediante el cambio de extensión del fichero que contenía aquella clase que implementaba el método que se mapeaba al servicio web. De esta forma esta extensión pasaba de ser java a jws. Con incluir este jws en el directorio raíz de la distribución de Axis, éste se encargaba de desplegarlo como un servicio web. Tras realizar un prototipo utilizando este método de despliegue, se pasó a utilizar un sistema de despliegue más flexible, utilizando las herramientas Java2WSDL y WSDL2Java. Mediante la primera de forma similar al primer método, se creaba un fichero WSDL a partir de una clase Java. Dicha clase Java contiene una serie de métodos, que se convertirán en servicios web, con la misma signatura de método que la indicada en esa clase. El fichero WSDL guarda información sobre la localización del servicio web (su URI), así como su espacio de nombres. Este comando, cuyo formato completo se incluye a continuación, generaba el fichero WSDL.

```
java org.apache.axis.wsdl.Java2WSDL -o MGrid.wsdl -l"URIdelservicioweb"-n
urn:MGrid -p"MGrid" urn:MGrid MGrid.MagicGrid
```

A su vez, este fichero WSDL se utilizaba para, mediante el comando WSDL2Java siguiente, generar las clases binding para el cliente y servidor, y dos ficheros Web Service Deployment Descriptor(WSDD), deploy.wsdd y undeploy.wsdd. Estos últimos se utilizan para desplegar el servicio en Axis, indicando como debe responder a las peticiones que llegan al servicio web con el que esos dos ficheros están asociados.

```
java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -p MGrid.ws
MGrid.wsdl
```

Una vez el fichero jar se encuentra en el directorio webapps/WEB-INF/lib, mediante el comando AdminClient y utilizando el fichero deploy.wsdd, se utiliza el siguiente comando para desplegar el servicio en Axis.

```
java org.apache.axis.client.AdminClient -hnombrehost -p8080 deploy.wsdd
```

Axis ofrece un gestor vía web, a través del cual se pueden comprobar qué servicios web están actualmente desplegados y su signatura. Para ello hay que entrar a la siguiente dirección, donde nombredelhost deberá ser sustituido por el nombre del equipo donde se aloja el Tomcat.

Tanto para el cliente como para el servidor se generaron las clases para realizar el binding de los servicios web, siendo necesario en el caso del servidor, modificar MGridSoapBindingImpl.java para que crease un objeto de la clase donde se encuentra la implementación de los métodos que invocan los servicios web (en este caso MagicGridImpl.java) de forma que la llamada a cada servicio se asociara a la llamada a cada uno de esos métodos. Una vez desplegados los servicios web en el servidor, en el cliente se puede invocar a un servicio de la siguiente manera, haciendo uso de las clases binding.

```
MagicGridService service = new MagicGridServiceLocator();
```

```
MagicGrid MGWS = service.getMGrid();
```

Pudiendo invocar los métodos de forma remota a través del objeto MGWS. Por ejemplo en este caso existía desplegado un método start(), que podría invocarse mediante `MGWS.start()`. También cabe resaltar la propagación de excepciones desde el servidor al cliente a través del servicio web.

Interfaz de Usuario

La interfaz de usuario se ha desarrollado con Java.Permite la configuración del fichero de entrada de Corsika, con las restricciones que para cada parámetro se indican en el manual de usuario de este programa. Estos parámetros se han agrupado en dos solapas, agrupando los que son de uso más común a la hora de hacer simulaciones en la pestaña *Basic*. La GUI cuenta también con un panel de texto donde se indica el estado del envío, o en el caso de surgir alguna excepción, la traza completa correspondiente a la misma. También se incluyen dos botones:

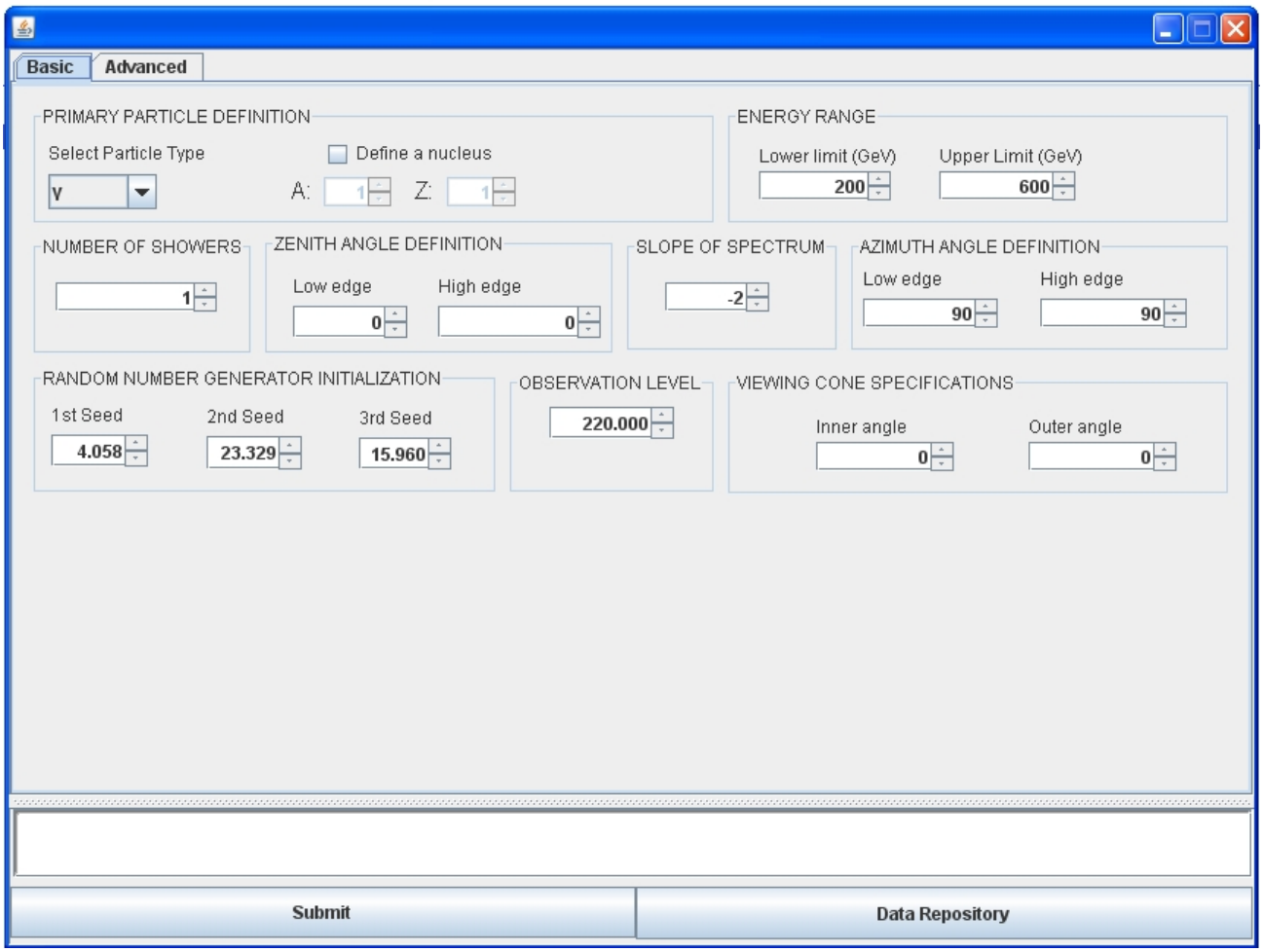

Figura 18. GUI Java del cliente de MAGICGrid

- El botón *Submit* el cual, como se ha descrito en la anterior sección, haciendo uso de

las clases de binding para los servicios web, hace una llamada al servicio web start, pasándole como argumento un fichero XML que contiene los valores de los parámetros introducidos a través de la GUI. Para crear este fichero, se ha creado un elemento XML para cada parámetro, incluyendo elementos XML anidados en los parámetros que contiene varios subparámetros.

▪El otro botón, *Data repository*, lanza el navegador web configurado por defecto en el sistema, abriendo el directorio donde se encuentra el repositorio de ficheros resultado de las simulaciones, con las extensiones cer y dat. Estos ficheros incluyen la fecha y la hora en la que fueron creados.

Este fichero se lanza a través de un fichero JNLP asociado, el cual permite cargar de forma remota aplicaciones Java utilizando la herramienta Java Web Start, que viene por defecto en el Java Development Kit. El fichero JNLP es similar al que se incluye a continuación, y en el se indican la información de configuración y recursos (archivos jar, con la completa de dónde se encuentra) necesarios para ejecutar la aplicación en remoto (cuya clase principal es *MGrid.gui*).

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+">
<information>
<title>MAGICGrid Client</title>
<vendor>INSA</vendor>
<homepage href="http://magicgrid.info" />
</information>
<offline-allowed/>
<security>
<all-permissions/>
</security>
<resources>
<j2se version="1.6+" />
<jar href="http://localhost/jnlp/MagicGridClient.jar"/>
<jar href="http://localhost/jnlp/axis.jar"/>
<jar href="http://localhost/jnlp/commons-discovery.jar"/>
<jar href="http://localhost/jnlp/commons-logging.jar"/>
<jar href="http://localhost/jnlp/commons-codec.jar"/>
<jar href="http://localhost/jnlp/activation.jar"/>
<jar href="http://localhost/jnlp/axis-ant.jar"/>
<jar href="http://localhost/jnlp/jaxrpc.jar"/>
<jar href="http://localhost/jnlp/log4j.jar"/>
<jar href="http://localhost/jnlp/mail.jar"/>
<jar href="http://localhost/jnlp/saaj.jar"/>
<jar href="http://localhost/jnlp/wsdl4j.jar"/>
</resources>
<application-desc main-class="MGrid.gui" />
</jnlp>
```

Dentro de los ficheros jar que se incluyen como recursos de la aplicación, está el que contiene a la aplicación en sí, en este caso MagicGridClient.jar, y todos aquellos jar que utiliza esta clase, en este caso aquellos relacionados con el uso de los servicios web en la

plataforma Apache Axis. Los ficheros jar deben ir firmados a través de la aplicación jarsigned incluida también en el JDK. En la mayoría de los casos estos ficheros suelen haber sido firmado previamente por el desarrollador del mismo. Firmar encima de dicha firma con la firma propia dará problemas a la hora de ejecutar la aplicación, por lo que habrá que eliminar dicha firma. Para poder hacer esto habrá que desempaquetar el fichero jar y eliminar todos aquellos ficheros con extensión RSA o SF, volviendo a empaquetar el jar tras la operación. Para firmar los ficheros jar será necesario crear un certificado utilizando la herramienta keytool, también incluida en el JDK.

Gridway y API Java

En la parte de servidor para enviar el trabajo al Grid, se utiliza el metaplanificador, Gridway, que, como se ha indicado anteriormente, puede ser configurado para que los trabajos sean enviados directamente a los computing elements. Estos computing elements deberán haber sido incluidos antes en el fichero etc/host.list del directorio donde haya sido descomprimida la distribución de Gridway. Para arrancar Gridway se utilizará el comando gwd, y para visualizar los trabajos en ejecución el comando gwps. También existe la posibilidad de ver la información, como sistema operativo instalado o características del hardware, con el comando gwhost. El uso de la API se hace a través del fichero drmaa.jar. En el siguiente ejemplo se ha generado a través de la API el siguiente fichero JDL que por defecto se utilizaba en el envío de trabajos por línea de comandos con Gridway.

```
EXECUTABLE=scriptSIM
STDIN_FILE=inputcard.card
STDERR_FILE=stderror.log
STDOUT_FILE=stdoutput.log
INPUT_FILES=inputcard.card, scriptSIM
OUTPUT_FILES=stderror.log, stdoutput.log, dat000001, cer000001
```

Que generado a través de la API en Java, quedaría de la siguiente forma

```
JobTemplate jt = session.createJobTemplate();
jobID = SessionImpl.DRMAA_GW_JOB_ID;
jt.setWorkingDirectory("/opt/tomcat/webapps/gw/");
jt.setRemoteCommand("scriptSIM");
jt.setInputPath("inputcard.card");
jt.setErrorPath("stderror.log");
jt.setOutputPath("stdoutput.log");
String inputFiles[] = {"inputcard.card","scriptSIM"};
jt.setInputFiles(inputFiles);
String outputFiles[] = {"stderror.log","stdoutput.log","dat000001","cer000001"};
jt.setOutputFiles(outputFiles);
```

Como se puede observar en el anterior ejemplo, es necesario fijar un directorio de trabajo, a través del método setWorkingDirectory, en donde Gridway guardará tanto los ficheros de log como los resultados de su ejecución. En este caso se ha fijado como directorio de trabajo un directorio dentro de Tomcat, los permisos de éste se han cambiado para que dicho directorio sea accesible desde Internet, a fin de poder descargar vía un navegador web los ficheros resultado de la simulación.

JDOM

Para enviar los parámetros de Corsika introducidos por el usuario a través de la interfaz gráfica desde el cliente al servidor, se ha creado un fichero XML con los mismos, utilizando los nombres de dichos párametros como el de los elementos del documento XML que contienen los valores de los mismos. En el caso de parámetros con varios subparámetros asociados, se han creado elementos XML con el nombre de estos subparámetros que son hijos de los elementos XML que representan al parámetro al que están asociados. Este fichero, una vez enviado a través del servicio web start al servidor, se procesa utilizando JDOM, para convertirlo al formato de un fichero de configuración de Corsika, que después se utilizará para enviar el trabajo, como se ha visto en el ejemplo anterior de la API en Java de Gridway. El uso de JDOM es sencillo, accediendo al árbol de elementos del documento XML desde el elemento raíz. Así, para sacar información sobre el parámetro ERANGE, que tiene como subparámetros a LLIMIT y ULIMIT, de un extracto del documento XML que incluye los parámetros de Corsika, como el siguiente.

```
<XML>
      <ERANGE>
           <LLIMIT>5</LLIMIT>
     </ERANGE>
</XML>
```

Se utilizaría JDOM de la siguiente forma, llamando el primer lugar al elemento raíz, consiguiendo la referencia al hijo que nos interesa, en este caso ERANGE, localizándole mediante su nombre, y posteriormente haciendo lo mismo con los hijos de éste, LLIMIT y ULIMIT, accediendo entonces bien al valor de dichos elementos (getValue) o a su nombre (getName).

```
cardWriter = new FileWriter("/opt/tomcat/webapps/gw/inputcard.card");
cardBuffWriter = new BufferedWriter(cardWriter);
builder=new SAXBuilder(false);
doc=builder.build("/opt/tomcat/webapps/gw/inputcard.xml");
Element xmlRootElement=doc.getRootElement(); Element
xmlRootElement=doc.getRootElement();
child = xmlRootElement.getChild("ERANGE");
cardBuffWriter.write(child.getName());
subchild = child.getChild("LLIMIT");
cardBuffWriter.write("\t"+subchild.getValue());
subchild = child.getChild("ULIMIT");
cardBuffWriter.write(" "+subchild.getValue());
cardBuffWriter.newLine();
```

Pruebas y automatización

Se han creado pruebas con los framework Junit, para pruebas generales, y Abbot, para las relacionadas con la interfaz gráfica. También se ha sustituido el builder por defecto de Eclipse por ANT, automatizando la compilación y despliegue tanto del cliente como del servidor. En el caso del cliente ANT además de crear el fichero jar que servirá para el despliegue y realizar el despliegue propiamente dicho, también se encarga de firmar los

ficheros jar necesarios para el funcionamiento de la aplicación cliente, preguntando al usuario la contraseña de su certificado de forma interactiva.

Capítulo 5. Perspectivas

Metadatos para la gestión de las simulaciones

El objetivo de portar al Grid las simulaciones Montecarlo no es sólo poder correr los trabajos que antes se ejecutaban en máquinas locales a nodos del Grid. Esto que en el fondo sólo añadiría paralelismo a la ejecución de estas simulaciones, pero no aportaría una ventaja en lo que a tiempo de ejecución se refiere. También aporta otras posibilidades como poder dividir un trabajo en subtrabajos de forma que éstos puedan ser distribuidos por el Grid y terminar así su ejecución antes. Este modo de funcionamiento exige el añadido de tener que, una vez terminados, volver a fusionar dichos subtrabajos para dar lugar al trabajo original. Dividir los trabajos implica mantener un control sobre los subtrabajos para comprobar que estos se ejecutan y devuelven los datos correctamente. En el entorno actual Grid de producción que proporciona EGEE esta tarea de gestión puede complicarse, debido al alto porcentaje de fallos en la mayoría de nodos. Como se analizaba en el anterior capítulo en lo que al envío de trabajos se refiere todo este problema de gestión y monitorización de la correcta ejecución de trabajos en el Grid se ha solventado en parte haciendo uso de un metaplanificador como Gridway, que se encarga de asegurarse de que el trabajo que ha sido enviado a través suyo termine, reenviando dicho trabajo si ocurre cualquier problema. El problema no sólo se limita a la monitorización de los trabajos, sino al almacenamiento de los resultados de los mismos. Dada la inversión que supone en tiempo llevar a cabo una simulación Montecarlo para digamos una noche de observación, como se ha visto en capítulos anteriores, conviene ofrecer al usuario la posibilidad de reutilizar aquellas simulaciones que cumpliendo sus requisitos ya han sido realizadas previamente, evitando su repetición. Este servicio supone crear un catálogo con los conjuntos de datos que han sido creados, y añadir a cada conjunto de datos de ese catálogo de datos una serie de metadatos. Un paso más allá sería utilizar toda esa producción para proyectos de la misma área, por ejemplo proyectos ligados al área de Físicas de Altas Energías que utilizan programas de simulación similares, para evitar el uso repetitivo de lo que al final son infraestructuras compartidas por los distintos proyectos, como es el caso de EGEE.

Una vez establecido el alcance y el uso de la base de datos que albergue dichos metadatos, también es necesario saber cual va a ser el lugar de dicha base de datos en la arquitectura del sistema de producción, si va a estar ligada a los nodos Grid, si va a estar centralizada en el nodo que se encargue de gestionar los trabajos, si estos van a ser gestionados de forma distribuida y la base de datos tendrá un único punto de acceso. Las soluciones son muchas, tantas como posibilidades ofrecen los actuales sistemas de bases de datos relacionales, cuyo uso se centra como se verá a continuación en MySQL y Oracle, debido sobre todo a su soporte a entornos distribuidos. Una vez definido el problema de los metadatos y su relación con la ejecución de trabajos de un alto consumo computacional en el Grid, queda plantear las dos cuestiones que todo proyecto que aborde una solución a este problema, y en concreto MAGIC, como se verá más adelante

en este mismo capítulo, debe abordar:

- ¿Qué es un trabajo atómico para el proyecto en concreto?. Se puede suponer como trabajo atómico un evento, pero este nivel de granuralidad realmente no es útil, y no merece el gasto de envío y gestión individual de un trabajo. Corresponde a los desarrolladores de la herramienta de producción, vistos los recursos, las características de su software de simulación y su experiencia en lo que se refiere a consumo de computación por parte de éste, para determinar en base a los parámetros de entrada del programa cual será el tamaño básico de un trabajo, para proceder a dividir los trabajos a ejecutar en base a este patrón
- ¿Qué metadatos guardar sobre una simulación Monte Carlo y los datos generados por la misma?. Los datos de interés pueden no sólo referirse a los valores de los parámetros con los que hemos generado dicha simulación y sus datos asociados, sino también pueden abarcar el tiempo de ejecución, la versión del programa u otro software asociado, por ejemplo una librería, o incluso información sobre las características y el estado de la máquina o máquinas donde ha tenido lugar dicha simulación, ¿qué información es realmente importante?

El área de las simulaciones Montecarlo en proyectos de Física de Altas Energías, como ya se ha visto en el repaso del estado del arte en el segundo capítulo, no es un área nueva y existen varias experiencias al respecto, también en lo referente al uso de metadatos, que abordan los problemas antes vistos. A continuación se verán algunas de las distintas propuestas a este respecto que han surgido en proyectos dentro de instituciones como el CERN o Fermilab, que cubren propuestas que abordan desde la descripción de producción hasta la gestión de conjuntos de datos para su uso en posteriores análisis.

Estudio de los sistemas de metadatos en sistemas de producción Montecarlo para proyectos de Física de altas energías

SAMGrid

SAMGrid [25] es un servicio de gestión de datos desarrollado en un primer momento para el experimento D0 del Fermilab, pero que posteriormente se adaptó también al experimento CDF del mismo laboratorio. Este sistema abarca toda la gestión de datos de dichos experimentos, tanto los generados a partir de simulaciones como los recopilados por el experimento en concreto, incluyendo el análisis de todos estos. Clave en el funcionamiento de SAMGrid es su gestión de los metadatos asociados a los datos que guarda, los cuales se almacenan de forma centralizada en un gestor de bases de datos Oracle, gracias a la cual se sirven en torno a 360 ficheros/minuto. Esta herramienta cubre los distintos casos de uso que experimentos del área de Física de Altas Energías, como HEPCAL, CDF, BABAR o ATLAS, han desarrollado respecto de la creación y gestión de metadatos. Dichos casos de uso se centran en tres áreas: análisis, gestión de trabajos y

gestión de datos. En lo que al análisis se refiere el caso de uso se basa en un usuario que desea analizar un conjunto de datos, para lo cual tendrá o bien que seleccionar los datos de una colección de estos o generarlos a través de una simulación Montecarlo, en el caso de que estos no hayan sido generados previamente. En lo referente a la gestión de trabajos, partiendo de una estimación previa de cual va a ser el coste de enviar dicho trabajo, en cuanto a recursos consumidos del sistema, el usuario quiere, una vez enviado el trabajo al Grid, ser capaz de recibir información sobre cual es el estado de la ejecución de dicho trabajo, de forma que el trabajo pueda ser recuperado en el caso de producirse un error en su ejecución. Por último la gestión de datos, debe permitir al usuario no sólo poder modificar los metadatos de los conjuntos de datos, sino que estos se adaptan a la necesidad de fusionar conjuntos de datos o dividirlos en subconjuntos más pequeños, sin perder por ello la información de metadatos que guardan. También debe ser posible para el usuario poder definir a priori los metadatos de un conjuntos de datos que va a ser producido, y poder añadir metadatos a un conjunto de datos cuando este ha sido bajado al directorio local del usuario y analizado por este.En la siguiente figura se muestran los metadatos asociados a un conjunto de datos, los principales como el nombre del fichero que aloja ese conjunto de metadatos, el formato y tamaño del mismo, y el valor (crc_value) del código de comprobación de redundancia cíclica, que sirve para comprobar si los datos han sido alterados durante su gestión.

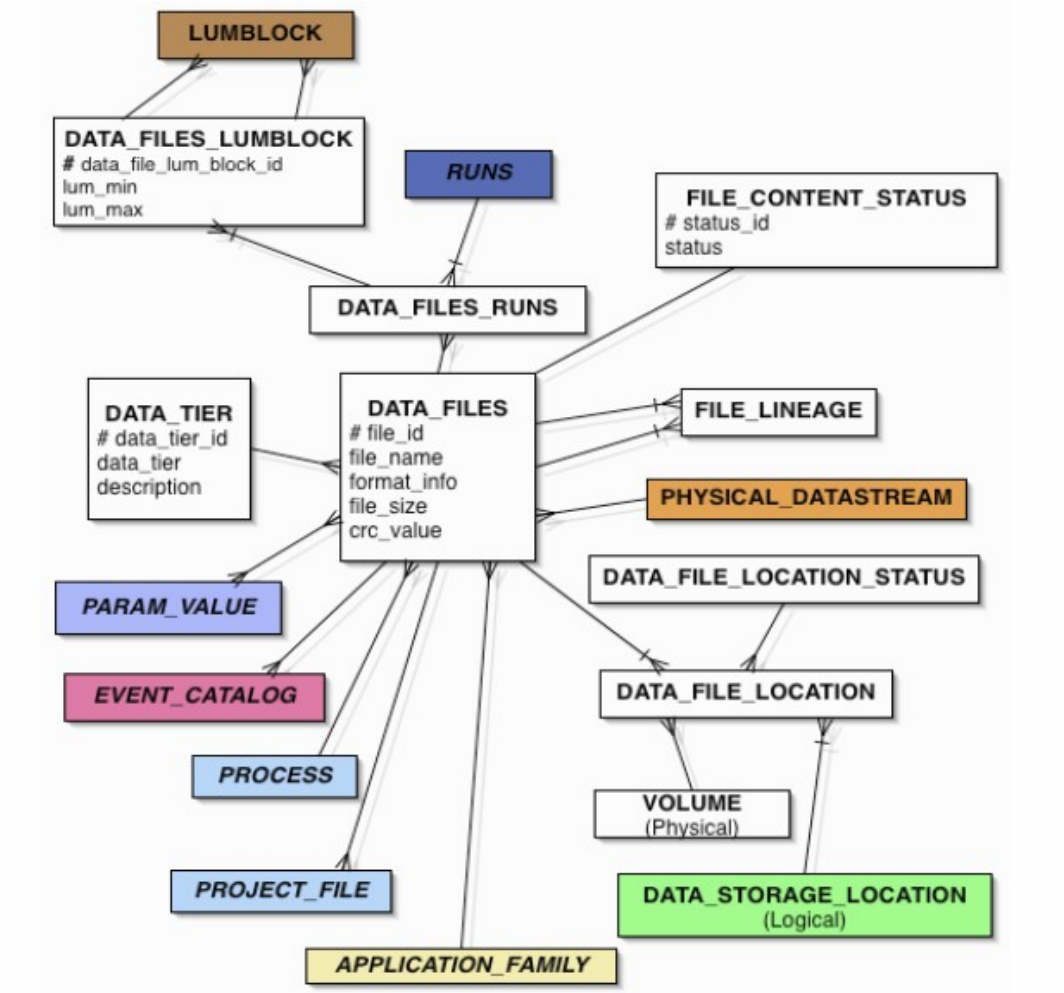

Figura 19. Metadatos asociados a un conjunto de datos en SAMGrid

A su vez este conjunto de datos tendrá otros metadatos asociados como el número de runs contenidos dentro de ese fichero, características físicas como la luminosidad, la fase del flujo de datos donde dicho conjunto de datos se encuentra (por ejemplo si ya ha sido analizado), y la localización o localizaciones asociadas al fichero. Estos son sólo algunos de los metadatos que se pueden asociar, ya que como se observa en la figura coloreados existen otros conjuntos de metadatos disponibles, como el valor de los parámetros físicos con los que se ha creado dicho conjunto de datos, el proyecto en el que ha sido creado, o metadatos sobre la versión de la aplicación con la que han sido generados.

Cuando el usuario va a hacer un análisis de un conjunto de datos ya disponible dentro del catálogo de SAMGrid, éste guarda información adicional sobre el trabajo que se está ejecutando para tal fin, de forma que sea capaz de reproducir el trabajo en caso de fallo. Entre esa información, como se puede observar en la siguiente figura, se encuentra el nombre de los procesos asociados al proyecto, información del proyecto en si, almacenada en ANALYSIS_PROJECT, de los conjuntos de datos utilizados, de la aplicación utilizada para ejecutar el proyecto y el estado de los ficheros utilizados.

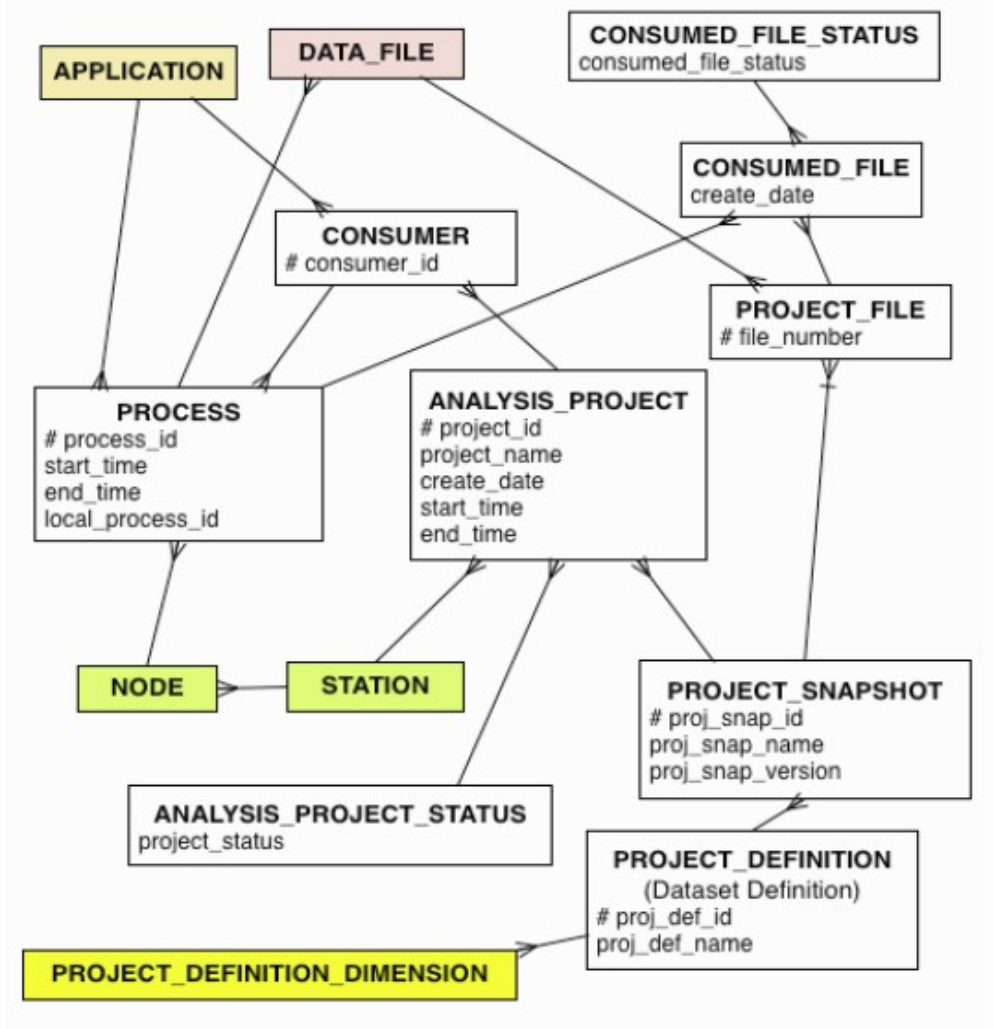

Figura 20. Metadatos asociados a un proyecto dentro de SAMGrid

SAMGrid además de los metadatos predefinidos, ofrece la posibilidad de crear metadatos propios a cada experimento, de forma que cada uno de estos pueda adaptar el conjunto de metadatos a sus necesidades concretas, por ejemplo valores distintos de los parámetros físicos a la hora de realizar las simulaciones Montecarlo. Para esto se utiliza el esquema mostrado en la siguiente figura que identifica cada posible descriptor de parámetro como un PARAM_TYPE, donde estos a su vez pueden estar agrupados en conjuntos, PARAM_CATEGORY, donde se guardan todos los parámetros referentes a un mismo proyecto. Cada uno de estos parámetros a su vez pueden tener un valor y unos datos de retorno asociados (PARAM_VALUE y DATA_TYPE), así como un fichero o conjuntos de datos de entrada (DATA_FILE), del cual se podrá conseguir más información, como sus metadatos, a través de la base de datos utilizando REQUEST_DETAIL.

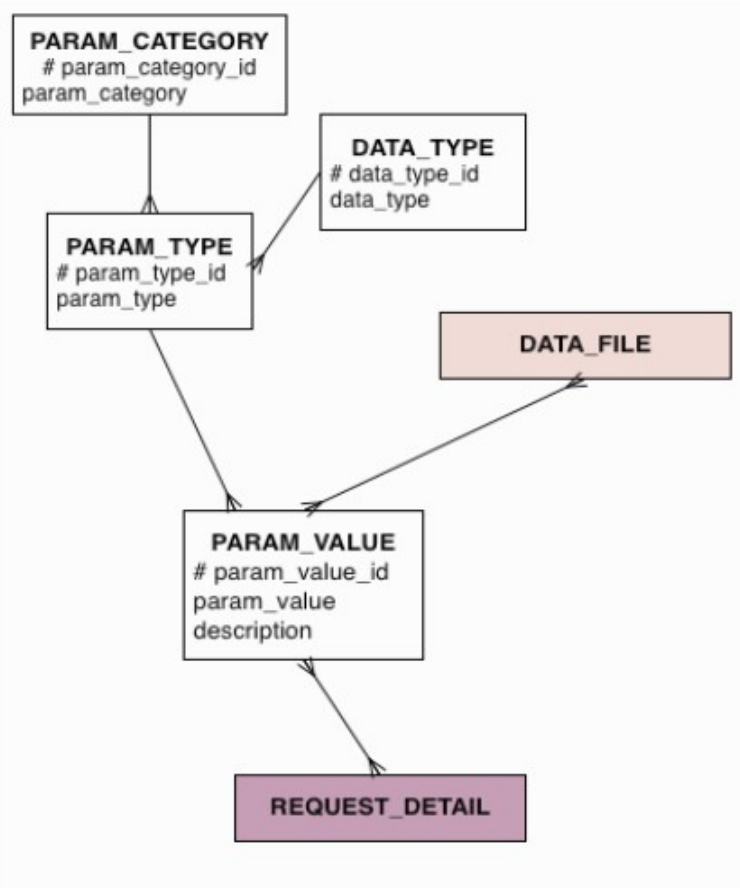

Figura 21. Creación de metadatos en SAMGrid

RefDB y BOSS

Otra de las arquitecturas de gestión de metadatos que hemos estudiado es la desarrollada para el experimento CMS del Large Hadron Collider en el CERN. Esta arquitectura consta de dos componentes, uno a nivel global para gestionar el envío de trabajos entre centros, RefDB [26], y otro para controlar dicho envío a nivel local de cada centro participante, en concreto de cada granja de servidores, BOSS [27]. RefDB permite el envío de trabajos para la producción de simulaciones Montecarlo, mediante la configuración de los parámetros asociados a estas, en especial los referentes a las características físicas de la simulación, vía web, permitiendo el uso de conjuntos de parámetros previamente seleccionados. Asimismo también funciona a modo de catálogo para saber que datos están disponibles. RefDB funciona sobre MySQL, accediéndose a ella mediante un conjunto de scripts desarrollados en PHP. El funcionamiento de RefDB es el que se muestra en la siguiente figura. Un usuario decide enviar una solicitud de producción por medio de la interfaz web de RefDB. Dicha solicitud es atendida por el coordinador del área de producción, que, una vez autorizada por su parte, la remite al operador de producción junto con un identificador. Dicho identificador será utilizado por el operador para indicar a la herramienta de planificación, en este caso McRunJob, IMPALA o CMSProd, que esta

solicite a RefDB la información necesaria para ejecutar dicho trabajo, y lo descomponga en trabajos de menor tamaño, si esto fuera necesario. Una vez el trabajo ha terminado correctamente se envia por correo electrónico los ficheros de log que serán utilizados para validar los datos y generar la información de metadatos asociada al trabajo. Todo este proceso no está automatizado debido a la necesidad de disponer un operador que esté atento a la ejecución del trabajo, a la disponibilidad de los recursos del centro en cuestión y principalmente a las prioridades en lo que a áreas de trabajo se refiere del personal de éste.

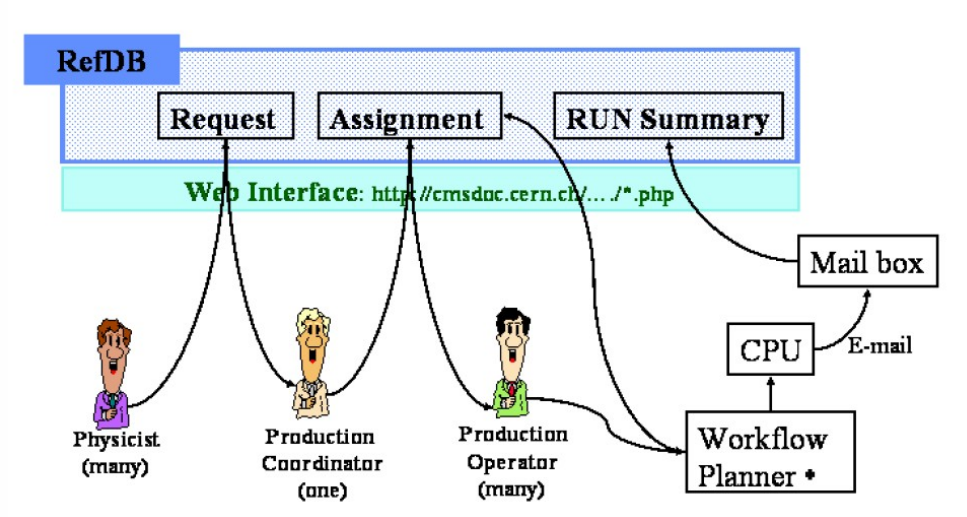

Figura 22. Funcionamiento de RefDB

Cada una de estas peticiones se compone de los siguientes cuatro elementos: el ejecutable que será utilizado para llevar a cabo la simulación, los parámetros de entrada para el programa de simulación, los datos de entrada y los datos de producción, estos últimos, que incluyen en otros el nombre de los datos de salida, la estrategia a seguir a la hora de dividir un trabajo en otros más pequeños o variables que cambian para cada trabajo como el valor de las semillas que se utilizarán en la simulación, no serán introducidos por el usuario sino por el coordinador de la producción. En cuanto al ejecutable, este será seleccionado por el usuario directamente desde RefDB en base a su nombre y versión, ya que todo el software para la producción se encuentra allí registrado. Esta información se guarda en RefDB junto con los esquemas para procesar los ficheros de log y realizar la monitorización del trabajo, que se utilizan localmente por BOSS. Como se puede observar en las dos figuras que se incluyen a continuación con este modelo de datos, también se asocia al ejecutable información sobre su gestión con DAR, una herramienta que se encarga de distribuir el binario seleccionado por el usuario a los centros de producción.

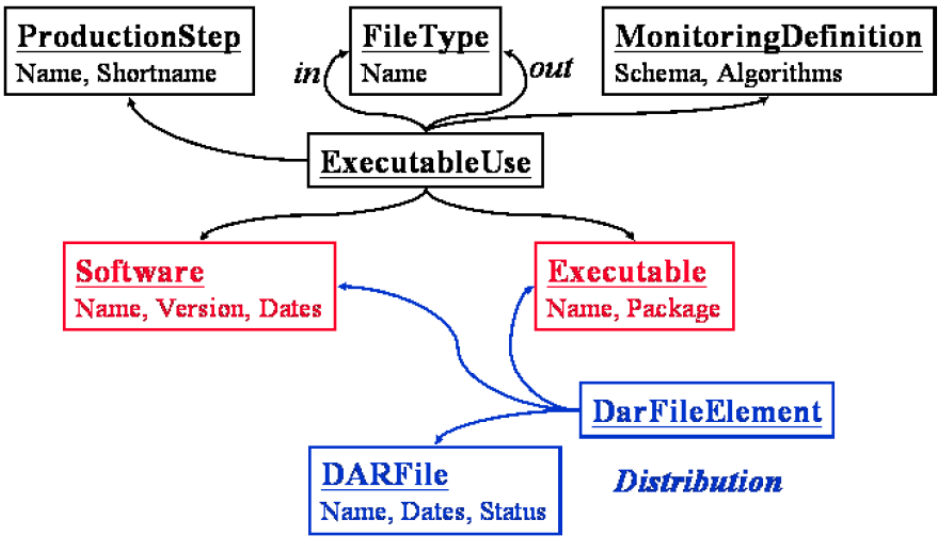

Figura 23. Tablas en RefDB que guardan información sobre el ejecutable

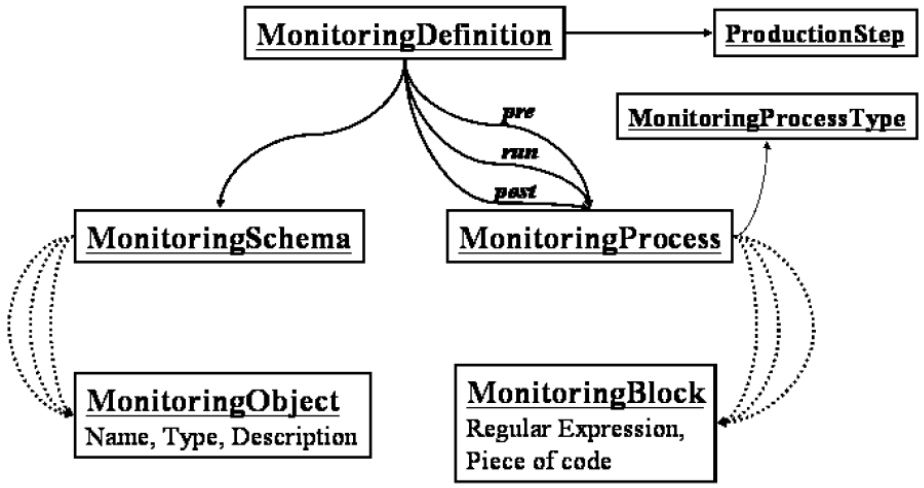

Figura 24. Tablas en RefDB que guardan información para BOSS

La modularidad del modelo de datos asociado a los parámetros físicos permite su reutilización a la hora de definir otros conjuntos de parámetros. Para definir estos parámetros se utiliza una interfaz web, que en el caso de coincidir el conjunto de parámetros, ofrece la posibilidad de reutilizarlos. La lista de parámetros se guarda, como se puede ver en el modelo de datos incluído en la siguiente figura, en un mapa de parámetros. Dichos parámetros pueden tener asociado uno o varios ficheros que a su vez tendrán asociado un programa determinado.

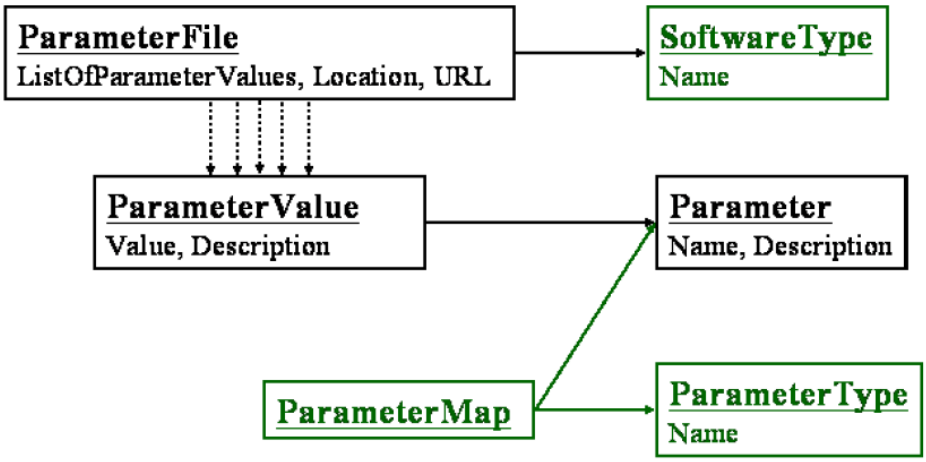

Figura 25. Tablas en RefDB que guardan información sobre los parámetros físicos

El usuario debe describir los datos que van a ser generados, los cuales deberán ser compatibles con el ejecutable incluído, lo cual se verificará a través de la información contenida en la tabla ExecutableUse de RefDB. Los conjuntos de datos se pueden agrupar en colecciones. Dichos conjuntos de datos se distinguen por la configuración del detector, con parámetros como la geometría o la especificación del campo magnético).

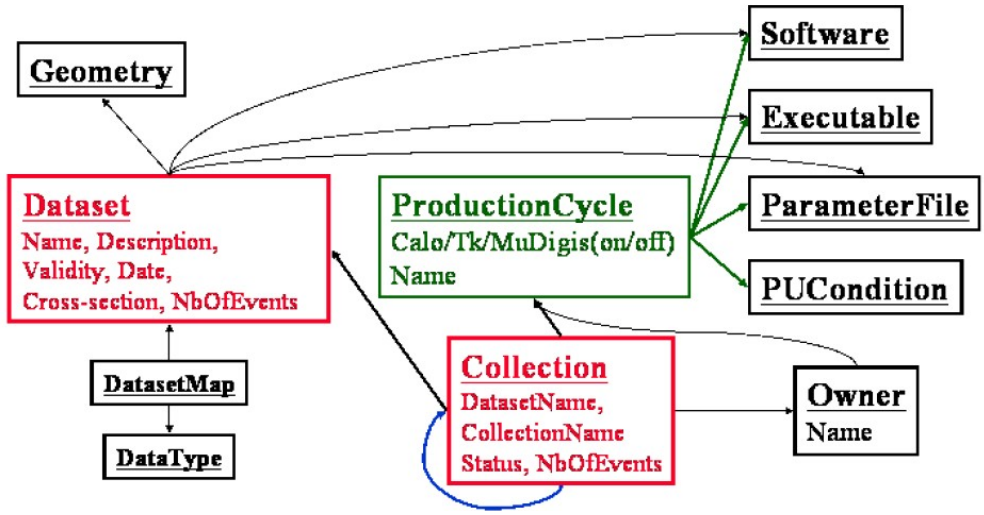

Figura 26. Tablas en RefDB que guardan información sobre los datos de entrada

Una vez definidos todos los parámetros de la petición ligada a la producción, dicha petición es asociada a una asignación de trabajo con un identificador. Esta asignación de trabajo tendrá los siguientes parámetros, que acompañarán al trabajo en las distintas fases de la producción: nombre del ejecutable, además del software a utilizar y la versión del mismo; la localización del fichero de parámetros que se modificará con cada trabajo en el caso de ser necesario; la localización de los distintos subtrabajos en los que se dividirá el trabajo en el caso de que sea necesario; el fichero para almacenar toda la información

sobre la monitorización, que incluirá los esquemas que necesita BOSS; el fichero para la configuración de los datos (como por ejemplo la geometría) y el nombre del conjunto de datos. Un resumen del modelo de datos referido a esta asignación se muestra en la siguiente figura.

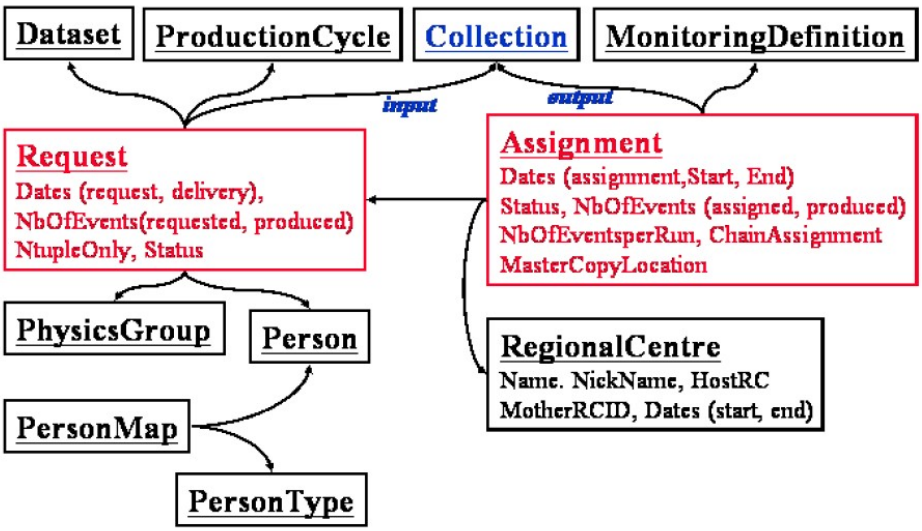

Figura 27. Tablas en RefDB sobre la asignación de una petición de producción

Para guardar información sobre la producción en si, se crea una tabla por cada conjunto de datos, de forma que contiene tantas filas como runs han sido necesarios para generarla. Cada fila describe uno de esos runs, y contiene la siguiente información: número de run, semillas utilizadas, nombre del fichero de salida, estado, identificador de asignación, número de eventos de entrada, número de eventos de salida, duración de la ejecución y tamaño de los datos de salida.

Una vez el usuario ha lanzado el trabajo desde RefDB, este junta con la información introducida por el usuario se transmite al centro que va a llevar a cabo la producción de las simulaciones, en concreto a BOSS, una aplicación que proporciona una capa de abstracción a los distintos gestores de colas que dicho centro puede tener instalados para enviar trabajos a su granja de servidores. Esta abstracción permite a la hora de lanzar trabajos incluir información sobre los mismos en la base de datos que BOSS tiene disponible para tal fin. BOSS consigue la información sobre el estado del trabajo directamente desde los ficheros de logs de salida, tanto el correspondiente a la salida estándar como el asociado con la de error. Para poder parsear la información contenida en estos ficheros, el usuario enviará desde RefDB, un esquema que contenga aquellas variables cuyo valor BOSS debe localizar, y tres scripts para sacar dicha información antes, después y durante la ejecución del trabajo. A la hora de mandar un trabajo BOSS, que corre sobre MySQL, se crea un envoltorio para el trabajo, un jobExecutor según la terminología de la herramienta, junto con el cual se ejecuta un dbUpdator encargado de guardar en la base de datos la salida sobre el estado del trabajo ya parseada. Al

ejecutarse dicho envoltorio se crea una entrada en la tabla correspondiente de la base de datos asociada a BOSS, con el identificador que tendrá dicho trabajo en BOSS asociándole el identificador que ese trabajo tiene en el gestor de colas. Cada vez que se crea un nuevo job_type, o conjunto de esquema más los scripts asociados, se crea en la base de datos una tabla en la que las columnas corresponden a las variables asociadas a dicho esquema junto con una columna adicional para el identificador del trabajo, de esta forma se puede monitorizar el estado de cada variable en un momento dado.

Las anteriores propuestas cubren prácticamente todos los aspectos que se encuentran en la literatura relacionada en lo referente al diseño de una base de datos que contenga los metadatos en un sistema de producción de simulaciones Montecarlo para un experimento de físicas de altas energias, como el que se aborda en este trabajo. Sin embargo dadas las características de MAGICGrid, la monitorización se va a limitar, al menos en una primera fase, a los trabajos enviados con Gridway a través del nodo Gris asociado al proyecto, por lo que un modelo más simple se adaptaría mejor a las necesidades del proyecto. Todo esto sin dejar de lado la posibilidad de poder dividir un trabajo en subtrabajos si el tamaño de este así lo recomienda. Para una primera fase en el desarrollo del modelo de computación para el experimento ATLAS del LHC, se desarrolló un diseño que se adapta a estos requerimientos. Por un lado este modelo permitía la creación de tareas, cada una de las cuales podía tener a su vez varios trabajos asociados, de forma que cubriría el caso de uso de dividir un trabajo en varios subtrabajos que se comentaba anteriormente. Por otro lado cada uno de esos trabajos a su vez pueden tener asociados uno o varios ejecutores, lo cual permite reenviar el trabajo en caso de error, aunque esta situación será bastante poco habitual dado el uso del metaplanificador. Otra propuesta siguiendo el espíritu de esta, es la que se realizo en la primera fase del proyecto MAGICGrid. Dicho diseño, que se muestra en la siguiente figura, no asume la partición de los trabajos en otros de menor tamaño, pero si mantiene una lista de trabajos que han fallado para su posterior gestión. Asimismo mantiene tablas separadas para los distintos parámetros que caracterizan a un trabajo, como puede ser su identificador global en el Grid o los datos de entrada asociados al mismo.

Dentro del proyecto MAGICGrid los objetivos con el uso de metadatos son los siguientes:

- Ofrecer la posibilidad de dividir un trabajo enviado por un usuario desde la aplicación en otros de menor tamaño, adecuándolos a las capacidades reales de los computing elements a los que se envían dichos trabajos, para ofrecer un tiempo de respuesta adecuado. Para ello habrá que encontrar que tamaño de trabajo es el más adecuado para definir como trabajo atómico de MAGICGrid. Para ello habrá que tener en cuenta los parámetros que el usuario introduce a Corsika, y en especial el referente al número de runs. El trabajo deberá ser capaz de identificar estos subtrabajos como suyos.
- De la misma forma que el trabajo se descompone en otros, para volver a fusionar esos subtrabajos será necesario monitorizar el estado de ejecución de todos ellos, de forma que la fusión se lleve a cabo cuando todos los subtrabajos han terminado.
- Otra de las funcionalidades básicas de MAGICGrid, en línea con una de las ventajas del concepto de observatorio virtual, que se verá más adelante en este mismo capítulo,

es ofrecer de forma automática al usuario la posibilidad de acceder al catálogo de simulaciones ya realizadas a través de la aplicación, de forma que no tengan que repetirse de nuevo simulaciones ya llevadas a cabo, esto es aquellas que coindicen en los valores de los parámetros que recibe Corsika. Este catálogo no sólo tendrá que guardar información sobre dichos parámetros sino la localización del conjunto de datos, ya que en la medida de lo posible dichas simulaciones se almacenarán de forma remota cerca del computing element que las ha generado, de manera que en el caso de querer llevar a cabo más adelante el análisis de las mismas, no se añada retardo por la transferencia de los datos a un tercer sitio más alejado. En aquellas situaciones donde el almacenamiento remoto no sea posible, este se llevará a cabo de forma local al servidor donde corre MAGICGrid, de forma que los datos sean accesibles a través de la interfaz web proporcionada por el servidor Tomcat.

- También sería necesario mantener una información más completa sobre el estado de cada trabajo que ha sido ejecutado o está siendo ejecutado con MAGICGrid, en especial de cara a aquellos trabajos que hayan fallado, por haber superado el número máximo de reenvios fijado en Gridway.

A estos requisitos, también habría que añadir aquellos ligados a una posible ampliación que en el futuro se pudiera hacer de esta herramienta, de forma que el análisis se pudiera llevar a cabo de forma remota. Actualmente, como ya se ha señalado en el anterior capítulo, esto no es así ya que el principal objetivo es minimizar el tiempo de cómputo de las simulaciones Montecarlo. En la siguiente figura se incluye la propuesta de diseño para esta base de datos de metadatos, está se restringe a los casos de uso relacionados con la producción de las simulaciones Montecarlo utilizando MAGICGrid.

Uso de agentes software en MAGICGrid

El entorno de Grid y los sistemas de agentes, como ya apuntaban Foster, Jennings y Kesselman en el artículo "Brain Meets Brawn: Why Grid and Agents Need Each Other" [28] hoy por hoy tienen necesidades complementarias. Así, el Grid necesita de un sistema flexible y autónomo como el que proporcionan los agentes, y los sistemas basados en agentes necesitan de una infraestructura robusta, como la que proporciona el Grid. Las últimas tendencias en ambas comunidades apuestan por una convergencia de ambas. Mientras que en el Grid las distintas fases de evolución en sus herramientas y estándares, la última de las cuales ha sido la adopción de los servicios web a la hora de comunicar los distintos componentes de este, dan por hecho que el próximo paso será la adopción generalizada de sistemas multiagentes para ciertas tareas como las que se verán a continuación, la comunidad detrás de los sistemas multiagentes ya están adaptando las plataformas y los estándares asociados para trabajar en sistemas distribuidos estándar (OMG, web services). Los agentes ya suponen un área de trabajo en varios de los servicios Grid, entre los más significativos se encuentran los siguientes, que se trataran más a fondo a continuación.

Modelado de Organizaciones Virtuales

La idea de la utilización compartida de recursos por parte de un grupo de personas con los mismos intereses se ven reflejadas en las organizaciones virtuales (VO en sus siglas en inglés) que se encargan de ofrecer los servicios necesarios para utilizar dichos recursos. Estas organizaciones se transforman constantemente dependiendo de cuales sean las necesidades de los usuarios del Grid. Entre otras funciones los agentes dentro de la organización virtual se encargan de monitorizar los recursos de ésta, aplicando en cada caso la política de uso que éstos tengan, así como de gestionar la confianza entre los distintos recursos que participan en una organización virtual. El agente tenderá a maximizar el uso de los recursos que gestiona, de forma que podría darse el caso de trabajos que no terminan por culpa de comportamientos "egoístas" por parte de los agentes, abandonando organizaciones virtuales cuando aún hay trabajos por terminar. En el artículo Policing Virtual Organizations [29] Oren, Norman, Preece y Chalmers proponen un sistema multiagente que se encargue de monitorizar a los agentes que forman parte de la VO, estableciendo políticas respecto de esta, de forma que los contratos rotos por los agentes influirán en la reputación de estos, lo cual reducirá su nivel de calidad de servicio y confiabilidad respecto del resto de miembros de dicha VO.

Gestión de recursos y negociación

A la hora de seleccionar que recurso va a encargarse de llevar un trabajo a cabo se ha tenido en cuenta una serie de factores que daban lugar a un coste asociado al uso de dicho recurso. El uso de sistemas multiagentes a la hora de seleccionar un recurso se atiene a un mayor número de variables, entre ellas la calidad del servicio que se va a ofrecer, el coste de dicho servicio y el tiempo de respuesta (o al menos una predicción del mismo). La idea detrás de este sistema es establecer una serie de niveles de calidad de servicio (SLAs) entre el agente encargado de gestionar los trabajos enviados por el usuario, y los agentes encargados de gestionar los distintos recursos de computación y almacenamiento, como los asociados con los recursos de red, de forma que se acuerde un conjunto de SLAs que cumplan con las restricciones del usuario.

Gestión de datos

La transferencia de datos entre distintos recursos de un Grid supone una de las mayores fuentes de problemas en la gestión de este tipo de entornos. Los sistemas de transferencia basados en protocolos de bajo nivel como GridFTP hacen que sea el usuario el que tenga que tener constancia del estado de la transferencia, pudiendo producirse así copias innecesarias que pueden llevar a la saturación de los elementos de almacenamiento (Storage Elements).

Reparto de carga

Otra funcionalidad común de los sistemas de agentes dentro del entorno Grid es el reparto de carga, actividad que va estrechamente ligada a la gestión de recursos. Al igual que en el caso de la gestión de recursos, cada recurso tiene asociado un agente que a través de varias fuentes de datos decide como gestionar la carga de ese recurso. A su vez los

agentes de distintos recursos se comunican entre si para repartir la carga global del sistema. Entre sus objetivos se encuentra la necesidad de cumplir con los plazos de terminación exigidos por la tarea en particular.

Modelado de Organizaciones Virtuales

Las organizaciones virtuales son uno de los servicios claves dentro de un Grid, de forma que los recursos de varios usuarios con objetivos comunes puedan ser compartidos entre ellos. Este nuevo tipo de entornos, viene a sustituir a los de la misma filosofía distribuida basados en tecnologías como CORBA, Java RMI o COM/DOM. Estas organizaciones se basan en una arquitectura dinámica, ya que estas se podrán crear o destruir en base a las necesidades concretas en cada momento de los usuarios del Grid; escalable, debido a que el número de usuarios de la organización virtual variará según esta vaya adquiriendo popularidad o vaya creciendo la comunidad no virtual sobre la que se asienta, siendo esta característica común de la mayoría de las organizaciones virtuales actuales al contrario que en el aspecto dinámico; y por último autónoma, de forma que la organización en si misma sea capaz de ofrecer servicios a sus miembros, tales como el balanceo de carga dentro de la propia organización virtual o la gestión de recursos. Las organizaciones virtuales necesitan de los agentes para suplir la falta de estado que sufren los servicios web sobre los que se asientan los actuales componentes de los distintos middleware Grid. Como se verá en los siguientes ejemplos de trabajos relevantes en este área, el trabajo actual en lo que a integración de sistemas de agentes en la gestión de organizaciones virtuales se centra principalmente en: monitorizar los recursos miembro, garantizar el cumplimiento de las políticas de uso y gestionar la confianza entre las distintas instituciones que forman parte de la VO.

Propuesta de Baohua, Yanbo y Hongcui [30] del Institute of Computing Technology de la Chinese Academy of Sciences, para la gestión de organizaciones virtuales.

Este sistema se centra en un componente llamado Business Collection en el que se registran todos los usuarios del Grid. Al registrarse cada uno de los usuarios registrará aquellos servicios que ofrece, sin especificar la dirección del servicio para evitar posibles usos indebidos por parte de usuarios no autorizados, y la dirección de un agente que se encargará de gestionar todo lo referente a ese usuario en la organización virtual. La organización virtual por su parte dispondrá de una definición en base a sus necesidades particulares (para la que los autores han desarrollado un lenguaje XML, VOML), y un agente asociado a la misma que se encargará de que las restricciones ahí impuestas se cumplan. En este sistema, los agentes juegan un papel principal a la hora de negociar que trabajos dentro de la VO puede realizar cada uno de los usuarios, para lo que utiliza la información proporcionada por estos al componente Business Collective, y fijando los detalles finales a través de una negociación basada en el esquema que proporciona el protocolo contract net [31].

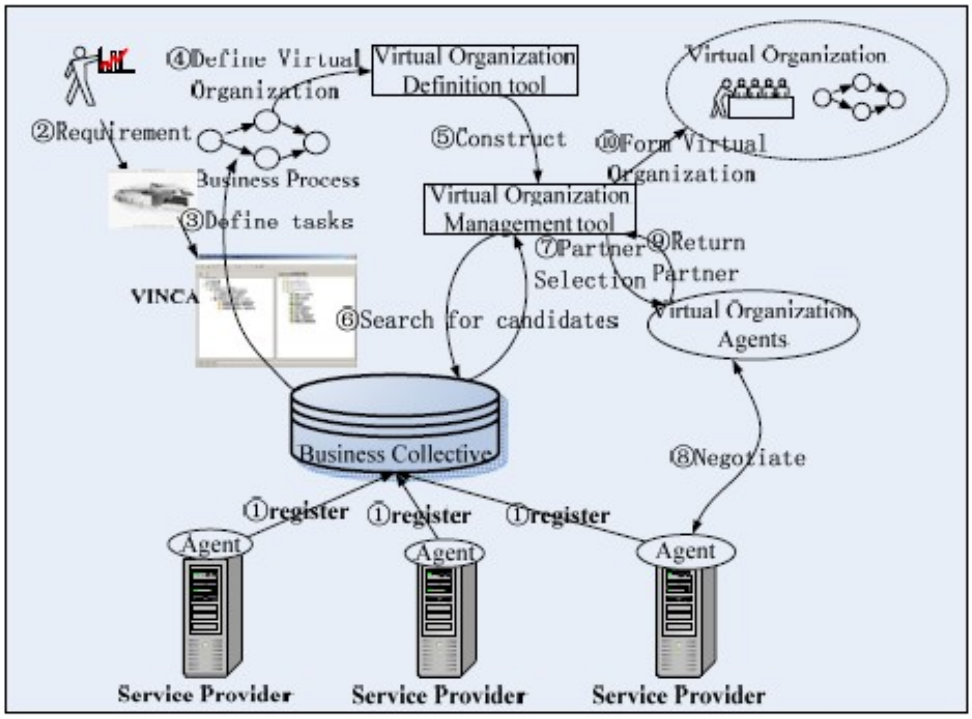

Figura 28. Arquitectura del framework

De esta forma el agente de la organización virtual enviará un mensaje al resto de agentes especificando la tarea a realizar, incluyendo las entradas que se recibirá y las salidas que se espera, junto con información sobre las condiciones del contrato, por ejemplo el coste en tiempo que esta tarea deberá tener. Si alguno de los agentes de los usuarios seleccionados en base a lo proporcionado por la información y restricciones que el usuario al que representa cumple las condiciones, en concreto aquellas referentes a las condiciones del contrato, este enviará un mensaje de respuesta indicando que puede llevar a cabo dicho trabajo. El agente de la organización virtual al final decidirá un ganador en base a las respuestas recibidas, que optimicen por ejemplo el coste en tiempo de la tarea, y enviará un mensaje afirmativo al ganador y negativo al resto de agentes participantes, enviándole finalmente el agente seleccionado un mensaje de confirmación a este último mensaje. Para generar los agentes se utiliza ZEUS, mientras que para la comunicación entre los mismos se utiliza FIPA.

Propuesta para la gestión y formación de organizaciones virtuales realizada dentro del proyecto COGNOISE (más información en http://www.cognoise.org)

Este sistema se centra en dos áreas: por un lado la formación de organizaciones virtuales basado en tres tecnologías: un sistema de decisión para un usuario, un sistema de subasta para la gestión de contratos y un sistema para encontrar servicios teniendo en cuenta la calidad del servicio que ofrecen estos; y por otro se aborda la gestión de organizaciones virtuales, tratada más a fondo dentro de COGNOISE-G, ofreciendo soluciones a la gestión

de políticas, a la reputación de los usuarios dentro de la organización virtual y a la monitorización de la calidad de servicio. A la hora de llevar a cabo la formación de la organización virtual se aplica el esquema que se muestra en la siguiente figura. Suponiendo que todos los proveedores de servicios dentro del Grid (SP) han registrado sus servicios a través del agente de páginas amarillas (YP), una vez el usuario ha especificado sus restricciones a través del agente encargado de gestionar estas restricciones, éste se encargará de negociar con el YP los proveedores que cumplen con las restricciones del usuario y elegir aquellos que ofrezcan unas mejores condiciones de servicio o calidad del mismo a través del agente encargado de valorar estas ofertas (CA). Una vez decididos los proveedores, el agente a cargo de gestionar las restricciones se convertirá en el encargado (VOM) de gestionar el correcto funcionamiento de la organización virtual creada.

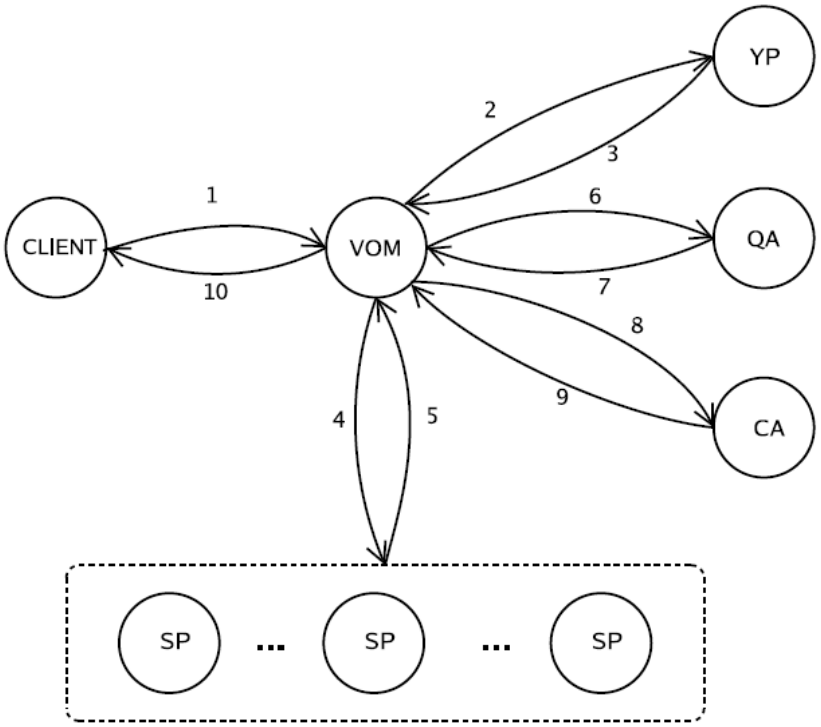

Figura 29. Comunicación seguida para la formación de una VO

Esta monitorización del servicio que están dando los proveedores seleccionados se llevará a cabo a través de un agente especializado en proporcionar información sobre la calidad del servicio de cada proveedor, información que servirá para romper el contrato en caso de no cumplimiento de la calidad acordada, en cuyo caso el VOM volverá de nuevo a crear una nueva organización virtual que si de cobertura a las necesidades del usuario.

Una vez el agente de restricciones ha enviado las solicitudes de servicio a los distintos proveedores de servicios dentro del Grid, estos utilizan una serie de métricas basadas en el tiempo que el proveedor puede utilizar un recurso para proporcionar un servicio, y el tanto por ciento de ese recurso que todavía no está siendo utilizado cuando llega la

petición desde el agente de restricciones. Como resultado de esta decisión el agente asociado al proveedor de servicio puede aparte de rechazar o aceptar la petición de servicio a título individual, utilizar alguna organización virtual de la que forme parte para dar servicio a esa petición, crear una nueva organización virtual a partir de la misma, o incluso romper un contrato de servicio en marcha para dar servicio a la nueva petición si esta resultara más rentable para el proveedor. Frente a esta última situación los mismos autores en el artículo "Policing Virtual Organizations" proponen un sistema que penalice al agente y a su proveedor asociado dentro de la organización virtual, cuando estos decidan romper el servicio a un usuario con el que han adquirido un contrato anteriormente.

Esta penalización consistirá en bajar su nivel de confianza y calidad de servicio dentro de la organización, de forma que su elección de cara a posibles futuros contratos se verá mermada. Una posible solución a este problema, aparte de incluir nuevos agentes dentro de la infraestructura como también proponen estos autores, sería la puesta en marcha de contratos más detallados, utilizando WS-SLA (Web Services – Service Level Agreement) como base, lo que incrementaría la complejidad del protocolo de negociación entre el agente de restricciones y los asociados a los proveedores de servicios. Por otro lado se encuentra la gestión de la reputación de los agentes que están asociados a la organización virtual. Para llevar a cabo esta gestión se utilizan dos herramientas, por un lado un nivel de confianza (entre 0 y 1) para cada uno de los agentes con los que se va a tratar, de forma que si no existe una historia previa de trabajo con un agente dado, se utilizará la información proporcionada por otros agentes. Frente a este modelo colaborativo, se ofrece una segunda herramienta, un agente dedicado a monitorizar la actividad de cada uno de los agentes de la organización virtual y calcular su confiabilidad.

Gestión de recursos y negociación

En un Grid cuando un usuario envía un trabajo el middleware le ofrece distintas posibilidades. Hasta ahora lo más común había sido que fuese el propio usuario el que pudiese enviar a uno u otro proveedor dentro del Grid en base a la carga de este, reflejada por una serie de estadísticas que ofrecía el mismo middleware. Esta aproximación ha ido variando siendo ahora los metaplanificadores, un nuevo tipo de componentes del middleware, los que se encargan de lleva a cabo toda esta toma de decisiones utilizando la funcionalidad proporcionada por el middleware y añadiendo nuevas herramientas como algoritmos de planificación. La planificación dentro del Grid se ha basado hasta hace poco en enviar el trabajo al proveedor que en ese momento tuviera recursos disponibles, sin tener en cuenta ninguna otra variable. Con la llegada de los agentes al área de la gestión de recursos y negociación dentro del Grid se han añadido otras variables de decisión a la hora de enviar un trabajo a un proveedor en concreto, como pueden ser el coste de dicho servicio, el rendimiento del proveedor y la calidad de servicio que ofrece éste, o incluso predicciones de tiempo sobre cual será el tiempo de respuesta de los distintos proveedores ante ese trabajo. Así todas estas variables entran dentro de un contrato o SLA entre el usuario y el proveedor, que cumplirá las restricciones del usuario. Las dos propuestas que se ven a continuación abordan esta problemática.

Propuesta del grupo encabezado por L. Nassif de la Federal University of Minas Gerais (UFMG) de Brasil para la gestión de recursos en un Grid [32]

Este sistema añade una capa de inteligencia a la hora de seleccionar un proveedor para que enviar un trabajo al Grid, utilizando una serie de técnicas de negociación que añaden nuevas variables a la hora de elegir el proveedor más adecuado, principalmente la calidad del servicio y la repuesta en tiempo que nos dará el mismo. En la figura se muestra los tres módulos en los que se divide este middleware, de nombre Mask: negociación, migración e interfaz. Este middleware utiliza JADE y se integra dentro de Globus. El funcionamiento del sistema es el siguiente, el usuario configura el agente de usuario indicándole los términos en los que puede negociar en cuanto a precio, calidad del servicio y duración del trabajo; indicándole los argumentos del trabajo y fijando los puntos a tener más en cuenta a la hora de negociar (ya sea este el precio, calidad del servicio o la duración del trabajo). Será entonces cuando el Monitoring and Discovery Service (MDS), uno de los componentes de Globus, le devuelva una lista de proveedores candidatos, de los que elegirá un grupo con los que negociar. Es entonces cuando la negociación entre el agente del usuario y el del proveedor con el que se va a negociar comienza. Al igual que el agente del usuario, el agente del proveedor negociará en base a las restricciones impuestas por este, reflejadas en las políticas de uso de los recursos fijadas por el proveedor. El objetivo es llegar a fijar una serie de SLAs (Service Level Agreement) que cumpla con los requisitos del usuario. Este conjunto de SLAs también incluirán aquellos que se negocien con los agentes asociados a los elementos de red del Grid, que también tomaran parte en este proceso.

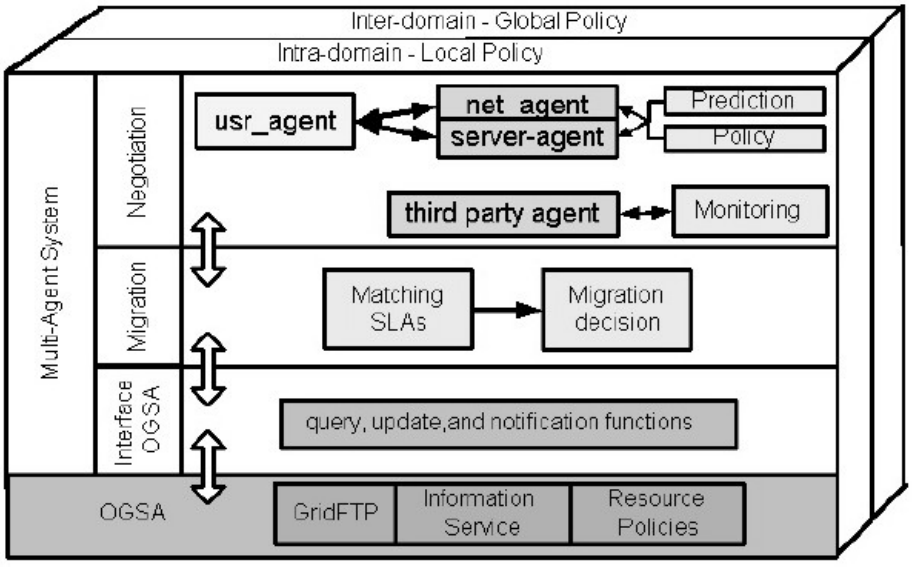

Figura 30. Arquitectura del sistema multiagente

En lo referente a la negociación el sistema ofrece tres tipos de negociación: bilateral, multiissue y chaining. La negociación bilateral implica a dos agentes, ya sean estos el

agente de usuario, el agente asociado al proveedor o el asociado a un elemento de red. Cuando en esta negociación se aborda más de un aspecto (por ejemplo coste y calidad del servicio) esta negociación pasa a llamarse también multiissue, mientras que cuando interviene un agente asociado a un elemento de red (en general el agente de usuario negociará en primer lugar con el agente asociado al proveedor, antes de hacerlo con los asociados a los elementos de red), la negociación también se denomina como chaining. La negociación es iniciada por el agente del usuario, y en caso de ser una negociación de tipo multiissue, éste se encargará de negociar los distintos términos implicados, uno a uno o incluyendo estos términos dentro de un paquete. El agente del proveedor por su parte seleccionará aquellos intervalos a los que ya haya asociado un menor número de SLAs, realizando asimismo una predicción del tiempo que le llevará correr ese trabajo en base a su experiencia previa ejecutando trabajos de una naturaleza similar. Por otro lado el agente del usuario tendrá que negociar finalmente con los distintos agentes asociados a los elementos de red, a fin de minimizar el tiempo que tardan los datos en ser transferidos desde el servidor que los contiene, hasta aquel con el que se ha alcanzado un SLA, de forma que el par de este tipo de servidores con mejor ancho de banda será el que se lleve el contrato. Esta conexión implicará a una serie de elementos de red con sus agentes asociados, que serán con los que el agente del usuario tendrá que negociar teniendo en cuenta que no deberá rebasar con los acuerdos alcanzados, más los que ya ha realizado con los agentes asociados a los proveedores, las restricciones en coste y tiempo impuestas por el usuario. A todo este proceso se le unirá un agente que se encargará exclusivamente de monitorizar los SLAs acordados.
En lo referente a la migración, el sistema por un lado agrupa los SLAs que forman parte de una misma propuesta (por ejemplo aquellos correspondientes a una serie de elementos de red y un servidor dados), y en base a esas agrupaciones de SLAs selecciona aquella que mejor se adapte a las necesidades del usuario. Es entonces cuando se hace la migración de los datos utilizando agentes móviles para las migrar las aplicaciones y GridFTP para migrar los datos. Una vez migrados, la información sobrelos SLAs activados se almacena en el MDS de Globus. Por último una última capa se encarga de comunicar al sistema multiagente con los componentes del Grid (representados en OGSA, que define la arquitectura del Grid basado en servicios).

Propuesta del grupo encabezado por S.S. Manvi del Basaveshwar Engineering College sobre la gestión de recursos en el Grid [33] .

Este sistema utiliza tres tipos de agentes. Por un lado los Job Agents (JA) que se encargan de representar al usuario del Grid, y que como tal guardan información sobre las restricciones de éste debidas a la prioridad del trabajo a enviar, el tipo de trabajo (uso intensivo de entrada/salida o de CPU), el tiempo de ejecución del trabajo, el tamaño de los buffers, el espacio de almacenamiento, la velocidad de procesamiento o el ancho de banda, entre otros. El JA se encarga de llevar a cabo las negociaciones con los agentes asociados a los recursos (Resource Brokering Agents, RBA). Por otro lado se encuentran los RBAs, que se encargan de gestionar los recursos de los proveedores, cuya información de estado reciben a través de los servicios de información del Grid. Los RBAs asociados a distintos recursos se mantienen en comunicación de cara a fijar el precio de los recursos que gestionan en base a la carga que en un momento determinado tenga el Grid. Así por

ejemplo, en casos donde la carga del Grid sea muy elevada un JA pagará un precio muy elevado por saltarse la cola de espera.
A la hora de gestionar las peticiones que llegan de los Jas, los RBAs utilizan un sistema de colas por área del conocimiento donde se enmarca el trabajo, y dentro de cada área sigue una estrategia FIFO a la hora de seleccionar trabajos. En caso de no poder atender un trabajo, el RBA correspondiente se lo redigirá al próximo RBA que será seleccionado bien aleatoriamente, bien circularmente, o bien mediante un esquema de árbol en el que cada nodo padre representa a un RBA que gestiona más recursos que el nodo hijo correspondiente, de forma que el trabajo se envía hacia arriba en el árbol. Por último se encuentra un tercer tipo de agentes, los Resource Monitoring Agents (RMA), que monitorizan los trabajos que se ejecutan dentro de cada uno de los proveedores, enviando dicha información a los servicios de información del Grid, información que será analizada posteriormente por los RBAs.

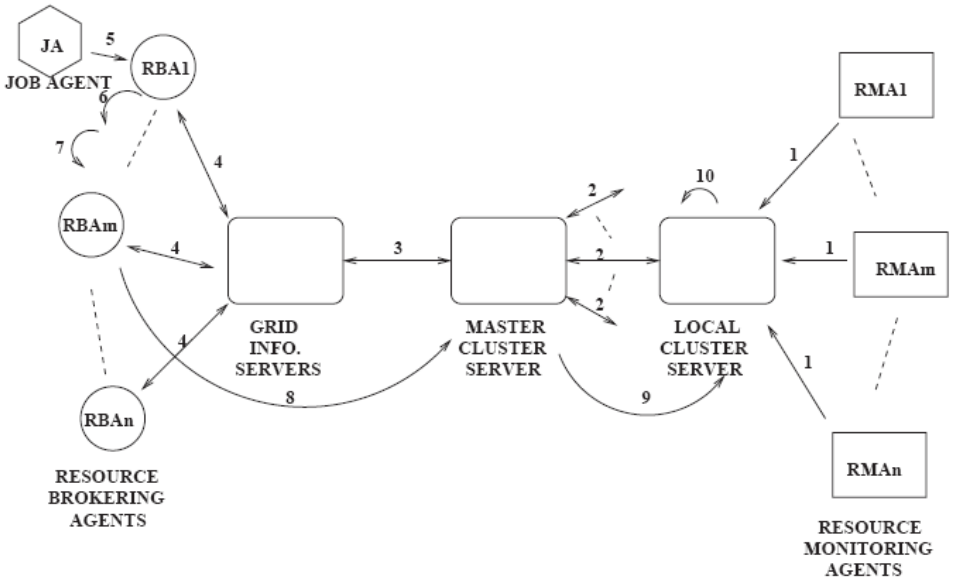

Figura 31. Secuencia de operaciones

A la hora de gestionar la llegada de un trabajo, como se muestra en la figura de la página anterior, en primer lugar los RMAs obtienen la información de los recursos internos de los proveedores, que envian a los servidores de información del Grid. Allí los RBAs se encargan de analizar esa información y crear los precios asociados a los recursos disponibles, de forma que cuando un JA solicite el procesamiento de un trabajo a alguno de los RBAs que tengan recursos disponibles este pueda enviar su trabajo al proveedor que tenga recursos disponibles.

Gestión de datos

Uno de los cuellos de botella en lo que al rendimiento del Grid se refiere es la

transferencia de datos entre los distintos proveedores del Grid. Aunque al hablar de Grid habitualmente se cita la capacidad de procesamiento de los proveedores asociados a este, también existen proveedores que se encargan únicamente de proporcionar capacidad de almacenamiento (son los llamados storage elements). Uno de los principales problemas a la hora de ejecutar un trabajo es transferir tantos los datos como los binarios de las aplicaciones implicadas al proveedor elegido para correr dicho trabajo, esto supone que en aplicaciones que corren miles de millones de trabajos pueda suponer un problema dicha transmisión en lo que al rendimiento del Grid se refiere. Es por ello que son necesarias soluciones que garanticen un almacenamiento que mantenga un ancho de banda lo más elevado posible con los proveedores que ejecutaran dichos datos, idealmente guardando dichos datos en la capacidad de almacenamiento que pueda ofrecer el proveedor que vaya a ejecutarlos. Así pues se hace necesario optimizar el uso del ancho de banda asociado a los elementos de red y a los elementos de almacenamiento.

La problemática asociada a este servicio del Grid se centra entre otros problemas en los fallos de transmisión. Hasta hace relativamente poco tiempo la gestión de los datos, así como la resolución de los errores que estos producían debían ser gestionados directamente por los usuarios. El problema surgía cuando el usuario se desconectaba del Grid, de forma que toda la información sobre la transacción en particular se perdía. Con la llegada de nuevos componentes a los paquetes de middleware, como RFT en Globus y FTS en Glite, y el uso de agentes en los mismos este problema ha desaparecido, en el que han aparecido componentes de gestión de datos que han permitido solucionarlo, permitiendo mejorar de paso la escalabilidad del Grid en lo que ha gestión de datos se refiere. El FTS permite enviar datos a través del Grid controlando al mismo tiempo el uso que se hace de los recursos de red sobre los que correrá esa transferencia, pudiendo así adaptarlos a las necesidades de la situación o a políticas de uso específicas. Su arquitectura se basa en una serie de agentes, que en base a la política de uso selecciona un trabajo de una cola de estos, para gestionarlo en base a una máquina de estados. Estos agentes estarán asociados bien a un recurso de red o a la propia organización virtual. PhEDEx se basa por su parte en un sistema multiagente donde cada agente está asociado a un sitio y tiene una tarea asignada, como por ejemplo replicación de ficheros o decisiones de enrutado, comunicándose a su vez todos los agentes a través del TMDB, una base de datos Oracle multi-servidor.

Otras aplicaciones

Como Foster señala en su artículo Brain Meets Brawn: Why Grid and Agents Need Each Other [28], el uso de agentes para mejorar los servicios ofrecidos por el Grid es el siguiente paso en la evolución de éste. Las tres áreas cubiertas arriba constituyen las que más atención están suscitando, aunque hay muchas otras iniciativas en marcha para introducir los agentes en otros servicios del Grid. Un buen ejemplo son los desarrollos orientados a ofrecer un sistema balanceo de carga dentro de organizaciones virtuales basado en agentes. La arquitectura se basa en asociar a cada proveedor un agente que se encargue de gestionar su carga de trabajo, de forma que los agentes asociados a varios

recursos que se encuentren dentro de una organización virtual, se organizan para repartir esa carga, a fin de minimizar el tiempo de espera del usuario y cumplir los plazos de terminación de las tareas. Aunque existen muchas otras aplicaciones como aquellas orientadas a gestionar la seguridad dentro del Grid.

El uso de agentes software en los servicios Grid es relativamente joven y corresponde a una necesidad de hacer más dinámica la configuración de ciertos servicios, que como se veía en la sección anterior, incluso seguían contando con la necesidad de actuación de un operador humano para la gestión de trabajos. Las necesidades de MAGICGrid a este respecto se centran en la gestión de trabajos y datos mediante estos agentes. A este respecto dos de los proyectos del LHC que se han ido desgranando a lo largo de este proyecto, CMS y ATLAS, cuentan con sendos sistemas de producción basados en un sistema multiagente que se acercan al uso que MAGICGrid quiere hacer de los agentes software. En la siguiente figura se observa la primera versión del sistema producción de ATLAS, este sistema cuenta con una serie de agentes software, llamados supervisores, que se encargan de gestionar los trabajos que los usuarios envían a través de la base de datos (prodDB). Estos supervisores tienen entre sus funciones:

- Transformar la definición genérica de trabajo que se encuentra en la base de datos en un fichero XML que incluya toda aquella información necesaria para su ejecución
- Gestionar los trabajos, enviándolos a los distintos ejecutores que se encargarán de enviarlos al Grid correspondiente, reenviándolos en el caso de haberse producido un error o gestionando adecuadamente los datos devueltos en el caso de que dicha ejecución haya sido correcta.
- Mantener actualizada la base de datos (prodDB), indicando información como el estado de los trabajos, o la localización de los resultados proporcionados por estos.

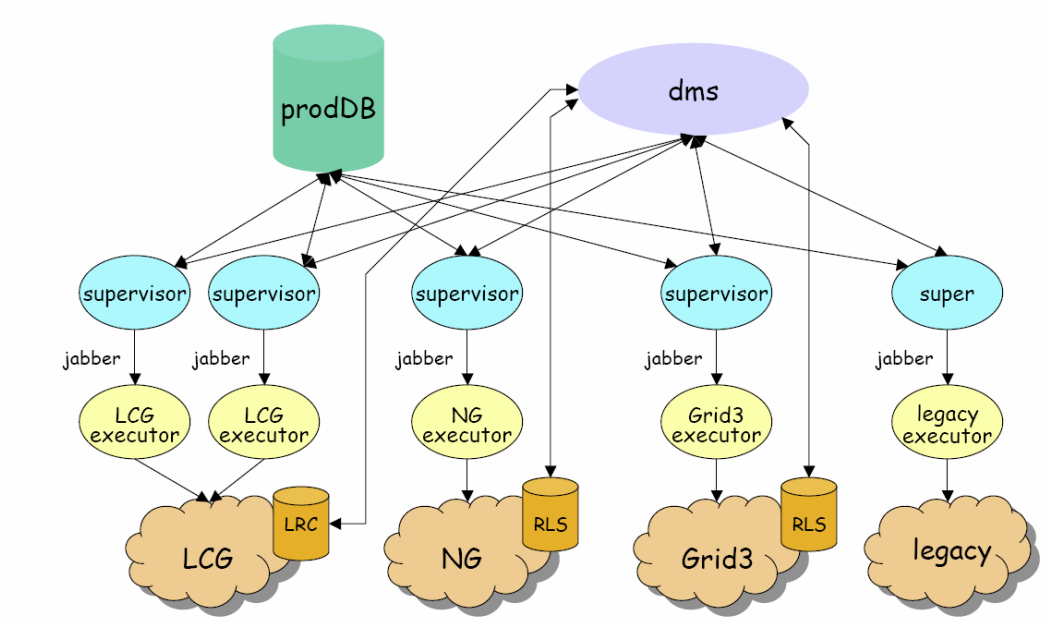

Figura 32. Sistema de producción de ATLAS basado en agentes

Existen dos versiones del supervisor, Windmill y Eowyn, esta última sucesora de la primera, frente a la cual añade un sistema de comunicación síncrona basada en objetos Python. Estos envían las descripciones de los trabajos a ejecutar a los ejecutores a través de XML. Los ejecutores se encargan de transformar dicha descripción del trabajo en formato XML al formato adecuado para el middleware Grid que están gestionando. Existen cuatro ejecutores, Lexor [34] para LCG, Dulcinea para NorduGrid (NG), CaponE para Grid3/OSG (Grid estadounidense) y Legacy para interaccionar con aquellas soluciones previamente utilizadas como sistemas de producción dentro de ATLAS. Todos los ejecutores son similares en cuanto a funcionalidad, siendo implementados en uno u otro lenguaje en base a la API que proporcionen los servicios del Grid al que preste servicio. En el caso de LCG, Lexor hace uso de la API del WMS para definir los JDL asociados a los trabajos que va a enviar. Para ello utiliza la información que le envía el supervisor, complementándola con otra información almacenada por Lexor que es común a todos los trabajos, por ejemplo añadiendo al JDL el requisito de que el nodo donde vaya a ser ejecutado ese trabajo disponga del software de simulación de ATLAS. Lexor, al igual que el resto de los ejecutores, dispone de una pequeña API para gestionar el envío de trabajos, proporcionando así dicho servicio al supervisor. Entre otras funciones se incluye la posibilidad de enviar trabajos, ver el estado del trabajo, transferir los resultados de un trabajo o cancelar dicho trabajo.

En CMS se puede encontrar un sistema similar al de ATLAS, donde los agentes interaccionan directamente con los recursos, como se puede ver en la siguiente figura. El objetivo de esta arquitectura era automatizar el proceso de simulación tanto como fuera posible, ofrecer un sistema escalable y dar soporte a múltiples infraestructuras Grid, en este caso a través del propio agente gracias a uno de los plugins asociados a uno de los componentes de dicho agente, JobSubmmiter. Dichos agentes (ProdAgent) consiguen los trabajos pidiéndoselo al ProdMgr, cuando los recursos asociados a losa agentes se encuentran libres, que se encarga de recolectarlos de los distintos sistemas de envío de trabajos. Para ello los agentes tienen que monitorizar en los distintos computing elements el número de CPUs libres en las colas dedicadas a la organización virtual asociada a CMS. Estos agentes también se encargan de gestionar la ejecución del trabajo y de dividir en subtrabajos y fusionar estos, aquellos trabajos que así lo requieran. Cada trabajo envía la información sobre su estado al ProdAgent a través de un fichero XML que le permitirá a este reenviar el trabajo si se produjera algún tipo de error.

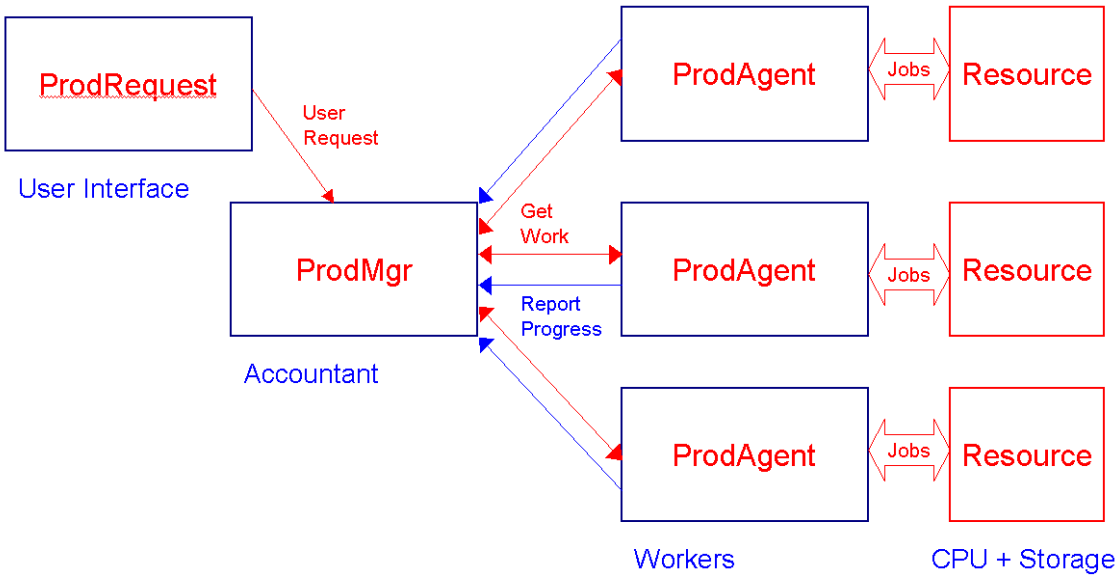

Figura 33. Sistema de producción de CMS basado en agentes

En MAGICGrid con el uso de esta tecnología se pretende ofrecer por un lado flexibilidad a la hora de gestionar los trabajos, permitiendo que los agentes se encarguen de dividir los trabajos en base a la disponibilidad y estado de los recursos del Grid, y a las prioridades de los gestores de la herramienta.

El Observatorio Virtual y su uso en proyectos de Física de altas energías

En el área de observación astronómica y astrofísica en general y en la de astrofísica de partículas en particular, el almacenamiento y la gestión de datos supone uno de los mayores problemas. El análisis de un conjunto de datos, no sólo supone la transmisión de TBs, que incluso contando con una conexión relativamente buena a Internet, puede suponer del orden de horas, sino también con el tiempo de computación necesario, por lo general muy alto y en algunos casos inabordable, debido al tamaño de los datos que se están manejando y a la falta por lo general de recursos computaciones que puedan gestionar adecuadamente este tipo de trabajos. Es por ello que uno de los primeros objetivos de los sistemas de producción Montecarlo fue minimizar dentro de lo posible el movimiento de datos, en especial desde el sistema al usuario. Para catalogar estos datos surgieron los catálogos que de forma similar a las bases de datos con metadatos sobre la producción de simulaciones, se encargaban de gestionar los conjuntos de datos ya creados. Junto a estos también se optó por un sistema más dinámico en lo que a gestión de datos se refiere, utilizando las soluciones que proporcionaba el mundo de los agentes software para diseñar las siguientes generaciones de sistemas de gestión de datos, buen ejemplo de esto es PhEDEx, el sistema de gestión de datos del experimenta CMS. El problema de esta forma de gestión más eficiente es que limitaba el uso de los datos a los grupos participantes en el experimento y a análisis referidos al mismo. Todo esta situación, junto con la necesidad de compartir datos entre los distintos proyectos,

posibilitó la creación del observatorio virtual. El objetivo del observatorio virtual es el de ofrecer al usuario la posibilidad de poder analizar datos de cualquier proyecto, posición, fecha y zona del espectro, desde su ordenador sin tener que instalar ningún tipo de software ni transferir datos. Además de proporcionar un catálogo a todo los datos generados por los distintos proyectos, el observatorio virtual también ofrece la posibilidad de poder visualizar datos de distintos proyectos, de forma que se pueda acceder a toda la información disponible sobre un objeto en particular. El esquema del observatorio virtual es en grandes rasgos el que se muestra en la siguiente figura.

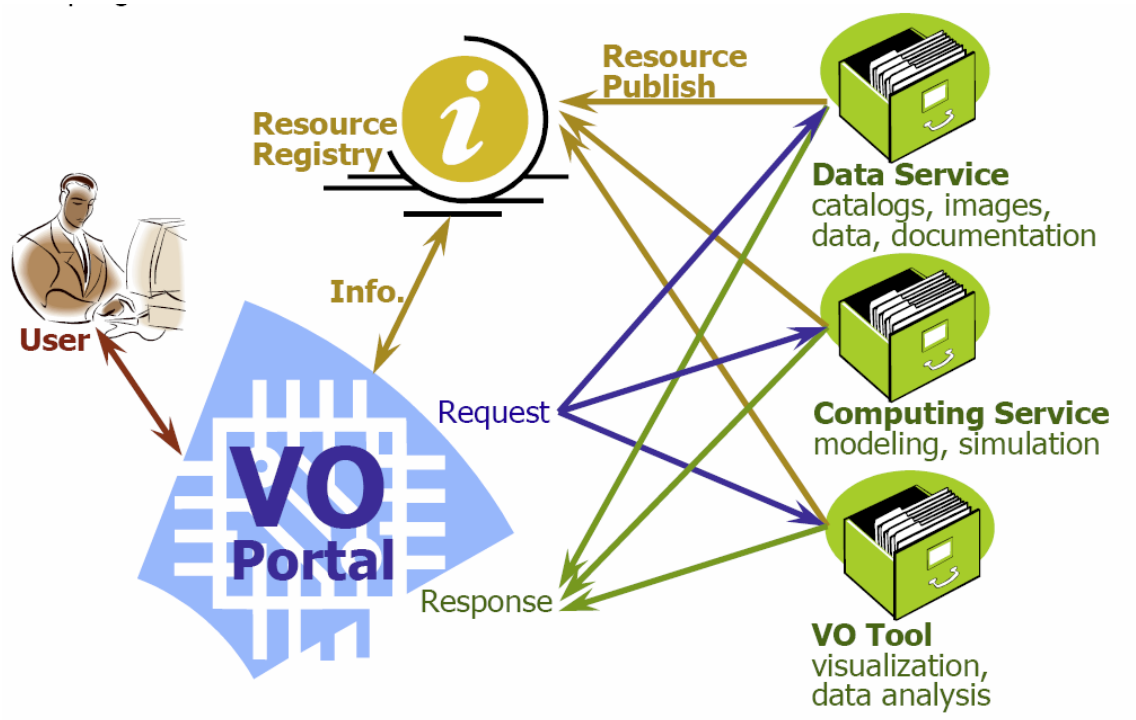

Figura 34. Componentes del sistema del Observatorio Virtual

El usuario accede a los servicios que ofrece la organización virtual a través de un portal web. Dichos servicios pueden ser desarrollados por cualquier organización adscrita al observatorio, la cual tendrá que registrarlos en el registro correspondiente para que los demás usuarios puedan acceder al mismo. De la misma forma que se registran estos servicios, dicho registro también guardará la información sobre que datos y recursos computacionales están disponibles y dónde se encuentran estos. Desde dicho portal, el usuario será capaz de descargar datos, e incluso combinaciones de datos provenientes de distintos proyectos, y acceder a las herramientas de análisis de los mismos. El objetivo final es que estos datos no tengan que ser descargados y que el usuario en vez de utilizar su herramienta de análisis de forma local, lo haga en remoto utilizando la que el observatorio virtual proporciona para tal fin. Entre los objetivos del observatorio también se incluye poder realizar simulaciones cuyos resultados sean posteriormente añadidos al observatorio para su análisis, debido sobre todo al interés que desde las organizaciones que gestionan el observatorio virtual en todo el mundo, se ha mostrado por el mundo

Grid, un área no tan nueva para los proyectos de física de altas energías.

En la colaboración MAGIC actualmente los datos se gestionan directamente desde repositorios, en los que según el interés del grupo o centro, y su papel dentro de la colaboración, se almacenan bien simulaciones, bien datos recién recibidos a través del telescopio o analizados. Esto supone que cada usuario que quiere analizar unos datos dentro de la colaboración debe descargárselos a su directorio local. Otra de las posibles ampliaciones de MAGICGrid sería poder poner los datos que ahí se generen a disposición de toda la comunidad científica en general y del resto de miembros de la colaboración MAGIC en particular. Para ello hay que tener en cuenta la arquitectura sobre la que se generan las simulaciones, siendo recomendable para evitar transferencias de datos innecesarias mantener los datos generados lo más cerca posible de los computing elements que los generan, siendo recomendable su almacenamiento el los storage elements que normalmente van asociados a los primeros. Para ello por cada conjunto de datos generado se tendría que crear un fichero XML siguiendo el estándar VOTable con la información sobre dichos datos.

Capítulo 6. Conclusiones y resultados

El objetivo con este proyecto era doble, por un lado poner en marcha de nuevo la organización virtual MAGIC dentro de EGEE y por otro gridificar las simulaciones Montecarlo realizadas dentro de la colaboración, utilizando para ello una aproximación basada en servicios web, que proporcionara una interfaz sencilla de utilizar.

En lo referente a la organización virtual MAGIC, como se indicaba en el tercer capítulo, se ha retomado la gestión de dicha organización, colaborando con tres centros: PIC, CIEMAT y TU Dortmund. Se han configurado para la organización virtual dos interfaces de usuario, desde donde ya se pueden enviar trabajos a la organización virtual. Se ha configurado el servicio VOMS, asignando nuevos roles y grupos, que ya figuran en las configuraciones de los recursos Grid asociados a estos tres centros, y poniendo en marcha de nuevo el sistema de suscripción a la organización virtual basado en certificados emitidos por las distintas entidades nacionales de certificados adscritas a EUGridPMA. Asimismo, se ha automatizado la instalación del software de simulación, Corsika, en los distintos computing elements, que es fácilmente adaptable a la instalación de cualquier otro software cuya ejecución sea necesaria gridificar dentro de la colaboración. Por último se han realizado pruebas de ejecución de simulaciones Montecarlo utilizando dicha infraestructura, automatizándolas mediante un conjunto de scripts, que permiten la creación de distintos trabajos en los que varíen un conjunto de los parámetros incluidos dentro del fichero de configuración de Corsika. Estos scripts ya han sido utilizados por otros usuarios dentro de la colaboración para ejecutar simulaciones Montecarlo.

Respecto del MAGICGrid, se ha puesto en marcha un nodo Grid basado en el middleware Globus Toolkit sobre el que se ha instalado y configurado para este tipo de instalación el metaplanificador Gridway. Esta configuración además de ser más flexible en lo que a instalación se refiere, permite el envío directo a los computing elements seleccionados, de los trabajos, sin tener que depender del tiempo de espera que el envío de un trabajo a través de una user interface supone, que en algunas ocasiones puede ser bastante elevado. Sobre este mismo nodo se ha creado una aplicación que permite mediante servicios web la definición de trabajos en el Grid, utilizando para ello la API en Java que proporciona Gridway, y el envío de los mismos mediante dicho metaplanificador. Asimismo se ha desarrollado una GUI en Java a través de la cual se pueden configurar los parámetros de Corsika, y remitir dicho fichero de configuración al servidor que se encargará de ejecutarlo. Dicha GUI se ejecuta automáticamente a través de cualquier navegador utilizando Java web start, una de las aplicaciones contenida en la JDK, evitando así la instalación de cualquier tipo de software por parte del usuario. Asimismo, una vez ejecutadas las simulaciones, los resultados de estas se han puesto a disposición de la colaboración a través del mismo servidor web (Apache Tomcat) sobre el que corren los servicios web, basados en Apache Axis.

Finalmente se han propuesto tres áreas para realizar ampliaciones en la herramienta MAGICGrid: uso de metadatos para la gestión de las simulaciones, sistemas multiagente para la gestión de los trabajos dentro de dicha aplicación y la gestión de los datos obtenidos a través del observatorio virtual. Para estas áreas se han estudiado trabajos relacionados, que puedan servir de base para abordar los problemas planteados.

Bibliografía

[1]. I. Foster, What is the Grid? A Three Point Checklist, Grid Today (2002)

[2]. I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid, International Journal of High performance Computing Applications, 15, 3 (2001)

[3]. I. Bird, EGEE-II Grid Operatings Cookbook

[4]. Jones, Bob. *An Overview of the EGEE Project*. 2005. LNCS 3664. P2P, Grid, and Service Orientation

[5]. Gagliardi, Fabrizio. Jones, Bob. Grey, François. Bégin, Marc-Elian. Heikkurinen, Matti. *Building an infrastructure for scientific Grid computing: status and goals of the EGEE project*. 2005. Philosophical Transactions of The Royal Society 363

[6]. Caballero Béjar, José. Tesis doctoral. *Computación Grid para el experimento CMS del LHC*. 2007. Universidad Complutense de Madrid

[7]. De los Reyes López, Raquel. Tesis doctoral. *Search for gamma-ray emission from pulsars with the MAGIC telescope: sensitivity studies, data check and data analysis*. 2008. Universidad Complutense de Madrid

[8]. López Moya, Marcos. Tesis doctoral. *Astronomía Gamma con el Telescopio MAGIC: Observaciones de la Nebulosa y Pulsar del Cangrejo*. 2006. Universidad Complutense de Madrid

[9]. A.M. Hillas, Cerenkov light images of EAS produced by primary gamma. 1985. International Cosmic Ray Conference

[10]. D. Heck, J. Knapp, J.N. Capdevielle, G. Schatz, T. Thouw, CORSIKA: A Monte Carlo Code to Simulate Extensive Air Showers

[11]. Kornmayer, H. Hardt, M., Kunze, M. Bigongiari, C. Mazzucato, M. deAngelis, A. Cabras, G. Forti, A. Frailis, M. Piraccini, M. Delfino, M. *A distributed, Grid-based analysis system for the MAGIC telescope*

[12]. H. Kornmayer, *Proposal for the introduction of Grid technology for the MAGIC telescope*

[13]. Stadie, H. Ernst, M. Ferrando, J. Mankel, R. Wrona, K. *Monte Carlo mass production for the ZEUS experiment on the Grid*. 2006. Nuclear Instruments and Methods in Physics Research A 559

[14]. González de la Hoz, Santiago. Sánchez, Javier. Lozano, Julio. Salt, Jose. Fassi, Farida. March, Luis. Adams, D.L. Poulard, Gilbert. Goossens, Luc and DC2 Production TEAM

(ATLAS Experiment). *ATLAS Data Challenge 2: A Massive Monte Carlo Production on the Grid*. 2005. LNCS 3470. EGC 2005

[15]. Goossens, L. De, K. *ATLAS production system in ATLAS Data Challenge 2*

[16]. Evans, D. Fanfani, A. Kavka, C. Van Lingen, F. Eulisse, G. Bacchi, W. Codispoti, G. Mason, D. De Filippis, N. Hernández, J.M. Elmer, P. *The CMS Monte Carlo Production System: Development and Design*. 2008. Nuclear Physics B.

[17]. Dahan, Maytal. Thomas, Mary. Roberts, Eric. Seth, Akhil. Urban, Tomislav. Walling, David. Boisseau, John R. *Grid Portal Toolkit 3.0 (GridPort)*. 2004. Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)

[18]. Bogdanski, Maciej. Kosiedowski, Michal. Mazurek, Cezary. Wolniewicz, Malgorzata. *Grid Service Provider: How to Improve Flexibility of Grid User Interfaces?.* 2003. LNCS 2657. ICCS 2003

[19]. *Migrating Desktop and Roaming Access.* Whitepaper. 2004. Poznan Supercomputing and Networking Center

[20]. Von Laszewski, Gregor. Gawor, Jarek. Lane, Peter. Rehn, Nell. Russell, Mike. *Features of the Java Commodity Grid Kit*. 2001. Concurrency and computation: practice and experience

[21]. EUGridPMA, http://www.eugridpma.org

[22]. Alfieri, R. Cecchini, R. Ciaschini, V. Dell' Agnello, L. Frohner, Ä. Gianoli, A. Lörentey, K. Spataro, F. *VOMS, an Authorization System for Virtual Organizations*

[23]. Corsika User's Guide, http://www-ik.fzk.de/corsika/usersguide/usersguide.pdf

[24]. Huedo, E. Montero, R.S. Llorente, I.M. *The GridWay framework for adaptive scheduling and execution on grids.* 2005. Scalable Computing: Practice and Experience

[25]. R. St.Denis, M. Burgon-Lyon, A.Doyle, S. Hanlon, P.Millar, A. Baranovski, G. Garzoglio, R. Herber, R. Illingworth, R. Kennedy, A. Kreymer, A. Kumar, L. Lueking, W. Merritt, L. Loebel-Carpenter, A. Lyon, I. Terekhov, J. Trumbo, S. Veseli, S. White, P. Vokac, M. Zimmler, S. Albrand, J. Fulachier, T. Barrass, S. Burke, O. Synge, S. Belforte, M. Leslie, V. Bartsch, S. Stonjek, C. Cioffi, A. Forti, F. Ratnikov, A.Sill, P. Kunszt, G. McCance, K. Nienartowicz, R. Rocha. *Housing Metadata for the Common Physicist using a Relational Database*

[26]. Lefébure, Véronique. Andreeva, Julia. *RefDB: The Reference Database for CMS Monte Carlo Production*. 2003. Computing in High Energy and Nuclear Physics (CHEP'03)

[27]. Grandi, C. *BOSS: a tool for batch job monitoring and book-keeping*. 2003. Computing in High Energy and Nuclear Physics (CHEP'03)

[28]. Foster, I. Jennings, N.R. Kesselman, C. ***Brain Meets Brawn: Why Grid and***

Agents *Need Each Other*, 2004. Third International Joint Conference on Autonomous Agents and Multiagent Systems.

[29]. Oren, N. Norman, T. Preece, A. Chalmers, S. *Policing Virtual Organizations*.

[30]. Shan, B. Han, Y. Wang, H. *Enabling Virtual Organizations with an Agent-Mediated Service Framework*.

[31]. *FIPA Contract Net Interaction Protocol Specification*, 2002. Foundation for Intelligent Physical Agents.

[32]. Nassif, L. Nogueira, J.M. Ahmed, M. Impey, R. Karmouch, A. *Agent-based Negotiation for Resource Allocation in Grid*.

[33]. Manvi, S.S. Birje, M.N. Prasad, B. *An Agent-based Resource Allocation Model for computational grids*, 2005. Multiagent and Grid Systems – An International Journal.

[34]. De Salvo, A. Negri, G. Rebatto, D. Vaccarossa, L. *LEXOR, the LCG-2 Executor for the ATLAS DC2 Production System*