Universidad Zaragoza
1542

Trabajo Fin de Máster

Integración dinámica de entornos de computación heterogéneos para la ejecución de *workflows* científicos

Autor

Sergio Hernández de Mesa

Director

Pedro Álvarez Pérez-Aradros

Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas
Grupo de Integración de Sistemas Distribuidos y Heterogéneos (GIDHE)

Junio 2012

Integración dinámica de entornos de computación heterogéneos para la ejecución de *workflows* científicos

RESUMEN

Los *workflows* científicos se caracterizan por estar compuestos por un elevado número de tareas computacionalmente muy costosas. Las necesidades planteadas por este tipo de *workflow* hacen necesaria la utilización de entornos de computación capaces de satisfacer estos requisitos de computación. En este contexto, la computación Grid ha emergido como un paradigma adecuado para la ejecución de *workflows* científicos gracias a la capacidad computacional y las comunicaciones en red de estos entornos. No obstante, esta nueva "sociedad" compuesta por Grids y *workflows* científicos todavía mantiene abiertos un amplio abanico de retos y dificultades.

La posibilidad de ejecutar *workflows* programados en diferentes lenguajes sobre un mismo entorno de computación, la integración de entornos de computación heterogéneos bajo una misma infraestructura, y la posibilidad de ejecutar diferentes partes de un mismo *workflow* en diferentes entornos de computación son algunos de los principales problemas existentes. Como primer paso para la resolución de estos problemas, se desarrolló una infraestructura que integra diferentes entornos de computación heterogéneos de forma transparente para el usuario y que permite ejecutar *workflows* programados en diversos lenguajes ampliamente aceptados por la comunidad científica. De esta forma, se proporcionó una infraestructura capaz de solucionar los retos anteriores. Un aspecto ortogonal a estos retos no considerado en la infraestructura propuesta es el proceso de asignación de tareas a los recursos disponibles en los diferentes entornos integrados (*meta-scheduling*). Este proceso es clave para la definición de soluciones maduras y completas a los problemas expuestos.

Para avanzar en el desarrollo de la solución propuesta y mejorar la infraestructura, en esta Tesis Fin de Máster se propone una estrategia de *meta-scheduling* basada en técnicas de simulación que permite asignar dinámicamente el entorno de ejecución a utilizar en cada una de las tareas de un *workflow*. Para ello, se ha integrado en la infraestructura un *meta-scheduler* que, para cada tarea, selecciona el entorno de ejecución más adecuado utilizando un algoritmo de optimización del tiempo de ejecución. La información utilizada para esta toma de decisiones proviene de los resultados de simular la ejecución de las tareas en los entornos de computación disponibles. Para soportar este proceso, se ha diseñado un simulador genérico, adaptable y extensible basado en Alea. Para cada entorno de computación, una instancia de este simulador ha sido customizada e integrada en la infraestructura. Asimismo, se ha definido una metodología para la creación de *workloads* dinámicos que permite simular las tareas en condiciones reales de carga. El uso de estos *workloads* y el propio diseño de los simuladores, capaces de capturar la complejidad inherente de los entornos de computación, han permitido obtener un elevado grado de precisión en las simulaciones, tal y como se ha demostrado en la validación experimental realizada. Como consecuencia, se ha conseguido mejorar el rendimiento de los *workflows* ejecutados.

Finalmente, la viabilidad y beneficios de la solución propuesta se muestran mediante su aplicación a un *workflow* real en el dominio de la computación científica, el *workflow* de análisis Inspiral. Para este caso de uso, la utilización de la infraestructura con la estrategia de *meta-scheduling* propuesta ha permitido obtener una mejora del rendimiento de un 59 % respecto a la ejecución del *workflow* completo en el *cluster* Hermes del I3A y una mejora de un 111 % respecto a la ejecución del *workflow* en el Grid AraGrid.

Índice

Capítulo 1 | **Introducción**

El creciente interés de la comunidad científica por automatizar de manera sistemática la ejecución de sus experimentos ha supuesto el impulso definitivo a la computación científica y, en particular, a los *workflows* científicos. Este tipo de *workflow* presenta unas características muy particulares que condicionan su ejecución: están compuestos por actividades complejas desde el punto de vista de los recursos computacionales necesarios para su ejecución, gestionan grandes volúmenes de datos como entrada/salida de las tareas ejecutadas, y necesitan gestionar adecuadamente la disponibilidad de recursos *hardware* y *software* heterogéneos. Estas características han alentado el uso de entornos de computación tipo Grid para el despliegue y ejecución de estos *workflows*, con el objetivo común de obtener el máximo aprovechamiento a colecciones de recursos heterogéneos y distribuidos geográficamente [1]. En este sentido, se ha avanzado notablemente en la comprensión de la naturaleza intrínseca de los *workflows* científicos y en los requisitos necesarios para su correspondiente ejecución en entornos Grid. No obstante, aún son numerosos los retos abiertos para este modelo de solución.

1.1 Problema a resolver

Algunos de los principales problemas existentes en el ámbito de *workflows* científicos y entornos de computación Grid son: la posibilidad de ejecutar *workflows* programados en diferentes lenguajes, la integración de entornos de computación heterogéneos bajo una misma infraestructura, o la ejecución de partes de un mismo *workflow* sobre diferentes entornos de ejecución.

Dentro de la línea de investigación del autor, se pretende proporcionar una solución integrada a los problemas anteriores. Como primer paso, se desarrolló un prototipo de infraestructura que permite integrar diferentes entornos de computación heterogéneos y ejecutar *workflows* programados independientemente del entorno de ejecución y utilizando diferentes lenguajes. Este trabajo fue presentado por el autor de esta Tesis Fin de Máster como Proyecto Fin de Carrera [2]. Sin embargo, la infraestructura desarrollada presenta limitaciones en la forma de asignar trabajos a diferentes entornos de ejecución. Estas limitaciones (ver Capítulo 2) provocan que el tiempo de ejecución de los *workflows* pueda verse penalizado en la mayoría de las ejecuciones.

En esta Tesis Fin de Máster se intenta dar solución al problema anterior mediante la definición de técnicas avanzadas de asignación de tareas a recursos y la utilización de información dinámica que ayude a seleccionar el entorno de ejecución más adecuado para cada tarea. Utilizando estos mecanismos, partes de un mismo *workflow* podrían ser ejecutadas en diferentes entornos, algo que a día de hoy no es posible con los sistemas existentes. Por tanto, este trabajo representa la evolución natural del Proyecto de Fin de Carrera realizado previamente, proponiendo una solución al tercero de los problemas comentados: la asignación dinámica de tareas a entornos de computación heterogéneos. En todo caso, este trabajo no soluciona todos los problemas y retos existentes, sino que todavía existen diferentes aspectos que deben ser estudiados e incorporados a la infraestructura (véase la sección 5.1).

1.2 Estado del arte

Han surgido diferentes trabajos de investigación que tratan de dar respuesta a los problemas anteriores dentro del contexto de programación de *workflows*, construcción de entornos de ejecución y desarrollo de arquitecturas que den soporte tanto a la programación como a la ejecución de *workflows* científicos. En esta sección, revisaremos los trabajos más relevantes exponiendo sus contribuciones principales.

1.2.1 Programación de *workflows* científicos

En la actualidad existe una amplia variedad de sistemas de gestión de *workflows*. La comparación detallada de estos sistemas ha puesto de manifiesto las diferencias existentes entre las distintas aproximaciones desde el punto de vista de la programación, despliegue y ejecución de *workflows* [3, 4]. Esta heterogeneidad provoca que las propuestas actuales presenten un alto grado de acoplamiento entre los *workflows* científicos y los entornos Grid concretos sobre los que serán ejecutados. En otras palabras, los *workflows* científicos son programados para ser ejecutados en un Grid concreto. Este acoplamiento limita la flexibilidad de las soluciones y provoca, desde el punto de vista del programador de *workflows* científicos, una serie de dificultades que requieren ser resueltas.

Entre estas dificultades destacan las siguientes: los *workflows* no son directamente portables entre diferentes entornos de ejecución; resulta costoso programar nuevos *workflows* reutilizando *workflows* más simples programados en distintos lenguajes (esto dificulta el aprovechamiento de iniciativas como myExperiment[1], un repositorio de *workflows* científicos de acceso público); es complejo programar *workflows* con fuertes requisitos de cómputo para que distintas partes de su flujo puedan ser ejecutadas en diferentes entornos de ejecución; el fuerte acoplamiento entre *workflow* y entorno de ejecución provoca en la mayoría de los casos que los administradores deban realizar tareas de configuración costosas previas al despliegue del experimento; y resulta difícil integrar en estas soluciones paradigmas de computación de interés que han aparecido recientemente y que permiten la optimización del uso de recursos en diferentes escenarios, como por ejemplo los denominados *Green/Cloud Computing*.

Aunque no existe una solución completa a las limitaciones previas, determinados trabajos abordan parcialmente alguna de ellas. Uno de los trabajos más destacados es el proyecto europeo SHIWA[2] (*SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs*). Su principal objetivo es el desarrollo de un conjunto de tecnologías de interoperabilidad que permitan compartir y reutilizar *workflows* entre comunidades de usuarios que habitualmente trabajen con distintos sistemas de gestión. Para la consecución de este objetivo se desarrollan dos ideas básicas. En primer lugar, un *workflow* podría ser encapsulado como un servicio que es ejecutado sobre un entorno concreto. La interfaz del servicio abstraería los detalles concretos y facilitaría su integración en otros *workflows* (interoperabilidad de grano grueso). Y, en segundo lugar, la posibilidad de traducir un *workflow* cualquiera a una representación común, llamada IWIR (*Interoperable Workflow Intermediate Representation*), y posteriormente disponer de herramientas que sean capaces de ejecutarla en distintos sistemas de gestión [5] (interoperabilidad de grano fino).

Por otro lado, independientemente del lenguaje concreto utilizado para modelar los *workflows*, deben proporcionarse mecanismos para abstraer al usuario de los detalles de bajo nivel de interacción con los entornos de ejecución y el lenguaje de representación utilizado por el *middlewares*. Esto permitiría al usuario centrarse en los detalles funcionales del *workflow* y

[1] http://www.myexperiment.org/
[2] http://www.shiwa-workflow.eu/

las relaciones existentes entre sus componentes [6], en lugar de en la interacción con el entorno de ejecución. Para ello, existe una interesante iniciativa de estandarización, llamada JSDL (*Job Submission Description Language*) [7], que define una representación estándar para los trabajos que son ejecutables sobre un entorno tipo Grid. La composición de trabajos concretos establece un *workflow*; por tanto, una evolución natural de esta especificación sería hacia la definición de un lenguaje estándar de *workflows*. En cualquier caso, el estándar en su estado actual ha motivado la aparición de propuestas que, aprovechando la orientación de servicios, facilitan la ejecución de trabajos JSDL en entornos heterogéneos [8, 9]. Una interfaz de servicio para el envío de estos trabajos abstrae al programador de los detalles específicos del entorno de ejecución responsable.

Como último paso para la solución de los problemas comentados, la descripción de los *workflows*, además de estar basada en lenguajes estándar y ser portable, reutilizable e interoperable, debe ser completamente independiente del entorno de ejecución empleado. Esto permitiría que los *workflows* fueran ejecutados en entornos de ejecución heterogéneos sin que fuese necesario realizar cambios sobre el *workflow*, lo cual, a su vez, facilitaría la reutilización y compartición de los *workflows*. Este tipo de solución ha sido explorada en portales de computación científica como P-GRADE [10] mediante el empleo de un lenguaje de alto nivel y la traducción automática del mismo al lenguaje concreto usado por el *middleware* correspondiente.

1.2.2 Entornos de computación heterogéneos para la ejecución de *workflows* científicos

El paradigma de computación Grid surgió como respuesta a las elevadas necesidades computacionales de los experimentos científicos. Prometía la puesta en funcionamiento de grandes entornos de computación mediante la integración y colaboración de diferentes entornos distribuidos a nivel mundial [11]. Sin embargo, a día de hoy nos encontramos muy alejados de dicha perspectiva debido a varios problemas que dificultan enormemente la creación de nuevos entornos de ejecución basados en la integración de entornos ya existentes.

Por un lado, la heterogeneidad de los entornos de computación Grid y la gran diversidad de sistemas *middleware* (Condor [12], Globus [13], gLite [14], etc.) que gestionen los mismos, dificultan el proceso de integración. Por otro lado, los entornos de computación existentes pertenecen a organizaciones y dominios administrativos diferentes que imponen barreras a su integración, colaboración y uso conjunto [15]. Estos problemas han limitado la aplicación de las diferentes soluciones propuestas a escenarios en los que no existen barreras administrativas y escenarios con diferentes entornos de ejecución homogéneos o, al menos, con *middlewares* interoperables. La dificultad para superar los problemas anteriores, ha llevado a la búsqueda de nuevas técnicas que permitan crear entornos de computación globales.

Con este propósito ha surgido la computación Cloud, o computación en la nube, como una forma de ver la computación como un servicio global [16]. Dentro de esta visión global, el Cloud se presenta como respuesta a numerosos problemas y necesidades a través de diferentes paradigmas: IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*) y SaaS (*Software as a Service*). Para la ejecución de *workflows* científicos, el paradigma IaaS propone la construcción de entornos de computación basados en la provisión bajo demanda de máquinas virtuales. La aplicación del mismo ha llevado al desarrollo de nubes de acceso público, o nubles públicas, y nubes de acceso privado, o nubes privadas.

Las nubes públicas, como Amazon EC2[3] (*Amazon Elastic Cloud*), corresponden a entornos de computación en los que un proveedor de servicios *Cloud* ofrece una visión de recursos infinitos, proporcionando al usuario cualquier número de recursos en forma de máquinas virtuales. De esta forma, el usuario puede cubrir sus necesidades pagando sólo por los recursos que utiliza.

[3] http://aws.amazon.com/es/ec2/

Las nubes privadas corresponden a una nueva forma más flexible de organizar entornos de computación tipo Grid en la que los usuarios pueden ejecutar sus trabajos en máquinas virtuales personalizadas. Asimismo, existen nubes híbridas que toman como base una nube privada y recurren al uso de nubes públicas cuando los usuarios solicitan más recursos de los disponibles.

Con este nuevo panorama, en el que ha aumentado enormemente la heterogeneidad de los entornos de computación al convivir *clusters*, Grids y Clouds, se hace necesaria la definición de nuevas herramientas que permitan integrar diversos entornos de una manera mucho más flexible y dinámica de acuerdo a las posibilidades, necesidades y requisitos de cada usuario concreto.

Para ello, han surgido alternativas que engloban diferentes entornos proporcionando un único punto de acceso, es el caso de los portales (P-GRADE [10], HPC-Europa [17], etc.) y el trabajo previo desarrollado como parte de esta línea de investigación [2]. Estos trabajos se basan en el desarrollo de una capa de mediación que abstrae los detalles específicos de cada entorno particular y proporciona una interfaz única para el despliegue y ejecución de *workflows*.

Sin embargo, estas soluciones presentan poca flexibilidad en cuanto al proceso de *scheduling*. Los portales obligan al usuario a seleccionar el entorno de ejecución a priori (el *scheduling* es estático y guiado por el usuario) y en nuestro trabajo previo los diferentes entornos de computación compiten por ejecutar los trabajos sin tener en cuenta cuál es el más adecuado. Obviamente, sería deseable que el usuario sólo fuera responsable de programar su experimento y la propia infraestructura de integración determinara el entorno de computación más apropiado en base a los requisitos indicados y la calidad de servicio requerida por el usuario. Este tipo de decisiones deberían de ser adoptadas por *meta-schedulers* [3, 18].

1.2.3 Arquitecturas de integración

Existen diferentes modelos arquitecturales para la construcción de las infraestructuras de integración anteriores. Una de las principales alternativas consiste en el desarrollo de una estructura jerárquica dominada por un *meta-broker* que integra diferentes entornos de computación y gestiona la ejecución de los trabajos enviados [18, 19]. Esta solución se basa en la definición de estándares de interoperabilidad que lidien con la heterogeneidad de los entornos de ejecución y permitan su integración. Por tanto, la necesidad de usar estándares hace que la solución sea inviable, al menos a corto plazo. Además, el uso de un *meta-broker* añade una nueva barrera administrativa al ser necesario determinar la organización encargada de su gestión.

Otra posibilidad es utilizar un esquema descentralizado y cooperativo en el que los diferentes entornos de ejecución se comuniquen siguiendo un estilo P2P. Este tipo de organización se ha usado para formar los denominados Grids globales y las federaciones de Grids [20, 21]. No obstante, esta solución se limita a que todas los entornos utilicen un mismo *middleware* o a que estos sean interoperables y al establecimiento de acuerdos que eliminen las barreras administrativas. Asimismo, pueden utilizarse modelos híbridos [15], los cuáles buscan beneficiarse de disponer de una estructura jerárquica y de una red descentralizada. En cualquier caso, este tipo de solución arquitectural también se ha visto limitado por los problemas anteriores.

Independientemente del modelo arquitectural utilizado, la idea de integrar una componente (*meta-scheduler*) que seleccione el recurso en el que ejecutar cada tarea no es nueva [22, 23]. No obstante, en el contexto de integración de entornos de computación heterogéneos, aún deben ser abordadas distintas cuestiones. La asignación de tareas a recursos debería realizarse de forma dinámica durante la ejecución del *workflow*, considerando la naturaleza evolutiva de los entornos integrados y la propia complejidad de los experimentos científicos. Esta asignación requiere conocer el estado de los recursos disponibles y sus prestaciones. La actualización de esta información es compleja y costosa y, además, son necesarios mecanismos de razonamiento complejos que sean capaces de aprovechar la misma [24].

La utilización de técnicas de predicción ha sido muy estudiada para este aspecto [25, 26, 27]. Estas técnicas pretenden obtener, a través del análisis de *logs* de ejecuciones previas, una estimación de cuál será el comportamiento de los trabajos en ejecuciones futuras. Sin embargo, suelen focalizarse en un número limitado de parámetros (tiempo de cola, tiempo de ejecución etc.) por lo que no son capaces de capturar toda la complejidad que entrañan los entornos Grid; como por ejemplo, la posibilidad de que los trabajos de un usuario sean expulsados para ejecutar trabajos de un usuario con mayor prioridad.

1.3 Objetivos

Como paso previo a este trabajo, el autor realizó como Proyecto Fin de Carrera un prototipo de infraestructura que permitía solucionar algunos de los problemas existentes en el ámbito de ejecución de *workflows* científicos en entornos de computación heterogéneos [2]. En dicho trabajo se abordó la posibilidad de ejecutar *workflows* programados en diferentes lenguajes utilizando varios entornos de computación heterogéneos.

Sin embargo, la infraestructura presentaba limitaciones en cuanto al proceso de selección del entorno utilizado para ejecutar cada una de las tareas de un *workflow* (*meta-scheduling*). Este proceso podía ser guiado por el usuario o determinado por la infraestructura, usando para ello un modelo en el que diferentes entornos de ejecución competían para ejecutar trabajos y se seleccionaba un entorno de manera no determinista. Al no tener en cuenta ningún aspecto referente a la idoneidad de ejecutar un tarea en cada uno de los entornos disponibles, este modelo podía provocar una disminución del rendimiento de los *workflows*.

Como respuesta a la limitación anterior, en esta Tesis Fin de Máster se plantea como objetivo principal integrar en la infraestructura mecanismos que permitan que sea la propia infraestructura la que decida el entorno más adecuado para ejecutar una tarea en base a información dinámica acerca del estado de los entornos disponibles. Concretamente, se plantea la integración de una componente de *meta-scheduling* dentro de la infraestructura propuesta. Esta componente debe encargarse de decidir el entorno de ejecución más adecuado en base a criterios dinámicos, que permitan realizar un óptimo aprovechamiento de los recursos disponibles, mejorando las prestaciones globales de los *workflows*.

Asimismo, se plantea la utilización de técnicas de simulación que permitan guiar las decisiones de *meta-scheduling* en base al comportamiento esperado tanto de los entornos de ejecución como de las propias tareas. Por tanto, se estudiará una alternativa que no ha sido todavía explorada en el contexto de *meta-scheduling*, pero que podría resultar adecuada para capturar y abordar la complejidad de los entornos Grid.

Finalmente, se pretende utilizar la infraestructura desarrollada para la ejecución de casos de estudio en el dominio de la computación científica. El objetivo es probar de manera experimental la validez y viabilidad del enfoque propuesto y comprobar sus beneficios.

1.4 Contexto del trabajo

Esta Tesis Fin de Máster ha sido realizada dentro del Grupo de I+D GIDHE[4] (Grupo de Integración de Sistemas Heterogéneos y Distribuidos) del Departamento de Informática e Ingeniería de Sistemas (DIIS) de la Universidad de Zaragoza, reconocido como grupo Emergente por el Gobierno de Aragón. Además, el trabajo está respaldado por el proyecto TIN 2010-17905 del Ministerio de Economía y Competitividad, a través de la Secretaría de Estado

[4] http://www.gidhe.es/

de Investigación, Desarrollo e Innovación, titulado "Una aproximación a la gestión flexible y dinámica de *workflows* científicos basada en Redes Objeto".

El grupo GIDHE tiene acceso y usa regularmente distintos entornos de computación heterogéneos: el *cluster* Hermes del Instituto de Investigación en Ingeniería de Aragón (I3A)[5] y, gracias a su participación en el proyecto internacional PIREGRID[6], los Grids AraGrid y PireGrid. También dispone de acceso al entorno de computación Cloud de Amazon (Amazon EC2)[7].

Este trabajo se enmarca dentro de una de las principales líneas de investigación del grupo, en la cual se plantea la integración de diferentes entornos de computación heterogéneos que permitan dar soporte a la ejecución flexible de *workflows* científicos. Dicha línea, comenzó con el desarrollo de un prototipo de infraestructura presentado como Proyecto Fin de Carrera por el autor de este trabajo y ha sido continuada con la realización de este Trabajo Fin de Máster.

Finalmente, las diferentes líneas abiertas por los trabajos anteriores pretenden ser abordadas durante el desarrollo de la tesis doctoral del autor, la cual tiene como objetivo desarrollar una infraestructura que proporcione una solución completa a los diferentes problemas y retos existentes en el ámbito de la ejecución de *workflows* científicos en entornos de computación dinámicos y heterogéneos. Para la realización de la tesis doctoral, el autor dispone de una beca de Formación y Contratación de Personal de Investigación concedida por la Diputación General de Aragón.

1.5 Organización de la memoria

El resto de esta memoria se organiza de la siguiente forma:

- En el Capítulo 2 se presenta la arquitectura en base a la cual se diseñó el prototipo de infraestructura de integración desarrollado como trabajo previo.
- En el Capítulo 3 se propone una técnica de *meta-scheduling* basada en simulación para la asignación dinámica de tareas en entornos de ejecución heterogéneos.
- En el Capítulo 4 se aplica la técnica de *meta-scheduling* propuesta a un caso de uso de un *workflow* real, el *workflow* de análisis Inspiral.
- En el Capítulo 5 se recogen las conclusiones extraídas de este trabajo y se presentan las líneas futuras de investigación abiertas por este trabajo.

Adicionalmente, se incluyen como anexo los artículos científicos en los que el autor de esta Tesis Fin de Máster ha participado como autor:

- En el Anexo A se proporciona el artículo presentado en la *III International Conference on Cloud Computing, GRIDs, and Virtualization* (CLOUD COMPUTING 2012) en su formato original.
- En el Anexo B se proporciona el artículo presentado en la *II International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH 2012) en su formato original.
- En el Anexo C se proporciona el artículo presentado en las VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2012) en su formato original.

[5] http://i3a.unizar.es/
[6] http://www.piregrid.eu/
[7] http://aws.amazon.com/es/ec2/

Capítulo 2 | Antecedentes: Una infraestructura para la integración dinámica de entornos de computación

Los problemas existentes en cuanto a la integración de entornos de computación heterogéneos para la ejecución de *workflows* científicos, hacen necesaria la definición de herramientas más flexibles y dinámicas que permitan resolver los retos planteados. Con este propósito, como trabajo previo a esta Tesis Fin de Máster, se elaboró una infraestructura orientada a servicios que permite dar solución a algunos de los problemas existentes [2, 28, 29]. Las principales contribuciones de esta solución son:

- La posibilidad de ejecutar *workflows* programados utilizando diferentes lenguajes ampliamente aceptados por la comunidad científica.
- La programación de *workflows* de manera independiente del entorno de ejecución.
- La integración de diferentes entornos de computación heterogéneos y componentes de gestión de forma dinámica y completamente transparente al usuario.
- La ejecución de diferentes tareas de un mismo *workflow* en diferentes entornos de computación.

Sin embargo, la infraestructura también presenta limitaciones. La principal limitación tiene que ver con el proceso de selección del entorno de computación utilizado para ejecutar cada tarea, para el cual se usa un modelo en el que los entornos capaces de ejecutar una tarea compiten por su ejecución. Este modelo no tiene en cuenta factores clave como la carga de las infraestructuras y puede llevar a una mala utilización de los recursos que se traduzca en una degradación de las prestaciones del *workflow*.

Para resolver esta limitación, en este trabajo se presenta una técnica de *meta-scheduling* que permite solucionar el problema anterior. La misma será presentada en el Capítulo 3. Mientras, para que el lector comprenda las características, virtudes y limitaciones de la infraestructura planteada, en este capítulo se van a presentar los aspectos fundamentales de la misma.

La Figura 2.1 muestra el diseño arquitectural de la infraestructura. El mismo consta de varias capas: en la parte superior de la figura, se refleja la interacción entre el programador y la infraestructura a través de diferentes lenguajes de descripción de *workflows*; en el centro se muestra la arquitectura interna de la infraestructura; y en la parte inferior, se indican los diferentes entornos de computación integrados junto con el *middleware* encargado de su gestión. A continuación, describiremos las características más relevantes de cada una de estas capas.

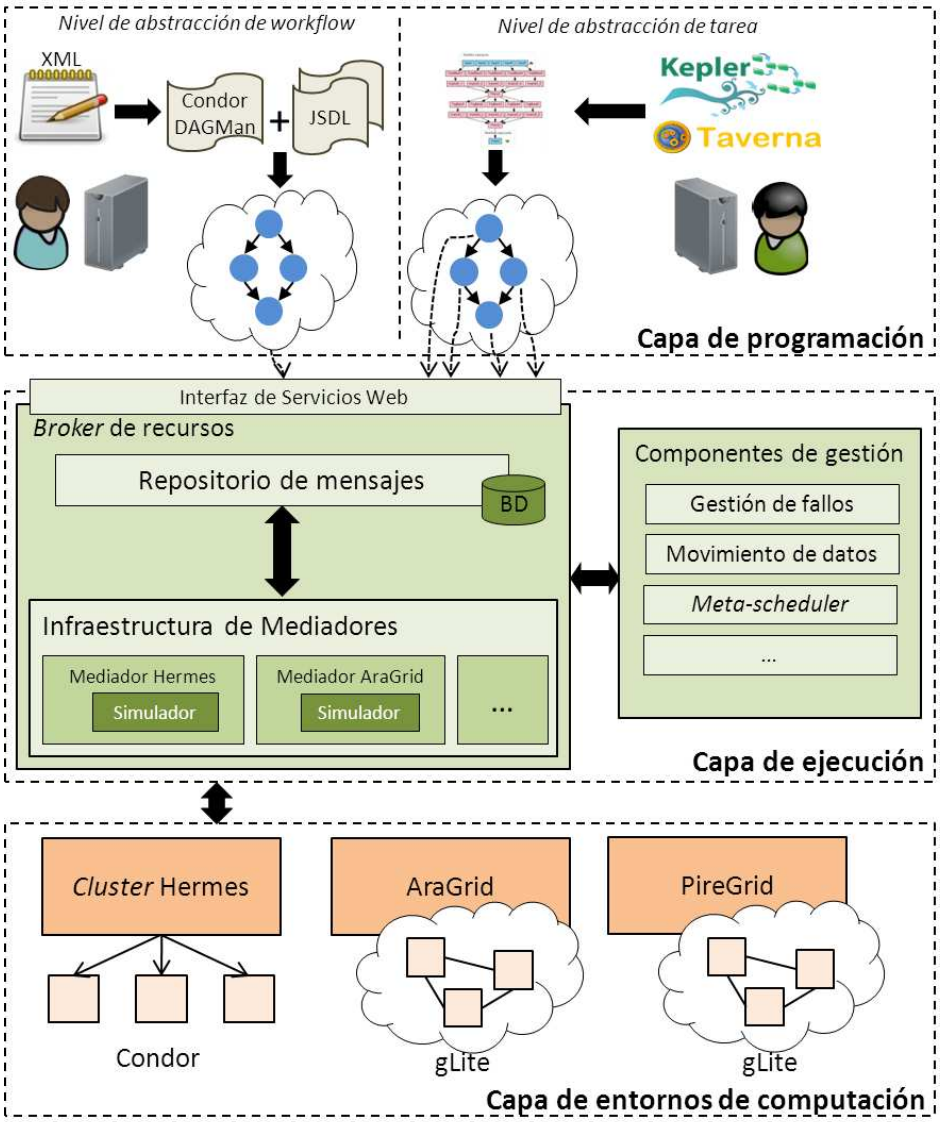

Figura 2.1: Diseño arquitectural de la infraestructura propuesta para la ejecución flexible de *workflows* científicos y la integración de entornos de computación heterogéneos.

2.1 Capa de programación de *workflows*

Desde el punto de vista de la programación de *workflows*, la infraestructura sigue una orientación a servicios [29]. Este enfoque ofrece al usuario la visión de la infraestructura como un servicio de ejecución de *workflows* y tareas computacionalmente costosas. Asimismo, para permitir diferentes tipos de interacción y cubrir las necesidades de diferentes usuarios, la infraestructura permite trabajar a dos niveles de abstracción: nivel de *workflow* y nivel de tarea.

El nivel de abstracción de *workflow*, reflejado en la parte superior izquierda de la Figura 2.1, permite solicitar la ejecución completa de un *workflow*, de forma que la infraestructura se encarga de la gestión de su ciclo de vida y el usuario sólo debe describir las relaciones existentes entre las tareas usando el lenguaje de Condor DAGMan [12].

Por su parte, el nivel de abstracción de tarea, reflejado en la parte superior derecha de la Figura 2.1, permite ejecutar *workflows* programados en el lenguaje utilizado por alguno de los diferentes sistemas de gestión de *workflows* existentes. De esta forma, pueden programarse

workflows usando, por ejemplo, redes de referencia (una subclase de las Redes de Petri de Alto Nivel) [30], Taverna [31] o Kepler [32]. En este caso, es el sistema de gestión de *workflows* el que se encarga de controlar el ciclo de vida del *workflow* y solicitar la ejecución individual de las tareas que componen el mismo bajo demanda.

Independientemente del nivel de abstracción empleado y el lenguaje utilizado para programar el *workflow*, se utiliza el lenguaje *Job Submission Description Language* (JSDL) para describir las tareas de un *workflow*. JSDL [7] es un lenguaje estándar propuesto por el *Open Grid Forum*[1] para la descripción textual de tareas, utilizando una sintaxis XML. El mismo ha sido ampliamente utilizado en entornos de computación Grid. En nuestro caso, su utilización, junto con el diseño desacoplado de la infraestructura, permite programar *workflows* de forma completamente independiente del entorno de ejecución. De esta forma, la infraestructura es la que selecciona internamente el entorno de computación a utilizar. En cualquier caso, se ofrece la posibilidad de que el usuario indique explícitamente la infraestructura de ejecución para no limitar el uso de la misma y proporcionar al programador la posibilidad de decidir dónde ejecutar cada tarea.

2.2 Capa de ejecución de *workflows*

Internamente, la infraestructura se diseña con una arquitectura de componentes que se traduce en un diseño flexible en el cual las diferentes componentes se encuentran desacopladas y pueden ser sustituidas, adaptadas o modificadas de forma dinámica y transparente para el usuario. Concretamente, la capa de ejecución está formada por un *broker* de recursos y un conjunto de componentes de gestión. El *broker* constituye el núcleo de la infraestructura, encargándose de gestionar la interacción con el exterior, conocer el estado de los entornos de ejecución y permitir la comunicación entre las diferentes componentes del sistema. Por su parte, las componentes de gestión ofrecen diferentes funcionalidades encaminadas a la gestión del ciclo de vida de los *workflows*.

El *broker* está formado por un repositorio de mensajes y una infraestructura de mediadores. La comunicación entre las diferentes componentes se realiza a través de mensajes que contienen información de diversa naturaleza y que se almacenan en el repositorio. La implementación del repositorio de mensajes se ha inspirado en el modelo de coordinación Linda [33]. Los mensajes se codifican como tuplas y son almacenados en un espacio de tuplas. La interfaz del repositorio proporciona una operación de escritura de tuplas y dos operaciones de lectura (destructiva y no destructiva) de acuerdo a la semántica de Linda. Este enfoque se traduce en que cada componente sólo interacciona con el repositorio y no tiene que comunicarse directamente con otras componentes. Por tanto, el uso del repositorio de mensajes permite desacoplar las diferentes componentes de la capa de ejecución de la infraestructura ya que la comunicación entre las mismas se realiza a través del repositorio. Como resultado, se otorga flexibilidad a la infraestructura al permitir que se añadan, modifiquen o eliminen componentes de forma dinámica sin que esto afecte al resto de componentes.

Por su parte, los mediadores encapsulan la heterogeneidad de un *middleware* determinado, teniendo completo conocimiento de sus capacidades y características. Este diseño elimina la necesidad de que el *broker* tenga que estar muy acoplado con la tecnología concreta del entorno Grid, permitiendo incorporar diferentes entornos de computación heterogéneos de forma sencilla y transparente para el programador de *workflows*. Internamente, el mediador es responsable de: i) tener información completa del entorno Grid que encapsula; ii) interaccionar con el repositorio de tuplas para obtener tareas a ejecutar; iii) enviar tareas al *middleware* para su ejecución

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

y controlar la transferencia de los datos de entrada y de salida; y iv) insertar tuplas en el repositorio de mensajes con el resultado de la ejecución de las mismas para que sea tratada por la componente adecuada. Se ha implementado un mediador para cada uno de los *middlewares* (Condor y gLite) utilizados por los entornos de computación disponibles, los cuales son, además, dos de los *middlewares* más utilizados en entornos Grid.

En cuanto a las componentes de gestión, éstas ofrecen diferentes funcionalidades encaminadas a gestionar el ciclo de vida de los *workflows* ejecutados. Se han desarrollado: una componente de gestión de fallos y una componente de movimiento de datos. El procedimiento de integración de estas componentes es similar al utilizado en los mediadores. Cada componente de gestión interacciona con el repositorio de mensajes para retirar mensajes con la etiqueta asociada a esa componente y procesarlos. Por lo tanto, la utilización de estas componentes puede ser debida a la necesidad de un procesado concreto (p. ej. *meta-scheduling*) o como respuesta al resultado de otra componente (p. ej. gestión de fallos), permitiendo la composición dinámica de complejas cadenas de acción. Con la integración de estas componentes, se consigue gestionar de forma completa el ciclo de vida de un *worfklow*. Pueden consultarse más detalles sobre las componentes de movimiento de datos y gestión de fallos en [2, 28].

A modo de ejemplo, para que el lector comprenda la interacción existente entre las componentes, mostraremos el proceso seguido para ejecutar una tarea. En primer lugar, la descripción de la tarea se almacena en el repositorio de mensajes. Los mediadores capaces de ejecutar dicha tarea compiten por su ejecución. Como resultado la tarea es asignada a un entorno concreto (este proceso se detalla en el siguiente párrafo). Antes de ejecutar la tarea, el mediador solicita el movimiento de los datos de entrada necesarios. Una vez transferidos, el mediador envía la tarea al Grid para que se ejecute. Cuando la tarea finaliza o falla, el mediador solicita el movimiento de los datos de salida a su ubicación final, recupera el log de ejecución e introduce dicha información en el repositorio de mensajes. Si la tarea ha finalizado correctamente se coloca la misma en el repositorio de mensajes hasta que es recuperada por el usuario. Si por contra, la tarea ha fallado, la componente de gestión de fallos obtiene la causa del fallo y toma alguna decisión al respecto, como por ejemplo, reejecutar la tarea en otro entorno o notificar al usuario del error que se ha producido. En caso de que la tarea sea reejecutada, se repite el proceso, mientras que si se decide avisar al usuario del fallo, se actúa como si hubiera acabado correctamente e indicando el error.

Por tanto, el proceso de *scheduling* de nuestra infraestructura se corresponde con el proceso anterior, en el que los mediadores compiten por ejecutar tareas. Este proceso se construye en base a la semántica de las operaciones de lectura definidas en el modelo Linda. Así, se establece que, cuando dos o más componentes realizan una operación de lectura sobre una misma tupla, se selecciona una de las componentes de manera no determinista, siendo ésta la que obtiene la tupla. De esta forma, el proceso de *scheduling* de la infraestructura consiste en que todos los mediadores capaces de ejecutar una tarea realizan una operación de lectura sobre la tupla que contiene la información de la tarea, siendo el propio repositorio el que selecciona uno de los mediadores de manera no determinista.

2.3 Capa de entornos de computación

En lo que corresponde a los entornos de computación, se han integrado: el *cluster* Hermes del Instituto de Investigación en Ingeniería de Aragón (I3A)[2], el cual es gestionado utilizando el *middleware* Condor; y dos Grids pertenecientes a la Iniciativa Grid Europea (EGI)[3]:

[2] http://i3a.unizar.es/
[3] http://www.egi.eu/

AraGrid[4] y PireGrid[5], gestionados por el *middleware* gLite y administrados por el Instituto de Biocomputación y Física de Sistemas Complejos (BIFI)[6].

La principal diferencia del modelo de integración propuesto, con respecto a otras soluciones [15, 18], reside en que la misma se realiza desde una perspectiva de usuario de los entornos de computación y no desde un punto de vista de administrador. Este enfoque permite evitar las barreras administrativas que surgen a la hora de integrar entornos gestionados por diferentes organizaciones ya que, en nuestro caso, esta integración es completamente transparente para el entorno de ejecución.

A su vez, el diseño desacoplado de la solución permite aislar al usuario de los detalles de ejecución referentes a cada entorno, siendo la infraestructura la encargada de lidiar con dicha complejidad. La capa de mediadores desarrollada, permite afrontar las diferentes características de cada una de estos entornos y gestionar su heterogeneidad. De la misma manera, este diseño facilita la integración de entornos de computación (puede realizarse de forma dinámica) y la reutilización de los mediadores desarrollados para gestionar otros entornos gestionados por los mismos *middlewares* (el mediador de un *middleware* puede ser utilizado en cualquier entorno gestionado por dicho *middleware*).

2.4 Breve resumen de las aportaciones de la infraestructura

En resumen, la orientación a servicios de la infraestructura ofrece al usuario una visión de servicio de ejecución de *workflows* y tareas computacionalmente costosas. De esta forma, se posibilita la ejecución de *workflows* programados en diferentes lenguajes. Además, el uso de JSDL, como lenguaje estándar de descripción de tareas, permite programar los *workflows* de manera independiente del entorno de ejecución.

Internamente, la naturaleza abierta y flexible de la solución propuesta se basa en el uso de un *broker* de recursos formado por un repositorio de mensajes basado en Linda y un conjunto de mediadores. El repositorio de mensajes facilita la integración y sustitución de mediadores y componentes de gestión de forma dinámica. Los mediadores encapsulan la heterogeneidad de los diferentes entornos de computación utilizados, desacoplan el *broker* de recursos de los detalles de los diferentes *middleware* de Grid, abstraen al usuario de la complejidad de los mismos y pueden ser reutilizados en entornos gestionados por un mismo *middleware*. El uso conjunto del repositorio de mensajes y los mediadores permite definir una política de competencia que posibilita la ejecución de las tareas de los *workflows* en diferentes entornos de ejecución utilizando las operaciones ofrecidas en el modelo Linda. Finalmente, la integración de varias componentes de gestión permite mejorar la gestión del ciclo de vida de las tareas ejecutadas ofreciendo servicios de movimiento de datos y gestión de fallos.

[4] http://www.araGrid.es/
[5] http://www.pireGrid.eu/
[6] http://bifi.es/es/

Capítulo 3 | *Meta-scheduling* basado en simulación

La infraestructura descrita en el capítulo anterior presenta una clara limitación en cuanto al proceso de asignación de tareas a entornos de computación. En dicho proceso no se utiliza información sobre el estado de los entornos de computación, si no que se utiliza un modelo no determinista Este modelo no utiliza ningún tipo de información para decidir el entorno concreto a utilizar. Esto puede llevar a una mala utilización de los recursos y a la disminución de las prestaciones del *workflow*. Para solucionar este problema, es necesario incluir en la infraestructura algún tipo de técnica de *meta-scheduling* que tenga en cuenta información dinámica sobre el estado de los entornos de ejecución y utilice dicha información para guiar el proceso de asignación de tareas a entornos de computación.

En este capítulo, se detalla la estrategia de *meta-scheduling* propuesta, se muestran sus beneficios y se detalla el proceso de asignación de tareas realizado al incorporar un *meta-scheduler*. Dicha estrategia utiliza técnicas de simulación para obtener datos que permitan tomar decisiones de *meta-scheduling*. Por tanto, debido a su importancia en el proceso, se detallan los aspectos fundamentales referentes a la simulación como es el diseño de los simuladores, la metodlogía de creación de los *workloads* y la validación de los resultados obtenidos. Finalmente, se proporcionan algunos detalles referentes a la implementación del *meta-scheduler*.

3.1 Proceso de asignación de tareas a entornos de computación

Como se indicó en el capítulo anterior, la infraestructura soporta dos mecanismos de selección del entorno a utilizar para ejecutar una tarea. El primero consiste en que el usuario indica el entorno de ejecución en el modelo del *workflow*. La utilización de este *scheduling* estático y guiado por el usuario plantea dos problemas: en primer lugar, el usuario no suele tener información para determinar cuál es el entorno más adecuado para ejecutar cada tarea, y en segundo lugar, el hecho de que la elección sea estática (se realiza al comienzo de la ejecución del *workflow*) implica que cuando se ejecuta cada tarea la situación será diferente de la inicial, y los criterios de selección pueden haber variado. Por tanto, ésta no es una alternativa adecuada para sacar el máximo partido a los entornos de computación integrados y asegurar los requisitos de calidad de servicio solicitados por los usuarios.

La segunda posibilidad es que sea el propio sistema el que asuma la responsabilidad de tomar dicha decisión. Para ello se propuso una estrategia de selección en la que los mediadores compiten por ejecutar trabajos y se selecciona uno de ellos de forma no determinista. Esta estrategia presenta varios inconvenientes:

- Los mediadores no consideran cuestiones de calidad de servicio (QoS) acerca de la ejecución de los trabajos en el entorno que representan. Esto puede llevar a que una tarea se ejecute en un entorno inadecuado, degradando el rendimiento de todo el *workflow*.

- Las decisiones de *scheduling* son adoptadas localmente por cada mediador. Por tanto, uno de ellos podría monopolizar la ejecución de todos los trabajos llevando a una situación de sobrecarga de uno de los entornos de computación mientras el resto permanecen vacíos. Esto provocaría que los trabajos se vieran afectados por elevados tiempos de espera en una situación en la que existen recursos libres.
- Los mediadores ignoran el estado actual de los recursos, la posible evolución de los mismos y el comportamiento de los trabajos ejecutados en el entorno. De nuevo, esto puede llevar a que se ejecuten trabajos en condiciones en las que el entorno está sobrecargado o a la elección de un entorno inadecuado para la ejecución de un determinado trabajo, degradando las prestaciones y realizando un mal aprovechamiento de los recursos disponibles.

Para solventar los inconvenientes anteriores, se propone incorporar en la infraestructura una estrategia de *meta-scheduling* basada en la utilización de técnicas de simulación [34]. La Figura 3.1 muestra las diferentes fases del proceso, el cual detallaremos a continuación:

1. Inicialmente, las tareas pendientes (abstractas) de ejecución, almacenadas en el repositorio de mensajes, son recuperadas por el *meta-scheduler*.
2. El *meta-scheduler* determina los entornos capaces de ejecutar dichas tareas y solicita a los mediadores correspondientes que simulen su ejecución.
3. Al recibir la petición de simulación, los mediadores obtienen el estado del entorno de ejecución y construyen un *workload* que refleje el estado de la misma y su evolución futura (la metodología de construcción de los *workloads* se detalla en la sección 3.3).
4. Los mediadores realizan la simulación de las tareas indicadas con el *workload* construido y devuelven el resultado al *meta-scheduler*.
5. Cuando el *meta-scheduler* ha recibido el resultado de todas las simulaciones, elige el entorno más adecuado en base a un algoritmo de optimización y almacena dicha información en la descripción de la tarea. De esta forma, la tarea se convierte en una tarea concreta.
6. Finalmente, el *meta-scheduler* envía la tarea al repositorio de mensajes para que sea recuperada y ejecutada por el mediador correspondiente en el entorno seleccionado

Esta solución, basada en la utilización de un *meta-scheduler* que asigna trabajos a diferentes entornos en base a la información proporcionada por simuladores, permite mejorar el rendimiento de los *workflows* al tener en cuenta el estado actual y futuro de los entornos de computación y aplicar algoritmos de optimización de los criterios deseados. Concretamente, en este caso se utiliza un algoritmo para optimizar del tiempo de ejecución. En cualquier caso, el análisis de diferentes algoritmos de optimización queda fuera del alcance de este trabajo.

3.2 Diseño de los simuladores

Para soportar el proceso anterior, es necesario extender los mediadores integrando un simulador dentro de los mismos. El simulador debe ser capaz de: i) modelar diferentes entornos de computación incluyendo la organización de los recursos, sus características (procesadores, memoria RAM, características de la red) y la política de *scheduling*; ii) permitir la construcción dinámica de *workloads* que contengan las tareas a simular y tareas que representen la carga de fondo del sistema; y, finalmente iii) simular la ejecución de tareas midiendo parámetros como el

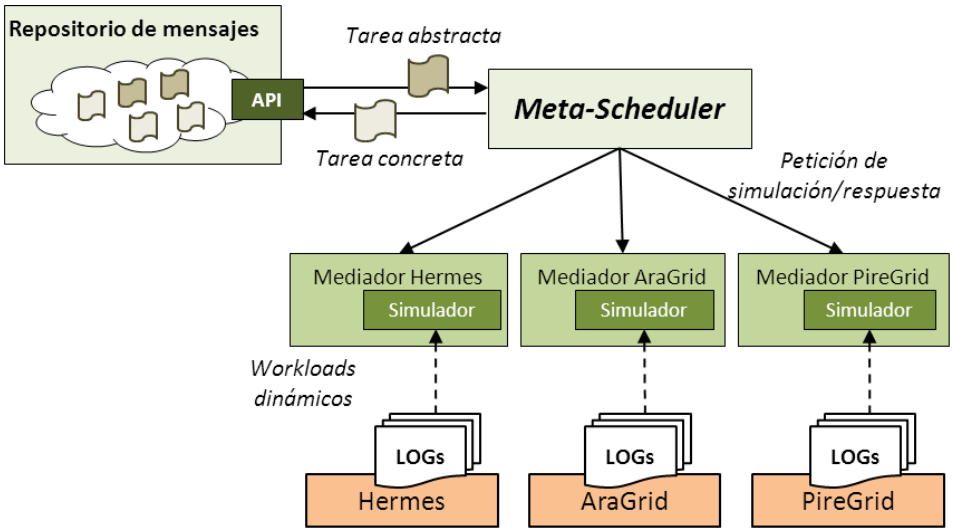

Figura 3.1: Componente arquitectural para realizar *meta-scheduling* basado en simulación.

tiempo de ejecución o el tiempo de cola. Asimismo, para facilitar el desarrollo de diferentes simuladores, el simulador debe proporcionar una interfaz común que sea independiente del entorno a simular. Por último, siguiendo la línea anterior, el diseño debe ser fácilmente adaptable y reutilizable para construir nuevos simuladores.

Para posibilitar la ejecución de trabajos en nuestra infraestructura se han construido simuladores para entornos gestionados por Condor y gLite. La elección de estos *middlewares* se debe a que son dos de los *middlewares* más empleados para gestionar entornos de computación de tipo Grid, y a que son los utilizados por los entornos integrados dentro de la infraestructura propuesta.

Como base para los simuladores desarrollados se ha utilizado Alea [35]. Alea es un simulador basado en eventos y construido sobre GridSim [36]. Alea extiende GridSim proporcionando un *scheduler* centralizado, mejorando algunas funcionalidades y aumentando la escalabilidad y la velocidad de la simulación. Además, Alea proporciona un entorno de experimentación fácil de configurar y utilizar, el cual ayuda en la adaptación del simulador a nuevos entornos. La implementación de Alea ha sido extendida como parte de este trabajo en aspectos como la posibilidad de definir requisitos de memoria, la definición de un modelo de Grid más completo o la definición de nuevas políticas de *scheduling*. A continuación, abordaremos el diseño de un simulador de Condor y su reutilización para construir un simulador de gLite.

3.2.1 Diseño del simulador de Condor usado en Hermes

Para ilustrar el diseño realizado, se va a detallar el diseño del simulador de Condor que se usa en el *cluster* Hermes. Hermes es un *cluster* de computación alojado por el Instituto de Investigación en Ingeniería de Aragón (I3A). Está formado por una gran variedad de recursos de computación heterogéneos y dispone de un total de 1308 procesadores y 2.56 TB de memoria RAM.

En lo que respecta al diseñado realizado, el cual puede observarse en la Figura 3.2, como entrada se proporciona el *workload* que indica las tareas a simular y un modelo del Grid sobre el que se van a ejecutar las mismas; mientras que, como salida se indican los resultados de la simulación de dichas tareas en el entorno indicado. Internamente, el simulador está formado por cuatro componentes fundamentales: la componente de carga de trabajos, la componente de carga de máquinas, el *scheduler* y el recolector de resultados.

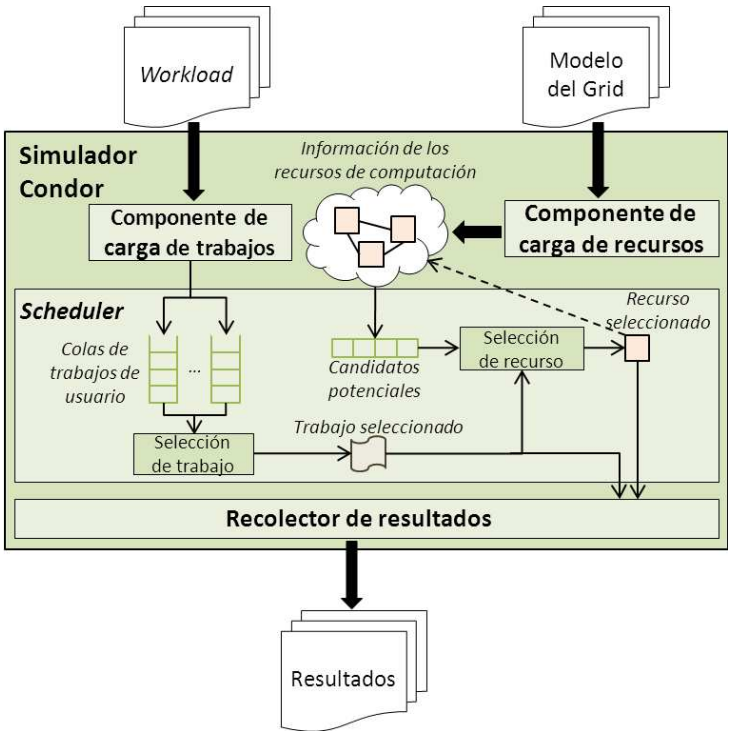

Figura 3.2: Diseño arquitectural del simulador de Condor.

El *workload* se representa utilizando el formato GWF (*Grid Workload Format*) propuesto por el archivo de *workloads* de Grid [37]. Este *workload* es el que contiene las tareas a simular y agrupa tanto las tareas objetivo de la simulación como otras tareas que modelan la carga de fondo del entorno. La metodología de construcción de los *workloads* se detallará más adelante.

El modelo del Grid corresponde a un fichero de texto que contiene la información de los nodos de computación del Grid. La representación usada en este modelo, se ha extendido para posibilitar la definición de modelos más detallados al usados por defecto en Alea. La representación de cada nodo incluye el número de máquinas que lo forman, el número de CPUs por máquina, la cantidad total de memoria por máquina, la arquitectura de las máquinas del nodo, su sistema operativo y las características de la red. Junto con este modelo, se incluye un modelo de fallos que permite reflejar cambios dinámicos en el entorno durante la simulación (caídas de nodos y fallos de máquinas).

La componente de carga de trabajos lee la descripción de los trabajos y se la envía al *scheduler*. Este módulo ha sido extendido para soportar la definición de requisitos de memoria y el usuario y grupo, u organización virtual, correspondiente a cada trabajo.

La componente de carga de máquinas es la responsable de obtener la descripción de los nodos del entorno de computación. Este módulo se ha extendido para ser capaz de tratar toda la información proporcionada dentro del modelo del Grid (anteriormente sólo se permitía indicar el número de máquinas y procesadores).

El *Scheduler* es la componente más compleja y la única que es necesario modificar para construir nuevos simuladores. La componente ha sido extendida para soportar la política de *scheduling* basada en prioridades de usuario que usa Condor. Esta política funciona del siguiente modo: cuando un trabajo llega al *scheduler*, el mismo es encolado en la cola de trabajos del usuario. Esta cola se ordena por la prioridad propia de los trabajos de ese usuario y el instante de llegada. Cuando el *scheduler* solicita la ejecución de un nuevo trabajo, los trabajos se ordenan por la prioridad de su usuario, seleccionándose el trabajo con mayor prioridad de todos los

disponibles. Entonces, se seleccionan las máquinas que tienen suficientes recursos (procesadores y memoria) para ejecutar ese trabajo y las máquinas que podrían tener los recursos solicitados si expulsasen alguno de los trabajos que están ejecutando en ese momento. Estas máquinas son seleccionadas como candidatas potenciales para ejecutar el nuevo trabajo. Cuando se ha completado la lista de candidatas potenciales, la misma se ordena de acuerdo a múltiples criterios (preferencias del trabajo, preferencias de la máquina, etc.) para encontrar el recurso más adecuado. En el caso de que no haya ningún recurso disponible el trabajo vuelve a ser encolado en la cola del usuario y el *scheduler* intenta ejecutar otro trabajo. Finalmente, cuando se ha podido seleccionar un trabajo y un recurso adecuado para su ejecución, el trabajo es enviado al recurso actualizándose el estado del mismo. Si para posibilitar la ejecución del trabajo en dicho recurso es necesario expulsar un trabajo, el *scheduler* se encarga de reencolar dicho trabajo para que pueda ser ejecutado en otro momento.

Finalmente, el recolector de resultados es la componente encargada de almacenar los resultados de la simulación y proporcionarlos como salida. Cuando un trabajo es enviado a un recurso, es expulsado o una máquina falla, el recolector de resultados almacena dicha información. Asimismo, cuando un trabajo finaliza su ejecución correctamente, el recolector almacena la información relativa al mismo en el fichero de salida de la simulación. Para cada trabajo, se indica el tiempo de llegada, el tiempo que ha permanecido el mismo en la cola, el tiempo de ejecución, el recurso en el que se ha ejecutado y el número de expulsiones que ha sufrido.

3.2.2 Diseño del simulador de gLite usado en AraGrid

La Figura 3.3 muestra la arquitectura del simulador de gLite utilizado en AraGrid. AraGrid es un Grid regional gestionado por el Instituto de Biocomputación y Física de Sistemas Complejos (BIFI). Está formado por cuatro *sites* que se encuentran geográficamente distribuidos en diferentes facultades (dos *sites* en Zaragoza, uno en Huesca y otro en Teruel). En total, el Grid dispone de 1728 procesadores y 4 TB de memoria RAM repartidos de forma homogénea entre los diferentes *sites*. Gracias a la extensibilidad y facilidad de uso del simulador desarrollado, se ha podido reaprovechar el simulador de Condor para la elaboración del simulador de gLite. De esta forma, la construcción de este nuevo simulador ha sido rápida y sencilla. En consecuencia, el diseño del simulador de gLite es análogo al del simulador de Condor presentado anteriormente (Figura 3.2).

Si comparamos los diseños arquitecturales de ambos simuladores (Figuras 3.2 y 3.3), se puede ver como sólo ha sido necesario modificar el *scheduler* ya que gLite utiliza un política de *scheduling* jerárquica, diferente de la de Condor. En dicha política, los trabajos enviados son gestionados por un *scheduler* global que envía los mismos a alguno de los *schedulers* locales dependiendo de los requisitos del trabajo, su *ranking*, la ocupación de los *sites* que forman el Grid y los recursos a los que puede acceder la organización virtual del usuario que envía el trabajo. Por su parte, los *schedulers* locales de los diferentes *sites* utilizan una política FCFS [38] (*First Come First Serve*) para ejecutar los trabajos dentro de los recursos del *site*.

3.3 Metodología para la creación de *workloads*

La creación de los *workloads* utilizados para reflejar el estado de cada entorno y su evolución es un aspecto clave en el proceso de simulación. Su elaboración se realiza mediante el análisis de información histórica de un largo período de tiempo (en nuestro caso se han utilizado *logs* de 1 año) y considerando sólo los momentos representativos; por ejemplo, las horas de carga pico de los días laborables en los meses de mayor utilización del entorno de ejecución [39]. Se asume que cuantos más datos se tengan en cuenta, más realista y representativo será el *workload* generado.

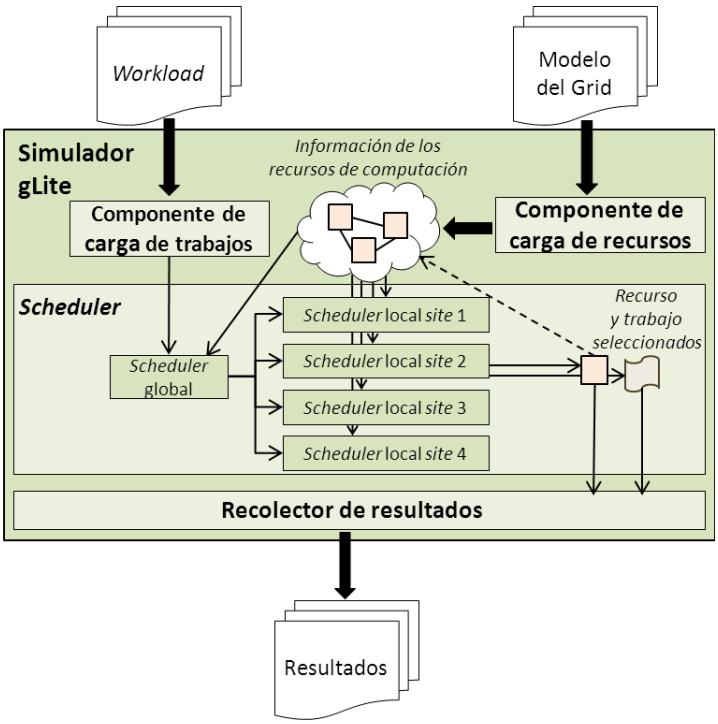

Figura 3.3: Diseño arquitectural del simulador de gLite con detalle de uno de los *schedulers* locales.

Una vez extraídos los datos relevantes se utilizan técnicas de regresión y ajuste de curvas para construir modelos probabilistas que modelen diferentes aspectos como el tiempo de ejecución, el tiempo de cola o los recursos utilizados.

La importancia de utilizar un *workload* apropiado ha sido identificada en varios trabajos [40, 41]. Los trabajos anteriores proponen la generación de un único *workload* que considere únicamente momentos de carga extrema o carga media del entorno de computación. Entonces, el *workload* obtenido se utiliza para customizar la configuración del entorno y obtener una mejora de sus prestaciones en la situación elegida (extrema o media) [39].

Sin embargo, con fines de simulación, estas aproximaciones no son válidas ya que debe considerarse el estado real de los recursos. Si se utiliza un *workload* medio o extremo como entrada del simulador, los resultados de la simulación no se ajustarán al comportamiento obtenido en la ejecución real de las tareas al no realizarse la misma en condiciones de carga reales. En consecuencia, como la información proporcionada por los simuladores se utiliza para tomar decisiones de *meta-scheduling*, el uso de un *workload* no representativo puede llevar a tomar malas decisiones de *meta-scheduling* que degraden las prestaciones del *workflow* en lugar de mejorar las mismas.

Nuestra propuesta consiste en construir varios *workloads* que representen diferentes situaciones representativas y que dependan de la carga de los recursos (carga baja, carga media y carga alta), la fecha (días laborables y festivos) y el momento del día (mañana, tarde y noche). De esta forma, cuando se va a realizar una simulación se obtiene el estado actual del entorno de ejecución y se selecciona el *workload* más adecuado. Además, a la información del *workload* se le añade la información de los trabajos que se encuentran actualmente en ejecución y encolados obteniendo un *workload* que representa el estado actual del entorno y su evolución.

De forma detallada, el proceso que se lleva a cabo cuando el *meta-scheduler* solicita que se simule la ejecución de una tarea es el siguiente:

1. Cuando un mediador recibe una petición de simulación, construye un *workload* que describe las tareas que se deben simular.
2. A continuación, el mediador consigue información sobre el estado del entorno de computación y los trabajos que se encuentran en ejecución actualmente o están encolados.
3. Con la información del estado de los recursos, se adapta el modelo predefinido del Grid a la situación actual, incluyendo los fallos existentes en los recursos.
4. Con la información de los trabajos que se están ejecutando o están en la cola y teniendo en cuenta la fecha actual, se selecciona el *workload* más adecuado a la situación actual.
5. Al *workload* anterior se le añade la información de las tareas que se están ejecutando actualmente y las que se encuentran encoladas. Con esto se obtiene un *workload* que representa el estado actual y la evolución futura del entorno.
6. Cuando se han creado los dos *workloads* (el *workload* con las tareas a ejecutar y el *workload* con el estado del entorno de ejecución), se combinan los mismos para crear un único *workload* que sirve como entrada del simulador.
7. Se realiza la simulación utilizando el *workload* anterior.
8. Cuando la simulación acaba, el mediador obtiene los resultados y filtra los mismos para proporcionar únicamente al *meta-scheduler* los resultados referentes a las tareas objetivo de la simulación.

La implementación de este proceso se encapsula dentro del mediador. Éste se encarga de obtener información acerca del estado del entorno de ejecución y realizar diferentes procesados de la información obtenida para adaptar el modelo del entorno de computación a la situación actual y construir el *workload* de entrada. Para facilitar la realización de las simulaciones, los simuladores ofrecen una interfaz común independiente de los detalles concretos del simulador y un formato común para la descripción de los datos de entrada y salida.

3.4 Validación de los simuladores

El objetivo de los simuladores desarrollados es utilizar los mismos como herramienta de decisión en el proceso de *meta-scheduling*. Por tanto, la validación de los resultados proporcionados por los mismos es fundamental para verificar la viabilidad y utilidad de la solución [42]. Para ello, se ha comparado la utilización de los recursos, la duración de los trabajos y los tiempos de cola en el entorno real y en el entorno simulado.

La Figura 3.4 muestra una comparación de la utilización media del *cluster* Hermes (gestionado por Condor), extraída de los *logs* de ejecución del último año, y la utilización obtenida de la simulación de esos mismos trabajos. La comparación se presenta en un ciclo diario, con el eje horizontal indicando la hora del día y el vertical mostrando el porcentaje de utilización de recursos. Como se puede observar, los resultados del entorno simulado son muy similares a los obtenidos en el escenario real. Ambas gráficas muestran la misma tendencia, siendo los resultados de la simulación ligeramente inferiores. En términos del error incurrido en la simulación, en media el error es del 15.09 % con una desviación típica del 8.03 %.

Para validar el indicador de rendimiento de los trabajos se exploran las métricas de tiempo de ejecución y tiempo de cola. La Figura 3.5 muestra la comparación de ambos parámetros mediante las correspondientes funciones de probabilidad acumulada. En ambos casos, se muestra el eje horizontal en escala logarítmica para dotar a las gráficas de una mayor claridad. La Figura 3.5-a

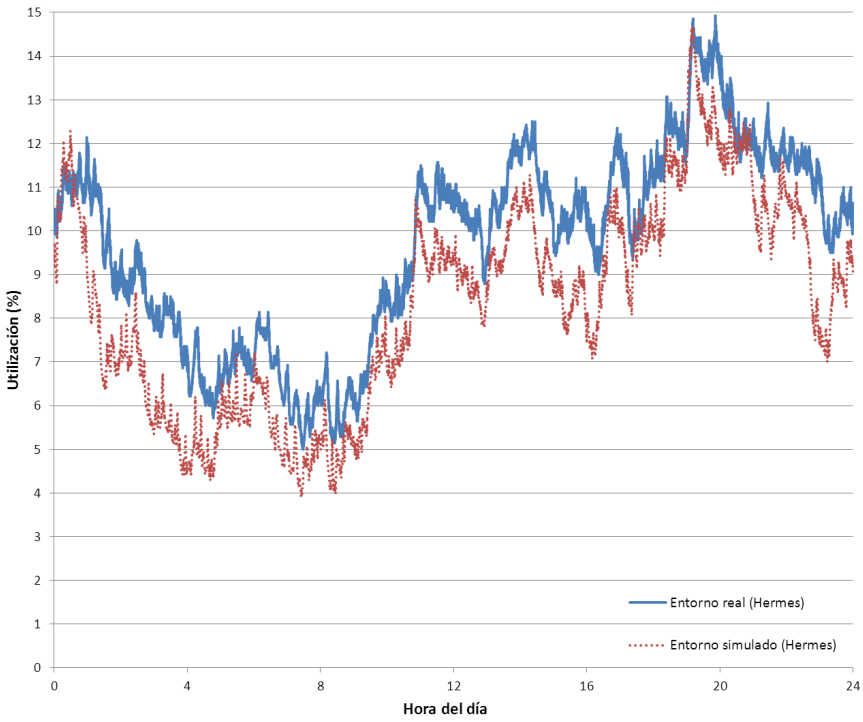

Figura 3.4: Comparación de la utilización del *cluster* Hermes entre el entorno real y el escenario simulado.

muestra la comparación entre el tiempo de ejecución en ambos entornos el cual es muy similar en ambos casos. En contraposición, si nos fijamos en la Figura 3.5-b, la distribución de los tiempos de cola experimentados en el entorno real y en el entorno simulado es diferente. Esto se debe a que el simulador es capaz de ejecutar trabajos sin ningún tipo de retardo, mientras que Condor se ve afectado por numerosos retardos como las notificaciones entre los diferentes componentes distribuidos, la duración del ciclo de *scheduling* o la actualización y propagación del estado. Para corregir este error y reducir su influencia en los resultados, se pueden emplear dos técnicas: la primera consiste en añadir un retardo sintético al tiempo de ejecución obtenido como resultado y la segunda consiste en añadir dicho retardo a la ejecución de los trabajos dentro del simulador. En cualquier caso, se está estudiando la manera más efectiva de incorporar estos aspectos al simulador (véase la Sección 5.1).

En el caso del simulador de gLite, se ha realizado el mismo tipo de análisis que el realizado para el simulador de Condor. Para ello se ha utilizado como entorno de prueba AraGrid. En este caso, la comparación en términos de utilización de los recursos depara resultados ligeramente mejores ya que la política de *scheduling* de gLite entraña menos complejidad y es más fácil de replicar. Concretamente, el error en el que se incurre en este caso es tan sólo del 1.19 % con una desviación típica del 0.85 %. Respecto al indicador de rendimiento de los trabajos, como era de esperar, se observa el mismo comportamiento que en Hermes. La distribución de los tiempos de ejecución es prácticamente la misma en ambos escenarios, mientras que la distribución de los tiempos de cola es diferente en el escenario simulado y el entorno real debido a que el simulador sólo es capaz de reflejar el tiempo que pasa un trabajo en cola debido a que no hay recursos disponibles.

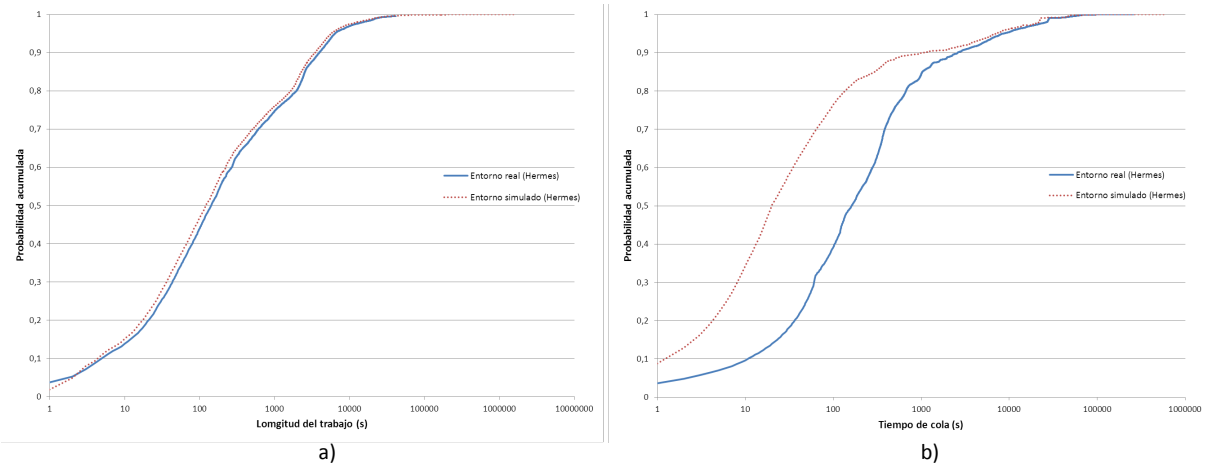

Figura 3.5: Comparación del rendimiento de los trabajos para el *cluster* Hermes en el entorno real y el escenario simulado en términos de: a) Tiempo de ejecución, b) Tiempo de cola.

3.5 *Meta-scheduler*

Como se ha comentado a lo largo de este capítulo, la integración de un *meta-scheduler* en la capa de ejecución de la infraestructura, permite mejorar el proceso de *scheduling* no determinista aplicado anteriormente. Ahora, los mediadores se limitan a obtener tareas que tienen como destino el entorno que representan (ya no compiten por tareas abstractas), dejando que el *meta-scheduler* obtenga las tareas abstractas y las transforme en tareas concretas asignándoles un entorno de ejecución concreto.

La transformación de las tareas de abstractas a concretas llevada a cabo por el *meta-scheduler* suele realizarse en base a algún tipo de algoritmo de optimización que considere aspectos como el rendimiento, el coste o la fiabilidad de la ejecución [43]. Asimismo, este proceso suele apoyarse en algún tipo de técnica que proporcione indicadores para los parámetros anteriores [25, 26, 27]. En este caso concreto, se ha decidido utilizar un algoritmo de optimización del tiempo de ejecución utilizando como indicadores los resultados obtenidos al simular la ejecución de una tarea en diferentes entornos de computación.

Por tanto, cuando el *meta-scheduler* obtiene una tarea abstracta, solicita a los mediadores que simulen su ejecución en el entorno que representan y espera los resultados. Cuando se reciben los resultados, se selecciona como entorno de ejecución aquel en el que se ha obtenido un menor tiempo de ejecución en la simulación. Por su puesto, esta interacción e intercambio de información se realiza a través del repositorio de mensajes.

No obstante, el *meta-scheduler* se ha diseñado para que pueda modificarse el algoritmo de selección de la infraestructura de forma sencilla y que el cambio sea transparente para el resto de la infraestructura. De esta forma, se facilita la utilización de diferentes algoritmos, siendo el estudio del algoritmo de *meta-scheduling* más adecuado uno de los trabajos futuros propuestos (veáse la sección 5.1).

Capítulo 4 | **Caso de estudio: Inspiral**

Para probar la viabilidad y utilidad del *meta-scheduler*, se va a ejecutar un *workflow* científico real en diferentes condiciones de carga. Para determinar la mejora obtenida, se va a comparar el tiempo de ejecución obtenido al usar la infraestructura respecto a la ejecución aislada del *workflow* en Hermes y AraGrid.

De esta forma, en este capítulo se va a ejecutar el *workflow* científico de análisis *Inspiral* [44] con la solución propuesta. En primer lugar, se presentará una descripción de alto nivel del *workflow* y su correspondiente implementación en Taverna. A continuación, se muestran los detalles referentes al entorno de experimentación y la configuración del experimento. Finalmente, se muestran y analizan los resultados obtenidos al ejecutar el *workflow* con la infraestructura propuesta comparando los mismos con los resultados que se habrían obtenido al ejecutar el experimento sin dicha infraestructura.

4.1 *Workflow* de análisis Inspiral

El *workflow* de análisis Inspiral es un *workflow* científico que analiza e intenta detectar ondas gravitacionales producidas por varios eventos en el universo utilizando datos obtenidos de la coalescencia de sistemas binarios compactos como estrellas binarias de neutrones y agujeros negros [44]. La Figura 4.1 muestra la estructura del *workflow* (Figura 4.1-a) y su implementación en Taverna (Figura 4.1-b). Como puede observarse, la relación entre cada una de las tareas que forman una fase en el diseño de alto nivel y la implementación correspondiente del *workflow* en Taverna es inmediata, siendo muy sencilla la composición del experimento. Internamente, cada una de las cajas que representan las tareas en Taverna encapsula varias operaciones sencillas que generan la información de la tarea, ordenan su ejecución utilizando la infraestructura y obtienen los resultados de la tarea.

El experimento consta de diferentes fases, cuya descripción detallada puede consultarse en [34, 44], que realizan el procesamiento de grandes conjuntos de mediciones generadas por un conjunto de sensores y detectores. Los trabajos *Inspiral* son los más complejos en términos computacionales y los que más recursos demandan. El resto de tareas aplican diferentes operaciones de filtrado, chequeo y transformación de los datos de entrada de cara a reconocer y validar las ondas gravitacionales proporcionadas por la tarea Inspiral.

4.2 Preparación del experimento

Como entornos de ejecución para la realización del experimento se han utilizado Hermes y AraGrid y se ha procedido a ejecutar el *workflow* Inspiral en dichos entornos de computación con la infraestructura y sin ella. Concretamente, se ha ejecutado el *workflow* durante un día entero (24 horas) y se ha repetido el experimento durante varios días para obtener resultados representativos.

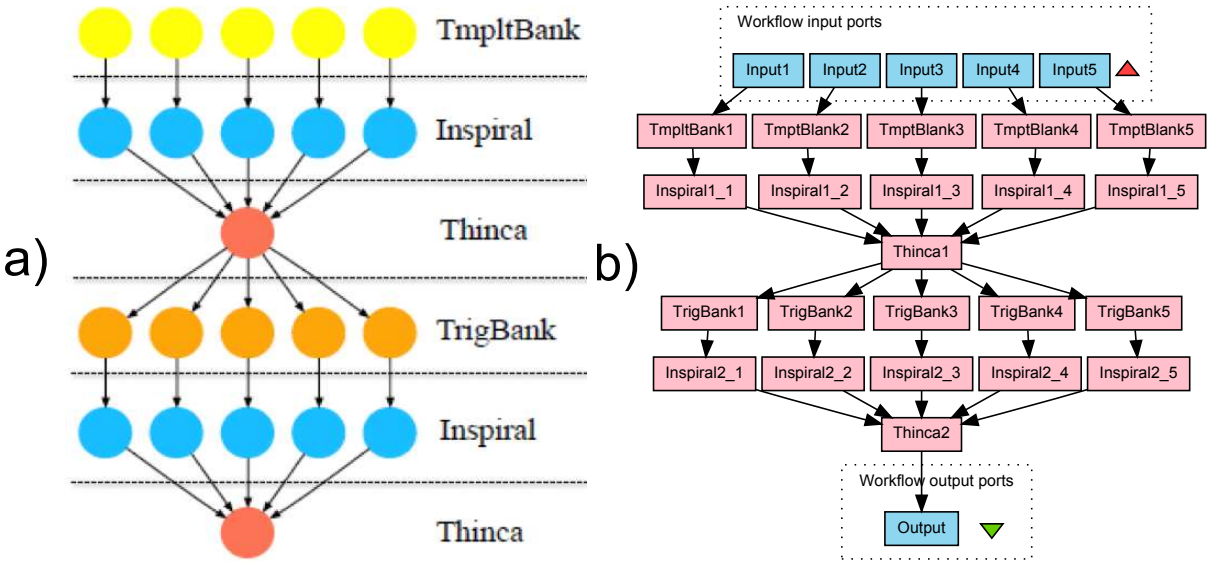

Figura 4.1: Workflow científico de análisis LIGO Inspiral: a) Descripción de alto nivel, b) Implementación en Taverna.

En lo que respecta a la configuración del experimento, no es necesario ningún tipo de configuración especial ya que la misma es generada automáticamente por las diferentes componentes de la infraestructura. Este diseño simplifica la utilización de la infraestructura, haciendo que el proceso de *meta-scheduling* basado en simulación sea completamente transparente para el usuario.

4.3 Análisis de resultados

Como se ha comentado anteriormente, el *workflow* Inspiral puede ser ejecutado tanto en Hermes como en AraGrid. Sin embargo, como se puede observar en la Figura 4.2, ambos entornos muestran una tendencia a tener diferentes niveles de carga a lo largo del día, lo que provoca que, dependiendo de la carga que presenten, sea más adecuado enviar los trabajos a un entorno o a otro. Concretamente, en los diferentes experimentos realizados, se ha observado que resulta más oportuno enviar los trabajos a Hermes por la mañana y durante la noche, mientras que por la tarde es más apropiado enviar los trabajos a AraGrid.

Evidentemente, la carga del sistema no es el único criterio a considerar ya que el rendimiento de un entorno Grid depende de muchos factores y su análisis es complejo. La utilización de un simulador como herramienta de decisión permite lidiar con esta complejidad y mejorar el rendimiento obtenido en la ejecución del *workflow* como se muestra en la Figura 4.3. Dicha figura muestra el tiempo de ejecución total de cada etapa del *workflow Inspiral* ejecutado de forma completa en cada entorno (la barra izquierda corresponde a Hermes mientras que la barra derecha corresponde a AraGrid) y ejecutado utilizando la infraestructura con la política de *meta-scheduling* propuesta (barra central de la figura).

Los resultados muestran que la utilización de la infraestructura de integración desarrollada junto con la política de *meta-scheduling* propuesta permite obtener una mejora del rendimiento en todas las tareas del *workflow*. Concretamente, se obtiene una mejora del 59 % respecto a la ejecución del *workflow* en Hermes y un 111 % respecto a la ejecución del mismo en AraGrid.

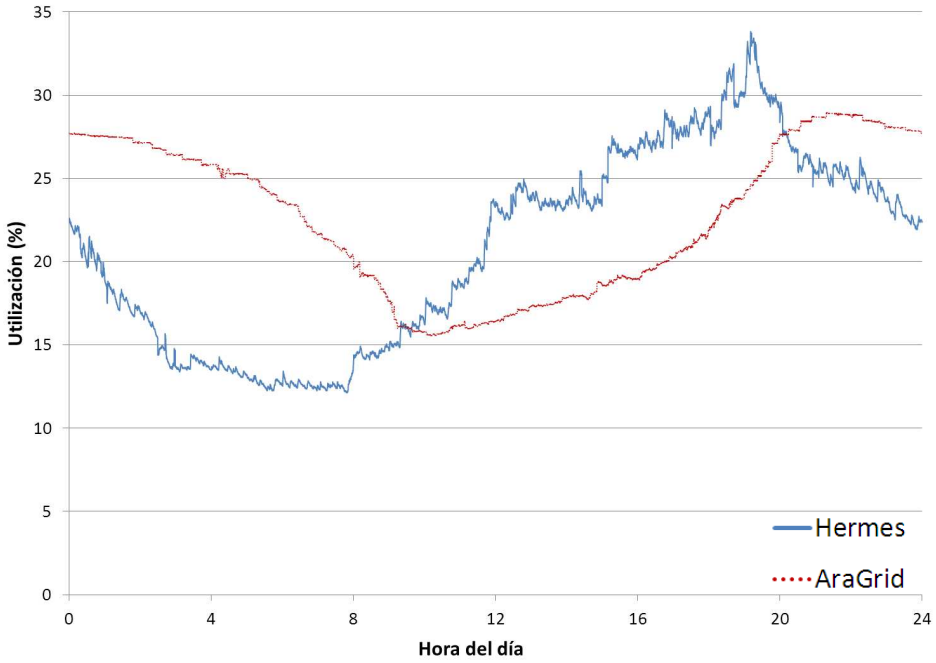

Figura 4.2: Carga media de los entornos de computación durante los experimentos realizados.

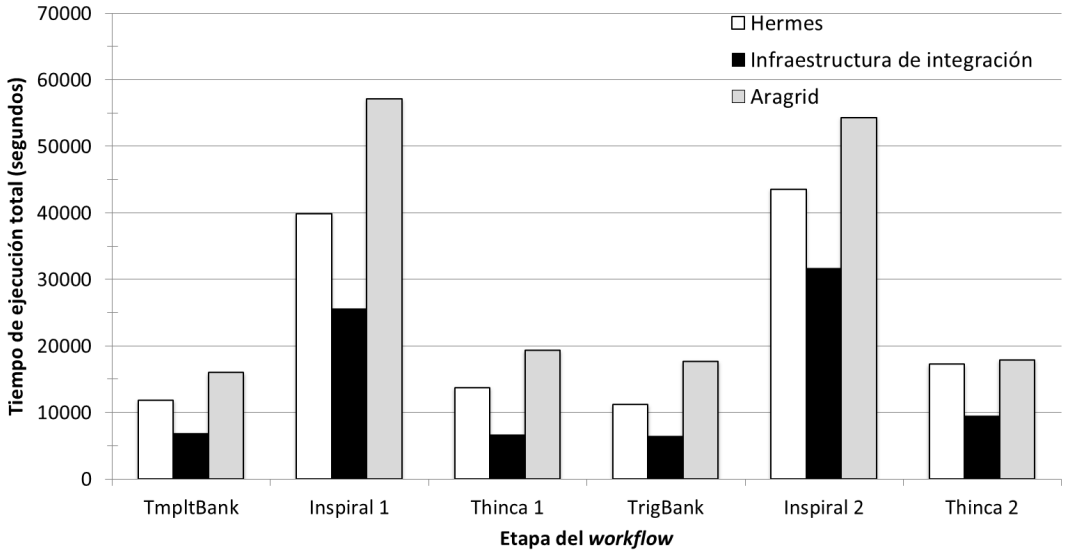

Figura 4.3: Resultados experimentales para cada una de las etapas del *workflow* Inspiral.

Respecto a la sobrecarga que introduce la simulación en términos de tiempo de ejecución, el proceso de simulación de Hermes es más complejo y tarda entre 3 y 4 minutos para una bolsa de 10000 tareas, mientras que para AraGrid lleva en torno a 1 minuto. Además, el tiempo de simulación es insignificante en comparación con el tiempo de ejecución de cada etapa y las transferencias de datos entre entornos usan enlaces de alta velocidad lo que implica que el tiempo de ejecución disminuya a pesar de la sobrecarga introducida.

Por tanto, queda claro que la utilización de la infraestructura permite realizar un mejor aprovechamiento de los recursos, lo que lleva a una mejora de las prestaciones de los *workflows* ejecutados. Esta mejora se observa independientemente del tipo de tarea a ejecutar, ya que la información proporcionada por los simuladores permite inferir el entorno en el que los trabajos se ejecutarán más rápido al pasar menos tiempo encolados y verse menos afectados por las expulsiones.

Capítulo 5 | Conclusiones y trabajo futuro

El trabajo realizado de forma previa a la elaboración de esta Tesis Fin de Máster se ha encaminado a resolver diferentes problemas en el ámbito de ejecución de *workflows* científicos en entornos de computación dinámicos [2, 28, 34, 29]. Por su parte, este Trabajo Fin de Máster ha permitido avanzar en el desarrollo de la solución propuesta. Concretamente, ha permitido mejorar el proceso de selección del entorno más adecuado para ejecutar cada tarea. En cualquier caso, todavía quedan diferentes retos abiertos en este campo, los cuales serán abordados durante el desarrollo de la tesis doctoral del autor.

En este capítulo, se pretende dar una visión general de algunas de las posibles líneas de trabajo futuro abiertas. Estas líneas incluyen la mejora de diferentes aspectos de la infraestructura y la integración en la misma de nuevas características que añadan distintas funcionalidades. Finalmente, se presentan las conclusiones obtenidas en esta Tesis Fin de Máster.

5.1 Líneas futuras

En primer lugar, uno de los aspectos a mejorar en la infraestructura presentada es la gestión de los fallos que se producen al ejecutar trabajos en entornos de computación tipo Grid. Éstos pueden deberse a diferentes causas como errores en la programación del *workflow*, fallos propios de la aplicación que se ejecuta o fallos debidos al propio entorno de ejecución. Para lidiar con estos fallos, se ha incluido una componente de gestión de fallos que trata los mismos de forma general. Sin embargo, sería deseable incluir un modelo de gestión de fallos que permita realizar un tratamiento más personalizado de los mismos a diferentes niveles. Una posible solución sería utilizar un sistema de gestión de fallos jerárquico. De esta forma, los propios mediadores deberían ser capaces de tratar fallos relacionados con su entorno de ejecución. Por su parte, la componente de gestión de fallos genérica se encargaría de los fallos que no pueden ser tratados por los mediadores y ofrecería políticas de más alto nivel para la reejecución de trabajos en otros entornos o la utilización de *workflows* sustitutos [45]. Por tanto, la utilización de un sistema jerárquico de fallos permitiría realizar un mejor tratamiento de los fallos ya que los mediadores pueden tener más información que la componente de gestión del fallo concreto que se ha producido y reducir la sobrecarga del *broker* en el caso de que los fallos puedan ser tratados por el mediador.

Otro punto delicado de la infraestructura de integración planteada es el *broker* de coordinación que gestiona los trabajos a ejecutar en los entornos disponibles. Dependiendo del número de *workflows* científicos que estén siendo ejecutados y su complejidad, este *broker* podría estar sujeto a situaciones de sobrecarga e, incluso, a potenciales fallos. En ambos casos, las consecuencias serían críticas desde el punto de vista de la solución de integración que se propone. Una posible solución sería el uso, durante estas situaciones, del servicio de colas Amazon

SQS[1] (*Amazon Simple Queue Service*), como alternativa más flexible al uso de Linda. Otra posibilidad sería utilizar un sistema de coordinación distribuido que incrementase la escalabilidad y tolerancia a fallos de la infraestructura [46].

También se pretende mejorar la política de *meta-scheduling* basada en simulación propuesta en este trabajo. Dicha mejora pretende aumentar la precisión de la simulación contemplando aspectos como el movimiento de datos entre diferentes entornos.

Asimismo, se plantea mejorar el proceso de generación de *workloads*. Generalmente, la construcción de *workloads* se realiza de forma manual debido a su elevada complejidad y a la dificultad para decidir el modelo estadístico que mejor se ajusta a los datos. Sin embargo, sería deseable utilizar algún tipo de técnica que permita generar *workloads* de forma automática sin que el usuario tenga que intervenir en el proceso [47]. Esto permitiría refinar los *workloads* en base a los nuevos datos obtenidos en cada ejecución y liberar al administrador de dicha tarea.

Para mejorar el proceso de asignación de tareas a infraestructuras, además de mejorar la precisión de los simuladores, se pretende mejorar el propio *meta-scheduler*. Para ello, se plantea explorar y comparar algoritmos de *meta-scheduling* que usen diferentes modelos y técnicas algorítmicas como modelos económicos [48], algoritmos genéticos [49] o algoritmos basados en inteligencia animal [50, 51]. El objetivo de esta comparación es determinar bajo que condiciones los algoritmos ofrecen un mayor rendimiento y permiten optimizar la solución propuesta.

Un aspecto muy importante ligado a los algoritmos de *meta-scheduling* es la definición de parámetros de calidad de servicio (QoS) y acuerdos de nivel de servicio (SLAs) [52]. En este punto, se plantea la utilización de algoritmos de negociación de contratos para cumplir los parámetros de calidad de servicio solicitados por el usuario [53]. También se pretende dar soporte a requisitos de calidad de servicio que no estén únicamente basados en métricas de rendimiento, si no también en otros aspectos menos habituales, pero que también son demandados por la comunidad de *workflows* científicos [54], es el caso de requisitos de coste o requisitos de tolerancia a fallos.

Otro aspecto clave en el ámbito de los *workflows* científicos es el denominado *provenance* [55], la recolección de datos que permitan reproducir los experimentos, compartir dichos experimentos y reutilizar las técnicas, herramientas y metodologías empleadas. En este punto, se plantea el registro de toda la información referente al proceso de ejecución de los *workflows* y la utilización de estándares (XES[2], OPM [56]) para almacenar dicha información y permitir la generación automática de *workflows* que faciliten las cuestiones anteriores.

De cara a dotar al *framework* de un mayor número de recursos computacionales, se plantea integrar entornos de computación basados en *Cloud* que permitan obtener recursos bajo demanda y ayuden a satisfacer la calidad de servicio requerida por el usuario. En esa misma línea, se pretende desarrollar nuevos mediadores que permitan integrar nuevos entornos de computación incluyendo Grids gestionados por diferentes *middlewares* (ARC [57], UNICORE [58], etc.), entornos de computación voluntaria y efímera a través de sus respectivos *middlewares* (p. ej. BOINC [59]) o *clusters* de computación basados en la utilización de tarjetas gráficas.

Finalmente, se pretende aplicar la infraestructura desarrollada a la resolución de problemas computacionalmente costosos ya sea dentro del ámbito académico o empresarial. Para ello, se pretenden establecer acuerdos de colaboración con otros grupos de investigación y con empresas y la realización de estancias en centros de investigación de referencia internacional.

[1] http://aws.amazon.com/es/sqs/
[2] http://www.xes-standard.org/

5.2 Conclusiones

En este trabajo se ha presentado una estrategia de *meta-scheduling* basada en el uso de técnicas de simulación. Además, se ha integrado un *meta-scheduler* que implementa dicha estrategia en una infraestructura que permite ejecutar *workflows* científicos, programados en diferentes lenguajes, en varios entornos de computación heterogéneos. Esta solución ha permitido abordar uno de los principales retos existentes en el ámbito de la ejecución de *workflows* científicos: la posibilidad de ejecutar diferentes partes de un mismo *workflow* en diferentes entornos de computación, obteniendo una mejora en el rendimiento global del *workflow*. Finalmente, por medio de un caso de uso real en el ámbito de la computación científica se ha validado experimentalmente la viabilidad de la solución y sus beneficios.

Para guiar el proceso de *meta-scheduling* se han utilizado técnicas de simulación, algo que no había sido todavía explorado para este propósito. Esta solución, ha permitido tener en cuenta diferentes parámetros importantes para la toma de decisiones como la carga actual y esperada de los entornos de ejecución o la política de *scheduling* utilizada en cada uno de ellos. El desarrollo de un simulador genérico, el cual ha sido diseñado para ser fácilmente reproducible y adaptable, ha permitido desarrollar simuladores para diferentes entornos de computación de forma sencilla. Asimismo, el desarrollo de una metodología para la construcción dinámica de *worklodas* ha permitido simular las tareas en condiciones de carga reales, mejorando la precisión de los resultados obtenidos en las simulaciones. Por tanto, la solución propuesta se ha mostrado como una estrategia válida y efectiva al ser capaz de abordar la complejidad inherente de los diferentes entornos de computación proporcionando resultados muy parecidos a los obtenidos en el entorno real. Este aspecto ha sido comprobado mediante la validación experimental de la técnica.

Este trabajo ha establecido las bases para un modelo de *scheduling* dinámico, transparente para el usuario y que trata de aprovechar todos los recursos disponibles para mejorar el rendimiento de los *workflows*. En cualquier caso, no se trata de un trabajo cerrado o finalizado, sino que se han abierto nuevas posibilidades y líneas de trabajo futuro. La utilización de diferentes algoritmos de *meta-scheduling* que no sólo optimicen el tiempo de ejecución, si no que permitan satisfacer varios tipos de requisitos de calidad de servicio y el establecimiento de acuerdos de nivel de servicio que garanticen al usuario un nivel mínimo de servicio son algunos de los aspectos que podrían incluirse en la solución presentada.

Finalmente, el trabajo de investigación ha dado lugar a tres publicaciones científicas, con revisión por pares, en dos congresos internacionales: el *III International Conference on Cloud Computing, GRIDs, and Virtualization* (CLOUD COMPUTING 2012) y el *II International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH 2012); y un congreso nacional: las VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2012), todos ellos de referencia en los diferentes campos abordados (computación Grid, simulación y servicios de computación, respectivamente).

Bibliografía

[1] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., 2003.

[2] S. Hernández, J. Fabra, and P. Álvarez, "Framework para el despliegue automático de workflows científicos en entornos grid," Proyecto Fin de Carrera, Universidad de Zaragoza, 2011.

[3] J. Yu and R. Buyya, "A taxonomy of workflow management systems for Grid computing," *Journal of Grid Computing*, vol. 3, no. 3-4, pp. 171–200, 2005.

[4] M. Rahman, R. Ranjan, R. Buyya, and B. Benatallah, "A taxonomy and survey on autonomic management of applications in grid computing environments," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 16, pp. 1990–2019, 2011.

[5] K. Plankensteiner, J. Montagnat, and R. Prodan, "IWIR: a language enabling portability across grid workflow systems," in *Proceedings of the 6th workshop on Workflows in support of large-scale science*, ser. WORKS '11, New York, NY, USA, 2011, pp. 97–106.

[6] I. Foster, "Service-Oriented Science," *Science*, vol. 308, no. 5723, pp. 814–817, 2005.

[7] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. Mcgough, D. Pulsipher, and A. Savva, "Job Submission Description Language (JSDL) Specification, Version 1.0," Global Grid Forum, Tech. Rep., 2005.

[8] A. Kulshrestha and G. Allen, "Service oriented architecture for job submission and management on grid computing resources," in *High Performance Computing (HiPC), 2009 International Conference on*. IEEE, 2009, pp. 13–19.

[9] J. Mulerikkal and P. Strazdins, "An soa approach to high performance scientific computing: Early experiences," in *High Performance Computing (HiPC), 2010 International Conference on*, 2010, pp. 1 –10.

[10] Z. Farkas and P. Kacsuk, "P-GRADE Portal: A generic workflow system to support user communities," *Future Generation Comp. Syst.*, vol. 27, no. 5, pp. 454–465, 2011.

[11] F. Berman, G. Fox, and A. J. G. Hey, *Grid Computing: Making the Global Infrastructure a Reality*, T. Hey, Ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[12] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience: Research articles," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[13] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, 1997.

[14] E. Laure, C. Gr, S. Fisher, A. Frohner, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, R. Byrom, L. Cornwall, M. Craig, A. D. Meglio, A. Djaoui, F. Giacomini, J. Hahkala, F. Hemmer, S. Hicks, A. Edlund, A. Maraschini, R. Middleton, M. Sgaravatto, M. Steenbakkers, J. Walk, and A. Wilson, "Programming the Grid with gLite," in *Computational Methods in Science and Technology*, 2006.

[15] A. Iosup, T. Tannenbaum, M. Farrellee, D. Epema, and M. Livny, "Inter-operating grids through delegated matchmaking," *Sci. Program.*, vol. 16, no. 2-3, pp. 233–253, 2008.

[16] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[17] A. Oleksiak, A. Tullo, P. Graham, T. Kuczynski, J. Nabrzyski, D. Szejnfeld, and T. Sloan, "HPC-Europa: Towards uniform access to European HPC infrastructures," in *6th IEEE/ACM International Conference on Grid Computing (GRID 2005)*. IEEE, 2005, pp. 308–311.

[18] A. Kertész and P. Kacsuk, "GMBS: A new middleware service for making grids interoperable," *Future Generation Computer Systems*, vol. 26, no. 4, pp. 542–553, 2010.

[19] I. Rodero, F. Guim, J. Corbalan, L. L. Fong, Y. G. Liu, and S. M. Sadjadi, "Looking for an evolution of grid scheduling: Meta-brokering," 2007.

[20] M. Rahman, R. Ranjan, and R. Buyya, "Cooperative and decentralized workflow scheduling in global grids," *Future Gener. Comput. Syst.*, vol. 26, no. 5, pp. 753–768, 2010.

[21] K. Leal, E. Huedo, and I. M. Llorente, "A decentralized model for scheduling independent tasks in federated grids," *Future Generation Computer Systems*, vol. 25, no. 8, pp. 840 – 852, 2009.

[22] M. Heidt, T. Dörnemann, K. Dörnemann, and B. Freisleben, "Omnivore: Integration of grid meta-scheduling and peer-to-peer technologies," in *Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid*, ser. CCGRID '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 316–323.

[23] J. Yu and R. Buyya, "A novel architecture for realizing grid workflow using tuple spaces," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, ser. GRID '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 119–128.

[24] E. Elmroth and J. Tordsson, "An interoperable, standards-based grid resource broker and job submission service," in *Proceedings of the First International Conference on e-Science and Grid Computing*, ser. E-SCIENCE '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 212–220.

[25] L. Adzigogov, J. Soldatos, and L. Polymenakos, "EMPEROR: An OGSA Grid Meta-Scheduler Based on Dynamic Resource Predictions," *Journal of Grid Computing*, vol. 3, pp. 19–37, 2005.

[26] R. M. Piro, A. Guarise, G. Patania, and A. Werbrouck, "Using historical accounting information to predict the resource usage of grid jobs," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 499 – 510, 2009.

[27] H. Li, D. Groep, and L. Wolters, "Mining performance data for metascheduling decision support in the grid," *Future Gener. Comput. Syst.*, vol. 23, no. 1, pp. 92–99, 2007.

[28] J. Fabra, S. Hernández, P. Álvarez, and J. Ezpeleta, "A framework for the flexible deployment of scientific workflows in grid environments," *To appear in The Third International Conference on Cloud Computing, GRIDs, and Virtualizations (CLOUD COMPUTING 2012)*, 2012.

[29] S. Hernández, J. Fabra, P. Álvarez, and J. Ezpeleta, "Una solución soa para ejecutar workflows científicos en entornos grid heterogéneos," *To appear in the VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2012)*, 2012.

[30] O. Kummer, "Introduction to petri nets and reference nets," *Sozionik Aktuell*, vol. 1, pp. 1–9, 2001.

[31] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services." *Nucleic acids research*, vol. 34, no. Web Server issue, pp. W729–732, 2006.

[32] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the Kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[33] N. Carriero and D. Gelernter, "Linda in context," *Commun. ACM*, vol. 32, no. 4, pp. 444–458, 1989.

[34] S. Hernández, J. Fabra, P. Álvarez, and J. Ezpeleta, "A simulation-based scheduling strategy for scientific workflows," *To appear in the Second International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2012)*, 2012.

[35] H. R. Dalibor Klusácek, "Alea 2 – job scheduling simulator," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010)*. ICST, 2010.

[36] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, and R. Buyya, "A toolkit for modelling and simulating data grids: an extension to gridsim." *Concurrency and Computation: Practice and Experience*, vol. 20, no. 13, pp. 1591–1609, 2008.

[37] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. Epema, "The grid workloads archive," *Future Generation Computer Systems*, vol. 24, no. 7, pp. 672 – 686, 2008.

[38] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Evaluation of job-scheduling strategies for grid computing," in *Proceedings of the First IEEE/ACM International Workshop on Grid Computing*, ser. GRID '00. Springer-Verlag, 2000, pp. 191–202.

[39] D. Feitelson, "Workload modeling for performance evaluation," in *Performance Evaluation of Complex Systems: Techniques and Tools*. Berlin / Heidelberg: Springer, 2002, pp. 114–141.

[40] E. Medernach, "Workload analysis of a cluster in a grid environment," in *Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science, D. Feitelson, E. Frachtenberg, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2005, vol. 3834, ch. 2, pp. 36–61.

[41] H. Li, D. Groep, and L. Wolters, "Workload characteristics of a multi-cluster supercomputer." Springer Verlag, 2004, pp. 176–193.

[42] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 2010 Winter Simulation Conference – WSC 2010*, 2010, pp. 166–183.

[43] F. Dong and S. G. Akl, "Scheduling algorithms for grid computing : State of the art and open problems," *Components*, vol. 202, no. 4, pp. 1–55, 2006.

[44] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Workflows for e-Science: Scientific Workflows for Grids*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[45] S. Hwang and C. Kesselman, "A flexible framework for fault tolerance in the grid," *Journal of Grid Computing*, vol. 1, pp. 251–272, 2003.

[46] J. Fabra, P. Álvarez, and J. Ezpeleta, "DRLinda: A Distributed Message Broker for Collaborative Interactions Among Business Processes," in *E-Commerce and Web Technologies*, ser. Lecture Notes in Computer Science, G. Psaila and R. Wagner, Eds. Springer Berlin / Heidelberg, 2007, vol. 4655, pp. 212–221.

[47] A. Iosup, O. Sonmez, and D. Epema, "DGSim: Comparing Grid resource management architectures through trace-based simulation," in *Proceedings of the 14th international Euro-Par conference on Parallel Processing*, ser. Euro-Par '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 13–25.

[48] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1507–1542, 2002.

[49] Y. Gao, H. Rong, and J. Z. Huang, "Adaptive grid job scheduling with genetic algorithms," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 151 – 161, 2005.

[50] H. Liu, A. Abraham, and A. E. Hassanien, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1336 – 1343, 2010.

[51] W.-N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 1, pp. 29 –43, 2009.

[52] R. Sakellariou, "Job scheduling on the grid: Towards sla-based scheduling," *Computer*, vol. 16, p. 207–222, 2008.

[53] V. Stantchev and C. Schröpfer, "Negotiating and enforcing qos and slas in grid and cloud computing," in *Advances in Grid and Pervasive Computing*, ser. Lecture Notes in Computer Science, N. Abdennadher and D. Petcu, Eds. Springer Berlin / Heidelberg, 2009, vol. 5529, pp. 25–35.

[54] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, "Examining the challenges of scientific workflows," *Computer*, vol. 40, no. 12, pp. 24 –32, 2007.

[55] S. B. Davidson and J. Freire, "Provenance and scientific workflows: challenges and opportunities," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1345–1350.

[56] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche, "The open provenance model core specification (v1.1)," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 743 – 756, 2011.

[57] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen, "Advanced resource connector middleware for lightweight computational grids," *Future Generation Computer Systems*, vol. 23, no. 2, pp. 219 – 240, 2007.

[58] D. W. Erwin, "UNICORE—a grid computing environment," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1395–1410, 2002.

[59] D. Anderson, "BOINC: a system for public-resource computing and storage," in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, 2004, pp. 4 – 10.