Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Computación de Altas Prestaciones sobre Entornos Grid en Aplicaciones Biomédicas: Simulación de la Actividad Eléctrica Cardiaca y Diseño de Proteínas

Tesis Doctoral

Requisito para la Obtención del Grado de Doctor en Informática
por la Universidad Politécnica de Valencia

Valencia, 3 de septiembre de 2007

Autor: **Germán Moltó Martínez**

Director: Dr. D. Vicente Hernández García

Title

High Performance Computing over Grid Environments in Biomedical Applications: Cardiac Electrical Activity Simulation and Protein Design

Abstract

The important research advances, in many scientific areas, have been fostered by an improvement of the computational strategies employed. As an example, High Performance Computing enables the collaborative usage of multiple processors to accelerate the resolution of scientific problems, and even to face large problems.

However, there are several scientific applications whose computational requirements can exceed the computing capacities of a single organisation. This way, the recent advances in the bandwidth of the communication networks have leveraged the idea of joining geographically distributed resources, creating a global computing infrastructure known as the Grid.

This thesis combines High Performance Computing and Grid Computing in order to accelerate the execution of scientific applications, and to allow solving problems that can not be solved, in a reasonable time, with the resources of a single organisation.

For that purpose, a system that offers an abstraction layer to simplify the execution of general scientific applications, in Grid infrastructures, has been developed. This system, called GMarte, provides metascheduling functionality for the concurrent execution of parallel applications on resources based on the Globus Toolkit, the standard software in computational Grids. Later, and according to the current trend towards service-oriented architectures, a metascheduler Grid service has been created, featuring interoperability and based on standard technologies. This Grid service offers metascheduling functionality to multiple clients, which use high-level graphical applications to interact with it, using security mechanisms for data protection. This way, it is possible to simplify and foster the usage of Grid technologies for the efficient execution of scientific applications.

The proposed computational approach has been applied to two biomedical applications: the cardiac electrical activity simulation and the protein design with targeted properties. First of all, a parallel simulation system of the action potential propagation on cardiac tissues has been developed. Secondly, an efficient system has been implemented for the design of proteins. In both cases, the usage of High Performance Computing has enabled to accelerate the executions as well as to face larger dimension problems.

Finally, executions of both applications have been performed over different Grid deployments, in order to evaluate the advantages of a strategy that combines both computationally advanced techniques.

Títol

Computació d'Altes Prestacions en Entorns Grid en Aplicacions Biomèdiques: Simulació de l'Activitat Elèctrica Cardiaca i Disseny de Proteines

Resum

Els importants avanços en la investigació, de nombroses àrees de coneixement, han vingut propiciats per una millora en les estratègies de computació emprades. A mode d'exemple, la Computació d'Altes Prestacions permet la utilització col·laborativa de múltiples processadors per a accelerar la resolució de problemes científics, i fins i tot abordar problemes de major dimensió.

No obstant, hi ha diverses aplicacions on els seus requisits computacionals poden excedir la capacitat de còmput d'una única organització. En aquest sentit, els increments en l'ample de banda de les xarxes de comunicacions han propiciat la idea d'unir recursos computacionals geogràficament distribuïts, proporcionant una infraestructura global de computació coneguda com el Grid.

En aquesta tesi es combina la Computació d'Altes Prestacions i la Computació en Grid amb l'objectiu d'accelerar l'execució d'aplicacions científiques, i permetre la resolució de problemes que no poden ser abordats, en temps raonable, amb els recursos d'una sola organització.

Per a això s'ha desenvolupat un sistema que ofereix una capa d'abstracció que simplifica l'execució d'aplicacions científiques generals sobre infraestructures Grid. Aquest sistema, anomenat GMarte, ofereix funcionalitat de metaplanificació de tasques per a l'execució concurrent d'aplicacions paral·leles sobre recursos basats en Globus Toolkit, el programari estàndard en Grids computacionals. Posteriorment, i d'acord amb la tendència actual cap a les arquitectures programari orientades a serveis, s'ha construït un servei Grid de metaplanificació genèric, interoperable i basat en tecnologies estàndard. Aquest servei Grid ofereix funcionalitat de metaplanificador a múltiples clients, que interactuen amb ell per mitjà de ferramentes gràfiques d'alt nivell, utilitzant mecanismes de seguretat per a la protecció de dades. D'aquesta manera s'aconsegueix simplificar i potenciar la utilització de les tecnologies Grid per a l'execució eficient d'aplicacions científiques.

L'estratègia de computació proposada s'ha utilitzat en dues aplicacions biomèdiques: la simulació de l'activitat elèctrica cardíaca i el disseny de proteïnes de propòsit específic. En primer lloc, s'ha desenvolupat un sistema de simulació de la propagació del potencial d'acció en teixits cardíacs. En segon lloc s'ha implementat un sistema eficient per al disseny de proteïnes de propòsit específic. En les dues aplicacions, la utilització de la Computació d'Altes Prestacions ha permès accelerar les execucions, amés d'abordar problemes de major dimensió.

Finalment s'han realitzat execucions d'ambdues aplicacions sobre diversos desplegaments computacionals Grid, per avaluar els avantatges d'aquesta estratègia de computació combinada.

Resumen

Los importantes avances en la investigación, de numerosas áreas de conocimiento, han venido propiciados por una mejora en las estrategias de computación empleadas. A modo de ejemplo, la Computación de Altas Prestaciones permite la utilización colaborativa de múltiples procesadores para acelerar la resolución de problemas científicos, e incluso abordar problemas de mayor dimensión.

Sin embargo, existen diversas aplicaciones cuyos requisitos computacionales pueden llegar a exceder la capacidad de cómputo de una única organización. En este sentido, los recientes incrementos en el ancho de banda de las redes de comunicaciones han propiciado la idea de unir recursos computacionales geográficamente distribuidos, proporcionando una infraestructura global de computación conocida como el Grid.

En esta tesis se combina la Computación de Altas Prestaciones y la Computación en Grid con el objetivo de acelerar la ejecución de aplicaciones científicas, y permitir la resolución de problemas que no pueden ser abordados, en tiempo razonable, con los recursos de una sola organización.

Para ello se ha desarrollado un sistema que ofrece una capa de abstracción que simplifica la ejecución de aplicaciones científicas generales sobre infraestructuras Grid. Este sistema, denominado GMarte, ofrece funcionalidad de metaplanificación de tareas para la ejecución concurrente de aplicaciones paralelas sobre recursos basados en Globus Toolkit, el software estándar en Grids computacionales. Posteriormente, y de acuerdo a la tendencia actual hacia las arquitecturas software orientadas a servicios, se ha construido un servicio Grid de metaplanificación genérico, interoperable y basado en tecnologías estándar. Este servicio Grid aporta funcionalidad de metaplanificador a múltiples clientes, que interactúan con él por medio de herramientas gráficas de alto nivel, utilizando mecanismos de seguridad para la protección de datos. De esta manera se consigue simplificar y potenciar la utilización de las tecnologías Grid para la ejecución eficiente de aplicaciones científicas.

La estrategia de computación propuesta se ha utilizado en dos aplicaciones biomédicas: la simulación de la actividad eléctrica cardiaca y el diseño de proteínas de propósito específico. Para ello se ha desarrollado, en primer lugar, un sistema de simulación paralelo de la propagación del potencial de acción en tejidos cardiacos. En segundo lugar se ha implementado un sistema eficiente para el diseño de proteínas de propósito específico. En ambos casos, la utilización de la Computación de Altas Prestaciones ha permitido acelerar las ejecuciones y abordar problemas de mayor dimensión.

Finalmente se han realizado ejecuciones de ambas aplicaciones sobre diversos despliegues computacionales Grid, con el objetivo de evaluar las ventajas de una estrategia que combina ambas técnicas computacionalmente avanzadas.

Agradecimientos

En primer lugar quiero dar las gracias a mis padres, por tantas razones que se podría llenar esta memoria sólo con ellas. Su apoyo a lo largo de toda mi vida ha sido determinante para llegar hasta este punto. Agradezco también a Clara su comprensión y paciencia por las horas que he invertido en este trabajo. Espero devolverle con creces el tiempo que se merece.

Quiero también agradecer a los miembros del Grupo de Redes y Computación de Altas Prestaciones (GRyCAP) el apoyo prestado a lo largo de mi trayectoria investigadora en este área tan apasionante como es la Computación de Altas Prestaciones y las tecnologías Grid. En especial a José Miguel Alonso y a Vicente Hernández, por confiar en mí desde el primer día y proporcionarme el soporte y la ayuda necesaria para llevar adelante este trabajo. Sin olvidarme de Andrés, Carlos, Damián, Daniel, David, Elizabeth, Fernando, Gabi, Gabri, Javier, José Román, José Vicente, Nacho, Miguel, Pedro y Roberto, todos ellos buenos compañeros ante el teclado.

Aprovecho para expresar mi agradecimiento a la gente del Departamento de Sistemas Informáticos y Computación (DSIC) de la UPV, por recordarme cada día que la docencia es igual de importante que la investigación. En especial a Inma, Marisa, Salva, Toni, Ramón y Quique, por hacer de las comidas quizá uno de los momentos más agradables del día.

Como olvidarme de aquellos compañeros, y al mismo tiempo amigos, que las circunstancias nos han obligado a separarnos, pero el contacto y el recuerdo permanece. Un abrazo para Carlos Campos, el mejor portugués que he conocido, y para mis compañeros de carrera y de fatigas Jorge Civera, Miguel Masmano y Sergio Grau.

Asímismo, quiero expresar mi agradecimiento a los miembros del Grupo de Bioelectrónica del Centro de Investigación e Innovación en Bioingeniería Ci^2B, en especial a Javier Saiz, Chema Ferrero, Beatriz Trénor, Lucía Romero y Marta Monserrat, sin cuya inestimable colaboración hubiera sido complicado alcanzar los objetivos marcados. Sin olvidarme de todos los estudiantes de doctorado en Bioingeniería con los que he tenido la oportunidad de colaborar: Oscar Alberto Henao, Catalina Tobón, Karen Cardona y Esteban Ramírez. A todos ellos, gracias por utilizar y sufrir el sistema de simulación CAMAEC.

Quiero agradecer también a Alfonso Jaramillo y a Pablo Tortosa, del laboratorio de Bioquímica de L'École Polytechnique, por su experiencia y ayuda en el área de la optimización de proteínas de

propósito específico, así como por la validación del sistema desarrollado.

Finalmente, y no por ello menos importante, quiero aprovechar el momento para agradecer a mis amigos de toda la vida los buenos momentos que hemos pasado, y los que nos quedan por disfrutar. Un abrazo especial para David, Jacinto, José Blas, José Puchades, Mayte, Ramón, Raúl y Vicente.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

"The art of programming is the art of organising complexity, of mastering multitude and avoiding its bastard chaos as effectively as possible". E. Dijkstra, Computational Scientist.

En este capítulo se presenta el ámbito de aplicación de la presente tesis doctoral y las principales líneas de actuación. En primer lugar se realiza una breve motivación y descripción de los objetivos a alcanzar. A continuación se describen las principales aportaciones de la tesis en cada una de las áreas relacionadas. Finalmente se comenta la estructura de la tesis, detallando la temática abordada en cada capítulo.

1.1. Motivación

Los importantes avances que se han producido en las últimas décadas, en el área de la computación, han propiciado un impulso notable en la investigación dentro de numerosos campos del conocimiento. Actualmente resulta incuestionable la necesidad de utilizar computadores para realizar actividades científicas, especialmente en el campo de la biomedicina y la bioinformática. Este tipo de aplicaciones requiere técnicas de computación avanzada para gestionar tanto el elevado coste computacional como la ingente cantidad de datos generada en los procesos de simulación.

Para ello, las técnicas de Computación de Altas Prestaciones permiten la utilización colaborativa de múltiples procesadores, bien sea para resolver de forma más rápida un problema o para abordar problemas de mayor dimensión. Uno de los casos de mayor repercusión de utilización de esta tecnología fue la secuenciación del genoma humano, llevado a cabo por la empresa Celera y publicado en la revista Science [1]. La utilización de una plataforma de computación basada en clusters de PCs permitió la resolución de un problema desafiante que provocó un importante paso adelante para la humanidad.

Sin embargo, el avance de la ciencia está propiciando abordar problemas científicos cuyos requisitos computacionales pueden exceder los recursos de una única organización. Por ello, y gracias a los recientes incrementos en los anchos de banda de las redes de comunicación, se ha impulsado la idea de unir diferentes recursos computacionales, geográficamente distribuídos a lo largo del mundo, para crear una infraestructura global de computación distribuida conocida como el *Grid*. De este modo, la Computación en Grid surge para satisfacer las elevadas demandas computacionales de numerosas aplicaciones. Se posibilita así abordar problemas de una envergadura tal que sería imposible su resolución, en un tiempo razonable, con los recursos computacionales de una única organización.

En esta tesis doctoral se combina la Computación de Altas Prestaciones con la Computación en Grid con el objetivo de acelerar la ejecución de aplicaciones científicas y permitir abordar problemas de mayor dimensión. Esta estrategia computacional se aplicará a dos campos diferentes dentro de la biomedicina: la simulación de la actividad eléctrica cardiaca y el diseño de proteínas de propósito específico.

En el campo de la electrofisiología cardiaca, los modelos matemáticos de propagación del potencial de acción están considerados como una herramienta muy potente, y de gran utilidad para comprender mejor su influencia en patologías cardiacas como la fibrilación ventricular, la más mortal de todas las arritmias. Por medio del modelado, y su posterior simulación en un computador, es posible realizar un estudio previo del origen y la evolución de la fibrilación, mediante la formulación y el análisis de multitud de hipótesis. Posteriormente, estas hipótesis serán contrastadas en vivo, reduciendo de esta manera el número de complejas pruebas intrusivas, convirtiendo muchas de ellas en simulaciones basadas en computador.

La simulación de la actividad eléctrica cardiaca es un proceso con unos elevados requisitos, tanto computacionales como de memoria, donde una simulación de propagación del potencial de acción sobre un tejido de tamaño medio puede requerir unos tiempos de simulación del orden de días en un PC estándar. Por lo tanto, el empleo de técnicas de Computación de Altas Prestaciones podrá reducir en gran medida el tiempo de simulación mediante el uso colaborativo de múltiples procesadores, posibilitando el estudio de tejidos cardiacos de mayor dimensión durante más tiempo.

Esta línea de trabajo surge fruto de la colaboración entre el Grupo de Redes y Computación de Altas Prestaciones (GRyCAP) del Instituto de Aplicaciones de las Tecnologías de la Información y las Comunicaciones Avanzadas (ITACA) y el Grupo de Bioelectrónica del Centro de Investigación e Innovación en Bioingeniería (Ci^2B), ambos pertenecientes a la Universidad Politécnica de Valencia. Los códigos de simulación cardiaca, desarrollados por este último grupo, resultaban excesivamente lentos al tratar de abordar problemas complejos, como por ejemplo el estudio del fenómeno conocido como *reentrada*, donde una propagación anormal de la señal eléctrica por el tejido cardiaco puede desembocar en un estado de fibrilación ventricular. Para ello, resulta imprescindible llevar a cabo simulaciones sobre tejidos de un tamaño elevado durante un tiempo considerable, precisando

una elevada cantidad de memoria y unos tiempos de computación del orden de varios días en una plataforma secuencial.

Existen además numerosos casos de estudio cardiacos, como la búsqueda de ventanas vulnerables en isquemia o el análisis de la influencia de nuevos fármacos, que requieren la ejecución de múltiples simulaciones paramétricas. Este tipo de simulaciones son muy apropiadas para su ejecución en una infraestructura Grid de alta productividad, donde todas ellas podrán ejecutarse de manera concurrente. Por lo tanto, la utilización de una estrategia de ejecución que combine la Computación de Altas Prestaciones, para acelerar una simulación, con la Computación en Grid, para acelerar la ejecución de múltiples simulaciones, proporciona herramientas eficientes para avanzar en el conocimiento de la electrofisiología cardiaca.

Por otra parte, el proceso de optimización de proteínas de propósito específico representa un nuevo campo dentro de la biología estructural, que permite el diseño de proteínas que cumplan una determinada funcionalidad. Sus aplicaciones son diversas y abarcan múltiples campos como la medicina, para el tratamiento de enfermedades neurodegenerativas, y la biotecnología, para el desarrollo de biosensores. Una de las posibles alternativas, para poder llevar a cabo el proceso de optimización, consiste en emplear técnicas de optimización combinatoria, para evaluar diferentes configuraciones de la proteína en base a sus constituyentes aminoácidos. Este proceso requiere de numerosas horas de CPU, debido principalmente al elevado número de combinaciones existentes y a la gestión de la información necesaria para afrontar el proceso.

Es por ello que la Computación de Altas Prestaciones puede aportar numerosas ventajas a este problema mediante la utilización, en primer lugar, de estructuras de datos eficientes que exploten la jerarquía de memoria de los computadores modernos y reduzcan la cantidad de memoria necesitada. Además, la utilización colaborativa de múltiples procesadores permitirá el diseño de proteínas de mayor dimensión con una mayor funcionalidad.

Esta línea de trabajo surge fruto de la colaboración entre el GRyCAP y el Laboratorio de Bioquímica de L'École Polytechnique, en París. Los códigos de optimización de proteínas secuenciales, desarrollados por este último grupo, presentaban dificultades para la optimización de proteínas de dimensiones elevadas, precisando técnicas de computación avanzada. Además, este proceso da lugar a tareas basadas en el modelo de alta productividad, para el estudio de diferentes configuraciones de una proteína, siendo el Grid la estrategia de computación apropiada para su eficiente ejecución.

Finalmente, la ejecución transparente y eficiente de aplicaciones científicas en infraestructuras Grid, precisa el desarrollo de componentes de alto nivel que faciliten la interacción del usuario con estos despliegues computacionales. En este sentido, resulta imprescindible el desarrollo de un sistema de Computación en Grid que permita, de forma sencilla, la ejecución eficiente de aplicaciones científicas basadas en Computación de Altas Prestaciones.

1.2. Objetivos y Principales Aportaciones

El principal objetivo de esta tesis es la utilización combinada de la Computación de Altas Prestaciones y la Computación en Grid para su aplicación concreta al problema de la simulación de la actividad eléctrica cardiaca y al proceso de optimización de proteínas de propósito específico.

Para ello, se plantea, en primer lugar, el desarrollo e implementación de un simulador paralelo y eficiente de la propagación del potencial de acción sobre tejidos cardiacos. Este simulador aprovechará la potencia de cómputo de un cluster de PCs para reducir el tiempo de ejecución de una simulación, mediante la distribución del cálculo entre los diferentes procesadores. Además, la utilización de una infraestructura de memoria distribuida permitirá la simulación de tejidos de mayor dimensión, mediante la correcta distribución de las estructuras de datos requeridas. Este simulador se desplegará en producción en el Grupo de Bioelectrónica del Ci^2B, permitiendo la ejecución de un mayor número de simulaciones por unidad de tiempo y aumentando de esta manera la productividad investigadora. La principal aportación de la tesis en este área supone, por lo tanto, la creación de un sistema de simulación de estas características. Esto permitirá avanzar en el conocimiento de la actividad eléctrica cardiaca y su influencia en problemas cardiacos como la fibrilación ventricular o la isquemia.

En segundo lugar, se implementará una aplicación eficiente orientada al proceso de diseño de proteínas de propósito específico usando técnicas de Computación de Altas Prestaciones. El objetivo y principal aportación de esta tesis consiste en mejorar la aplicación secuencial existente, desarrollada en el Laboratorio de Bioquímica de L'École Polytechnique, mediante técnicas de gestión eficiente de la memoria, así como permitir abordar problemas más grandes mediante el uso colaborativo de múltiples procesadores. La utilización de una herramienta de estas características, por parte de dicho grupo de investigación, permitirá diseñar proteínas de mayor funcionalidad que, hasta el momento, no han podido ser diseñadas con las herramientas existentes.

El tercer objetivo fundamental de esta tesis consiste en el diseño, desarrollo e implementación de un sistema de gestión de la ejecución eficiente de aplicaciones científicas, de carácter general, sobre un Grid Computacional. De manera particular, el sistema acelerará la ejecución de los casos de estudio cardiacos y de proteínas mediante el aprovechamiento eficiente de una infraestructura de computación distribuida. En ese sentido, el desarrollo de una capa software, que simplifique y potencie el uso de las tecnologías Grid, supondrá un gran avance en el auge e impacto que estas tecnologías están teniendo actualmente en el ámbito científico. Además, el sistema deberá ser suficientemente genérico para ser utilizado en la ejecución de otras aplicaciones científicas sobre infraestructuras Grid.

Más aún, y con el objetivo de construir componentes genéricos, se plantea el diseño e implementación de un servicio Grid de metaplanificación de tareas. Este servicio ofrecerá, como funcionalidad, la gestión de tareas ante múltiples clientes mediante interfaces gráficos y herramientas de alto nivel, apropiadas para su utilización por parte de usuarios no expertos en el Grid. Por lo tanto, la aporta-

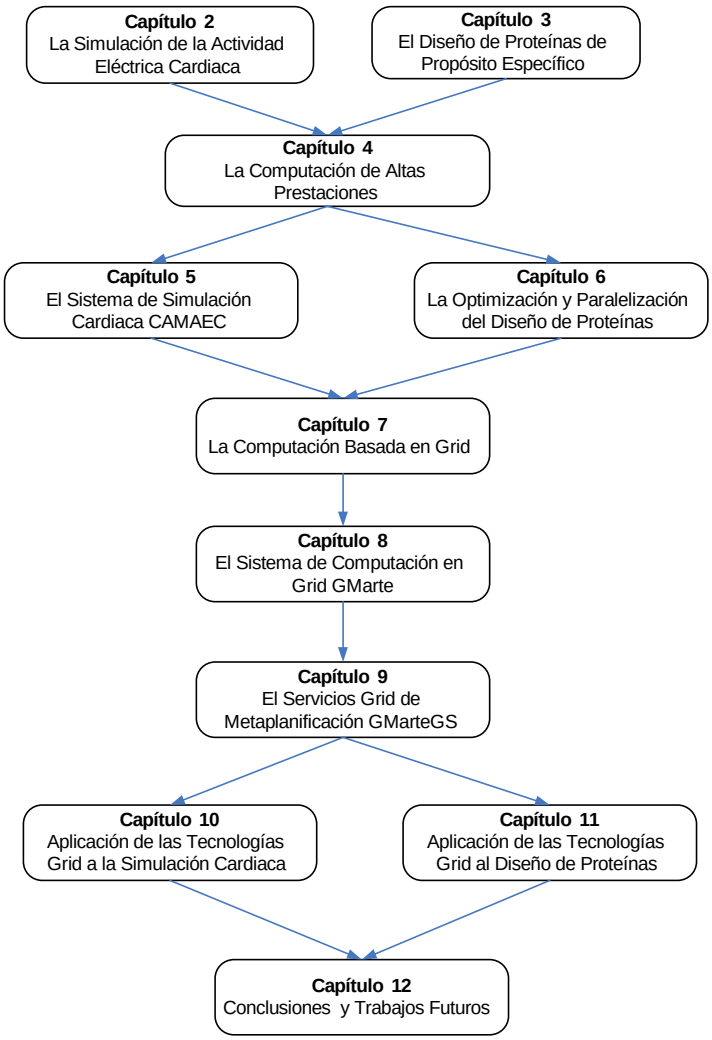


Figura 1-1: Estructuración de la tesis en capítulos y relación de dependencias entre ellos.

ción de la tesis en este área supone la consecución de un componente genérico e interoperable, basado en estándares, para la ejecución transparente de tareas científicas sobre Grids computacionales. De esta manera, el sistema desarrollado podrá ser utilizado no sólo para las aplicaciones descritas en esta tesis, sino para cualquier otra aplicación.

Además, se pretende integrar ambas aplicaciones paralelas con el sistema de Computación en Grid para poder realizar múltiples ejecuciones concurrentes, cada una de ellas ejecutándose en paralelo en las máquinas multiprocesador de la infraestructura Grid empleada.

Finalmente, el proceso llevado a cabo permitirá evaluar las ventajas de combinar la Computación de Altas Prestaciones con la Computación en Grid para la ejecución eficiente de aplicaciones. Para ello, se realizarán ejecuciones de casos de estudio realistas, compuestos por múltiples simulaciones, sobre plataformas Grid geográficamente distribuidas.

1.3. Estructura de la Tesis

Esta tesis está estructurada en base a capítulos que presentan una relación de dependencia tal y como se muestra en la Figura 1-1. En primer lugar, el capítulo 2 realiza una introducción al problema de la simulación de la actividad eléctrica cardiaca, destacando el estado del arte y las principales estrategias de resolución de este problema. A continuación, el capítulo 3 presenta el diseño de proteínas de propósito específico y revisa el estado del arte en dicho campo.

Más adelante, el capítulo 4 describe las técnicas de Computación de Altas Prestaciones y las diferentes librerías empleadas en el desarrollo de las aplicaciones realizadas en esta tesis. El capítulo 5 describe la arquitectura y funcionalidad del sistema de simulación cardiaca llamado CAMAEC, abordando la estrategia de paralelización aplicada y evaluando sus prestaciones paralelas. A continuación, el capítulo 6 resume el desarrollo de una aplicación eficiente, llamada gBiObj, para la optimización en paralelo de proteínas.

Posteriormente, el capítulo 7 introduce el concepto de Computación en Grid, describiendo el estado del arte de esta tecnología, así como los principales proyectos de investigación relacionados. A continuación, el capítulo 8 presenta a GMarte, el sistema de Computación en Grid desarrollado para soportar la ejecución de aplicaciones científicas sobre infraestructuras Grid. En esta linea, el capítulo 9 describe la adaptación de GMarte a una arquitectura orientada a servicios para la creación de un metaplanificador multiusuario genérico e interoperable.

El capítulo 10 describe la aplicación de las tecnologías Grid, basadas en GMarte, al problema de la simulación cardiaca, realizando la ejecución de diversos casos de estudio a fin de evaluar las ventajas de la estrategia de computación propuesta. De manera similar, el capítulo 11 describe la aplicación del sistema GMarte a la optimización de proteínas.

Finalmente, el capítulo 12 resume las principales aportaciones de la tesis, destacando los proyectos de investigación en los que se ha enmarcado y citando las diferentes publicaciones de investigación a las que ha dado resultado. Por último, se incluyen también las líneas futuras de actuación en cada una de las áreas abiertas de investigación

Capítulo 2

La Simulación de la Actividad Eléctrica Cardiaca

"The heart has reasons that reason cannot know". Blaise Pascal, Mathematician, Phylosopher and Physicist.

En este capítulo se describe el comportamiento eléctrico del corazón, enfatizando su influencia en la aparición de determinadas patologías. Posteriormente se presenta el estado del arte de la simulación de la actividad eléctrica cardiaca, exponiendo las principales estrategias existentes para abordar el problema. A continuación, la discusión se centra en el problema matemático subyacente que se pretende resolver, describiendo el elevado coste computacional que involucran las simulaciones de la actividad eléctrica cardiaca. En este sentido, se justifica la necesidad de utilizar un esquema combinado, basado en Computación de Altas Prestaciones y Computación en Grid, para acelerar el proceso de simulación. Finalmente, se comentan los trabajos relacionados en el área.

2.1. Introducción

Según la Red Europea del Corazón [2], las enfermedades cardiovasculares provocan casi la mitad de todas las muertes ocurridas en Europa. Además, el número de muertes causadas por enfermedades cardiovasculares derivadas del consumo de tabaco aumentó un 13 % durante los años 1990 y 2000. Este aumento en la tasa de mortalidad ha provocado que, tanto la sociedad como las autoridades sanitarias, se planteen ampliar el conocimiento existente sobre las diferentes patologías cardiacas.

Dentro de las enfermedades cardiovasculares, las arritmias ventriculares son una de las principales causas de mortalidad en los países desarrollados. A pesar de las numerosas investigaciones llevadas a cabo, los mecanismos de generación, mantenimiento y terminación de estas arritmias no están totalmente aclarados. En la actualidad, es mucha la información disponible acerca de la actividad

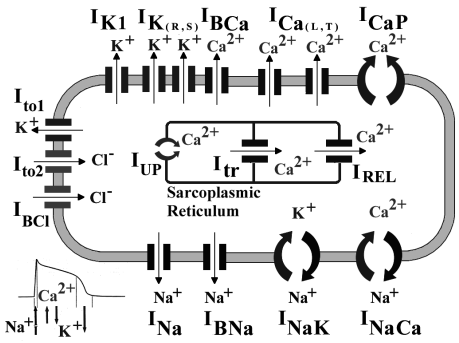


Figura 2-1: Esquema de una célula cardiaca, detallando las principales corrientes iónicas que atraviesan su membrana.

eléctrica cardiaca, sobre todo a nivel celular, pero todavía existe mucho desconocimiento en niveles de modelización más altos como, por ejemplo, en tejidos cardiacos de una dimensión adecuada o incluso en un corazón completo.

Las células cardiacas presentan una forma aproximadamente cilíndrica, con una longitud típica de 30 a 100 μm y un diámetro de 8 a 20 μm [3], estando rodeadas por una membrana que delimita el *medio extracelular* del *medio intracelular*. Estas células presentan una diferencia de potencial eléctrico, entre el medio intracelular y el extracelular, conocido como *potencial de membrana*, debido a las diferentes concentraciones de iones entre el interior y el exterior de la célula. Esto provoca un flujo de intercambio mutuo de iones, entre ambos medios, a través de los *canales iónicos* existentes en la membrana celular, dando lugar a las *corrientes iónicas*. La Figura 2-1 muestra un esquema de una célula cardiaca con las principales corrientes iónicas que atraviesan la membrana celular. En ella se pueden observar, entre otras, las corrientes producidas por los flujos de iones de Calcio (Ca^{2+}), Potasio (K^+) y Sodio (Na^+).

Las células cardiacas presentan un comportamiento excitable, es decir, reaccionan ante la ocurrencia de un estímulo eléctrico de una determinada amplitud y, bajo ciertas condiciones, pueden desarrollar un *potencial de acción*. Se conoce como *estímulo umbral* a aquel impulso eléctrico con una amplitud suficiente para desencadenar un potencial de acción. La Figura 2-2 muestra un ejemplo del potencial de acción desarrollado por una célula cardiaca, al ser excitada por medio de un estímulo eléctrico. Como se puede apreciar, un potencial de acción viene representado por la secuencia de valores de diferencia del potencial eléctrico al que está sometida la membrana celular a lo largo del tiempo, típicamente expresado en milisegundos.

En un potencial de acción se pueden distinguir tres fases principales. En primer lugar, la fase de *depolarización* es provocada por la inyección de un estímulo o cuando el potencial de membrana se vuelve positivo. La fase de depolarización debe ser lo suficientemente grande para poder alcanzar un cierto valor umbral a partir del cual se obtiene propagación eléctrica [4] y se caracteriza por

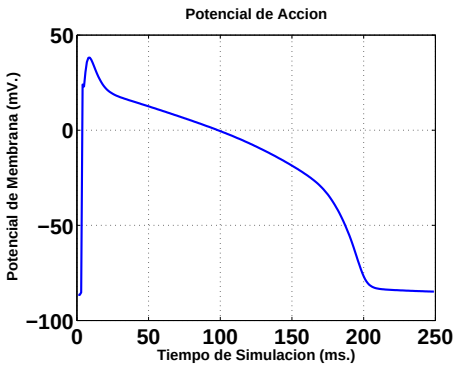


Figura 2-2: Potencial de acción desarrollado por una célula cardiaca durante 250 ms.

un aumento rápido del potencial de membrana. A continuación se produce la fase de *plateu*, que suele durar unos 100-200 ms. dependiendo del tipo de célula, caracterizada por un lento descenso del potencial de membrana. Por último, la fase de *repolarización* provoca un rápido descenso del potencial de membrana, hasta alcanzar nuevamente el estado estacionario inicial, equivalente al *potencial de reposo* de la célula.

Una célula que ha sido estimulada eléctricamente permanece en el denominado *periodo refractario* durante un cierto tiempo. Este intervalo de tiempo se divide en dos etapas. Cuando una célula se encuentra en el *periodo refractario absoluto*, ningún estímulo provocará el desarrollo de un potencial de acción. Sin embargo, transcurrido un cierto tiempo, cuando la célula alcanza el *periodo refractario relativo*, es posible desencadenar un potencial de acción mediante una estimulación eléctrica suficientemente intensa.

Las células cardiacas están unidas entre sí por medio de unas estructuras, denominadas *miofibrillas*, formando las fibras musculares cardiacas. Estas estructuras tienen la peculiaridad de que, ante la presencia de iones de calcio, reaccionan mediante una rápida contracción. En este sentido, es conocido que un potencial de acción libera iones de calcio que causan dicha reacción. Sin embargo, la contracción de una fibra muscular requiere la acción simultánea de todas sus miofibrillas. Por lo tanto, para poder generar un latido y bombear adecuadamente la sangre del corazón, es precisa la contracción síncrona de las fibras musculares. Para ello, es necesario que todas las células desarrollen un potencial de acción, debiendo ser convenientemente excitadas a lo largo del tejido cardiaco.

Dicho de otro modo, las células del nodo sinoauricular (Figura 2-3), conocido como el marcapasos natural del corazón, generan una serie de impulsos eléctricos periódicos que se propagan a través de todo el tejido cardiaco. Esto desencadena un frente de onda eléctrico encargado de excitar a las células cardiacas, de modo que la liberación asociada de calcio provoca la contracción síncrona de las miofibrillas, dando lugar a un latido. De esta manera se consigue cumplir con la función de bombeo de sangre al resto del cuerpo.

Sin embargo, es posible que la existencia de alguna patología en el tejido provoque una contracción

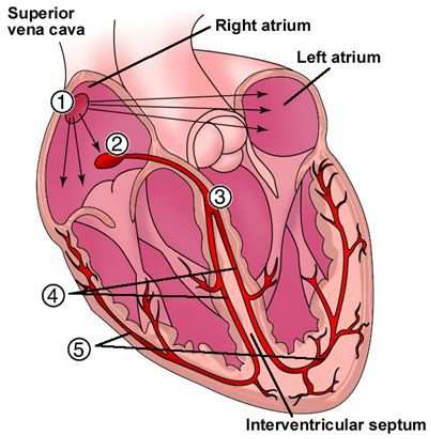


Figura 2-3: Esquema de un corazón completo seccionado verticalmente, mostrando el nodo sinoauricular (1), el nodo atrioventricular (2), el haz atrioventricular de His (3), con sus correspondientes ramas izquierda y derecha (4), y las fibras de Purkinje (5).

asíncrona del mismo, debido a una propagación deficiente del frente de onda eléctrico. Si esto ocurre, las células serán excitadas pero no se conseguirá bombear la sangre de manera adecuada, siendo además posible que el corazón esté fibrilando. De hecho, el principal objetivo de un desfibrilador consiste en inyectar un impulso de corriente, en el músculo cardiaco, que lleve a las células a un estado eléctrico controlado, a fin de que el nodo sinoauricular vuelva a generar los impulsos eléctricos que marquen el ritmo del corazón.

Dada la importancia del comportamiento de las células cardiacas, y su influencia en patologías cardiacas que se manifiestan a niveles más altos, como son los tejidos, resulta importante comprender su funcionamiento interno. En este sentido, a lo largo de las últimas décadas han surgido numerosos modelos celulares que tratan de obtener aproximaciones al comportamiento eléctrico de las células cardiacas. Son los conocidos como modelos celulares cardiacos.

2.1.1. Modelos Celulares Cardiacos

Las observaciones experimentales de las propiedades electrofisiológicas de las células cardiacas se han venido llevando a cabo desde las últimas seis décadas. La invención de la técnica de fijación de potencial por Hodgking, Huxley y otros [5], a finales de los años 40, y la más reciente técnica de *patch-clamp* por Sakmann y Neher [6] ha permitido a los electrofisiologistas grabar y medir corrientes iónicas a través de la membrana celular. Estos métodos se han convertido en herramientas muy útiles para la comprensión de la generación del potencial de acción.

De hecho, los datos obtenidos con la ayuda de estas técnicas han permitido la formulación de modelos matemáticos referentes al comportamiento eléctrico de células excitables. Más concretamente, la actividad eléctrica de las células cardiacas ha sido cuantitativamente descrita, desde principio

de los años 60, por modelos cada vez más detallados a medida que se iban descubriendo nuevas corrientes iónicas, así como las diferentes propiedades de los canales.

En 1977, Beeler y Reuter (BR) [7] formularon un conjunto de ecuaciones que reproducían el potencial de acción ventricular. Este modelo, que fue posteriormente modificado por Drouhard y Roberge en 1982 [8], fue usado hasta que los modelos de Luo y Rudy (LR) fueron formulados en los años 90. El modelo LR se publicó en dos fases. La primera versión, conocida como el modelo Luo-Rudy Fase I [9], reformulaba algunas de las corrientes del modelo BR, mientras que la segunda versión, conocida como Luo-Rudy Fase II [10], era mucho más completa y estaba basada en datos recientes obtenidos mediante la técnica de patch-clamp, describiendo una amplia variedad de corrientes iónicas. De hecho, la Figura 2-1 citada anteriormente, muestra las principales corrientes que atraviesan la membrana de una célula según el modelo Luo-Rudy Fase II. La versión más actual de este modelo es el Luo-Rudy 2000 [11].

Hoy en día, los modelos de potencial de acción están resultando extremadamente útiles en el campo de la electrofisiología cardíaca, debido a su habilidad para simular con un preciso detalle los eventos eléctricos que tienen lugar a través de la membrana celular. Dado que es imposible registrar por medios experimentales todas las corrientes iónicas y el potencial de membrana, las simulaciones por computador proporcionan una alternativa muy deseable para explorar los mecanismos electrofisiológicos básicos en células bajo condiciones normales y patológicas.

Además, si el modelo celular que describe toda la dinámica de las corrientes se combina con una representación de las características eléctricas del tejido, el modelo matemático resultante, basado en un sistema de ecuaciones diferenciales, puede ser usado para simular la actividad eléctrica cardiaca.

2.2. Modelos de la Simulación Eléctrica Cardiaca

Actualmente, la simulación de la actividad eléctrica cardiaca por medio de un computador es un campo bastante amplio que involucra diferentes estrategias de resolución del problema. Dado que, a menudo, cada estrategia tiene un campo de aplicación diferente, resulta importante describir el estado del arte de dicha simulación.

Existen dos modelos principales que describen el fenómeno de la propagación eléctrica en el tejido cardiaco. En primer lugar, el modelo bidominio [12] relaciona el potencial intracelular ϕ_i con el potencial extracelular ϕ_e, a través de la densidad de corriente que atraviesa la membrana celular I_m. Utilizando la notación de [13] el modelo bidominio se puede describir mediante las siguientes expresiones:

$$\begin{aligned} \nabla \cdot \bar{\sigma_i} \nabla \phi_i &= \beta I_m \\ \nabla \cdot \bar{\sigma_e} \nabla \phi_e &= -\beta I_m - I_e \\ I_m &= C_m \cdot \frac{\partial V_m}{\partial t} + I_{ion} - I_{trans} \end{aligned} \tag{2.1}$$

En el sistema de ecuaciones 2.1, $\bar{\sigma_i}$ y $\bar{\sigma_e}$ son, respectivamente, los tensores de conductividad del medio intracelular y del extracelular, β es el ratio volumen-superficie de las células cardiacas, I_{trans} es la densidad de corriente de un estímulo transmembrana proporcionado por un electrodo intracelular, I_e es una densidad de corriente causada por un estímulo extracelular, C_m es la capacidad de membrana y V_m es la diferencia de potencial entre el medio intracelular y el extracelular (definido, por tanto, como $\phi_i - \phi_e$). El termino I_{ion} representa la densidad de corriente que fluye a través de los canales iónicos, y se computa en base a un modelo iónico celular como los descritos en la sección 2.1.1.

Como se puede observar, el modelo bidominio representa las corrientes eléctricas tanto en el medio intracelular como en el extracelular, y consta de una ecuación parabólica no lineal en ϕ acoplada con una ecuación elíptica en el potencial extracelular (ϕ_e) [14]. Sin embargo, partiendo del modelo bidominio y asumiendo que:

1. La conductividad es la misma en cualquier dirección para todos los dominios, es decir, considerando un medio isótropo.

2. Los ratios de conductividad entre las regiones intracelular y extracelular son constantes.

3. El dominio extracelular es altamente conductor y, por lo tanto, se considera como conectado a una toma de tierra ($\phi_e = 0$),

entonces el sistema 2.1 se puede simplificar, dando lugar al modelo monodominio expresado mediante la Ecuación 2.2.

$$\nabla \cdot \sigma \nabla V_m = C_m \cdot \frac{dV_m}{dt} + I_{ion} + I_{st} \tag{2.2}$$

El modelo monodominio lleva asociado una complejidad numérica y computacional menor que el bidominio. Es por esto que, a menudo, se suele emplear éste modelo para el estudio de la propagación del potencial de acción con modelos celulares más detallados. Es posible encontrar estudios de propagación realizados con el modelo bidominio, entre otros, en [15, 16, 17, 18].

En la Ecuación 2.2, V_m representa el potencial de membrana de las células y C_m indica la capacidad de membrana de la célula, dado que ésta actúa como un pequeño condensador. I_{st} especifica la estimulación eléctrica inyectada al tejido para desencadenar el potencial de acción. El término I_{ion},

tal y como se ha comentado anteriormente, se obtiene a partir de un modelo celular y representa la suma de corrientes iónicas que entran y salen de cada una de las células del tejido a través de la membrana celular.

La Ecuación 2.2 es una ecuación en derivadas parciales parabólica que modela un fenómeno de reacción-difusión [19]. La parte asociada a la reacción está determinada por el término I_{ion} que viene gobernado por el modelo celular. La parte de difusión modela la propagación del potencial de acción en el medio sobre el que se resuelve la ecuación.

Independientemente del modelo celular empleado, las ecuaciones que definen el fenómeno de la propagación del potencial de acción se deben discretizar sobre un dominio concreto. El caso más sencillo consiste en la simulación de la actividad eléctrica de una célula aislada. Posteriormente, es posible considerar la propagación unidimensional a lo largo de una fila de células. Como extensión, la propagación sobre un dominio 2D permite aumentar el nivel de realismo ya que considera un tejido cardiaco. La propagación sobre un tejido 3D, donde las células estén conectadas unas con otras mediante resistencias, permite considerar la influencia en la actividad eléctrica de diferentes capas de tejido. Finalmente, introducir características más realistas, como la orientación de las fibras musculares cardiacas y geometrías irregulares, permite asemejar cada vez más un modelo matemático a la realidad física.

Por lo general, el empleo de geometrías complejas suele incrementar en gran medida la talla del problema, ya que aumenta el número de puntos en los que se debe calcular una solución. Por ello, su uso se suele combinar con modelos iónicos celulares sencillos o métodos sencillos de propagación eléctrica, que compensen el coste global del proceso de simulación. Por ejemplo, en [20] se utiliza un modelo geométrico complejo de todo el miocardio incluyendo las direcciones de las fibras musculares pero como modelo de propagación se utiliza el FitzHugh-Nagumo [21], un modelo extremadamente sencillo y con un coste computacional mínimo, comparado con el cálculo de la propagación empleando el modelo monodominio.

El trabajo desarrollado en esta tesis se centra en la utilización de un esquema de propagación monodominio utilizando modelos celulares complejos, como el Luo-Rudy (Fase II y 2000), sobre una geometría bidimensional, permitiendo la simulación detallada del comportamiento interno de las células cardiacas. Es importante destacar que el trabajo, en esta parte de la tesis, surge como resultado de la colaboración con el Grupo de Bioelectrónica de la UPV, cuyo principal interés radica principalmente en investigar el funcionamiento de las células en base a modelos celulares sofisticados y computacionalmente complejos. En este sentido, la utilización de un esquema de propagación monodominio permite alcanzar los objetivos de estudio de propagación eléctrica en tejidos cardiacos.

2.3. Aproximación Computacional al Problema Matemático

Para un tejido cardiaco bidimensional la Ecuación 2.2 de propagación del potencial de acción puede reescribirse de la siguiente manera:

$$\frac{a}{2} \cdot \left(\frac{1}{R_{ix}} \cdot \frac{\partial^2 V_m}{\partial x^2} + \frac{1}{R_{iy}} \cdot \frac{\partial^2 V_m}{\partial y^2}\right) = C_m \cdot \frac{\partial V_m}{\partial dt} + I_{ion} \tag{2.3}$$

I_{ion} es la suma de las corrientes iónicas que atraviesan la membrana de la célula, a es el radio de la célula, C_m es la capacidad específica de membrana, V_m es el potencial eléctrico al que está sometido la membrana celular y R_{ix} y R_{iy} son las resistividades longitudinales y transversales. Se puede observar que los tensores de conductividad se han substituido por las correspondientes medidas inversas de resistividad entre cada par de células del tejido. En efecto, el tejido se modela como un conjunto de células interconectadas mediante resistencias.

Para poder resolver la Ecuación 2.3 en un computador es necesario realizar una discretización de la misma, tanto en espacio como en tiempo. El uso común de técnicas de separación de operadores (*operator splitting* [22]) permite desacoplar, en cierta medida, el cómputo del término de reacción con el de difusión. De esta manera se simplifica la actualización de las células y la obtención de las corrientes iónicas (término de reacción), mediante la ejecución del modelo celular, y se facilita el acoplamiento de diferentes modelos celulares. Por lo tanto, esta estrategia permitirá la implementación de diferentes modelos celulares, aprovechando la parte del cálculo dedicada a la propagación del potencial de acción (término de difusión). Además, esta forma de proceder permite utilizar métodos de integración diferentes para ambos términos, dado que es conocido que las escalas de tiempo de ambos subproblemas son muy diferentes [23].

Una de las principales técnicas empleadas para resolver las ecuaciones de propagación monodominio es el método de las diferencias finitas, donde las derivadas espaciales se aproximan por expresiones de diferencias finitas [17, 24]. El método de las diferencias finitas, para la discretización espacial, destaca por su sencillez de implementación, así como por los bajos requisitos de memoria que presenta. No obstante, la discretización de un tejido de dimensiones considerables puede dar lugar a un elevado número de puntos sobre los que obtener una solución, de manera que se superen los recursos de una sola máquina.

Para el modelo monodominio, Bogar y Keener [25] afirman que el método de Crank-Nicolson [26], para la discretización temporal, es el más estable de entre todos los métodos semi-implícitos, incluso para pasos de tiempo relativamente grandes. El principal beneficio de Crank-Nicolson es que ofrece mayor precisión que los métodos semi-implícitos convencionales. Además, en cada paso de tiempo únicamente se deben resolver sistemas de ecuaciones lineales. Sin embargo, las restricciones de los métodos explícitos y semi-implícitos en cuanto a estabilidad al aumentar el paso de tiempo de integración, todavía se mantienen [14]. La idea que subyace a esta técnica consiste en resolver de

manera implícita el término de difusión, en la Ecuación 2.3, mientras que el término de reacción se resuelve mediante métodos explícitos. Aplicando el método de Crank-Nicolson, como en [27], a la Ecuación 2.3, se llega a un sistema de ecuaciones como el que se muestra a continuación:

$$A_L \cdot V_m^t = A_R \cdot V_m^{t-1} + I_{ion}^{t-1} + I_{st}^{t-1} \tag{2.4}$$

El vector I_{ion} de la Ecuación 2.4 se calcula como la suma de las corrientes iónicas que atraviesa la membrana celular, que depende tanto de su potencial de membrana como de su estado interno, especificado por medio de un modelo celular. El vector I_{st}^{t-1} representa un estímulo dependiente del tiempo, que puede inyectarse en el tejido para desencadenar un potencial de acción. El vector V_m^t representa el potencial de membrana de cada una de las células en el instante de tiempo t. Las matrices A_L y A_R involucran la información de conductividad en el tejido cardiaco, y permanecen inalteradas a lo largo del proceso de simulación.

Por lo tanto, la simulación se convierte en un proceso iterativo donde se parte de unos valores iniciales de potencial de membrana, para cada una de las células del tejido (vector V_m). En cada paso de simulación se obtienen los valores del potencial de membrana de cada una de las células del tejido, mediante la resolución de un sistema de ecuaciones lineales.

Concretamente, para cada uno de los pasos de simulación se deben llevar a cabo las siguientes operaciones:

1. Realizar un producto matriz-vector y sumar otro vector ($A_R \cdot V_m^{t-1} + I_{ion}^{t-1}$). Además, en el caso de la inyección de un estímulo eléctrico, se debe considerar la suma de otro vector (I_{st}^{t-1}). De esta forma, se genera el vector de términos independientes del sistema de ecuaciones lineales.

2. Resolver el sistema de ecuaciones lineales para obtener el potencial de membrana de las células en el instante de tiempo actual (vector V_m^t).

3. Actualizar el estado de las células una vez conocido el potencial de membrana del estado actual, de acuerdo al modelo celular empleado, obteniendo así el nuevo vector de las corrientes iónicas que atraviesan la membrana de cada una de las células (vector I_{ion}^t).

Este proceso de simulación consta de una serie de fases, que se deben calcular para cada uno de los instantes de tiempo del proceso de simulación. Por ejemplo, una simulación de un potencial de acción de una célula cardiaca, descrita mediante el modelo Luo-Rudy Fase II, requiere un total de 250 ms. Empleando un paso de tiempo típico de 10 μs se deben realizar 25000 pasos de simulación, donde cada paso consta de las fases anteriormente descritas.

Además, una discretización espacial de un tejido bidimensional de 1000x1000 células, da lugar a un sistema lineal de 1 millón de ecuaciones y 1 millón de incógnitas, lo que proporciona una idea de la elevada dimensión del problema. De hecho, la simulación de la mayoría de fenómenos arrítmicos

requiere el empleo de modelos que representen el comportamiento de tejidos relativamente grandes, con un elevado número de células conectadas en una geometría bidimensional o tridimensional. Además, el estudio de determinadas patologías, como las situaciones de isquemia, requieren simulaciones de varios segundos que involucran varios días de ejecución en una plataforma secuencial.

El principal coste computacional viene dominado, fundamentalmente, por la ejecución del modelo celular, que incluye la resolución de varias ecuaciones (14 ecuaciones diferenciales en el modelo celular Luo-Rudy Fase II), y la resolución del sistema de ecuaciones lineales resultante de la discretización de la Ecuación 2.3 que describe la propagación del potencial de acción a lo largo del tejido.

Además del tiempo requerido por este tipo de simulaciones es importante destacar sus elevados requisitos de memoria. El modelo celular Luo-Rudy Fase II requiere almacenar, aproximadamente, 60 variables de doble precisión para cada una de las células del tejido. Por ejemplo, un tejido bidimensional de 1000x1000 células requiere 457.76 Mbytes de memoria para albergar el estado del tejido, sin tener en cuenta el espacio requerido para almacenar las matrices resultantes de la discretización de la Ecuación 2.3 (A_L y A_R).

Esta situación implica la necesidad de utilizar técnicas computacionales avanzadas, que permitan gestionar toda la complejidad involucrada en cada una de las simulaciones cardiacas. Para ello, la utilización de Computación de Altas Prestaciones supone una herramienta fundamental para la consecución de un simulador eficiente de propagación del potencial de acción.

2.3.1. Casos de Estudio Cardiacos

Además de la inherente complejidad computacional y espacial asociada a una única simulación de propagación del potencial de acción, existen multitud de estudios cardiacos que requieren la ejecución de un elevado número de simulaciones. Estos estudios precisan la ejecución de un conjunto de simulaciones que analizan el comportamiento eléctrico de un tejido, bajo determinadas circunstancias de simulación que varían ligeramente de una ejecución a otra. De hecho, un caso de estudio suele involucrar la ejecución de aplicaciones paramétricas, donde un cierto parámetro de estudio se recorre en un rango de valores, realizando diferentes simulaciones cardiacas para estudiar el impacto del parámetro en la propagación eléctrica.

Un ejemplo de este tipo de aplicaciones es el estudio del mecanismo de reentrada en condiciones de isquemia [28]. La isquemia de miocardio es un estado del corazón derivado de la falta de oxígeno en alguna zona del tejido cardíaco. En estas condiciones, donde existe una zona dañada del tejido, la propagación del potencial de acción se bloquea al llegar a la zona afectada, rodeándola y reentrando sobre la misma. La Figura 2-4 muestra un ejemplo de reentrada donde el potencial de acción se bloquea al llegar a la zona central, propagándose repetidamente en forma de espiral sobre la zona afectada. Este fenómeno puede derivar en un estado de fibrilación ventricular, de ahí la importancia de su estudio. Los estudios de ventana vulnerable en condiciones de isquemia, requieren variar

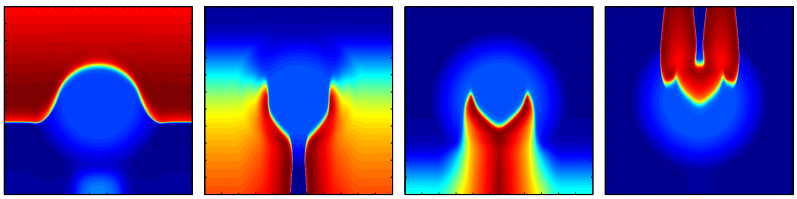


Figura 2-4: Bloqueo de la propagación del potencial de acción al llegar a una zona isquémica (zona central) y posterior rodeo de la zona afectada, para acabar estimulándola dando lugar a una propagación perpetua en espiral.

el intervalo de tiempo entre dos estímulos eléctricos consecutivos, para poder detectar el rango de valores en los que se produce reentrada. Esto se puede conseguir por medio de simulaciones paramétricas variando dicho intervalo de aplicación.

Otro ejemplo de caso de estudio cardiaco consiste en el análisis de los efectos de la isquemia tardía, proceso en el que es preciso variar las resistencias de acoplamiento en todas las dimensiones del tejido, y observar la evolución de la actividad eléctrica para diferentes ratios de anisotropía.

Finalmente, otra área que requiere de estudios paramétricos es la evaluación del impacto que tiene un medicamento sobre el comportamiento de una determinada patología cardiaca. En estos estudios se varía la concentración del fármaco dentro de un rango determinado, y se realizan simulaciones de propagación para estudiar como se ven afectadas las características de amplitud, duración y forma del potencial de acción.

Por tanto, los casos de estudio cardiacos presentan unos requisitos computacionales que, a menudo, superan los recursos de computación disponibles en un solo centro de investigación. El número de simulaciones involucradas suele estar entre 10 y 100 por caso de estudio, dando una idea de la magnitud del problema. De hecho, la ejecución de un caso de estudio, sobre una plataforma secuencial, suele requerir tiempos de computación del orden de semanas, por lo que resulta imprescindible la utilización de técnicas de computación avanzada, como la Computación en Grid, que permitan soportar su ejecución eficiente. Para ello, se plantea la ejecución concurrente de las múltiples simulaciones de un caso de estudio cardiaco sobre una infraestructura Grid, permitiendo reducir en gran medida el tiempo involucrado en la ejecución global.

2.4. Estado del Arte

La Computación de Altas Prestaciones y la Computación en Grid se utilizan en diversos campos de la ingeniería y, en este sentido, el campo de la simulación de la actividad eléctrica no está exento de estas aportaciones. A continuación se resumen las principales contribuciones a este campo que tienen una relación directa con los objetivos de esta tesis.

En [29] se realiza un estudio de diferentes estrategias de paralelización, basadas en memoria

distribuida para el problema de la simulación cardiaca bidimensional monodominio. Para ello, se realizan diversas implementaciones basadas en el paradigma maestro-esclavo, comparando los resultados obtenidos mediante la ejecución en clusters de PCs con los correspondientes incrementos de velocidad teóricos. Sin embargo, los resultados de eficiencia descritos distan mucho de ser alentadores, reportando un incremento de velocidad de 1.78 con 7 procesadores para una simulación sobre una malla de 2^{12} nodos.

En esta línea, en [30] se utiliza la computación paralela mediante memoria distribuida para resolver sistemas de reacción-difusión en tres dimensiones, aplicados tanto al modelo monodominio, como el bidominio. Como modelos celulares se utilizan el FitzHugh-Nagumo y el Luo-Rudy fase 1. Se realizan ejecuciones en un IBM SP4 con hasta 128 procesadores, obteniendo medidas de tiempo de ejecución, incrementando el tamaño del problema de manera proporcional al número de máquinas. Se realizan simulaciones de hasta $O(10^7)$ nodos.

Por otra parte, en [13] se utiliza una estrategia de paralelización del problema de simulación cardiaca monodominio (y bidominio), empleando el paradigma de memoria compartida mediante directivas OpenMP. Las ejecuciones se llevan a cabo en una máquina de memoria compartida SGI Origin 2000, obteniendo incrementos de velocidad de entre 2 y 3 al simular una malla desde 108031 hasta 500000 nodos en 4 procesadores. Al aumentar el número de procesadores, según los autores, el incremento se hace más pequeño.

Tal vez la aproximación de paralelización más eficiente al proceso de simulación la encontramos en [18] con el simulador CardioWave, que emplea paralelización mediante memoria distribuida para acelerar los cálculos al resolver las ecuaciones bidominio. Se obtienen eficiencias cercanas al 90 %, al emplear 16 procesadores sobre una máquina Cray T3E, durante la realización de una simulación monodominio.

Por otra parte, la aplicación de las tecnologías Grid al proceso de la simulación cardiaca todavía no es muy común actualmente. Recientemente, el proyecto Integrative Biology [31, 32] pretende desarrollar la infraestructura de computación Grid necesaria para la ejecución de aplicaciones de simulación relacionadas con el estudio del cáncer y del corazón. En el proyecto se propone una arquitectura para aunar el estudio biológico a múltiples niveles (celular, tejido, órgano, ser vivo). Sin embargo, todavía no existe una implementación en producción de la arquitectura propuesta.

2.5. Conclusiones

En este capítulo se ha realizado una breve introducción al problema de la simulación de la actividad eléctrica cardiaca, introduciendo los principales conceptos involucrados. Además, se ha descrito tanto el problema matemático subyacente como la aproximación llevada a cabo para su resolución, es decir, un esquema de propagación bidimensional monodominio empleando modelos

celulares detallados. En este sentido, se ha justificado la necesidad de utilizar Computación de Altas Prestaciones para abordar la ejecución de las complejas simulaciones cardiacas. También se ha descrito el concepto de caso de estudio cardiaco, destacando las ventajas que puede suponer la utilización de la Computación en Grid para la aceleración de la ejecución de los mismos.

Posteriormente, se ha realizado una revisión del estado del arte de la simulación cardiaca, analizando algunas de las estrategias de simulación existentes. A continuación, se ha revisado en la literatura la aplicación de técnicas computacionales avanzadas para la simulación de la propagación del potencial de acción.

Aunque existen aproximaciones paralelas al proceso de simulación cardiaca, todavía existen posibilidades de mejora. Por un lado, la consecución de una paralelización eficiente del problema supone un incremento de las prestaciones del sistema. Por otro lado, no se ha encontrado en la literatura ninguna estrategia que trate de combinar diferentes técnicas de computación avanzada a múltiples niveles. Para ello, la Computación de Altas Prestaciones permitirá acelerar la ejecución de una simulación en una máquina paralela mientras que la Computación en Grid permitirá la ejecución concurrente de múltiples simulaciones paralelas sobre una infraestructura distribuida.

Por lo tanto, resulta necesario adoptar estrategias integrales que combinen diferentes técnicas computacionales para la simulación de la actividad eléctrica cardiaca. De esta manera, se conseguirán herramientas que permitan acelerar los procesos de investigación, permitiendo al mismo tiempo aumentar la productividad investigadora de los expertos biomédicos. Esto redundará en un mayor conocimiento de los fenómenos subyacentes a las patologías cardiacas más comunes, permitiendo la definición de tratamientos más específicos que los actualmente existentes.

Capítulo 3

El Diseño de Proteínas de Propósito Específico

> *"Protein engineering is not only a promising path, but it provides a very compact and (I think) persuasive argument for being able to build molecular machines, because proteins are molecular machines"*. K. Eric Drexler, Researcher.

En este capítulo se introduce el diseño de proteínas de propósito específico basado en computador, un nuevo campo de la biología estructural que requiere métodos avanzados de optimización combinatoria. Para ello, se describen los principales conceptos relacionados con el plegamiento de proteínas y se aborda la historia de este campo hasta llegar a las técnicas actuales basadas en optimización combinatoria. Para ello, se revisa el estado del arte haciendo énfasis en la colaboración mantenida con el Laboratorio de Bioquímica de L'École Polytechnique y la aproximación llevada a cabo por este grupo en el campo del diseño de proteínas. A continuación, se describen las actuales limitaciones de las implementaciones existentes analizando la idoneidad de las técnicas de computación avanzada para mejorar el proceso de diseño de proteínas de propósito específico.

3.1. Introducción

Estructuralmente, una proteína es una macromolécula formada por un conjunto de aminoácidos unidos mediante enlaces peptídicos. Los aminoácidos son moléculas que están formadas por una serie de átomos cuya disposición en el espacio puede variar. La estructura espacial tridimensional de un aminoácido se conoce con el nombre de rotámero.

La secuencia de aminoácidos de la proteína se conoce como estructura primaria y ésta predispone a la proteína a plegarse en una forma predeterminada. Por lo tanto, el plegamiento de proteínas [33]

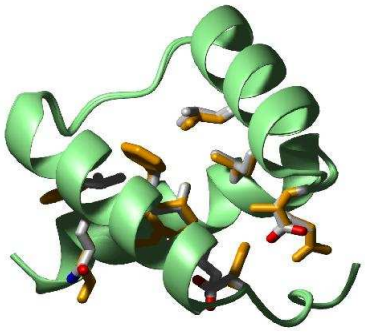


Figura 3-1: Resultado del diseño computacional del plegado de una proteína. Cortesía de Jaramillo y Wodak.

es el proceso que permite a una proteína alcanzar una estructura tridimensional estable, conocida como estado nativo. Por ejemplo, la Figura 3-1 muestra un ejemplo de proteína plegada diseñada por computador [34]. Por lo general, las proteínas únicamente pueden desarrollar su función biológica cuando se han plegado completamente ya que la forma tridimensional de las proteínas, en su estado nativo, tiene una relación directa con su comportamiento. De ahí la importancia de conocer los procesos involucrados en el plegamiento de proteínas, dado que resulta imposible desarrollar un propósito biológico si la proteína no alcanza el estado nativo.

En los últimos años se ha producido un importante avance en el uso de procedimientos automáticos, basados en ordenadores, para el diseño de proteínas. Esto es debido, principalmente, tanto a un mejor conocimiento y modelización de las interacciones físicas de los átomos que conforman una proteína, como a la reformulación del proceso como un problema de optimización combinatoria.

De hecho, en las últimas décadas se ha observado un progreso considerable en la habilidad para desarrollar proteínas con cierta estabilidad y propiedades funcionales [35, 36]. Una forma interesante de abordar este problema consiste en partir de una estructura 3D de una proteína conocida (estado nativo) y buscar la secuencia de aminoácidos compatible con ella (estructura primaria). Este procedimiento se conoce como el problema de plegamiento inverso [37] y supuso el comienzo de un interesante campo de investigación. Inicialmente, se trató de obtener el diseño de proteínas siguiendo reglas sencillas observadas a partir de otras estructuras de proteínas ya conocidas. Adicionalmente se utilizaron métodos de fuerza bruta basados en mutagénesis y selección [38].

Posteriormente, se desarrollaron diversas técnicas computacionales, combinadas con estudios experimentales, que trataban de identificar las secuencias compatibles con una estructura 3D [39]. Estos mecanismos basados en reglas resultaron ser parcialmente satisfactorios ya que las proteínas resultantes raramente mostraban la propiedad de desplegamiento cooperativo, indicando que las secuencias diseñadas no eran proteínas reales. Recientemente se han obtenido mejores resultados empleando procedimientos computacionales más sofisticados para la selección de secuencias, apli-

cándose al diseño de pequeñas proteínas. El grupo del Prof. Mayo (Caltech, USA) fue el primero en conseguir de manera satisfactoria el diseño de la secuencia de una proteína artificial que pudiera plegarse de una determinada manera [40]. Partiendo de la estructura en alta resolución de una proteína se utilizó un método combinatorio para mutar simultáneamente todos los residuos de cualquier manera posible. Un residuo se corresponde con cada una de las posiciones de una proteína, donde cada posición puede albergar un rotámero de un determinado conjunto de rotámeros posibles.

Recientemente, el grupo del Prof. Baker (Univ. of Washington, USA) utilizó una metodología más empírica para diseñar una proteína que pudiera plegarse en una estructura conocida [41], lo que abrió las puertas a la nanotecnología basada en proteínas. Posteriormente, el grupo del Prof. Hellinga [42] realizó un gran avance: Emplearon sus métodos computacionales para introducir actividad catalítica en una proteína, mediante la mutación simultanea de 20 residuos. En este sentido, es posible utilizar mutagénesis dirigida a sitios, una técnica de biología molecular que permite introducir una mutación en un determinado sitio de una molécula. Por ejemplo, en [43] se consiguió mutar una isomerasa en una enzima proteolítica. Sin embargo, este método no puede generalizarse debido a la pérdida de estabilidad producida por las mutaciones.

Alternativamente, la evolución dirigida permite la exploración de muchas mutaciones simultaneas para la estabilidad de la proteína. Esta técnica, utilizada en ingeniería de proteínas, permite evolucionar proteínas con determinadas propiedades deseables. En este caso, el número de secuencias exploradas, es decir, el tamaño de la librería combinatoria, es mucho más pequeño y puede ser conseguido mediante las técnicas computacionales de optimización combinatoria más recientes. Como desventaja, la aproximación computacional puede simular situaciones antinaturales. Sin embargo, una metodología computacional permite mejorar el conocimiento que tenemos sobre la estructura y funcionalidad de las proteínas.

En realidad, los métodos computacionales y la evolución dirigida son altamente complementarios: La combinación de métodos de búsqueda en el espacio de secuencias del primero, gracias a la modelización, y la optimización final en el sistema real del último proporcionará probablemente los mejores resultados.

Una mayor investigación en este nuevo campo proporcionará respuestas a preguntas fundamentales relativas a la génesis y evolución del plegado de proteínas y funciones enzimáticas. Esto supondrá un importante impacto en áreas tan diversas como:

- Medicina. El tratamiento de enfermedades neurodegenerativas [44], cáncer (a través del diseño de anticuerpos terapéuticos que se enlacen a determinantes antigénicos asociados a tumores que, al mismo tiempo, mantengan una inmunogenicidad relativamente pequeña) y enfermedades autoinmunes, así como el desarrollo de vacunas basadas en péptidos.

- Biotecnología. Desarrollo de biosensores [45] y biocatalizadores (mediante el diseño de enzi-

mas activos en entornos no naturales), ciencias del medio ambiente (mediante el diseño de enzimas que reduzcan la contaminación y toxicidad provocadas por residuos), circuitos genéticos/metabólicos diseñados o dispositivos basados en proteínas.

Las estrategias computacionales necesarias para la selección de secuencias compatibles con una estructura dada, requiere técnicas eficientes para el escaneo de largas secuencias de números así como la puntuación de su calidad en base a la estructura objetivo. El problema requiere una elevada complejidad computacional ya que, incluso para proteínas pequeñas, el número de posibles secuencias es muy elevado. Entre todas estas secuencias se deben seleccionar aquellas que adoptará la estructura.

En este sentido, el objetivo que se pretende alcanzar en esta parte de la tesis consiste en la mejora del proceso de diseño de proteínas llevado a cabo por el grupo del Prof. Jaramillo (École Polytechnique, Francia), mediante la utilización de técnicas de Computación de Altas Prestaciones y Computación en Grid. Esto permitirá acelerar el proceso de optimización e incluso abordar la optimización de proteínas de mayor dimensión.

3.2. Protocolo de Optimización de Proteínas

La estrategia utilizada por el grupo del Prof. Jaramillo permite una aproximación sistemática al problema de introducir capacidad de enlace a una proteína base (scaffold) inerte, con el objetivo de diseñar nuevas proteínas con una función específica. Para ello, se utiliza un modelo atómico detallado basado en los principios físico-químicos, y un procedimiento computacional automático, para explorar el espacio de búsqueda formado por las diferentes secuencias que conforman la proteína.

Como se ha visto anteriormente, el campo del diseño de enzimas ha experimentado una actividad frenética y, lo que es más importante, cabe esperar un aumento en su interés en los próximos años debido a la cantidad de información genética y estructural que va apareciendo y por el alto potencial científico e industrial de las enzimas.

La aproximación llevada a cabo por este grupo de investigación requiere la combinación de diferentes disciplinas: La física de la estructura de proteínas, para su modelización en el nivel atómico. Matemática algebraica, para la gestión de sitios activos y conformaciones de péptidos. Matemática aplicada a la optimización combinatoria, para adaptar los métodos actuales de Monte Carlo [46] y ramificación y poda para buscar los sitios activos y la modelización de conformaciones locales. Química y física, para la modelización de los mecanismos de la actividad enzimática y para la obtención de los parámetros energéticos para la descripción del enlace del ligando. Bioquímica general, para extraer los datos relevantes de los experimentalistas y proporcionar soluciones a algunos de sus problemas. Finalmente, es necesario utilizar conceptos de biología general y medicina para utilizar el trabajo de investigación publicado en los campos de la medicina y la biología que requieren la aproximación molecular utilizada.

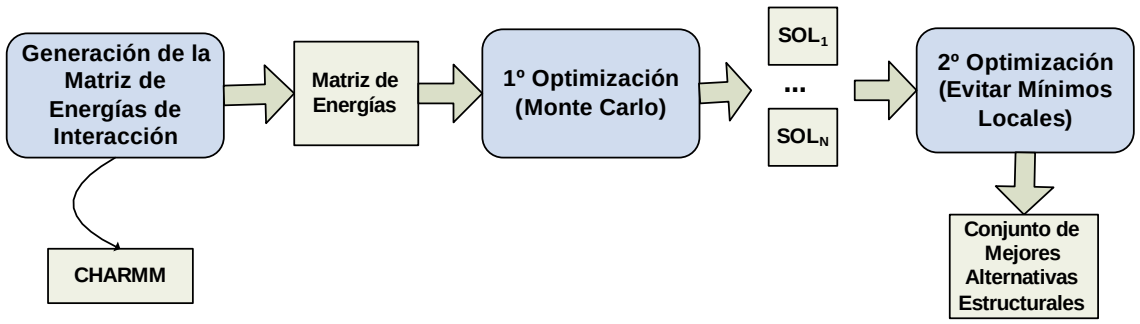


Figura 3-2: Esquema del proceso global de diseño de proteínas.

Para poder alcanzar estos objetivos, este grupo co-desarrolló el protocolo de diseño automático DESIGNER [47]. Esta herramienta utiliza la estructura atómica en alta resolución de una proteína, y técnicas de optimización combinatoria, para encontrar las posibles secuencias de aminoácidos que satisfacen uno o más objetivos, por ejemplo la estabilización del correspondiente plegado. La puntuación obtenida para valorar la bondad de las secuencias mutadas es una aproximación a la energía de plegamiento (*folding free energy*), que se computa modelando en el nivel atómico el estado de la proteína tanto sin plegar como plegada, es decir, el estado en el que la proteína tiene una forma estable y actividad bioquímica.

El sistema de optimización actual se encarga de diseñar las proteínas mediante el ensamblaje de los aminoácidos que la constituyen. La Figura 3-2 muestra un esquema del proceso global de diseño de proteínas. Para poder ensamblar de forma eficiente cualquier posible proteína, en primer lugar, se calculan todas las posibles interacciones energéticas entre rotámeros usando una aplicación que invoca a la librería CHARMM [48] para realizar las tareas de mecánica molecular (minimización y evaluación de energía). Este proceso genera una matriz de energías de dimensión igual al número de rotámeros y que representa la energía de interacción entre cada par de rotámeros. Dado que cada interacción entre dos rotámeros puede ser evaluada de manera independiente, esta tarea se realiza ya de manera eficiente mediante el uso de un cluster de PCs, en el que cada procesador genera parte de dicha matriz de energías. De esta manera, se obtiene un incremento de velocidad casi lineal con el número de procesadores, debido a la ausencia de comunicaciones entre ellos.

Posteriormente, el programa de optimización lee la matriz de energías y la utiliza para definir una función de rango adecuada para el proceso de optimización combinatoria, que se basa en métodos de Monte Carlo. Actualmente, el sistema de optimización es bi-objetivo, considerando tanto el plegamiento (folding) como el enlace (binding) de la proteína. Este método permite la obtención de la combinación de rotámeros que deben formar parte de la proteína para que ésta tenga una energía de plegamiento mínima y que permita que la proteína se enlace con otra molécula.

Una vez finalizado el primer proceso de optimización, dando lugar a un conjunto de secuencias de

rotámeros para la proteína a optimizar, se realiza un segundo proceso de optimización. Éste comienza desde cada una de las secuencias obtenidas anteriormente para tratar de explorar los mínimos locales que puedan haber en el subespacio de búsqueda considerado.

Por lo tanto, DESIGNER permite el diseño de proteínas funcionales mediante el acoplamiento del diseño automatizado de proteínas y el escaneo de sitios activos. Simultáneamente inspecciona todos los posibles sitios activos y optimiza la secuencia de aminoácidos del entorno para permitir el enlace. De hecho, no sólo optimiza la estabilidad de la proteína diseñada sino también su funcionalidad, es decir, el enlace con el ligando o actividad catalítica.

Esta herramienta define el ligando como un rotámero generalizado y utiliza técnicas de minimización para generar un conjunto de posibles ligandos. Esta aproximación innovadora permite explorar únicamente el conjunto posible de conformaciones de ligandos. Además, en el proceso de diseño se controla la desestabilización de los ligandos competitivos utilizando un algoritmo de puntuación multi-objetivo para seleccionar simultáneamente dos puntuaciones competitivas. Esta ventaja no ha sido incluida de manera consistente anteriormente y representa un paso crucial para la obtención de la elevada actividad enzimática.

Recientemente, el software DESIGNER se utilizó para inyectar de manera automática el caso más simple de un sitio activo en una proteína inerte, involucrando únicamente un residuo catalítico. Usando tiorredoxina como proteína base se comprobó experimentalmente en el laboratorio del Prof. Sanchez-Ruiz (U. Granada, Spain) que se obtuvo una actividad similar con respecto a [49, 50].

Dado que la metodología empleada en DESIGNER está basada únicamente en parámetros estándar de mecánica molecular y en un método de resolución atómica empírico, este puede ser aplicado a casos donde otros métodos computacionales de diseño de proteínas tienen menos oportunidades. La aproximación generalizada de rotámeros explicada anteriormente es bastante probable que tenga garantía de éxito dado que la función de puntuación basada en energía libre no cambia. Por lo tanto, el objetivo es utilizar un procedimiento computacional automatizado que proporcione una estimación estructural para una nueva proteína funcional con una determinada actividad o enlace.

Una ventaja adicional de la metodología computacional de DESIGNER, con respecto a otras previamente empleadas, es que, en contraprestación al incremento del modelado físico, tiene mayor probabilidad de ser satisfactorio con un plegado de proteína arbitrario. Esta metodología original ya ha sido testeada para rediseñar siete plegados [47] y mediante el diseño de péptidos que se enlazan a proteínas MHC-I [51]. Para poder desarrollar una metodología para el desarrollo de proteínas funcionales es necesario utilizar cuanta más información estructural sea posible, combinada con nuevas técnicas que permitan buscar el mayor número de alternativas. A medio plazo, esta aproximación producirá mejores resultados que aquellas basadas únicamente en conocimientos parciales.

3.3. Objetivos de Mejora del Proceso de Diseño de Proteínas

La implementación actual del proceso de optimización de proteínas descrito presenta una importante limitación. Para realizar la optimización se debe leer toda la matriz de energías para tenerla disponible en memoria, ya que se utiliza para el cálculo de la energía de la proteína en base a las interacciones energéticas entre los rotámeros que la constituyen.

Esta implementación introduce una limitación al tamaño del problema abordable, cuya cota superior queda fijada por la cantidad de memoria RAM del sistema de computación utilizado. Por otra parte, cada vez resulta más interesante el diseño de proteínas de mayor tamaño, que presenten mayor funcionalidad. Sin embargo, conforme aumenta la dimensión de la proteína, así lo hace la matriz de energías. Por lo tanto, resulta preciso el desarrollo de herramientas computacionales avanzadas que superen dicha limitación para poder abordar problemas de mayor dimensión.

En este sentido, en esta parte de la tesis se plantea la utilización de técnicas de Computación de Altas Prestaciones. Para ello, en primer lugar se realizará una implementación secuencial de las estructuras de datos que almacenan la matriz de energía para permitir un almacenamiento y un acceso eficiente a dicha información. De esta manera se aprovechará el uso de la jerarquía de memorias para acelerar el acceso a la información de la interacción energética entre rotámeros. En segundo lugar, se aplicarán técnicas de computación paralela para permitir una distribución de la matriz de energías entre múltiples procesadores de un cluster de PCs. De esta manera, es posible que múltiples procesadores trabajen de forma colaborativa en la optimización de una proteína, superando así la limitación en el consumo de memoria.

Posteriormente, el uso de la computación en Grid permitirá gestionar de manera eficiente la ejecución de las múltiples sub-optimizaciones, necesarias para el proceso global de optimización, sobre un Grid computacional. En este sentido, la aplicación de una estrategia conjunta basada en Computación de Altas Prestaciones y Computación en Grid permitirá diseñar proteínas más grandes y con mayor funcionalidad que las que se han podido diseñar hasta el momento.

3.4. Estado del Arte

En la actualidad, existe mucha actividad relacionada con la aplicación de técnicas computacionales avanzadas al proceso de diseño de proteínas.

En [52] se realizan simulaciones de dinámica molecular [53] para computar el enlace (docking) entre proteínas y ligandos. Para ello se plantea un algoritmo de enlace, comparando las prestaciones obtenidas frente a métodos tradicionales como, entre otros, AutoDock [54]. Los autores argumentan que es necesario el desarrollo de algoritmos que sean flexibles y adaptables para su utilización en Grid. En su caso, el hecho de intentar diferentes aproximaciones de enlace requiere la ejecución de intentos independientes. Se concluye que la utilización de múltiples PCs permitiría reducir el tiempo

en alcanzar la solución y que, en este sentido, el algoritmo es fácilmente adaptable para la utilización de una infraestructura Grid.

En [55] se plantea la utilización de métodos de Templado Simulado (Simulated Annealing) [56] para la optimización de proteínas, con el objetivo de solventar la problemática de los mínimos locales en la búsqueda de proteínas con mínima energía. Además, se utiliza una aproximación basada en Computación en Grid para realizar múltiples procesos de optimización, utilizando diferentes parámetros del algoritmo de optimización. Para ello se utiliza MyGrid [57], una plataforma basada en tecnologías punto-a-punto (peer-to-peer) para la ejecución remota de tareas.

En [58] se utiliza una aproximación que combina computación paralela y computación en Grid. Para ello se utiliza el método de intercambio de réplicas [59], donde múltiples simulaciones de la proteína se ejecutan en paralelo, cada una de ellas bajo una condición de temperatura diferente. Periódicamente, las simulaciones que se ejecutan con temperaturas vecinas son evaluadas e intercambiadas. Esta aproximación introduce un acoplamiento entre las diferentes simulaciones, ya que el retardo en el cómputo de una temperatura puede afectar al resto. Como herramienta computacional se utiliza un cluster Condor formado por unos 450 PCs de sobremesa con Windows NT 5.1.

En [60] se utiliza una aproximación multiparamétrica para el estudio de enlaces entre proteínas y ligandos, utilizando computación en Grid. De acuerdo a los autores, la idea es escanear las posibles conformaciones de un ligando y una proteína, así como encontrar sus posiciones óptimas con respecto a ambas. Para ello se modela su tendencia a enlazarse por medio de una función de coste basada en energía libre. Para la ejecución se utiliza la herramienta Nimrod [61] para escanear las posibles alternativas y distribuir los trabajos en un Grid de máquinas.

Con respecto a proyectos relacionados con este área, es posible encontrar el Human Proteome Folding Project [62], una iniciativa desarrollada por la Universidad de Nueva York en colaboración con IBM. En este proyecto se plantea utilizar la potencia computacional de millones de ordenadores para predecir la forma de las proteínas humanas. A partir de esta forma detallada, se espera aprender sobre las funciones de esas proteínas, dado que su forma está inherentemente relacionada con su funcionamiento en el cuerpo humano. La base de datos resultante de estructuras de proteínas permitirá a los científicos comprender cómo funcionan las enfermedades que involucran esas proteínas.

En este sentido, el proyecto Swiss Bio Grid [63] es una iniciativa Grid a nivel nacional que permitirá soportar aplicaciones computacionalmente intensivas en el campo de la bioinformática, biosimulación y ciencias biomédicas, mediante la utilización de computación distribuida de altas prestaciones, redes de alta velocidad, colecciones de datos masivas así como las herramientas necesarias. Una de las líneas de investigación involucra la proteómica, para identificar proteínas mediante simulación molecular.

Por otra parte, el proyecto GPS@ (Grid Protein Sequence Analysis) es un portal Grid integrado dedicado a la informática biomolecular. Este proyecto es la evolución del proyecto NPS@ (Network

Protein Sequence Analysis), un servidor web interactivo dedicado al análisis de secuencias de proteínas, disponible para la comunidad de biólogos. La versión actual se encuentra en desarrollo y está desplegada sobre los recursos de LCG-2 [64].

3.5. Conclusiones

El campo de la biología estructural ha experimentado una revolución en los últimos años debido, principalmente, a la combinación de técnicas y estrategias computacionales eficientes que han permitido obtener resultados en áreas tan diversas como la medicina o la biotecnología. En particular, el diseño de proteínas requiere técnicas avanzadas de optimización combinatoria para evaluar las diferentes conformaciones de una proteína, con el objetivo de que realice una función específica.

Aunque son muchos los grupos de investigación trabajando en este campo, no existen muchas aproximaciones que combinen diferentes estrategias computacionales al problema del diseño de proteínas. Para ello se plantea realizar un proceso de optimización a múltiples niveles donde, en primer lugar, se realice una gestión eficiente de la memoria, con el objetivo de acelerar los cálculos y reducir el consumo de memoria, al tiempo que permita abordar problemas de mayor dimensión.

A continuación se plantea la utilización de Computación Paralela para la optimización colaborativa de proteínas, donde múltiples procesadores cooperen de manera coordinada en este proceso. Esto permitirá que diferentes elementos de proceso optimicen partes de una proteína e intercambien información para optimizar el problema global.

Posteriormente, la utilización de una estrategia combinada de Computación de Altas Prestaciones y Computación en Grid, basada en middlewares Grid estándar, permitirá realizar múltiples procesos de optimización de una proteína. Esto permitirá realizar mayor número de ejecuciones por unidad de tiempo, con el objetivo de incrementar la bondad de la solución.

La realización de un proceso de optimización de la aplicación a múltiples niveles permitirá conseguir un salto cualitativo en las estrategias computacionales empleadas actualmente para el diseño de proteínas.

Capítulo 4

La Computación de Altas Prestaciones

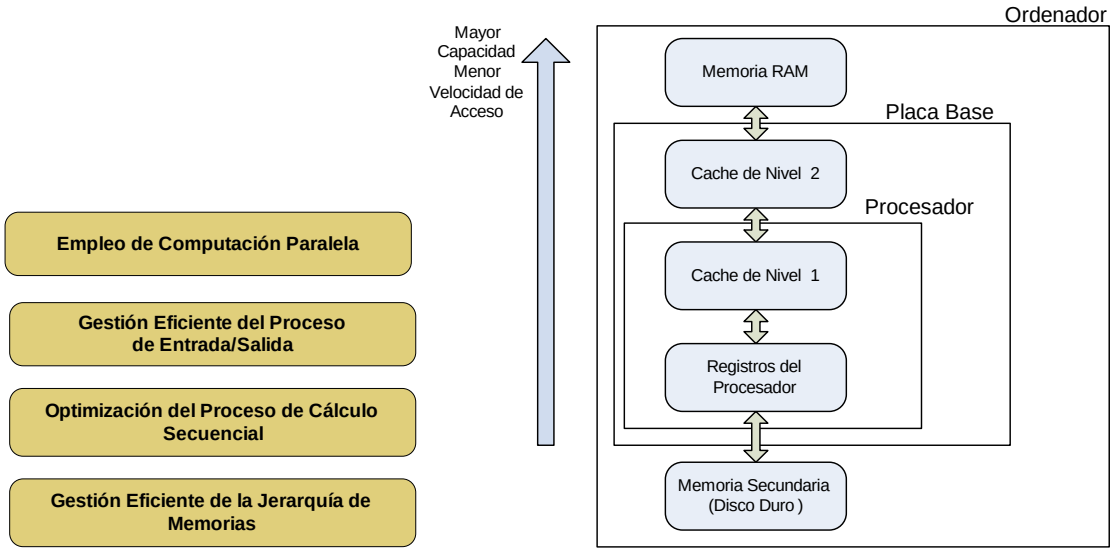
"If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?". Seymoure Cray, Creator of the first super computer Cray I.

En este capítulo se realiza una breve introducción a la Computación de Altas Prestaciones, describiendo los principales paradigmas y objetivos de la programación paralela. Además, se destaca la optimización a múltiples niveles como mecanismo para obtener las máximas prestaciones de un sistema de computación. Posteriormente, se detallan las librerías de cálculo científico que han sido empleadas en el desarrollo de los sistemas descritos en esta tesis.

4.1. Introducción

En general, la historia de la computación siempre ha venido marcada por la búsqueda de las máximas prestaciones. Hasta hace unos años era bastante aceptado que, para conseguir altas prestaciones, era necesario realizar una fuerte inversión económica en una máquina de propósito dedicado. Desde que los computadores secuenciales aparecieron en el mercado, su potencia ha ido aumentando con el paso de tiempo siguiendo la ley de Moore, que predice que el número de transistores de los circuitos integrados se dobla cada 18 meses aproximadamente. Esto implica que cada vez existen máquinas más potentes, con mayor capacidad de proceso.

Sin embargo, las tendencias actuales apuntan a que las prestaciones de las computadores están comenzando a saturarse [65]. Por lo tanto, una manera sencilla de solventar este problema consiste en la utilización de la potencia computacional de un conjunto de procesadores. En ese sentido, la Computación de Altas Prestaciones (CAP) permite el desarrollo de aplicaciones que aprovechan el



(a) Computación de Altas Prestaciones. (b) Jerarquía de Memoria.

Figura 4-1: Visión de la Computación de Altas Prestaciones como utilización eficiente de un sistema de computación.

uso colaborativo de múltiples procesadores con el objetivo de resolver un problema común [65, 66]. A menudo se suele hablar indistintamente de CAP y computación paralela. Sin embargo, el primero es un término más general que abarca la obtención de las máximas prestaciones, en múltiples niveles, de un sistema de computación.

Como se puede ver en la Figura 4-1.a, la CAP principalmente se apoya sobre una gestión eficiente de la jerarquía de memorias. La existencia de diferentes niveles de memoria (ver Figura 4-1.b) implica que el tiempo empleado en realizar una operación de acceso a la memoria de un nivel aumenta con la distancia al nivel más rápido, que es el acceso a los registros del procesador. En efecto, una mala gestión de la memoria provoca numerosos fallos de página, que abusan del sistema de memoria virtual, realizando costosos accesos al disco. Por lo tanto, el diseño de aplicaciones eficientes requiere aprovechar el conocimiento de la estructuración de la memoria.

Siguiendo con la Figura 4-1, la optimización del proceso de cálculo secuencial implica, entre otros, una adecuada selección de las estructuras de datos a emplear para representar la información relativa al problema, atendiendo a criterios de eficiencia computacional. Por ejemplo, el uso de técnicas de almacenamiento de matrices dispersas cae dentro de esta categoría, permitiendo almacenar únicamente los elementos no nulos de una matriz. Además, este nivel incluye la selección de métodos eficientes para la resolución del problema.

Un fase importante de las aplicaciones, especialmente para aquellas orientadas a procesos de simulación, corresponde a la grabación de datos en disco. Dado que el acceso a un disco duro presenta tiempos de acceso muy superiores a los correspondientes a memoria RAM, resulta indispensable realizar una gestión eficiente del proceso de Entrada/Salida (E/S) para que tenga el mínimo impacto

sobre el proceso de simulación.

Finalmente, y por encima del resto de capas, aparece la utilización de computación paralela. En este sentido, una buena estrategia de partición de tareas que garantice una distribución equitativa de la carga entre los procesadores involucrados, así como la minimización del número de comunicaciones entre los mismos, permite reducir los tiempos de ejecución. Además, con la partición y distribución de las principales estructuras de datos de la aplicación, entre los diferentes procesadores, se puede conseguir la resolución de problemas de mayor dimensión.

Como se puede observar, la CAP engloba la utilización de computación paralela además de otro tipo de técnicas que permiten realizar una implementación eficiente de las aplicaciones. De esta manera es posible conseguir buenas prestaciones globales.

4.2. Computación Paralela

Como se ha comentado, la computación paralela permite el desarrollo de aplicaciones que aprovechan el uso de múltiples procesadores de manera colaborativa con el objetivo de resolver un problema común [66]. De hecho, el objetivo fundamental que persiguen éstas técnicas es conseguir reducir el tiempo de ejecución de una aplicación mediante el empleo de múltiples procesadores. Adicionalmente, también es posible desear resolver problemas de mayor dimensión mediante el aprovechamiento de las diferentes memorias de los procesadores involucrados en la ejecución. Básicamente, la computación paralela se basa en dos conceptos fundamentales:

1. Particionamiento de Tareas: Consiste en dividir una tarea grande en un número de diferentes subtareas que puedan ser completadas por diferentes unidades de proceso (procesadores).

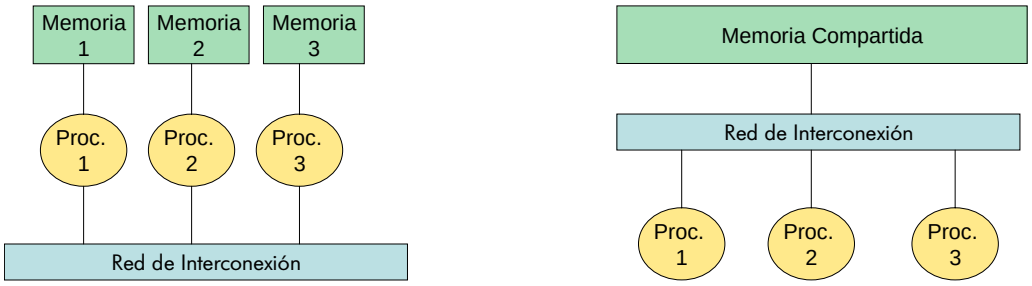
2. Comunicación entre Tareas: Aunque cada procesador realice una subtarea, generalmente será preciso que éstos se comuniquen para poder cooperar en la obtención de la solución global a un problema determinado.

4.2.1. Paradigmas de Programación Paralela

En la actualidad, existen dos grandes paradigmas de programación dedicados a la construcción de algoritmos paralelos: programación en *memoria distribuida* y programación *en memoria compartida*.

Memoria Distribuida

La Figura 4-2.a muestra un esquema general del paradigma de programación paralela de memoria distribuida. Bajo este esquema, cada elemento de proceso dispone de una memoria local que únicamente puede ser accedida por él. La comunicación entre procesadores se realiza por medio de la red de interconexión, mediante el paso de mensajes entre los diferentes elementos de proceso.



(a) Memoria Distribuida. (b) Memoria Compartida.

Figura 4-2: Paradigmas de computación paralela.

En este esquema, los datos del usuario se encuentran típicamente repartidos entre las diferentes memorias locales a cada elemento de proceso. Esta repartición no es transparente al usuario y, por lo tanto, la aplicación es responsable de la distribución de los datos y la comunicación de los mismos intercambiando mensajes entre los elementos de proceso.

Memoria Compartida

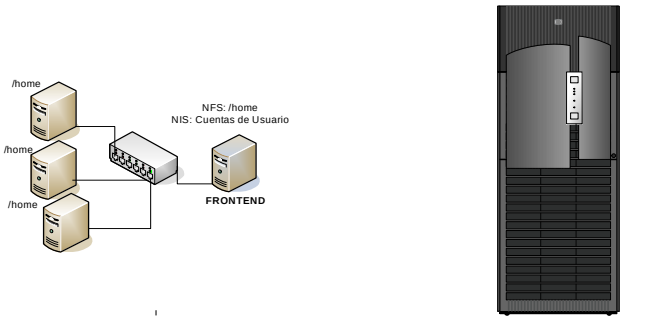
La Figura 4-2.b muestra un esquema general del paradigma de memoria compartida, donde todos los procesadores comparten una misma memoria global que es accedida por medio de la red de interconexión. En este caso, la comunicación entre procesos se hace mediante áreas de memoria compartidas. Es responsabilidad del programador la gestión de las estructuras de datos para realizar la compartición de datos entre los diferentes procesos.

4.2.2. Principales Arquitecturas Paralelas

Los paradigmas de programación paralela son soportados por las implementaciones hardware existentes. En la actualidad, los dos principales sistemas dedicados a computación paralela se pueden clasificar en máquinas multiprocesadores de memoria compartida y clusters de PCs.

Multiprocesadores de Memoria Compartida

Los multiprocesadores de memoria compartida tienen una relación directa con el paradigma de programación de memoria compartida y, generalmente, constan de un conjunto de procesadores con una pequeña memoria local (*cache*) que acceden a una gran memoria global compartida entre todos los procesadores. Un ejemplo de este tipo de arquitectura es el SGI Altix 3000, disponible en las instalaciones de la Universidad Politécnica de Valencia y que consta de 48 procesadores Intel Itanium II. Esta máquina es un sistema de memoria compartida, aunque físicamente la memoria está distribuida en diferentes bloques. Esto provoca que el acceso a la memoria global compartida



(a) Conjunto de PCs. (b) En rack.

Figura 4-3: Diferentes visiones de un cluster de PCs.

dependa del procesador empleado y la zona de memoria que sea accedida. A este tipo de sistemas se les denomina NUMA (Non Uniform Memory Access).

El principal problema de las máquinas de memoria compartida suele ser su elevado coste y su limitada escalabilidad.

Clusters de PCs

Como alternativa a las máquinas de memoria compartida, es posible construir unidades de computación de bajo coste que ofrecen unas buenas prestaciones, como los clusters de PCs.

Un cluster, aunque generalmente viene presentado en un *rack* (Figura 4-3.b), no es más que un conjunto de PCs interconectados por medio de una red de alta velocidad, tal y como muestra la Figura 4-3.a. Uno de los nodos, denominado *frontend*, suele realizar ciertas labores de gestión y representa el punto de entrada al cluster. Generalmente, los clusters de PCs cumplen una serie de características:

1. Todos los PCs comparten un sistema de archivos común. De esta manera, cualquier proceso ejecutándose en cualquier nodo tiene la misma visión del sistema de ficheros. Generalmente, el nodo principal del cluster suele exportar el directorio de usuarios (/home) para que los nodos internos de cómputo puedan tener acceso al mismo.

2. Las cuentas de usuario están compartidas entre todas las máquinas. Generalmente se emplean herramientas como NIS (Network Information Service) para poder exportar las cuentas de usuario, creadas en el nodo principal, al resto de nodos del cluster. De esta manera se simplifica la administración en la gestión de usuarios.

3. Implementan soporte para la ejecución de trabajos paralelos, desde librerías de paso de mensajes entre procesos como MPI (Message Passing Interface) [67], a gestores de trabajos para la compartición de recursos como PBS (Portable Batch System) o Torque.

Los clusters de PCs se han convertido en unas máquinas que ofrecen un alto rendimiento y un excelente ratio precio/prestaciones. Por lo tanto, son considerados como una buena opción para la paralelización de códigos mediante el paradigma de memoria distribuida.

Es importante destacar que, en la actualidad, el uso de un paradigma de programación paralela no está vinculado directamente a la arquitectura de la máquina subyacente. De hecho, es totalmente posible la utilización de una librería de paso de mensajes sobre una máquina de memoria compartida, así como la utilización de un esquema de programación de memoria compartida que simule el acceso a las memorias locales de varias máquinas como si se tratara de una memoria global.

4.2.3. Librerías de Soporte a la Computación Paralela

Para dar soporte a la programación paralela existen diferentes librerías y herramientas de ayuda. Teniendo en cuenta los estándares de facto actuales, MPI se utiliza para la programación con memoria distribuida, mientras que OpenMP suele ser la opción habitual para la programación en máquinas de memoria compartida. A continuación se describen brevemente ambas aproximaciones.

El estándar MPI

Los comienzos de la computación paralela se caracterizaron por una marcada tendencia hacia la experimentación, tratando de reimplementar las aplicaciones para cada nueva computadora que aparecía en el mercado. De hecho, todo el soporte de comunicaciones sobre el que se construían las aplicaciones debía ser implementado nuevamente. Ante esta situación, existía claramente la necesidad de definir una infraestructura básica de comunicaciones que ofreciera una interfaz común a las aplicaciones, independiente de la máquina.

Durante los años 1993 y 1994, un grupo de representantes de la industria de computadores, laboratorios gubernamentales y gente del entorno universitario se reunieron para desarrollar una interfaz estándar para el paradigma de programación de memoria distribuida, mediante paso de mensajes. Esta organización, conocida como *MPI Forum* finalizó su trabajo en 1994, produciendo el estándar conocido como MPI [67]. Éste define la interfaz de las rutinas empleadas para la comunicación mediante paso de mensajes entre los procesadores de una ejecución paralela. Las operaciones de comunicación soportadas se clasifican en:

1. Comunicaciones punto a punto. Permiten el envío de un mensaje de un proceso a otro.

2. Comunicaciones colectivas. Transfieren datos entre todos los procesos pertenecientes a un grupo. A su vez, pueden ser catalogadas en:

   a) Difusión. Un proceso envía los mismos datos al resto de procesos.

   b) Difusión personalizada. Un proceso distribuye una serie de datos entre el resto de procesos.

c) Multidifusión. Todos los procesos envían un dato al resto de procesos.

d) Multidifusión personalizada. Todos los procesos distribuyen una serie de datos entre el resto de procesos.

El estándar MPI es únicamente una especificación. Por lo tanto, es necesario alguna implementación del estándar para utilizar su funcionalidad. Una de las implementaciones con mayor repercusión es MPICH [68, 69], desarrollado en la División de Matemáticas e Informática del Laboratorio Nacional Argonne, en los EE.UU.

Además, existen implementaciones de MPI específicas para la optimización de las comunicaciones bajo determinadas redes de intercomunicación. Por ejemplo, *Scali MPI Connect* es una implementación de MPI, realizada por la empresa Scali [70], optimizada para el uso de la red SCI (Scalable Coherent Interface), una red de muy baja latencia empleada en clusters de PCs.

Por lo tanto, el uso del estándar MPI, como mecanismo de comunicación entre procesos, facilita la migración de una aplicación a otras plataformas para las que exista una implementación de MPI. En este sentido, la utilización de estándares facilita las labores al programador de aplicaciones paralelas.

La librería OpenMP

Para desarrollar programas paralelos, utilizando el paradigma de memoria compartida, una de las alternativas más utilizadas es el estándar OpenMP [71]. La especificación de OpenMP soporta programación paralela en memoria compartida en múltiples arquitecturas. OpenMP es una especificación de directivas de compilador, librería de rutinas y variables de entorno que pueden ser utilizadas para especificar paralelismo de memoria compartida en programas escritos tanto en C/C++ como en Fortran.

Inicialmente, las directivas de programación empleadas para el desarrollo de aplicaciones paralelas en memoria compartida no estaban estandarizadas. De hecho, cada fabricante mantenía su propia especificación de directivas con ligeros cambios tanto en sintaxis como en semántica. Aunque en 1994 se terminó un borrador de estándar conocido como ANSI X3H5, que trataba de solucionar el problema, nunca fue adoptado de manera formal. Esto fue debido principalmente a la popularidad de las máquinas de memoria distribuida y que el estándar MPI atraía buena parte de la atención de la comunidad científica dedicada a la computación en paralelo.

La primera especificación de OpenMP surgió en Octubre de 1997 para el lenguaje Fortran. En ella se consolidaron el conjunto de directivas existente hasta la fecha para unificar criterios, tanto sintácticos como semánticos. Básicamente, OpenMP define una serie de directivas que se especifican en el código fuente de la aplicación del usuario. Éstas son posteriormente transformadas en código por parte de un compilador que admita la especificación OpenMP como, por ejemplo, los compiladores de Intel. De esta manera, OpenMP se constituye como un estándar en las comunicaciones entre

procesos bajo el paradigma de memoria compartida. En este sentido, y al igual que ocurre con MPI, una aplicación que utilice las directivas de OpenMP será fácilmente portada a otra máquina que disponga de un compilador con soporte OpenMP.

Obviamente, el uso de estándares facilita la adopción de herramientas por parte de la comunidad científica y, en este sentido, tanto MPI como OpenMP suponen un importante avance en la consecución de herramientas eficientes y genéricas de aplicación de computación paralela a problemas científicos.

4.3. Librerías de Cálculo Científico

En los comienzos de la informática, el uso de un ordenador para la resolución de problemas en ingeniería precisaba del diseño e implementación de todas y cada una de las fases necesarias para acometer satisfactoriamente el proceso. Esta situación implicaba la necesidad de invertir tiempo en la resolución de problemas comunes por parte de numerosos científicos. Por ejemplo, dos investigadores trabajando en problemas diferentes, que precisaran realizar multiplicaciones de matrices, debían realizar sus propias implementaciones del algoritmo de producto de matrices.

Es fácil observar que esta forma de trabajar provoca una serie de problemas. En primer lugar, el tiempo dedicado a la implementación de operaciones básicas de bajo nivel (por ejemplo la multiplicación de matrices) es tiempo no dedicado a la resolución del problema global. Esto implica una ralentización del proceso de desarrollo de código. En segundo lugar, realizar múltiples implementaciones de algoritmos conocidos fomenta la aparición de errores en el código, disminuyendo la posibilidad de reutilizar implementaciones ya existentes que hayan sido testeadas y verificadas previamente.

Debido a estas razones, la comunidad científica decide aunar esfuerzos para la consecución de librerías estándar de resolución de problemas comunes. A continuación se describen brevemente las características de algunas de las librerías que se han utilizado en el marco de esta tesis doctoral.

4.3.1. Los Núcleos Computacionales BLAS y LAPACK

Las librerías BLAS y LAPACK surgen con el objetivo de crear librerías estándar para el tratamiento y resolución de problemas algebraicos y numéricos comunes en ingeniería.

BLAS

La librería BLAS (Basic Linear Algebra Subprograms) [72, 73, 74], consta de un conjunto de implementaciones eficientes de operaciones básicas utilizadas normalmente con vectores y matrices. Además, esta librería pretende proporcionar una interfaz estándar para la descripción de operaciones de álgebra lineal. Por ello, un programa que utilice las funciones especificadas por BLAS puede

ser fácilmente portado a otra máquina que disponga de una implementación de esta librería. Las operaciones implementadas en BLAS se agrupan en 3 niveles diferentes:

- Nivel 1. Operaciones vector-vector. Incluye operaciones como el escalado de vectores, copia de vectores y suma de vectores.
- Nivel 2. Operaciones matriz-vector. Incluye operaciones como el producto de una matriz por un vector, donde la matriz puede estar afectada por alguna operación adicional (transpuesta o transpuesta conjugada).
- Nivel 3. Operaciones matriz-matriz. Incluye operaciones como el producto de dos matrices.

Aunque la primera versión de BLAS fue desarrollada en 1979, es a partir de los años 80 cuando fue adoptado por la comunidad científica como un estándar *de facto*. Posteriormente, en 1988 aparece el nivel 2 de BLAS, para dar solución a los problemas que no eran abordables empleando operaciones de nivel 1. En 1990 aparece BLAS de nivel 3, para realizar operaciones entre matrices y supone la culminación de un paquete dedicado a la realización eficiente de operaciones entre vectores y matrices. En la actualidad existen implementaciones de BLAS para diferentes lenguajes de programación como C, C++, Fortran y Java.

LAPACK

La librería LAPACK [75] proporciona una colección de subrutinas desarrolladas en Fortran 77 a finales de los años 80 (1987) para resolver problemas de álgebra lineal numérica. La librería implementa métodos de resolución de sistemas de ecuaciones lineales, solución mediante mínimos cuadrados de sistemas de ecuaciones lineales, problemas de valores propios y problemas de valores singulares. Esta librería se encarga de proporcionar la implementación de las factorizaciones matriciales necesarias para resolver esos problemas (LU, Cholesky, QR, SVD y Schur).

Las rutinas de LAPACK generalmente realizan llamadas a rutinas de BLAS para aprovechar su eficiencia y robustez. Las rutinas ofrecidas por LAPACK, disponibles tanto en simple como en doble precisión para números reales y complejos, se clasifican en dos tipos diferentes:

- Rutinas completas (*driver routine*). Son las encargadas de resolver un problema de manera completa, como por ejemplo la resolución de un sistema de ecuaciones lineales.
- Rutinas auxiliares (*auxiliary routine*). Son rutinas encargadas de resolver parte de un problema como, por ejemplo, una subtarea requerida durante un algoritmo estructurado a bloques.

Esta librería trata de explotar la jerarquía de memoria de los procesadores mediante la división de los datos de trabajo, generalmente matrices, en bloques. En este sentido, estas rutinas pretenden trabajar la mayor parte del tiempo con los datos situados en los niveles de memoria más cercanos

al procesador. Esta manera de proceder responde al principio de *localidad espacial*, por el que generalmente se tienden a procesar datos cuyas direcciones de memoria están cercanas unas de otras, situación muy frecuente cuando se trabaja con vectores y matrices.

Implementaciones Alternativas

En la actualidad, existen diversas implementaciones de BLAS y LAPACK con el objetivo de aprovechar al máximo las prestaciones de la máquina en la que se ejecuta una aplicación. Por ejemplo, Intel dispone de implementaciones optimizadas de BLAS dentro de la Math Kernel Library (MKL) [76], específicas para algunas plataformas Intel como Itanium 2, Xeon y Pentium 4.

Además, existen alternativas que tratan de investigar de manera automática las capacidades de la máquina del usuario, para generar un conjunto de rutinas de BLAS y LAPACK que sean óptimas para la arquitectura subyacente. Este es el caso de ATLAS (Automatically Tuned Linear Algebra Software) [77], que proporciona compatibilidad completa con la especificación de rutinas de BLAS, así como un subconjunto pequeño de operaciones de LAPACK. Esta librería permite la optimización automática de software de cálculo numérico para arquitecturas de procesador que presentan jerarquía de memorias y unidades funcionales segmentadas, de manera que, probando diferentes valores de configuración para determinadas rutinas de cálculo numérico, se escoge la configuración adecuada para una arquitectura concreta.

Se puede observar que los núcleos computacionales de niveles más bajos son extremadamente útiles, por razones de eficiencia y reaprovechamiento de código. Además, en la actualidad, el campo de la computación científica está tendiendo al empleo de librerías generales para la resolución de problemas. De esta manera, es posible aprovechar el conocimiento de un grupo de expertos que ha dedicado su tiempo al desarrollo y verificación de librerías útiles y eficientes para la resolución de determinados tipos de problemas. Un ejemplo representativo de librería científica genérica, para la resolución de problemas de ingeniería, es la librería PETSc.

4.3.2. La librería PETSc

La librería PETSc [78, 79] (Portable, Extensible Toolkit for Scientific Computation) es un conjunto de estructuras de datos y rutinas que proporcionan bloques computacionales para la implementación tanto de códigos paralelos basados en MPI como secuenciales, para la resolución de Ecuaciones en Derivadas Parciales.

Esta librería incluye una completa colección de métodos paralelos iterativos de resolución de sistemas de ecuaciones lineales y no lineales, así como solvers de ecuaciones diferenciales que pueden ser empleados desde aplicaciones escritas en Fortran, C y C++. La librería está organizada jerárquicamente, permitiendo a los usuarios emplear el nivel de abstracción que sea más apropiado para un problema particular.

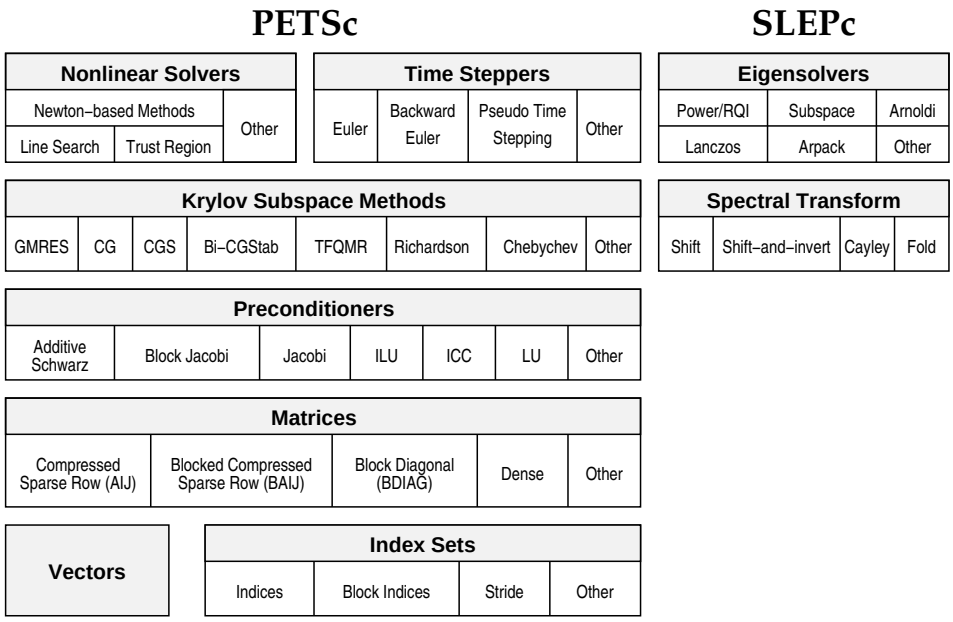


Figura 4-4: Diagrama de componentes y funcionalidad de la librería PETSc.

Esta librería está pensada para su uso en problemas de gran dimensión, y en la actualidad se está empleando en numerosos proyectos de diferentes campos de la ingeniería, como puede ser la nano-simulación, biología, problemas de fluidos, problemas de optimización y diseño de algoritmos.

La librería PETSc está desarrollada en C, siguiendo un esquema de programación mediante una estructura orientada a objetos. De esta manera se consigue facilitar el uso de la librería, donde el usuario encuentra objetos para representar vectores, matrices, etc. y operaciones de alto nivel que trabajan sobre esos objetos.

PETSc está principalmente destinada a ser utilizada con matrices dispersas y, por lo tanto, implementa esquemas de almacenamiento eficientes en los que únicamente se almacenan en memoria aquellos elementos no nulos de la matriz. Por motivos de eficiencia, el usuario suele proporcionar una estimación del tamaño que va a ocupar una matriz y posteriormente se encarga de insertar en ella aquellos elementos no nulos.

La Figura 4-4 muestra el diagrama de componentes y funcionalidad de esta librería. Uno de los aspectos más importantes que ofrece la librería PETSc es la utilización de métodos iterativos de resolución de sistemas de ecuaciones lineales basados en subespacios de Krylov [80]. La gran variedad de métodos iterativos, ofrecidos por esta librería, permite evaluar la eficiencia de diferentes métodos de resolución de sistemas de ecuaciones lineales desde un mismo entorno.

Quizás una de las características más importantes que presenta PETSc es la posibilidad de integrar librerías externas para aumentar su funcionalidad. En [81] se detalla el proceso llevado a cabo por el doctorando para la integración y evaluación de diferentes librerías externas de resolución

de sistemas de ecuaciones lineales, mediante métodos directos, así como la integración de una librería para la utilización de precondicionadores paralelos.

4.3.3. La librería MUMPS

MUMPS [82] (MUltifrontal Massively Parallel sparse direct Solver) es una librería para la resolución de sistemas de ecuaciones lineales, mediante métodos directos, de la forma $Ax = b$, donde la matriz A es dispersa y puede ser simétrica y definida positiva, simétrica general o incluso no simétrica.

Las principales características de este paquete incluyen la especificación de la matriz en formato ensamblado, ya sea distribuido o centralizado, análisis de errores, refinamiento iterativo y escalado de la matriz original. Además, MUMPS implementa diversos algoritmos de reordenación de ecuaciones con el objetivo de reducir el llenado, es decir los elementos nulos que pasan a tener un valor distinto de cero, producido en la matriz tras realizar la fase de factorización. Además, permite la utilización de librerías externas de reordenación como PORD [83] y METIS [84], incluso la especificación de la reordenación por parte del usuario. Además, la librería MUMPS permite trabajar en simple y doble precisión, tanto para matrices reales como complejas. MUMPS está implementada en Fortran 90, aunque existe una interfaz para su uso desde el lenguaje C. La versión paralela de MUMPS requiere MPI para la comunicación entre procesos, BLAS, BLACS (librería de comunicaciones) y ScaLAPACK [85]. Alternativamente, existe una versión secuencial de MUMPS que únicamente depende de BLAS.

La librería MUMPS distribuye el trabajo entre los procesadores, pero uno de ellos es el encargado de realizar la mayor parte de la fase de análisis, distribuir la matriz al resto de procesadores, en el caso de que la matriz esté centralizada, y la recogida de la solución. El sistema de ecuaciones $Ax = b$ se resuelve en 3 pasos principales:

1. Análisis. Uno de los procesadores (procesador principal) lleva a cabo una fase de ordenación, para tratar de mantener el nivel de dispersidad, y realiza la factorización simbólica. Posteriormente, se computa un mapeo del grafo computacional multifrontal y la información simbólica se transfiere del procesador principal al resto de procesadores. A partir de esta información, los procesadores obtienen una estimación de la memoria necesaria para la factorización y la solución.

2. Factorización. La matriz original se distribuye entre los procesadores que participan en la factorización numérica. A continuación se procede a realizar la factorización numérica en cada matriz frontal por un procesador maestro (determinado por la fase de análisis) y uno o más procesadores esclavos (determinados dinámicamente). Cada procesador reserva la memoria necesaria para los bloques contribuidos y los factores, que deben ser almacenados para la fase

de solución. Posteriormente, se realiza una factorización multifrontal con control de estabilidad basado en pivotación parcial.

3. Solución. El vector parte derecha b es enviado desde el procesador principal al resto. Estos procesadores computan parte del vector solución x usando los factores distribuidos calculados en el paso 2.

Por último, comentar que la librería MUMPS está en constante actualización y se trata de una librería de dominio público.

4.3.4. La librería CVODE

La librería CVODE [86] está incluida dentro de la suite SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers) [87] y contiene los solvers CVODE, KINSOL e IDA, así como variantes de estos paquetes. CVODE resuelve problemas de valores iniciales para ecuaciones diferenciales ordinarias. KINSOL se encarga de resolver sistemas algebraicos no lineales. Finalmente, IDA resuelve problemas de valores iniciales para sistemas de ecuaciones algebraico-diferenciales (DAE). Así mismo, se incluye una librería llamada CVODES que resuelve sistemas de ODEs, realizando un análisis de sensibilidad posterior. Existen versiones tanto secuenciales como paralelas de las 4 librerías mencionadas. La suite SUNDIALS al completo, desarrollada por el Laboratorio Nacional de Lawrence Livermore, está disponible bajo licencia BSD, que permite tener acceso al código fuente e incluso modificarlo siempre que se preserven los derechos de autor.

En particular, CVODE es una librería desarrollada en C para la resolución de problemas de valores iniciales para sistemas de ecuaciones diferenciales ordinarias (ODEs). Sirve tanto para problemas de valores iniciales rígidos como no rígidos, que pueden escribirse de la siguiente manera:

$$\dot{y} = \frac{\partial y}{\partial t} \tag{4.1}$$

Los métodos de integración empleados en CVODE son versiones de coeficientes variables del método de Adams-Moulton y BDF (Backward Differentiation Formula). La solución numérica de 4.1 se genera como valores discretos y_n en puntos de tiempo t_n. Los valores calculados siguen una fórmula lineal multipaso:

$$\sum_{i=0}^{K_1} \alpha_{n,i} y^{n-i} + h_n \sum_{i=0}^{K_2} \beta_{n,i} \dot{y}^{n-i} = 0. \tag{4.2}$$

Los valores y^n se calculan como aproximaciones a $y(t_n)$, mientras que $h_n = t_n - t_{n-1}$ es el paso de tiempo y $\alpha_{n,0} = -1$. Para el uso en problemas no rígidos, la fórmula de Adams-Moulton se caracteriza por $K_1 = 1$ y $K_2 = q$, donde el orden q varía entre 1 y 12. Para problemas rígidos, CVODE incluye el método BDF donde $K_1 = q$ y $K_2 = 0$, y el orden q varía entre 1 y 5 [88].

El uso de una librería de estas características permite la utilización de diferentes esquemas de integración de ODEs, y evaluar las ventajas obtenidas por su aplicación.

4.4. Conclusiones

En este capítulo se ha realizado una introducción a la Computación de Altas Prestaciones, que trata de obtener las máximas prestaciones del sistema de computación utilizado. Para ello, se aprovecha el conocimiento de la jerarquía de memoria, se utilizan métodos eficientes de resolución de problemas y de almacenamiento en disco, y se utiliza la computación paralela para resolver problemas de manera colaborativa y eficiente entre múltiples procesadores.

Se han descrito las diferentes estrategias de paralelización principales, basadas en el paradigma de memoria distribuida y el de memoria compartida, describiendo su relación con, respectivamente, los clusters de PCs y las arquitecturas multiprocesador. En este sentido, la existencia de estándares de comunicaciones entre procesos como MPI u OpenMP supone un importante avance en la portabilidad de las aplicaciones paralelas.

Además se ha comentado la tendencia actual, en el campo del álgebra numérica, hacia el desarrollo de librerías computacionales que permitan resolver problemas comunes en diferentes campos de la ingeniería. De esta manera, se consigue reducir el número de errores de programación, acelerar el desarrollo de aplicaciones científicas y utilizar estrategias de resolución eficientes. En este sentido, se han descrito los núcleos computacionales BLAS y LAPACK así como las librerías PETSc, MUMPS y CVODE que, posteriormente, serán empleadas en la presente tesis.

Este tipo de librerías se están convirtiendo en estándares *de facto* en computación científica. Por ello, resulta indispensable tratar de construir aplicaciones sobre estas herramientas, aprovechando el esfuerzo realizado por otros centros de investigación en la creación de herramientas eficientes.

Capítulo 5

El Sistema de Simulación Cardiaca CAMAEC

"First learn computer science and all the theory. Next develop a programming style. Then forget all that and just hack". George Carrette, Software Engineer.

En este capítulo se describe el sistema de simulación cardiaca CAMAEC. En primer lugar se aborda la arquitectura y se detallan los principales componentes. A continuación, se describe la estrategia de paralelización empleada para acelerar la ejecución de una simulación mediante el uso de un cluster de PCs. Posteriormente, se describen las implementaciones realizadas de los modelos iónicos celulares, detallando las principales técnicas de aceleración empleadas. Luego, se describe el sistema de gestión de grabación de datos, que emplea técnicas automáticas de generación de código y Entrada/Salida paralela, para acelerar este proceso. Seguidamente, se aborda la adaptación del sistema de simulación para su utilización en geometrías tridimensionales sencillas. A continuación, se evalúan las prestaciones de este sistema mediante su ejecución en diferentes plataformas paralelas. Finalmente, se describen las diferentes líneas de investigación en las que se está utilizando actualmente el sistema de simulación.

5.1. Introducción

El sistema CAMAEC (Computación Avanzada en la Modelización de la Actividad Eléctrica Cardiaca) realiza simulaciones de propagación del potencial de acción, en tejidos cardiacos bidimensionales monodominio, empleando Computación de Altas Prestaciones basada en un paradigma de memoria distribuida sobre un cluster de PCs. La Figura 5-1 presenta un esquema de la arquitectura principal del sistema de simulación. Inicialmente, la aplicación recibe como entrada la información

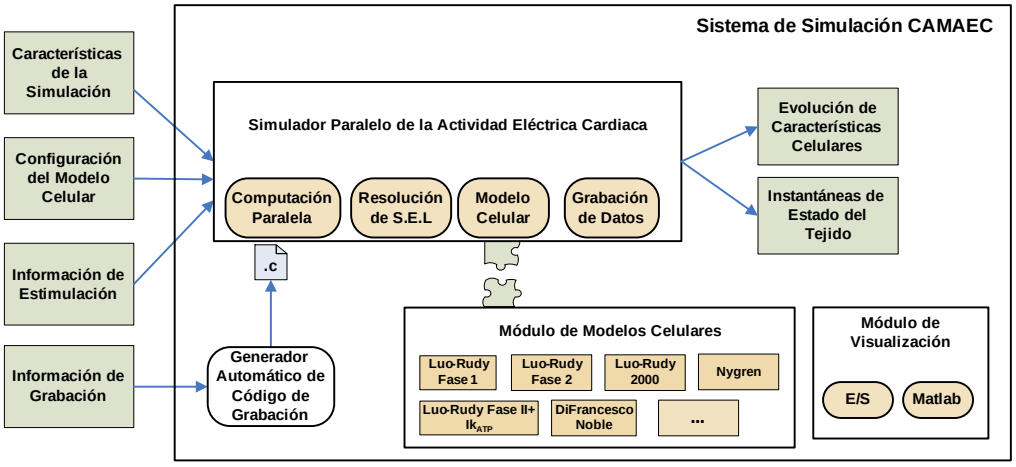


Figura 5-1: Arquitectura principal del sistema de simulación CAMAEC.

que describe la simulación a realizar:

- Características de la Simulación. Incluye información como el paso de tiempo de discretización temporal, el instante de tiempo final de la simulación, el tamaño del tejido y el modelo celular a emplear.

- Configuración del Modelo Celular. Aquí se describe la información específica del modelo celular empleado e incluye información sobre los valores iniciales de las concentraciones iónicas, los factores de bloqueo de las corrientes y demás atributos específicos dependientes del modelo celular. Estos valores permiten establecer una configuración inicial del tejido celular.

- Información de Grabación. Permite especificar, para cada una de las células del tejido, la grabación periódica de cualquier característica, a lo largo de una ventana de tiempo, durante el proceso de simulación. También permite la grabación de instantáneas del estado de cualquier característica, para todas las células, a lo largo de la simulación.

- Información de Estimulación. Permite la definición de patrones de estimulación eléctrica en el tejido cardiaco, mediante la especificación de un rectángulo de tejido donde se aplicará un tren de impulsos de una amplitud y duración determinada.

El sistema de simulación, durante la ejecución, produce una serie de resultados que dependen de la información de grabación especificada por el usuario:

- Evolución de Características Celulares. Se obtienen tantos ficheros como características y células ha decidido grabar el usuario. Cada fichero contiene una secuencia de los valores que ha ido tomando una característica de una célula concreta, a lo largo de un periodo de tiempo, grabada de manera periódica.

- Instantáneas de Estado del Tejido. Estos ficheros contienen una secuencia de valores de una determinada característica, para todas las células del tejido, a lo largo del periodo de grabación definido por el usuario. Esta funcionalidad se suele utilizar, principalmente, para determinar los patrones de propagación del potencial de membrana en el tejido, permitiendo la generación de secuencias animadas.

Para conseguir los resultados, tal y como se muestra en la Figura 5-1, el sistema dispone de varios módulos:

- Computación Paralela. Gestiona la utilización del paralelismo, permitiendo el uso transparente de múltiples procesadores para abordar de manera conjunta la simulación de propagación del potencial de acción.

- Resolución de Sistemas de Ecuaciones Lineales. Encargado de resolver de manera eficiente el sistema de ecuaciones lineales que surge al discretizar la ecuación de propagación del potencial de acción.

- Modelo Celular. Implementa diferentes modelos celulares iónicos que pueden ser seleccionados por el usuario, para el estudio de diversos tipos de tejidos cardiacos. Este módulo permite una sencilla incorporación de nuevos modelos celulares por parte de usuarios que no dispongan de elevados conocimientos de programación.

- Grabación de Datos. Coordina la grabación de los datos en disco, de manera colaborativa, entre los múltiples procesadores de una ejecución. Permite la obtención de resultados parciales a lo largo del proceso de simulación que, posteriormente, serán analizados por expertos biomédicos.

Adicionalmente, el sistema incorpora un módulo de visualización, basado en Matlab, que permite procesar eficientemente los resultados del proceso de simulación. Las herramientas desarrolladas permiten graficar el comportamiento de las variables a estudiar, así como la obtención de vídeos en el formato estándar MPEG (Motion Pictures Expert Group), para que puedan ser posteriormente visualizados en cualquier equipo. Para ello se han utilizado técnicas de procesamiento a bloques de la información, con el objetivo de obtener métodos con un consumo de memoria acotado e independiente de la cantidad de datos de entrada. Esto supone una importante ventaja, especialmente para las simulaciones que generan varios GBytes de información.

El simulador paralelo de la actividad eléctrica cardiaca consta actualmente de casi 100 ficheros, que aglutinan un total de más de 30000 lineas de código fuente escritas en el lenguaje de programación C. El módulo de visualización consta actualmente de unos 20 ficheros escritos en el lenguaje de Matlab. A lo largo de las siguientes secciones se describen con mayor detalle los módulos de este sistema de simulación.

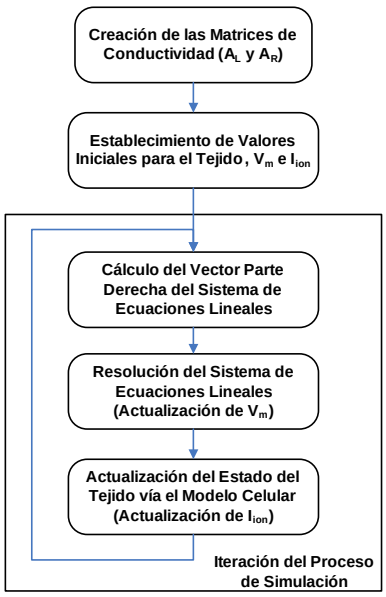


Figura 5-2: Principales fases involucradas en el proceso de simulación de la propagación del potencial de acción con el sistema CAMAEC.

5.2. Resolución del Problema Computacional

La Ecuación 2.4 de propagación eléctrica puso de manifiesto el carácter iterativo del proceso de simulación. La Figura 5-2 resume las principales fases que se realizan durante el proceso de simulación cardiaca con CAMAEC. En ella se observa que hay una fase inicial de generación de las matrices de conductividad que, posteriormente, se complementa con la asignación de valores iniciales, dependientes del problema, a las principales estructuras de datos: El vector de potencial de membrana (V_m), el vector de corrientes iónicas (I_{ion}) y el tejido cardiaco.

Posteriormente, tal y como se describió en la sección 2.3, comienza un proceso iterativo en el que, para cada paso de tiempo, se realizan varias fases. En primer lugar se computa el vector parte derecha del sistema de ecuaciones. A continuación, se resuelve el sistema de ecuaciones lineales que permite obtener el nuevo potencial de membrana de las células del tejido. Finalmente, el estado de cada célula y, por extensión, el tejido, se actualiza en base al modelo celular de la misma y al nuevo potencial de membrana.

Antes de abordar la paralelización del problema, resulta importante determinar las fases del proceso de simulación en las que se invierte la mayor parte del tiempo de ejecución. Para ello, la Tabla 5.1 muestra el porcentaje del tiempo global que requiere cada una de las principales fases del proceso iterativo de simulación cardiaca, para dos tamaños de tejido diferentes, sobre una plataforma secuencial (Xeon, 2.0 GHz, 1 GByte de RAM). El modelo celular empleado es el Luo-Rudy Fase II y el sistema de ecuaciones se resuelve mediante el método del Gradiente Conjugado [89] sin precondicionado. Los cálculos se han obtenido a partir de una simulación de 100 pasos de tiempo,

Tabla 5.1: Porcentaje de tiempo, respecto al tiempo total de ejecución, involucrado en cada una de las principales fases del proceso de simulación de propagación del potencial de acción.

Fase Principal	**Tejido 100x100**	**Tejido 500x500**
Cálculo Vector Parte Derecha	1.22	0.88
Resolución Sistema de Ecuaciones	8.3	10
Ejecución Modelo Celular	90.4	89

en la que inicialmente se inyectó un estímulo supraumbral, es decir, suficientemente intenso como para desencadenar un potencial de acción.

Se puede observar que las dos fases que requieren la mayor parte del tiempo corresponden a la resolución del sistema de ecuaciones y la ejecución del modelo celular, siendo esta última la más costosa con diferencia. Estas medidas de tiempo son coherentes con los resultado obtenidos en [29], donde se muestran tiempos de ejecución de una simulación de características similares, empleando también el modelo celular Luo-Rudy Fase II. En el artículo se menciona que la resolución del sistema de ecuaciones precisa un 12 % del tiempo global de simulación, mientras que la ejecución del modelo celular requiere un 85 %. En efecto, los modelos celulares cardiacos implican una elevada complejidad computacional debido a los numerosos cálculos de expresiones matemáticas complejas que, además, involucran operaciones costosas de resolver en un ordenador.

5.2.1. Estrategia de Paralelización

Para conseguir una adecuada paralelización del problema es necesario maximizar la utilización de todos los procesadores involucrados en la ejecución. Para ello, es necesario realizar un apropiado balance de la carga computacional que trate de conseguir los siguientes objetivos [90]:

1. Realizar una distribución de datos equitativa, entre los diferentes procesadores, de manera que el tiempo de ejecución requerido por cada procesador sea similar.

2. Reducir el número de comunicaciones entre los procesadores. Una operación de comunicación tiene un coste mucho mayor al de cualquier operación realizada dentro de un procesador y, por lo tanto, las comunicaciones afectan negativamente al tiempo global de ejecución.

3. Realizar una distribución de los requisitos globales de memoria entre los diferentes procesadores. De esta manera, será posible abordar tamaños de problema mayores que los posibles en una plataforma secuencial.

Con respecto al problema que nos ocupa, el proceso de actualización de una célula únicamente depende del nuevo potencial de membrana de la misma y no precisa interacción alguna con el resto de células del tejido cardiaco. Además, esta fase requiere el mayor tiempo de ejecución. Por lo tanto,

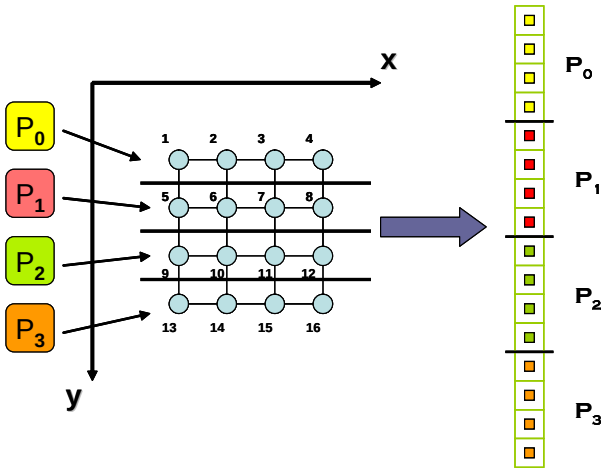


Figura 5-3: Estrategia de paralelización. Ejemplo de división de un tejido bidimensional de 4x4 células entre 4 procesos. Adicionalmente, se muestra la partición correspondiente al vector de corrientes iónicas I_{ion}.

una estrategia eficiente para la distribución de datos consiste en repartir la tarea de actualización del tejido entre todos los procesadores involucrados.

Para abordar la paralelización se ha utilizado el paradigma de programación en memoria distribuida. Para ello se ha diseñado una aplicación con estructura SPMD (Single Program, Multiple Data [91]), donde los procesadores realizan la misma función sobre diferentes datos. La arquitectura escogida es un cluster de PCs. La elección de este tipo de arquitectura paralela responde a varias razones. En primer lugar, un cluster de PCs presenta un excelente ratio coste/prestaciones y resulta muy sencillo aumentar las capacidades del sistema sin más que añadir nuevos nodos de computación. En segundo lugar, la memoria del sistema global aumenta de manera proporcional con el número de nodos, por lo que es factible aumentar la memoria del sistema mediante la incorporación de nuevos nodos de computación. Finalmente, las plataformas de computación basadas en clusters de PCs suelen estar disponibles en los centros de investigación. Esta situación facilita en gran medida el proceso de migración del sistema de simulación a otras máquinas.

La estrategia de paralelización empleada consiste en realizar una distribución del tejido cardiaco entre los diferentes procesadores. Para ello, cada procesador se encarga de la actualización de un conjunto de células consecutivas. Esta división se realiza de manera que todos los procesadores tengan un conjunto de células de tamaño similar, balanceando así la carga computacional entre todos los procesadores. La Figura 5-3 muestra un ejemplo de descomposición de un tejido bidimensional de 4x4 células entre 4 procesadores. En ella se puede observar el esquema de numeración de células empleado: primero el eje x y luego el y. Si el número de células no es múltiplo del número de procesadores, las células sobrantes se reparten entre los procesadores de manera consecutiva comenzando desde el último procesador.

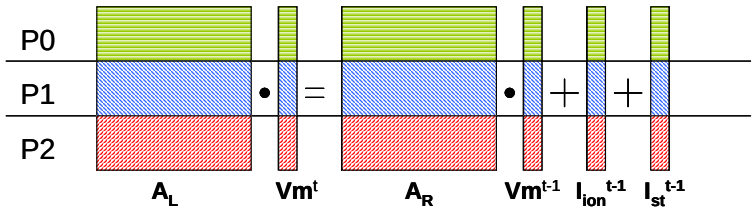


Figura 5-4: Ejemplo de partición de las principales estructuras de datos involucradas en el proceso de simulación cardiaca para 3 procesadores.

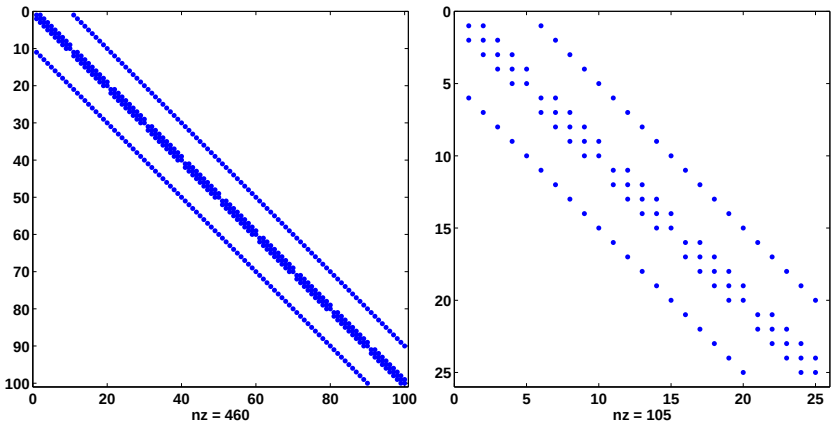
Por ejemplo, si el tejido consta de 15 células y la ejecución involucra a 4 procesadores (P0, P1, P2 y P3) entonces cada procesador recibe $15/4 = 3$ células (división entera). Las restantes 3 células ($15\,\%4 = 3$) son repartidas entre todos los procesadores comenzando por el último. De esta manera, P0 se encargaría de las células $[0-2]$, P1 de las células $[3-6]$, P2 de las células $[7-10]$ y P3 de las células $[11-14]$, proporcionando un balance equilibrado de la carga.

Las principales estructuras de datos, requeridas por el proceso de simulación, también han sido convenientemente particionadas, entre todos los procesadores, empleando una distribución por bloques de filas consecutivas, tal y como muestra la Figura 5-4. En efecto, la i-ésima fila de la matriz se corresponde con la i-ésima célula del tejido de acuerdo a la numeración empleada. Las matrices involucradas, así como el vector de potencial de membrana y de corrientes iónicas, se generan en paralelo de manera eficiente puesto que es posible calcularlos sin realizar ninguna comunicación entre los procesos.

5.2.2. Sistema de Ecuaciones Lineales

La fase correspondiente a la resolución del sistema de ecuaciones lineales es computacionalmente costosa debido a la magnitud del problema y, por lo tanto, requiere mecanismos eficientes de resolución que traten de reducir el tiempo de ejecución. En este sentido, resulta importante conocer las características de la matriz de coeficientes para averiguar las propiedades del sistema de ecuaciones lineales. Este conocimiento ayudará en la elección de métodos de resolución específicos para el tipo de matriz de coeficientes [80].

La Figura 5-5 muestra dos ejemplos de matriz de coeficientes del sistema de ecuaciones, correspondientes a dos tejidos de diferente tamaño. La dimensión de la matriz equivale al número de nodos de la malla de discretización que, en general, corresponde al número de células del tejido. La matriz es simétrica, definida negativa, y muestra una estructura regular con un elevado nivel de dispersidad, ya que se trata de una matriz pentadiagonal. El patrón de dispersidad de la matriz depende del ordenamiento empleado para las incógnitas que, en nuestro caso, corresponde a la ordenación natural. La forma pentadiagonal de la matriz es debida a la utilización de un esquema de diferencias finitas que considera 5 vecinos para el tejido bidimensional. El hecho de que la matriz original sea definida



(a) Tejido 10x10 (b) Tejido 5x5

Figura 5-5: Matriz de coeficientes asociada al sistema de ecuaciones lineales.

negativa no supone ningún problema para la resolución del sistema de ecuaciones, ya que en lugar de resolver el sistema $Ax = b$ se resuelve el sistema $-Ax = -b$ donde la matriz $-A$ es simétrica y definida positiva.

Con el objetivo de reducir la cantidad de memoria necesaria para el almacenamiento de la matriz se utilizan técnicas de almacenamiento disperso, disponibles en el marco de la librería PETSc, en el que únicamente se realiza el almacenamiento de las entradas no nulas de la matriz.

Condicionamiento de la Matriz

El estudio del condicionamiento de la matriz permite, por un lado, estudiar los errores que se pueden introducir en el proceso de resolución del sistema de ecuaciones, para estudiar el impacto de los mismos en el resultado final. Por otro lado permite obtener una idea de la bondad de la aplicación de métodos iterativos de resolución [80]. Para ello se analizará el número de condición de la matriz.

$$cond(A) = ||A|| \cdot ||A^{-1}|| \tag{5.1}$$

La expresión 5.1 muestra la definición teórica del número de condición de una matriz, donde la norma matricial es inducida por una norma vectorial. El cálculo del número de condición de A requiere, por tanto, el cálculo de A y de A^{-1}. Obviamente, el cálculo de la inversa de una matriz, cuando la matriz tiene una dimensión muy elevada, puede ser impracticable y por lo tanto esa expresión no se suele computar de manera explícita.

Para ello, se ha empleado Matlab que calcula el número de condición aplicando la 2-norma (el ratio del mayor valor singular de la matriz con respecto al menor). Un valor elevado de número de condición indica que la matriz es casi singular. El mejor número de condición es 1, correspondiente a la matriz identidad.

Tabla 5.2: Número de condición de la matriz de coeficientes para diferentes tamaños de tejido.

Tamaño del Tejido	Matriz de Coeficientes	Número de Condición
50x50	2500x2500	1.2914
100x100	10000x10000	1.2914

Para matrices dispersas, sin embargo, Matlab calcula una cota inferior para el número de condición basado en la 1-norma de una matriz cuadrada, con el objetivo de reducir el coste global del proceso. Para el caso de un tejido de 50x50 células (matriz de 2500x2500), el número de condición real computado (calculado como $cond(full(A))$) es 1.2911, mientras que el número de condición estimado (calculado como $condest(A)$) es 1.2914. En el primer caso se requieren 193 MBytes de memoria RAM y 84.9 segundos para calcular el resultado, mientras que, en el segundo caso, el resultado se obtiene en tiempo despreciable. Obtener una estimación del número de condición supone una importante ventaja, sobre todo para el caso de matrices dispersas de gran dimensión donde su correspondiente matriz densa puede sobrepasar holgadamente las capacidades de la máquina. La Tabla 5.2 muestra el número de condición estimado por Matlab, para diferentes tamaños de matriz de coeficientes correspondientes a diferentes tejidos.

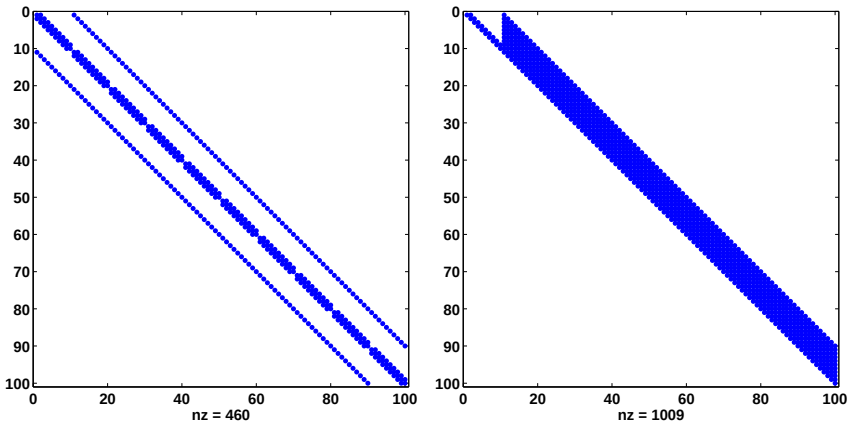
A la vista de los resultados se puede observar que la matriz de coeficientes del sistema de ecuaciones está muy bien condicionada. Esta información indica fundamentalmente dos cosas: En primer lugar, el buen condicionamiento del problema garantiza que, ante pequeñas variaciones en los datos de entrada, no es de esperar grandes cambios en la solución. Esta característica es muy deseable en procesos de simulación de fenómenos físicos. En segundo lugar, un número de condición pequeño permite intuir una rápida convergencia a la solución mediante la aplicación de un método iterativo de resolución de sistemas de ecuaciones [92].

Métodos de Resolución del Sistema de Ecuaciones Lineales

En la actualidad los métodos empleados para la resolución de un sistema de ecuaciones lineales se pueden agrupar en métodos directos [93] y métodos iterativos [80]. Los métodos directos se han aplicado tradicionalmente a la resolución de sistemas donde la matriz de coeficientes tiene una estructura densa, es decir, donde la mayoría de sus entradas son diferentes de 0. Mediante estas técnicas generalmente se trata de buscar la solución de un problema permutado [94], donde las filas se intercambian con el objetivo de reducir el llenado producido al realizar la factorización. Por ejemplo, sea un sistema de ecuaciones lineales dado por la siguiente expresión:

$$Ax = b \tag{5.2}$$

Emplear un método directo, como la eliminación Gaussiana, implica realizar una descomposición



(a) Matriz Original (b) Factor de Cholesky

Figura 5-6: Efecto del llenado al aplicar una factorización de Cholesky (Tejido 10x10).

de la matriz A en dos factores L y U de manera que $A = LU$. Esto permite transformar la Ecuación 5.2 en dos sistemas de ecuaciones:

$$\begin{aligned} Ux &= y \\ Ly &= b \end{aligned} \tag{5.3}$$

donde la matriz U es triangular superior y la matriz L es triangular inferior unidad. Además, si se pretende resolver un sistema de ecuaciones con la misma matriz de coeficientes, pero con múltiples vectores parte derecha, entonces resulta ventajoso almacenar los factores L y U, ya que únicamente se deberán resolver los dos sistemas triangulares para obtener respectivamente y y el vector solución x. De esta manera, es posible realizar una única vez la costosa factorización y, posteriormente, reaprovecharla para el resto de resoluciones.

Sin embargo, la utilización de un método directo en matrices dispersas provoca el llenado de componentes en ambos factores, tras realizar la descomposición. Básicamente, el ancho de banda de la matriz se mantiene pero el patrón de dispersidad interno a la banda se pierde. Esta situación requiere un aumento en el espacio de almacenamiento, ya que entradas de la matriz que antes eran nulas y, por lo tanto, no precisaban ser almacenadas, ahora dejan de serlo y requieren ser guardadas. Además, el llenado provoca un aumento en el número de cálculos a realizar, incrementando la complejidad computacional del problema.

Por ejemplo, la Figura 5-6 muestra los efectos del llenado al realizar una factorización de Cholesky sobre una matriz de coeficientes procedente de un tejido de 10x10 células. Se puede observar que la matriz de coeficientes inicial tiene 460 elementos no nulos, mientras que el factor de Cholesky resultante contiene 1009 elementos, provocando un incremento de memoria considerable. En la figura

se puede observar que el llenado únicamente se produce dentro de la banda, provocando la aparición de nuevos elementos no nulos.

La alternativa al uso de un método directo supone la aplicación de un método iterativo, que explota el patrón de dispersidad de la matriz y no requiere demasiado espacio extra para realizar los cálculos. Además, una de las ventajas de los métodos iterativos frente a los directos es que la estructura de la matriz permanece inalterada durante todo el proceso por lo que no es preciso utilizar memoria adicional.

Los métodos iterativos parten de una aproximación inicial a la solución y generan una secuencia de vectores $x^{(k)}$, que se espera que converjan a la solución x. La carga computacional de estos métodos suele ser modesta, cuando se utilizan técnicas iterativas eficientes y, por lo general, únicamente involucran productos matriz-vector. Como desventaja frente a los métodos directos, cuando se pretende resolver un sistema de ecuaciones con la misma matriz de coeficientes y diferentes vectores parte derecha, un método iterativo no puede aprovechar información de ninguna resolución anterior, mientras que el método directo puede aprovechar el resultado de la factorización.

En las siguientes secciones se detalla el proceso de resolución del sistema de ecuaciones lineales, empleando varios métodos iterativos, y la evaluación de sus prestaciones para elegir el método más eficiente. Además, también se utiliza un método directo para comparar su coste frente a la alternativa iterativa elegida.

5.2.3. Resolución Empleando Métodos Iterativos

Para la resolución del sistema de ecuaciones lineales, mediante métodos iterativos, se ha optado por emplear diversos métodos basados en subespacios de Krylov [80]: Gradiente Conjugado (Conjugate Gradient, CG), Gradiente BiConjugado (BiConjugate Gradient, BICG), Residuo Mínimo Generalizado (Generalized Minimal Residual, GMRES) y Residuo Casi Mínimo Sin Transpuesta (Transpose-Free Quasi-Minimal Residual, TFQMR). Todos ellos están disponibles en el marco de la librería PETSc.

Los métodos basados en subespacios de Krylov realizan productos matriz-vector en cada iteración, así como algunas operaciones con vectores (producto escalar y actualizaciones de vectores) por lo que son altamente paralelizables. Estos métodos se utilizan generalmente para adaptarse a dos requerimientos básicos. Por un lado, minimizar cierta norma del vector residuo sobre un subespacio de Krylov generado por la matriz del sistema y que dé lugar a una convergencia suave sin grandes fluctuaciones. Por otro lado, el objetivo consiste en reducir el coste computacional por iteración sin exigir una excesiva capacidad de almacenamiento [95].

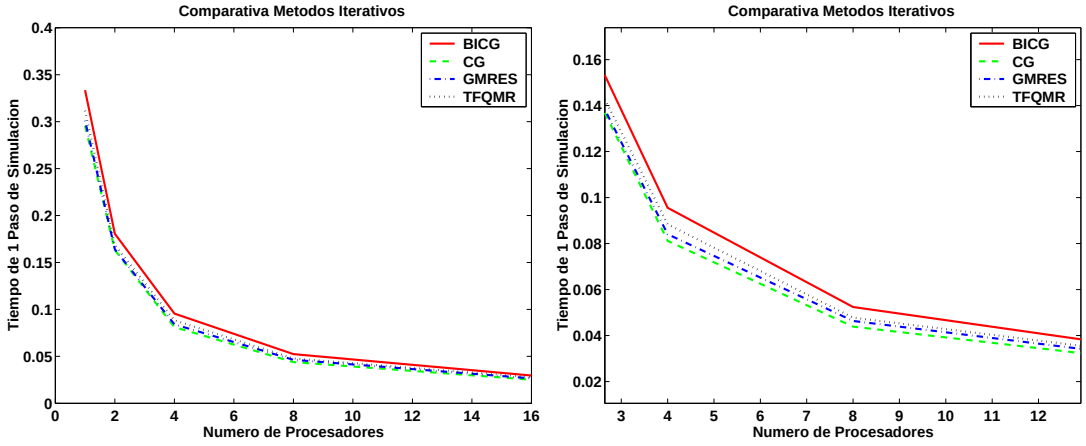
El método del Gradiente Conjugado, propuesto por Hestenes y Stiefel en 1952 [89], y desarrollado a partir de 1970, se aplica a matrices simétricas y definidas positivas. Teóricamente y, salvo errores de redondeo, este algoritmo alcanza la solución exacta en un número de iteraciones igual a la dimensión

del sistema. El Método del Residuo Mínimo Generalizado [96] es un método de proyección basado en minimizar la norma del residuo. Este método puede ser utilizado en sistemas de ecuaciones donde la matriz de coeficientes no es simétrica. GMRES requiere un espacio de almacenamiento superior a CG y el algoritmo es susceptible de obtener peores prestaciones paralelas que CG [18]. Para las pruebas se ha empleado un reinicio de 30 usando la ortogonalización de Gram-Schmidt sin modificar. El método de Gradiente BiConjugado [97] es un método de biortogonalización, es decir, un método de proyección pero intrínsecamente no ortogonal. Los métodos de biortogonalización presentan la ventaja de que las formulas de recurrencia son reducidas y el almacenamiento no aumenta con el número de iteraciones. Sin embargo, tienen un comportamiento irregular en la convergencia, con oscilaciones que pueden dar lugar a criterios de parada con condiciones erróneas [98]. Este tipo de métodos se fundamentan en el algoritmo de biortogonalización de Lanczos [97]. Por último, el método TFQMR [99] es también un método de biortogonalización, que presenta la ventaja de no requerir productos de matriz por vector con A^T.

En todos los métodos empleados, el criterio de parada ha consistido en obtener un error relativo inferior a 1e-10 o realizar como máximo un total de 10000 iteraciones. Dado que el objetivo de esta tesis no es profundizar en la implementación de algoritmos iterativos de resolución de sistemas de ecuaciones lineales, es posible encontrar una explicación ampliada de estos y otros métodos iterativos en [98].

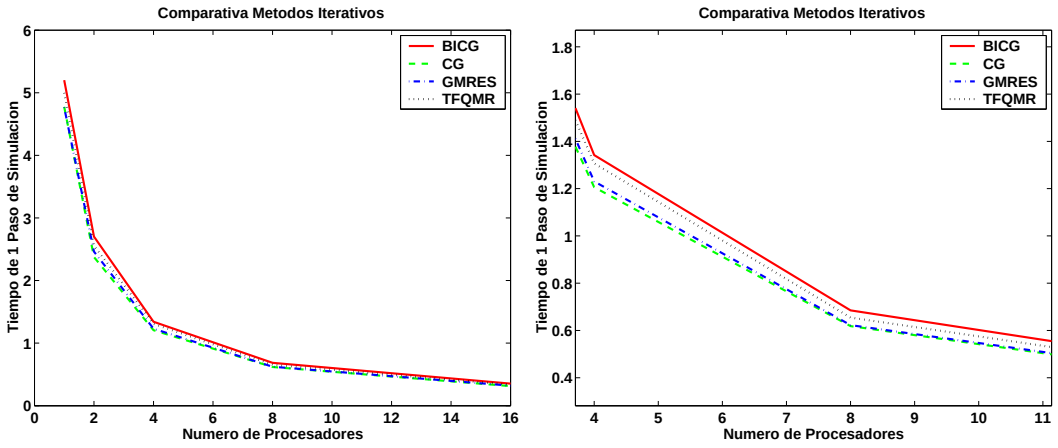
La Figura 5-7 muestra una comparativa del tiempo de ejecución requerido, en cada paso de simulación, utilizando cada uno de los 4 métodos iterativos descritos, en una simulación de propagación del potencial de acción sobre un tejido de tamaño medio, de 200x200 células, que da lugar una matriz de 40000x40000. Las ejecuciones se han llevado a cabo en el cluster Kefren (ver Anexo A). Los resultados se han obtenido simulando un total de 10 ms con un paso de tiempo de 10 μs, es decir, promediando el tiempo de ejecución sobre un total de 1000 pasos de simulación. En el instante de tiempo 0 se inyectó un estímulo supraumbral para inducir variabilidad numérica en el vector parte derecha del sistema de ecuaciones, lo que puede afectar a la convergencia del método iterativo. La Figura 5-8 muestra los mismos resultados para un tejido bidimensional grande de 800x800 células que da lugar a una matriz de coeficientes de 640000x640000.

A la vista de los resultados obtenidos se pueden obtener diversas conclusiones. En primer lugar se observa que el tiempo empleado por cada uno de los métodos es bastante similar entre ellos. Esto es debido a las buenas propiedades de la matriz, ya que, al estar bien condicionada, la convergencia en la resolución del sistema de ecuaciones se alcanza en pocas iteraciones. Sin embargo, se observa que el método del Gradiente Conjugado es el que resuelve el sistema de ecuaciones lineales en menor tiempo, independientemente del número de procesadores empleado (en el ejemplo, hasta con 16 procesadores). Las ejecuciones realizadas reflejan que la convergencia del método CG se obtiene como máximo en 7 iteraciones y, en promedio, en unas 5 iteraciones.



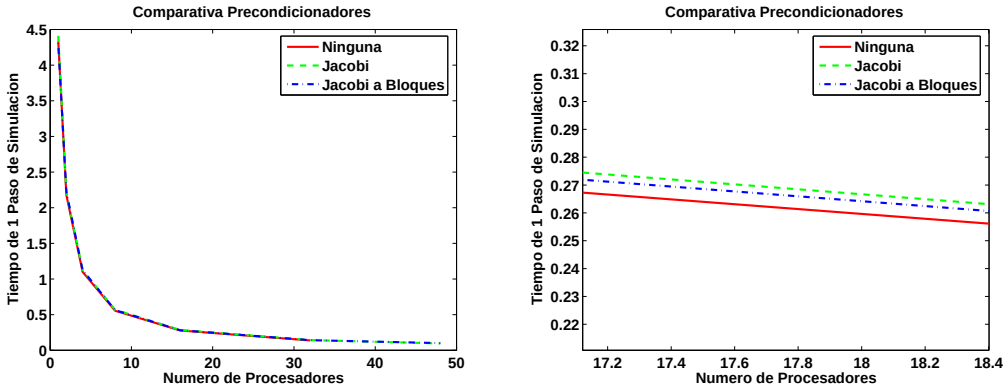
(a) General (b) Detalle

Figura 5-7: Comparativa de métodos iterativos (Tejido 200x200). El tiempo se expresa en segundos.



(a) General (b) Detalle

Figura 5-8: Comparativa de métodos iterativos (Tejido 800x800). El tiempo se expresa en segundos.



(a) General (b) Detalle

Figura 5-9: Utilización de precondicionadores de Jacobi y Jacobi a bloques (Tejido 800x800).

En efecto, dado que la matriz de coeficientes es simétrica y definida positiva el método del Gradiente Conjugado, aplicable únicamente a este tipo de matrices, verifica las propiedades de minimizar la norma del vector residuo sobre el espacio de Krylov así como ofrecer un bajo coste computacional por iteración, sin exigir una excesiva capacidad de almacenamiento.

Por otra parte, la tasa de convergencia de los métodos de proyección basados en subespacios de Krylov, para un sistema, depende de su espectro de valores propios [78]. En este sentido, el precondicionamiento es una herramienta usada para alterar el espectro y, de esta manera, acelerar la tasa de convergencia de los métodos iterativos. En realidad, el precondicionamiento permite transformar el sistema lineal original en otro sistema, con la misma solución, pero más fácil de resolver [80]. Sin embargo, para el caso que nos ocupa, la rápida convergencia del método desaconseja la utilización de precondicionadores, ya que la sobrecarga introducida en su aplicación puede no compensar la mejora en la convergencia del método.

La Figura 5-9 muestra el resultado de utilizar técnicas de precondicionamiento en la resolución del sistema de ecuaciones de un tejido 800x800 (matriz 640000x640000) con las mismas condiciones que en el caso anterior. Se ha empleado un precondicionado de Jacobi y de Jacobi a bloques. Los resultados confirman la hipótesis inicial de que no es necesaria la utilización de un precondicionado, para acelerar la convergencia del método iterativo, dada las buenas propiedades de la matriz.

5.2.4. Resolución Empleando Métodos Directos

En esta sección se describe la aplicación de un método directo para la resolución del sistema de ecuaciones lineales disperso y de gran dimensión. Aunque las buenas propiedades de la matriz apuntan a una rápida convergencia de los métodos iterativos, la utilización de un método directo puede ser muy útil para este tipo de problemas.

Para ello, es importante recordar que la matriz de coeficientes no varía a lo largo de la simulación

y, por lo tanto, es posible aprovechar el resultado de la factorización de la matriz, que únicamente deberá realizarse al principio. Posteriormente, los pasos de simulación únicamente involucrarán la resolución de dos sistemas triangulares. De esta manera, es posible que la resolución de los sistemas triangulares tenga un coste temporal inferior al necesario para alcanzar la convergencia del método iterativo. Bajo estas circunstancias, con un número suficientemente grande de pasos de simulación, el coste de la factorización podría despreciarse del coste global del proceso.

Para ello se ha empleado la Factorización de Cholesky Multifrontal [100, 101] como método directo paralelo, en el marco de la librería MUMPS. Esta factorización ha demostrado ser, para una amplia gama de matrices dispersas, una de las más eficientes y escalables sobre plataformas con memoria distribuida, presentando una sobrecarga de comunicación inferior a cualquier otra formulación paralela basada en métodos directos y aplicada sobre matrices dispersas. Antes de realizar la factorización y, para evitar el llenado, la matriz de coeficientes se reordena empleando el algoritmo paralelo de Disección Anidada Multinivel [102] (DAM), que utiliza a su vez el algoritmo de Bisección de Grafos Multinivel [103]. El método directo empleado está compuesto por las siguientes cuatro fases consecutivas paralelizadas:

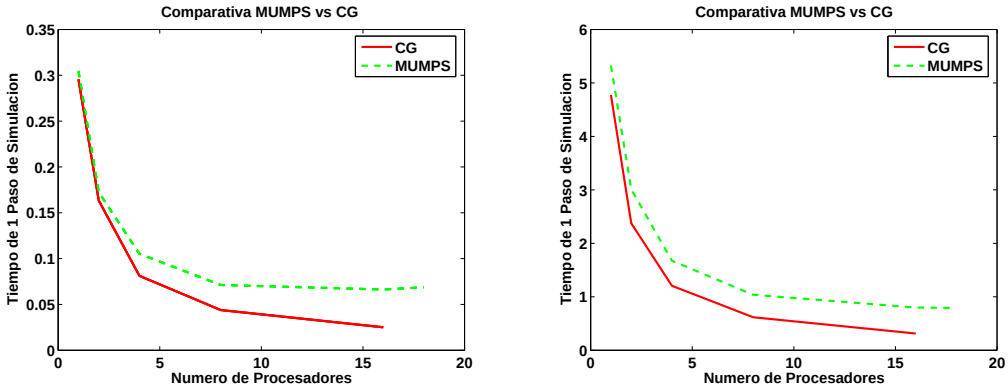
1. Ordenación. Consiste en calcular una matriz de permutación P de manera que la matriz PAP^T tenga un mínimo llenado en la fase de factorización de la matriz de coeficientes, reduciendo los requerimientos de memoria y el número de operaciones durante la misma, al tiempo que incrementa el grado de paralelismo. Esto supone que dicha etapa sea un factor clave a la hora de determinar la eficiencia de la implementación paralela del resto de fases.

2. Factorización Simbólica. En esta etapa se determina la estructura de elementos distintos de cero del factor de Cholesky triangular L. El papel de dicha factorización simbólica es incrementar las prestaciones de la siguiente fase.

3. Factorización Numérica. Durante esta fase tienen lugar las operaciones necesarias para calcular los valores de L que satisfacen que $PAP^T = LL^T$, empleando el algoritmo de la Factorización de Cholesky Multifrontal en paralelo.

4. Resolución de Sistemas Triangulares. Finalmente, la solución del sistema se calcula resolviendo dos sistemas triangulares, $Ly = b'$ seguido por $L^T x' = y$, donde $b' = Pb$ y $x' = Px$. El primero de dichos sistemas se resuelve mediante el algoritmo de Eliminación Progresiva, mientras que el segundo utiliza el algoritmo de Sustitución Regresiva. Finalmente, la solución del sistema se obtiene como $x = P^T x'$.

En nuestro caso, la ordenación y la factorización simbólica y numérica se realizan una única vez al comienzo de la simulación, puesto que la matriz de coeficientes A_L permanece constante a lo largo



(a) Tejido 200x200 (b) Tejido 800x800

Figura 5-10: Comparativa del método de Gradiente Conjugado frente a la resolución de sistemas triangulares después de una Factorización de Cholesky Multifrontal. El tiempo se expresa en segundos.

de todo el proceso de simulación. De este modo, para cada paso de tiempo se deben resolver en paralelo dos sistemas de ecuaciones triangulares.

La Figura 5-10 muestra el tiempo involucrado en un paso de simulación, comparando el método directo descrito con el método iterativo del Gradiente Conjugado sin precondicionar. Para ello se ha realizado, al igual que en la prueba de los métodos iterativos, una simulación de 10 ms empleando un paso de tiempo de 10 μs, es decir, realizando un total de 1000 pasos de simulación y obteniendo un promedio del coste temporal de un paso. Las pruebas se han realizado para un tamaño de tejido pequeño (200x200 células) y para un tamaño de tejido grande (800x800 células).

Con respecto al método directo, el tiempo de la fase de ordenación, factorización simbólica y factorización numérica han sido descartados, dado que la matriz de coeficientes del sistema permanece constante a lo largo del proceso de simulación y el resultado de la factorización puede ser reutilizado. Por lo tanto, únicamente se incluye el tiempo involucrado en la resolución de los sistemas triangulares.

A la vista de los resultados se puede observar que, para un número reducido de procesadores, la resolución de los sistemas triangulares llevada a cabo por MUMPS resulta muy competitiva respecto a la resolución empleando el método del Gradiente Conjugado. Sin embargo, conforme aumenta el número de procesadores la resolución mediante el método iterativo es cada vez más rápida con respecto al método directo proporcionado por MUMPS. De hecho, para el tejido de tamaño pequeño (200x200), se puede observar como a partir de 16 procesadores ya no se obtiene una reducción significativa en el tiempo de resolución.

La Tabla 5.3 muestra los tiempos requeridos por la librería MUMPS en factorizar la matriz de coeficientes asociada al sistema de ecuaciones lineales. En ella se puede observar como, para una matriz de tamaño pequeño (40000x40000), el método no consigue escalar adecuadamente y aumentar el número de procesadores no supone una reducción en el tiempo de la factorización. De hecho, a

Tabla 5.3: Coste temporal, en segundos, de la factorización realizada por MUMPS para dos matrices de coeficientes asociadas a tejidos de 200x200 células y 800x800 células.

Número de Procesadores	Matriz 40000x40000	Matriz 640000x640000
1	4.26	768.15
2	4.07	426.52
4	3.68	231.94
8	4.33	151.14
16	12.82	117.87

veces incluso puede verse afectado debido a las comunicaciones entre procesos. Para una matriz de tamaño grande (640000x640000) la factorización presenta mejores resultados de escalabilidad, permitiendo reducir el tiempo al aumentar el número de procesadores.

Resulta importante destacar que, para tamaños de tejido grandes, el coste espacial requerido por el proceso de factorización puede superar la cantidad de memoria disponible en la máquina lo que supone una desventaja frente a los métodos iterativos, que no precisan requisitos adicionales de memoria. Además, no hay que olvidar que, en la comparativa, se ha eliminado el coste temporal de la factorización. Sin embargo, este coste no debería ser obviado para simulaciones cortas, ya que puede suponer una fracción importante con respecto al tiempo de simulación global.

Por lo tanto y, a la vista de los resultados, se puede concluir que el método del Gradiente Conjugado sin precondicionar es el que obtiene las mejores prestaciones en la resolución del sistema de ecuaciones lineales.

5.3. Modelos Iónicos Celulares

De todo el proceso de simulación cardiaca, la fase correspondiente a la ejecución del modelo celular, para la actualización del estado de las células, es, con diferencia, la más costosa en tiempo. Los modelos celulares se encargan de modelar el comportamiento interno de las células y, para ello, utilizan expresiones matemáticas complejas que involucran operaciones computacionalmente costosas. Además, se utilizan ecuaciones diferenciales ordinarias (ODEs) para simular el carácter dinámico de las concentraciones iónicas así como las variables de compuerta que regulan la transferencia de iones a través de la membrana celular.

En la actualidad existen multitud de modelos celulares cuya utilización depende, fundamentalmente, del tipo de células a simular y de la capacidad computacional disponible. CellML [104] constituye un excelente repositorio de modelos celulares que incluye modelos relacionados con la electrofisiología cardiaca.

En los modelos celulares de ventrículo, existen aproximaciones sencillas como el de Beeler and Reuter (BR), del año 1977 [7], que describe el potencial de acción ventricular de mamíferos y donde

únicamente se consideran 4 corrientes individuales. Posteriormente, en 1991 Ching-Hsing Luo y Yoram Rudy realizaron una actualización del BR, dando lugar al modelo conocido como Luo-Rudy Fase I [9]. En 1994, este modelo fue complementado con una descripción matemática de los procesos que regulan la concentración de calcio intracelular y el movimiento de los iones de calcio, a través de la membrana celular, dando lugar al modelo conocido como Luo-Rudy Fase II [10]. La versión más reciente del modelo de Luo-Rudy es del año 2000 [11]. Por otra parte, el modelo DiFrancesco-Noble [105] permite modelar las células correspondientes a las fibras de Purkinje. Con respecto a los modelos celulares de aurícula, uno de los más completos es el de Nygren [106], que presenta un modelo matemático de las respuestas electrofisiológicas en la aurícula humana.

Todos estos modelos celulares han sido implementados dentro del sistema de simulación CAMAEC. Además, una importante ventaja de este sistema es que permite la realización de simulaciones no homogéneas que combinen la utilización de células de diferentes modelos. Esto permite simular, por ejemplo, un tejido que incluya células ventriculares y de Purkinje permitiendo así la modelización de la parte conductiva de los estímulos nerviosos por el tejido cardiaco.

En las siguientes secciones se detallan algunos aspectos sobre la implementación de los modelos celulares, con el objetivo de conseguir su ejecución eficiente.

5.3.1. Aceleración de Modelos: Tablas de Lookup

Las funciones de activación de las compuertas, en los modelos celulares iónicos, requieren el cálculo de expresiones matemáticas que únicamente dependen del potencial de membrana y que involucran el uso de la función exponencial. Por ejemplo, la Ecuación 5.4 muestra una de las funciones de activación de la compuerta m, encargada de regular el paso de iones de Sodio, para el modelo celular Luo-Rudy Fase II.

$$\alpha_m(V_m) = \frac{0{,}32 * (V_m + 47{,}13)}{1{,}0 - e^{-0{,}1*(V_m+47{,}13)}} \tag{5.4}$$

Es conocido que, en un computador, no todas las operaciones matemáticas requieren el mismo tiempo de cómputo. De hecho, entre las operaciones más costosas se encuentra el cálculo de la función exponencial, sobre todo para exponentes reales [107], que es precisamente el tipo de operación utilizada por las funciones de activación.

Un modelo celular complejo, como el Luo-Rudy Fase II, involucra 6 compuertas y, cada una de ellas, dos funciones de activación. En cada paso del proceso de simulación se realiza la actualización del estado de las células del tejido y, por lo tanto, el cálculo de estas expresiones supone una parte importante del tiempo de ejecución del modelo celular. En este sentido, una reducción en el tiempo de cómputo de estas expresiones tiene un impacto directo en el tiempo de ejecución global de la simulación.

En la actualidad se han desarrollado técnicas para la aceleración del cálculo de la operación exponencial, como la propuesta por Schraudolph [108], que combina de forma ingeniosa el uso de estructuras en lenguaje C y el conocimiento de la representación interna de los números de doble precisión en la máquina (formato IEEE-754), para calcular rápidamente la exponencial de un número real. De esta manera, se obtiene un modelo de la función exponencial, con una precisión aceptable, pero significativamente más rápida que invocando a la operación `exp` disponible en la librería matemática de cualquier sistema Linux.

Sin embargo, para este caso, dado que las funciones de activación únicamente dependen del potencial de membrana, es posible aplicar el concepto de las tablas de búsqueda (del inglés Lookup Tables). Las tablas de búsqueda permiten transformar rápidamente un determinado valor en otro, cuyo cálculo en tiempo de ejecución sería temporalmente costoso. Este concepto es muy conocido en el campo de la informática gráfica aplicada al procesamiento de imágenes, sobre todo para la implementación de mapas de colores.

Para su utilización en los modelos celulares, se ha implementado una tabla de búsqueda para cada una de las expresiones matemáticas involucradas en cada función de activación. Esto permite precalcular el valor de las variables dentro de un rango de potenciales de membrana. Se ha estimado experimentalmente que, en condiciones normales, el potencial de membrana de una célula está comprendido entre el rango de valores [-90, 50] mV. Por lo tanto, las tablas de búsqueda han sido diseñadas para cubrir este rango, con un equiespaciado de 0.1 mV. Además, se utiliza interpolación lineal entre los dos puntos más cercanos al solicitado lo que ofrece una precisión adicional a un coste muy bajo. Con este espaciado y, empleando interpolación lineal, se ha comprobado experimentalmente que las soluciones globales de la simulación son precisas hasta el tercer decimal.

Para el modelo Luo-Rudy Fase II se utilizan 12 tablas de búsqueda, para los parámetros α y β de cada una de las 6 compuertas, ocupando 131.25 Kbytes de memoria RAM. Las ejecuciones realizadas para diferentes simulaciones indican que la ejecución global del modelo celular resulta, en promedio, un 33 % más rápida debida al uso de las tablas de búsqueda, lo que supone una ventaja substancial frente al cómputo explícito de las expresiones.

5.4. Integración de Ecuaciones Diferenciales Ordinarias

Uno de los principales problemas que introducen los modelos celulares complejos, como el Luo-Rudy Fase I y, en mayor medida, el Luo-Rudy Fase II, consiste en los problemas de estabilidad numérica que acontecen cuando el paso de tiempo de integración se aumenta por encima de cierto umbral. Es conocido que este problema se debe a la rigidez de las ODEs de los modelos celulares [23, 109]. Esta limitación fuerza a reducir el paso global de simulación, para mantener la estabilidad de las ODEs durante la integración del modelo celular.

Uno de los métodos de integración más utilizado en este tipo de simulaciones es el de Euler explícito debido a su facilidad de implementación. Sin embargo, se ha comprobado experimentalmente que este método no permite aumentar el paso de tiempo de manera estable por encima de unos 10 μs. Por lo tanto, resulta importante utilizar métodos de integración estables que permitan aumentar el paso de tiempo de simulación. En efecto, si se consigue duplicar este valor el tiempo de ejecución se reduce a la mitad, puesto que el número de pasos globales de simulación se reduce en esa proporción. Dado que el modelo Luo-Rudy Fase II es el más utilizado en el marco de esta tesis, los estudios se han realizado con este modelo. Sin embargo, es posible extrapolar los resultados obtenidos a otros modelos celulares.

El modelo celular Luo-Rudy Fase II es un modelo encargado de representar el comportamiento dinámico del potencial de acción ventricular en base a corrientes y concentraciones iónicas. El modelo precisa cerca de 60 variables de doble precisión para almacenar el estado de una célula, definiendo más de 20 corrientes iónicas intermedias. Básicamente, este modelo se encarga de realizar tres tareas principales:

1. Actualiza las compuertas encargadas de definir la permeabilidad de la membrana celular a los diferentes iones (Na^+, Ca^{2+}, K^+, etc), mediante 6 ecuaciones diferenciales ordinarias.

2. Actualiza las concentraciones iónicas, empleando 8 ecuaciones diferenciales ordinarias.

3. Actualiza el valor de las corrientes iónicas, de acuerdo a los valores obtenidos en los puntos anteriores.

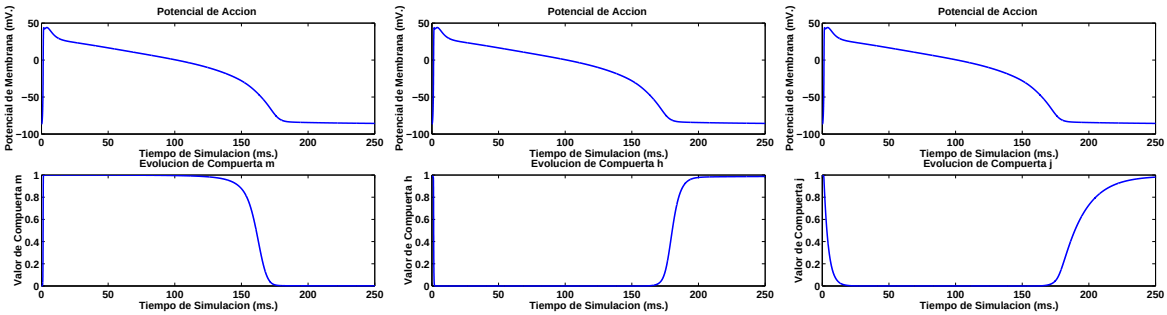
Las ODEs encargadas de integrar las compuertas se rigen de acuerdo a la siguiente expresión:

$$\frac{dy}{dt} = \alpha_y(V_m) \cdot (1 - y) - \beta_y(V_m) \cdot y \tag{5.5}$$

donde los parámetros α_y y β_y son las funciones de activación que dependen exclusivamente de la compuerta y integrada y del potencial de membrana de la célula, en el momento de realizar su actualización. Por otro lado, las ODEs encargadas de integrar las concentraciones iónicas vienen dadas por la siguiente expresión:

$$\frac{d[B]}{dt} = \frac{-I_B \cdot A_{cap}}{V_c \cdot z_B \cdot F} \tag{5.6}$$

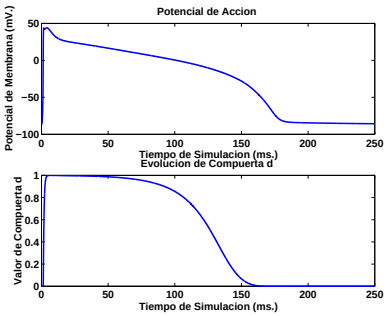
donde $[B]$ es la concentración del ion B, I_B es la suma de las corrientes iónicas que transportan el ion B, A_{cap} es el área de la membrana capacitiva, V_c es el volumen del compartimiento donde se está actualizando $[B]$, z_B es la valencia del ion B y F es la constante de Faraday.



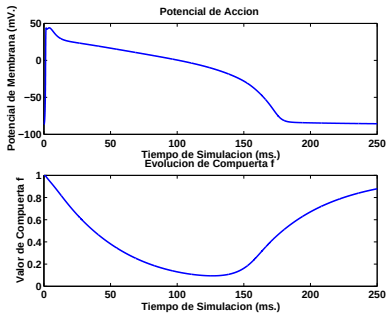
(a) Compuerta m

(b) Compuerta h

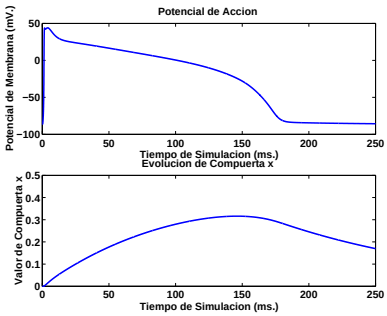
(c) Compuerta j



(d) Compuerta d



(e) Compuerta f



(f) Compuerta x

Figura 5-11: Evolución de las compuertas m, h, j, d, f y x a lo largo de un potencial de acción de 250 ms para el modelo Luo-Rudy Fase II.

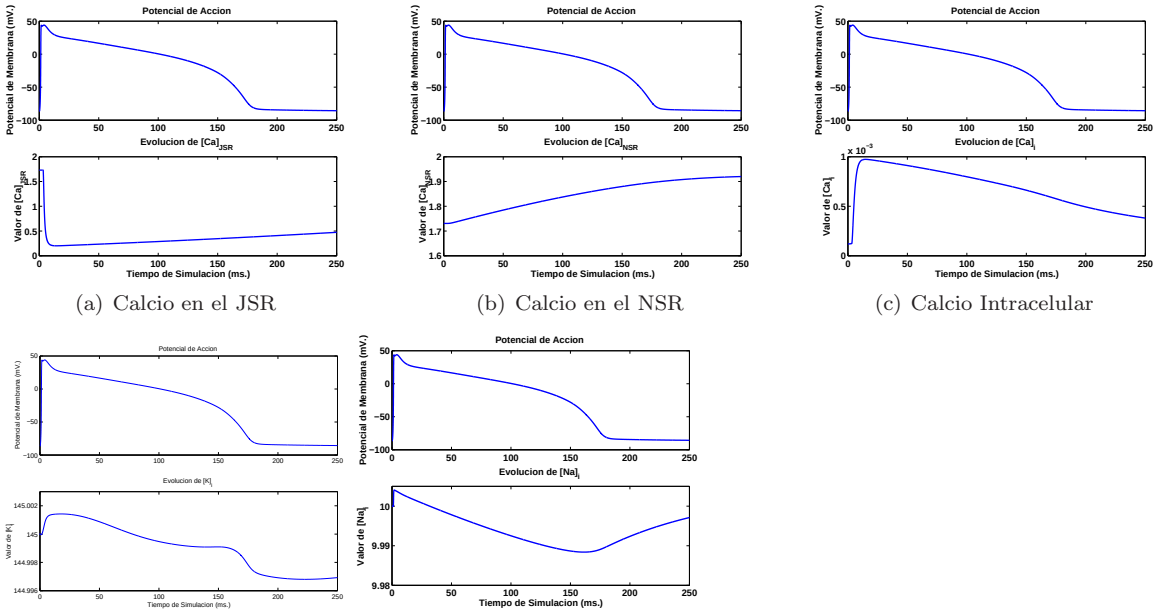
5.4.1. Comportamiento de las Variables a Integrar

En este apartado se muestra gráficamente el comportamiento que presentan las variables a integrar a lo largo de un potencial de acción. Para la realización de las gráficas se ha empleado el método de integración de Euler explícito con un paso de tiempo de 8 μs. Este paso es suficientemente pequeño para garantizar la estabilidad del método y la corrección de la solución obtenida.

Variables de Compuerta

Las variables de compuerta, como se ha comentado anteriormente, definen la permeabilidad de la membrana frente a ciertos iones. Por ejemplo, la compuerta m (Figura 5-11.a) controla el paso de iones de Sodio (Na^+) a través de la membrana celular. El valor que toma la variable m indica el porcentaje de apertura de dicha compuerta. Un valor de 0 impide el paso de iones de Sodio a través de la membrana celular, es decir que la compuerta está totalmente cerrada, mientras que un valor de 1 permite la libre circulación de iones de Sodio a través de la membrana, es decir que la compuerta está totalmente abierta.

A la vista de las gráficas se observa que las compuertas presentan escalas de tiempo muy cortas, donde se produce una rápida variación en los valores. Por ejemplo, la compuerta m presenta una subida muy brusca cuando comienza el potencial de acción, que puede llegar a comprometer la



(a) Calcio en el JSR (b) Calcio en el NSR (c) Calcio Intracelular

(d) Potasio Intracelular (e) Sodio Intracelular

Figura 5-12: Evolución de las concentraciones durante un potencial de acción.

estabilidad de un método de integración si se emplea un paso de tiempo elevado.

Concentraciones Iónicas

La Figura 5-12 muestra las principales concentraciones iónicas del modelo celular Luo-Rudy Fase II. Se han omitido las gráficas correspondientes a la concentración de sodio intersticial ($[Na]_c$), la concentración de calcio intersticial ($[Ca]_c$) y la concentración de potasio intersticial ($[K]_c$), por permanecer casi constantes durante todo el potencial de acción a los valores 140, 1.8 y 5.4 mM respectivamente.

La variación de valores para las concentraciones presenta, por lo general, escalas de tiempo bastante más largas, comparadas con las de las compuertas. Además, el rango de valores que toman las concentraciones es bastante acotado y para algunas de ellas, como el potasio intracelular, el cambio es muy leve. Quizá lo más destacable es la rápida variación en la concentración de calcio intracelular al comienzo del potencial de acción. El hecho de que las concentraciones presenten escalas de tiempo más largas hace que el problema se centre, fundamentalmente, en controlar la estabilidad en las ecuaciones diferenciales ordinarias con escalas de tiempo más rápidas, es decir, las asociadas a las compuertas.

5.4.2. Métodos de Integración Empleados

En este apartado se describen los métodos empleados para realizar la integración de las ODEs de las compuertas.

Método Explícito de Euler

El método de Euler explícito tiene como objetivo obtener la solución aproximada al problema de valores iniciales:

$$\frac{dy}{dt} = f(t, y) \tag{5.7}$$

donde $y(t_0) = y_0$ y $t_0, y_0 \in R^N$. Si denotamos el tiempo en el paso *n-ésimo* como t_n y la solución calculada en ese mismo paso como y_n, es decir que $y_n = y(t = t_n)$, y asumimos un paso de tiempo constante $h = t_n - t_{n-1}$, entonces dado un par (t_n, y_n), el método explícito de Euler calcula y_{n+1} como:

$$y_{n+1} = y_n + h \cdot f(y_n, t_n) \tag{5.8}$$

El método de Euler explícito está basado en una expansión en serie de Taylor truncada donde se han eliminado los términos de segundo orden en adelante. Por lo tanto, se está cometiendo un error en el cálculo de la solución, para cada paso de tiempo, que, además, se acumula de una etapa a la siguiente. Para este método, el error local de truncamiento es de $O(h^2)$. Este error define al método como una técnica de primer orden ya que, generalmente, un método con un error local de truncamiento de $O(h^{k+1})$ se dice que es de orden k.

La principal ventaja del método de Euler es su facilidad de implementación y su bajo coste, ya que únicamente precisa de una evaluación de la función a integrar y realizar un par de operaciones aritméticas para obtener el nuevo valor de la función. Sin embargo, para este problema, el método de Euler explícito deja de comportarse de manera estable cuando el paso de tiempo de integración aumenta por encima de los 10-11 μs.

Métodos de Runge-Kutta

Los métodos de Runge-Kutta suponen una mejora frente al método explícito de Euler ya que presentan un menor error de truncamiento. El método de Runge-Kutta de orden 2 calcula un paso de prueba en el instante intermedio del paso de integración de manera que, el nuevo valor, se calcula mediante los siguientes pasos:

$$\begin{aligned} k_1 &= h \cdot f(t_n, y_n) \\ k_2 &= h \cdot f(t_n + \frac{1}{2} \cdot h, y_n + \frac{1}{2} \cdot k_1) \\ y_{n+1} &= y_n + k_2 \end{aligned} \tag{5.9}$$

Realizar este paso intermedio permite cancelar el error de primer orden, convirtiendo este proceso

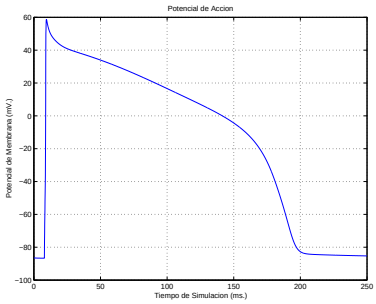


Figura 5-13: Potencial de Acción para una célula donde la integración de las ODEs se realiza mediante el método de Adams-Moulton, con un paso de tiempo de 80 μs.

en un método de segundo orden ($O(h^3)$). Se pueda observar en la Ecuación 5.9 que el coste viene dominado por la necesidad de evaluar 2 veces la función a integrar, siendo el doble de costoso que el método explícito de Euler. Este método, al ser de mayor orden que Euler explícito, permite pasos de tiempo de manera estable hasta 14 μs. Para ello se ha comparado la forma del potencial de acción con respecto a la solución considerada como correcta, obtenida con un paso de 8 μs.

El método de Runge-Kutta de orden 4 tiene una formulación análoga a la presentada para el método Runge-Kutta de orden 2 pero requiere 4 evaluaciones de la función a integrar. Sin embargo, las pruebas realizadas indican que la estabilidad ofrecida no es muy superior a la ofrecida por la versión de orden 2.

Adams-Moulton

Dada la poca estabilidad ofrecida por los métodos explícitos al aumentar el paso de tiempo global de simulación se han probado métodos implícitos desarrollados en el marco de la librería CVODE. Para ello, se ha empleado el método de Adams-Moulton. En este caso la Ecuación 4.2 se particulariza para los valores $K_1 = 1$ y $K_2 = q$, donde q representa el orden del método y suele tomar valores entre 1 y 12, dando lugar a la siguiente expresión:

$$\alpha_{n,0} \cdot y_n + \alpha_{n,1} \cdot y_{n-1} + h_n \sum_{i=0}^{K_2} \beta_{n,i} \cdot \dot{y}_{n-i} = 0. \tag{5.10}$$

donde $\alpha_{n,0} = -1$. Es posible particularizar la expresión para que el número de pasos sea 2 ($K_2 = 2$), dando lugar a la siguiente expresión.

$$y_n = \alpha_{n,1} \cdot y_{n-1} + h_n(\beta_{n,0} \cdot \dot{y}_n + \beta_{n,1} \cdot \dot{y}_{n-1} + \beta_{n,2} \cdot \dot{y}_{n-2}) = 0 \tag{5.11}$$

El método de Adams-Moulton es un método implícito, tal y como se puede ver en (5.11), y CVODE lo emplea para integrar problemas no rígidos. Mediante la aplicación de este método numérico es posible superar el límite impuesto por Euler explícito aumentando considerablemente el paso de

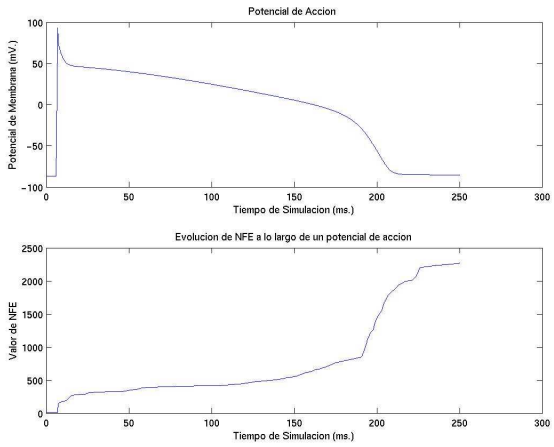


Figura 5-14: Número de evaluaciones de la función f a lo largo del potencial de acción. La gráfica muestra los valores acumulados a lo largo del tiempo. (dt=8 μs).

tiempo. La Figura 5-13 muestra el aspecto de un potencial de acción utilizando este método de integración con un paso de 80 μs, que es 10 veces mayor que el paso de tiempo original. Se puede observar que la forma del potencial de acción queda respetada. Si se aumenta todavía más el paso de tiempo, hasta 1000 μs, la librería CVODE indica que ha tenido que superar el número máximo de pasos internos de integración antes de poder integrar el intervalo especificado. En realidad CVODE utiliza este método para problemas no rígidos, cuando este problema es claramente rígido, por lo que la utilización de métodos de integración más sofisticados pueden aportar una mejora significativa.

La Figura 5-14 muestra el número de evaluaciones acumulado de la función f a lo largo de un potencial de acción. Se observa que en la fase de repolarización se precisa mayor número de evaluaciones de la función, ya que la pendiente es acusada durante un gran intervalo de tiempo.

Observando las estadísticas finales mostradas por CVODE, se han precisado 1582 pasos de tiempo de integración y 2266 evaluaciones de la función, para la integración de un potencial de acción durante 250 ms.

Backward Differentiation Formula (BDF).

La librería CVODE proporciona una implementación del método BDF, donde la Ecuación 4.2 se particulariza para los valores $K_1 = q$ y $K_2 = 0$, siendo q el orden del método, que suele variar entre 1 y 5. Es posible particularizar esa ecuación para el caso de orden 2, obteniendo la siguiente expresión:

$$y_n = \alpha_{n,1} \cdot y_{n-1} + \alpha_{n,2} \cdot y_{n-2} + h_n(\beta_{n,0} \cdot \dot{y}_n) \tag{5.12}$$

CVODE emplea este método para integrar problemas rígidos, por lo que es de esperar que permita

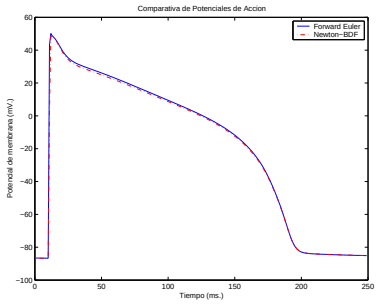
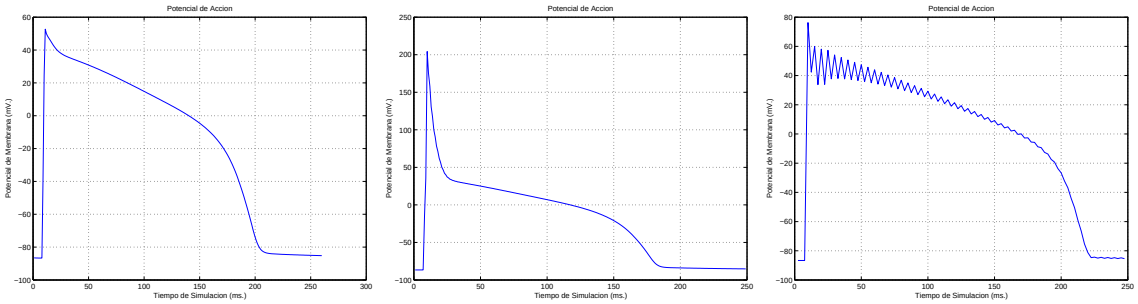


Figura 5-15: Comparativa de potencial de acción entre Euler explícito y BDF para un paso de tiempo pequeño.



(a) dt = 80 us (b) dt = 1000 us (c) dt = 2500 us

Figura 5-16: Potencial de acción para una célula con diferentes pasos de tiempo (μs). Integración mediante BDF.

utilizar pasos de tiempo mayores. La Figura 5-15 muestra dos potenciales de acción calculados con un paso de tiempo pequeño, de 8 μs, para el método de Euler explícito y el método BDF iterando mediante Newton. Se observa que la forma del potencial de acción es casi idéntica, lo que indica que el método de integración está funcionando correctamente. Hay que tener en cuenta que, aunque se produzcan diferencias numéricas entre ambos métodos, interesa que la forma del potencial de acción sea similar y que el método empleado se comporte de manera estable, permitiendo pasos de tiempo por encima del límite impuesto por Euler explícito.

En la Figura 5-16 se muestra el comportamiento del método al aumentar el paso de tiempo. Se observa que el método es estable, incluso con pasos de tiempo muy grandes (1 ms en la Figura 5-16.b), aunque en este caso la solución obtenida se deforma bastante con respecto a la solución considerada correcta. Para un paso de tiempo de 80 μs, el potencial de acción tiene la forma esperada. Si se fuerza el método de integración con un paso de tiempo de 2500 μs, el método comienza a sufrir problemas de estabilidad.

Es importante destacar que, conforme aumenta el paso de tiempo la solución obtenida será menos precisa. Sin embargo, es importante que el método sea estable para que el usuario pueda decidir entre una simulación rápida, con menor precisión, frente a una solución detallada que involucre horas de cálculo empleando un paso de tiempo más pequeño.

Las estadísticas finales apuntan que, para realizar la integración de todo un potencial de acción, durante 250 ms con un paso de tiempo de 8 μs, se precisaron 1873 pasos de integración y 2452 evaluaciones de la función. Estas cifras apuntan a que el método BDF es más costoso que el de Adams-Moulton, aunque ofrece una mayor estabilidad.

5.4.3. El Método de Euler Implícito

El método de Euler implícito viene dado por la siguiente expresión:

$$y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1}) \tag{5.13}$$

Se observa que el valor de y_{n+1} se define de manera implícita en términos de una función de y_{n+1}. Generalmente, la aplicación de un método implícito lleva asociado la resolución de un posterior sistema de ecuaciones, aumentando así el coste de cada paso de integración. Sin embargo, la buena estabilidad de un método implícito puede llegar a compensar el sobrecoste, ya que permite el empleo de pasos de tiempos mayores.

La ecuación diferencial ordinaria asociada a las compuertas viene dada por la Ecuación 5.5, cuya notación simplificada puede expresarse de la siguiente manera:

$$\frac{dy}{dt} = (1 - y)\alpha_y - y\beta_y \tag{5.14}$$

Es posible reescribir la Ecuación 5.14 de la siguiente manera:

$$\frac{dy}{dt} = f(t, y) = \alpha_y - y(\alpha_y + \beta_y) \tag{5.15}$$

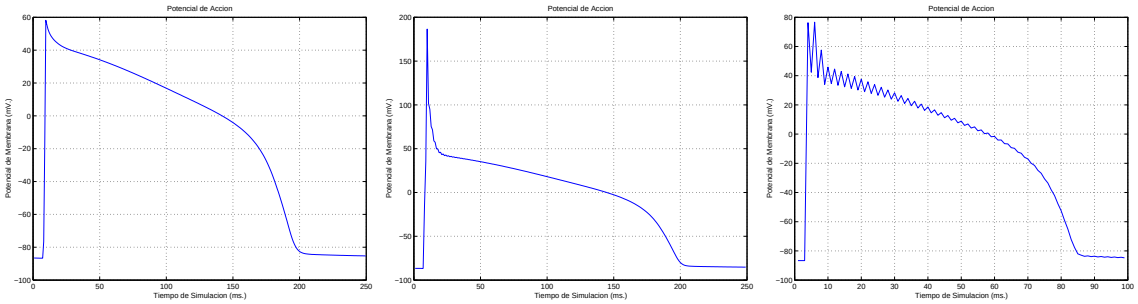
por lo que, si se aplica la expresión de Euler implícito, se obtiene la siguiente expresión:

$$y_{n+1} = y_n + h \cdot (\alpha_y - y_{n+1}(\alpha_y + \beta_y) \tag{5.16}$$

En la Ecuación 5.16 es posible despejar el termino y_{n+1}, en función de y_n, dando lugar a la Ecuación 5.17 que define la aplicación del método de Euler implícito para integrar las ODEs de las compuertas.

$$y_{n+1} = \frac{y_n + h\alpha}{1 + h(\alpha + \beta)} \tag{5.17}$$

La Ecuación 5.17 muestra la sencilez de la expresión de integración de las compuertas y, aunque se trate de un método implícito, no requiere la resolución de un sistema de ecuaciones. De hecho, únicamente se precisa una evaluación de la función, lo que implica un coste computacional comparable al método de Euler explícito. En la Figura 5-17 se muestra el comportamiento del método



(a) dt = 80 us (b) dt = 1000 us (c) dt = 2500 us

Figura 5-17: Potencial de acción de una célula para diferentes pasos de tiempo. Integración mediante Euler implícito.

para diferentes pasos de tiempo. Resulta interesante observar que el método de Euler Implícito tiene un comportamiento muy similar al obtenido por BDF. Los límites de estabilidad en ambos casos son muy similares, donde un paso de tiempo por encima de 2.5 milisegundos comienza a alterar completamente la forma del potencial de acción.

De todos los métodos utilizados, el método de Euler implícito es el que ofrece mayor número de ventajas. En primer lugar, combina la estabilidad ofrecida por un método implícito, con el bajo coste computacional de un método explícito, permitiendo el empleo de pasos de simulación más grandes que los permitidos por Euler explícito.

En efecto, el hecho de que las ODEs de compuerta estén desacopladas ha permitido que la aplicación del método de Euler implícito no requiera la resolución de un sistema de ecuaciones por lo que el coste del método se reduce en gran medida. Para evaluar la expresión únicamente se requieren 6 operaciones aritméticas además del coste requerido para evaluar los parámetros α y β, que es un coste común para cualquiera de los métodos de integración.

Validación de Soluciones

La utilización de un método implícito ha permitido aumentar el paso de tiempo de manera estable, aunque esto no implica que se realice sin introducir errores en la solución. Por ello es importante comprobar que los resultados obtenidos son electrofisiológicamente correctos.

Para validar las soluciones, se han realizado una serie de pruebas comparando el potencial de acción obtenido empleando el método de Euler explícito con un paso de tiempo pequeño (8 μs), considerado como la solución correcta, con el potencial de acción obtenido mediante el método de Euler implícito con un paso de tiempo más elevado.

No existe ninguna consenso sobre cual es el máximo paso de tiempo que se puede emplear para capturar tanto los acontecimientos que subyacen en la célula como las características de propagación del potencial de acción en el tejido. En la tesis de Hooke [109] el paso de tiempo variable que emplea permite alcanzar los 0.15 ms. Pormann [18] también emplea un paso de tiempo adaptativo que varía

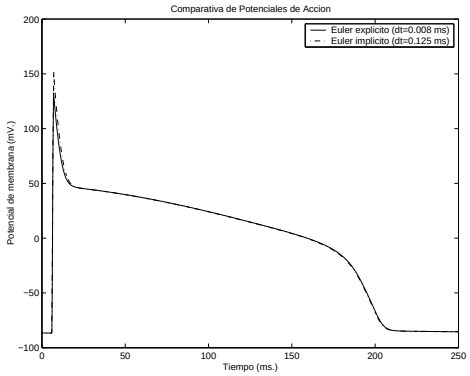


Figura 5-18: Potencial de acción para el método de Euler explícito y Euler implícito. Tejido de 1x1.

dentro del rango [0.00001, 0.1] ms. Finalmente, en la tesis de Lines [23] se apunta a que un valor de 0.125 ms permite representar adecuadamente los comportamientos internos de la célula.

Consideremos una simulación de un potencial de acción completo durante 250 ms. Si se emplea un paso de tiempo de 8 μs se deberán realizar 31250 pasos de simulación y, utilizando un paso de tiempo de 125 μs, se deberán realizar 2000 pasos de simulación. Debido al coste ofrecido por el método de Euler implícito, es posible afirmar que la simulación cardiaca realizada utilizando el método implícito se ejecutará 15 veces más rápida, al precisar menos pasos de tiempo. Este factor de velocidad deberá ser tenido en cuenta a la hora de comparar la solución obtenida por el método.

Para ello se ha realizado una simulación de potencial de acción sobre un tejido 1x1 inyectando un estímulo de corriente en el intervalo de simulación [6,7] ms. La Figura 5-18 muestra una comparativa del potencial de acción obtenido con dos simulaciones. En la primera de ellas se emplea el método de Euler explícito con un paso de tiempo de 8 μs y en la otra se emplea el método de Euler implícito con un paso de tiempo de 125 μs. Cualitativamente, ambos potenciales de acción son muy semejantes. La única diferencia apreciable aparece en el pico del potencial de acción, donde para el método implícito aumenta unos mV más por encima de la solución correcta. Sin embargo, esa fase únicamente corresponde a la fase de depolarización del potencial de acción, y, durante el resto del potencial, se observa que la solución obtenida es prácticamente idéntica.

Considerando que el uso del método de Euler implícito para integrar las compuertas permite aumentar de manera estable el paso de tiempo hasta llegar a reducir en un factor de 15 el tiempo global de simulación, es posible afirmar que se ha obtenido una gran ventaja. No obstante, es responsabilidad del experto biomédico valorar el riesgo que supone aumentar el paso de tiempo, dado el impacto que puede tener en la solución obtenida.

Como comentario adicional, esta estrategia ha supuesto una ventaja substancial en un caso práctico como la detección de la ventana vulnerable en situaciones de isquemia. En este caso, es posible ejecutar de forma rápida diferentes simulaciones, con un paso de tiempo elevado, hasta detectar los límites de la ventana vulnerable donde se produce una reentrada. Posteriormente, se

confirman los límites de la ventana mediante simulaciones más precisas, con un paso de tiempo más pequeño, para contrastar los resultados. De esta manera, es posible ahorrar mucho tiempo en simulaciones costosas, lo que redunda en un aumento de la productividad investigadora.

5.5. Sistema de Grabación de Datos

La grabación de datos es una de las labores fundamentales del sistema de simulación. El principal objetivo de las simulaciones consiste en analizar la propagación eléctrica y el estado del tejido, bajo determinadas patologías cardiacas. Por lo tanto, resulta imprescindible obtener información periódica a lo largo del proceso de simulación.

En este tipo de simulaciones, el análisis de la información puede realizarse en múltiples niveles. Por ejemplo, el usuario puede querer observar la evolución del potencial de acción para detectar la presencia de patrones de propagación anómalos. Para ello, hay que almacenar el valor del potencial de membrana, de cada una de las células, a lo largo de la simulación. Una representación animada de esta información ofrece información visual muy valiosa para analizar dichos patrones. Por otra parte, un estudio concreto puede requerir el análisis detallado del comportamiento de una o varias características de una célula del tejido para estudiar su evolución a lo largo de la simulación.

Dado el amplio abanico de posibilidades de grabación, resulta imprescindible utilizar técnicas sofisticadas y versátiles que puedan gestionar esta complejidad. En este sentido, las técnicas de generación automática de código permiten la creación del código necesario para satisfacer los requisitos del usuario.

La idea principal radica en partir de una declaración en alto nivel de los requisitos de grabación del usuario y generar, de forma automática, el código fuente necesario para realizarla. De esta manera, se consiguen dos objetivos fundamentales. Por un lado, el mecanismo de grabación se vuelve versátil ya que la generación de código puede adecuarse a la expresividad de los requisitos del usuario. Por otro lado, la propia gestión de la información de grabación se optimiza, puesto que únicamente se ejecuta el código específico para los requisitos de grabación del usuario.

Para facilitar el proceso de generación de código, a partir de la información de grabación especificada por el usuario, se ha optado por emplear el formato de datos XML (eXtensible Markup Language) [110] para la descripción de la información de grabación. La elección de XML responde a varias razones: En primer lugar es un lenguaje en modo texto que puede ser fácilmente procesado por humanos ya que está basado en etiquetas. En segundo lugar, es fácilmente procesable de manera programática por computadores. Finalmente, XML es un estándar del W3C (World Wide Web Consortium) lo que facilita la interoperabilidad y el intercambio de información entre diferentes usuarios.

La Figura 5-19 muestra un ejemplo de documento que define la información de grabación de

```
<?xml version="1.0" encoding="UTF-8"?>
<Grabacion habilitada="si" >
        <Tejido ncel_l="200" ncel_t="1" tipo_cel="celula_LRII_t" />
        <Parametros>
                <Iniciograbacion>  0   </Iniciograbacion>
                <Fingrabacion>  90000  </Fingrabacion>
        </Parametros>
        <Quegrabar>
                <Celula x="0" y="0">
                        <Caracteristica id="Vm"/>
                </Celula>
                <Celula x="99" y="0">
                        <Caracteristica id="jNaCa"/>
                </Celula>
                <Celula x="109" y="0">
                        <Caracteristica id="Vm"/>
                </Celula>
        </Quegrabar>
  <Mallas>
     <Malla id="Vm" bytescalar="no"/>
  </Mallas>
</Grabacion>
```

Figura 5-19: Ejemplo de documento XML con información de grabación.

datos. Se observa que el usuario puede definir una ventana temporal de grabación (*Iniciograbacion*, *Fingrabacion*), expresada en milisegundos. Actualmente, se pueden definir dos tipos de información de grabación:

- Instantáneas de Estado del Tejido. Se definen en la sección *Mallas* y permiten obtener un fichero con la evolución de una característica determinada, para todas las células del tejido, a lo largo de la ventana de grabación. Se suelen utilizar para la generación de vídeos de propagación del potencial de acción en el tejido. Un ejemplo de nombrado de fichero generado por dicha definición es `Malla_Vm.out`.

- Evolución de Características. Se definen en la sección *Quegrabar* y permiten obtener un fichero con la evolución de una característica determinada, de una célula concreta, a lo largo de la ventana de grabación. Se utiliza para analizar con detalle el comportamiento interno de algunas células. Un ejemplo de nombrado de fichero generado por dicha definición es `jNaCa_99-0.out`. Se observa que el nombre del fichero incluye, además de la característica, la posición de la célula en el tejido cardiaco.

Tal y como se vió en la Figura 5-1 el usuario especifica la información de grabación a través de un documento XML. Este documento es procesado por un generador automático de código de grabación que produce un fichero de código C. Este módulo de generación de código es una aplicación Java que utiliza la funcionalidad de la herramienta XMLWrappers, desarrollada por el doctorando en el marco de su proyecto final de carrera [111], para realizar enlace de datos con el fichero XML.

Brevemente, el objetivo principal de la herramienta XMLWrappers consiste en simplificar el acceso a la información de un documento XML desde un entorno de programación Java. Generalmente,

un documento XML está sujeto a las reglas de un esquema, que dicta la sintaxis y tipos de los valores que puede albergar un documento XML, que se considere instancia de ese esquema. Aunque existen varios tipos de esquemas, los más utilizados son los *W3C XML Schemas* [112].

La herramienta XMLWrappers, tras procesar el W3C XML Schema asociado a la información de grabación, genera de manera completamente automática un conjunto de clases Java que realizan funcionalidad de *wrapper*, o envoltorio software. Este envoltorio queda constituido como una capa entre la aplicación y el documento XML. De esta manera, tanto las consultas como las modificaciones realizadas al documento XML se llevan a cabo a través de los métodos que exportan las clases del wrapper a las aplicaciones. Estas clases se encargan de ocultar la complejidad del documento XML a las aplicaciones y, de esta manera, se simplifica en gran medida el acceso a la información XML del documento.

En primer lugar, este módulo se encarga de acceder al documento XML de grabación a través del wrapper. Posteriormente, procesa la información y genera el código de 4 rutinas que serán invocadas por el sistema CAMAEC para llevar a cabo la grabación de datos:

1. InicializaGrabacionDatos. Esta rutina se invoca una sola vez y permite la creación de los ficheros necesarios para albergar los datos de las características especificadas por el usuario. Además, reserva la memoria necesaria para los vectores auxiliares empleados en otras rutinas.

2. GrabaCaracteristicasCelula. Esta rutina se invoca en cada paso de simulación y permite detectar si el instante de tiempo actual está dentro de la ventana de grabación. De ser así, graba las características especificadas por el usuario.

3. GrabaMallasCaracteristicas. Esta rutina se invoca en cada paso de simulación y, al igual que la función anterior, detecta si el instante de tiempo actual está dentro de los límites de grabación. De ser así, cada procesador almacena en un vector auxiliar los valores de la característica para cada una de las células de su tejido. Posteriormente, esa información será grabada en disco. Si se han definido múltiples mallas, entonces se utilizarán múltiples vectores auxiliares.

4. LiberaGrabacionDatos. Esta rutina se invoca una sola vez con el objetivo de cerrar los descriptores de ficheros abiertos y liberar la memoria asociada a los vectores auxiliares.

Una vez emitido todo el código C en un fichero llamado `grabacarac.c`, este fichero es compilado, junto al resto de código del sistema de simulación, dando lugar a un ejecutable adaptado a las necesidades de grabación del usuario. De esta manera, es posible combinar una amplia expresividad y versatilidad, para la especificación de la información, con la eficiencia en la gestión del proceso de grabación.

Persistencia en las Simulaciones: Copias de Seguridad

Dado que el simulador desarrollado debe abordar simulaciones largas, que a menudo involucran varias horas e incluso días de ejecución, se ha implementado un sistema de copia de seguridad durante el proceso de simulación. El objetivo es guardar completamente el estado de la simulación cardiaca en disco, ya sea periódicamente o bajo petición del usuario, para poder retomar la simulación más tarde desde el instante en el que se realizó la grabación. Este mecanismo de *checkpoint* permite proteger parte de una simulación frente a fallos en el sistema de computación, cortes eléctricos y otras eventualidades.

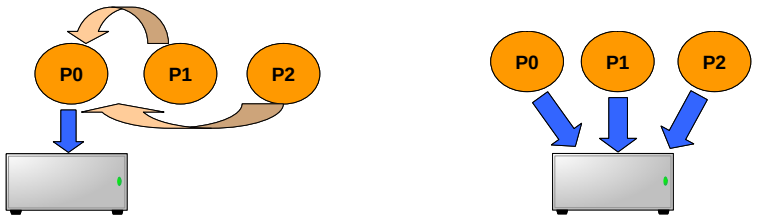
Adicionalmente, esta funcionalidad demuestra su utilidad para la realización de simulaciones que incluyen una parte común. Por ejemplo, a la hora de realizar búsquedas de ventana vulnerable bajo situaciones de isquemia, resulta necesario, en primer lugar, acometer un proceso de estabilización del tejido. Para ello se simulan unos cuantos milisegundos y, a continuación, se inyecta un primer estímulo eléctrico. Posteriormente, tras la espera de un determinado tiempo, que suele ser variable, se aplica un estímulo adicional. En este sentido, la simulación correspondiente a la primera fase (estabilización e inyección del primer estímulo) es común a todas las ejecuciones y, por lo tanto, puede ser realizada una sola vez. Por lo tanto, es posible aprovechar los ficheros de copia de seguridad para retomar diferentes simulaciones desde el mismo punto, en las que el segundo estímulo se aplicará en instantes de tiempo diferentes.

La información de copia de seguridad involucra tres ficheros:

1. Un fichero que almacena el vector de corrientes iónicas.
2. Un fichero que almacena el vector del potencial de membrana de todas las células del tejido.
3. Un fichero que almacena el estado del tejido, es decir, el valor de todos los parámetros que definen a una célula, para todas las células del tejido.

El almacenamiento de todo el estado del tejido se realiza mediante una estrategia bastante sofisticada. El tejido está internamente representado en el simulador como un vector de estructuras en C. El vector tiene un tamaño igual al número total de células del tejido y cada estructura almacena el estado de una célula cardiaca, es decir, todos los atributos definidos por el modelo celular al que pertenece dicha célula. Para poder diseñar un método de grabación genérico que permita guardar cualquier tipo de célula, independientemente del modelo celular, se utiliza una herramienta llamada AutoMap [113], que permite la generación automática de un tipo de datos MPI (*MPI_Datatype*) [114] a partir de cualquier estructura en lenguaje C. Este tipo de datos MPI posteriormente se utiliza para especificar la información que debe ser guardada en disco.

Por lo tanto, cualquier nuevo modelo que se introduce en el sistema requiere la utilización de dicha herramienta para generar automáticamente su tipo de datos MPI. Como medida básica de



(a) E/S Secuencial (b) E/S Paralela

Figura 5-20: Diferencia conceptual entre la E/S secuencial y la E/S paralela en la grabación de datos en disco.

detección de errores, al final de cada fichero se almacena un número mágico que permite averiguar si los datos se han grabado de manera correcta o si, por el contrario, la grabación de los datos fue abortada abruptamente, dando lugar a un fichero de datos corrupto.

La estrategia utilizada para generar los ficheros de copia de seguridad implica que el código de grabación es independiente del modelo celular ya que únicamente depende de su tipo de datos MPI asociado. Por lo tanto, añadir un nuevo modelo celular permite utilizar de manera automática las características de copia de seguridad sin programar ni una sola línea de código adicional. Este aspecto es fundamental para los usuarios de CAMAEC ya que facilita la inclusión de nuevos modelos celulares, obviando la implementación de código dedicado a esta funcionalidad.

Además, con el objetivo de realizar el proceso de grabación de manera eficiente en disco se han empleado técnicas de Entrada/Salida paralela, soportadas por el estándar MPI-2 I/O [115].

5.5.1. Grabación de Datos Mediante E/S Paralela

El proceso de E/S requiere almacenar los datos en un disco duro, cuyo tiempo de acceso es mucho mayor que el de la memoria RAM. Por lo tanto, resulta importante que el proceso de grabación de datos sea realizado de forma eficiente para evitar afectar demasiado a las prestaciones del sistema de simulación.

En este sentido, se ha optado por un esquema de E/S paralela. La Figura 5-20 muestra las diferencias conceptuales entre la E/S secuencial y la paralela. En el primer caso (Figura 5-20.a), un solo proceso es el encargado de acceder al sistema de archivos. Por lo tanto, el resto de procesos deben enviarle los datos que desean grabar y, este proceso, se encarga de guardarlos en disco de manera ordenada. Realizar una lectura de datos requiere un proceso análogo donde solo uno de los procesos es el encargado de leer todos los datos y, posteriormente, realizar una distribución de los mismos al resto.

Este esquema de E/S secuencial presenta algunos inconvenientes. Por un lado, disponer de un solo proceso encargado de acceder al disco implica que debe tener suficiente espacio en memoria para albergar los datos, tanto del resto de procesos como los suyos, antes de guardarlos de manera

positioning	synchronism	coordination	
		noncollective	*collective*
explicit offsets	*blocking*	MPI_FILE_READ_AT MPI_FILE_WRITE_AT	MPI_FILE_READ_AT_ALL MPI_FILE_WRITE_AT_ALL
	nonblocking & split collective	MPI_FILE_IREAD_AT MPI_FILE_IWRITE_AT	MPI_FILE_READ_AT_ALL_BEGIN MPI_FILE_READ_AT_ALL_END MPI_FILE_WRITE_AT_ALL_BEGIN MPI_FILE_WRITE_AT_ALL_END
individual file pointers	*blocking*	MPI_FILE_READ MPI_FILE_WRITE	MPI_FILE_READ_ALL MPI_FILE_WRITE_ALL
	nonblocking & split collective	MPI_FILE_IREAD MPI_FILE_IWRITE	MPI_FILE_READ_ALL_BEGIN MPI_FILE_READ_ALL_END MPI_FILE_WRITE_ALL_BEGIN MPI_FILE_WRITE_ALL_END
shared file pointer	*blocking*	MPI_FILE_READ_SHARED MPI_FILE_WRITE_SHARED	MPI_FILE_READ_ORDERED MPI_FILE_WRITE_ORDERED
	nonblocking & split collective	MPI_FILE_IREAD_SHARED MPI_FILE_IWRITE_SHARED	MPI_FILE_READ_ORDERED_BEGIN MPI_FILE_READ_ORDERED_END MPI_FILE_WRITE_ORDERED_BEGIN MPI_FILE_WRITE_ORDERED_END

Figura 5-21: Rutinas que ofrece el estándar MPI-2 I/O para realizar E/S paralela.

ordenada en el sistema de archivos. En el caso del simulador CAMAEC, cada proceso puede necesitar almacenar varios centenares de Mbytes dependiendo del tamaño del tejido. Por lo tanto, el hecho de enviar todos los datos a un solo proceso puede superar su memoria disponible. Por otro lado, hay que considerar el tiempo invertido en realizar el envío de los datos al proceso encargado de guardarlos en disco, lo que aumenta inevitablemente el tiempo de la operación.

La utilización de un esquema de acceso al disco basado en E/S paralela (Figura 5-20.b) permite que cada proceso mantenga su propia copia de los datos a almacenar y que todos los procesos participen de manera cooperativa en el acceso al disco. En ese sentido, el estándar MPI-2 [115] surge en 1997 como complemento al estándar existente, MPI 1.1, del año 1995. La idea principal de este nuevo estándar fue ampliar su funcionalidad, tanto para los usuarios como para los desarrolladores de librerías. Entre otras características en él se define MPI-2 I/O, un estándar para E/S paralela. Al igual que MPI, MPI-2 es únicamente un estándar y, por lo tanto, para poder emplear su funcionalidad, es preciso utilizar alguna de las implementaciones existentes. Para ello se ha elegido la librería ROMIO [116, 117], que ofrece una implementación casi completa de toda la parte de E/S del estándar MPI-2.

En MPI-2 I/O los datos se mueven entre los procesos y los ficheros mediante llamadas de lectura y escritura. Hay tres aspectos que definen el acceso a los datos en disco:

- Posicionamiento. Utilización de un desplazamiento positivo en las rutinas de acceso o empleo de un puntero implícito a fichero.
- Sincronismo. Llamadas bloqueantes, que devuelven el control una vez finalizadas, o no bloqueantes.
- Coordinación. Llamadas colectivas, que involucran a todos los procesos, o no colectivas.

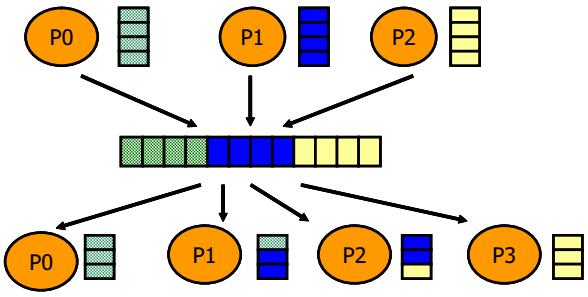


Figura 5-22: Lectura y escritura de datos de manera independiente del número de procesadores mediante las facilidades de MPI-2 I/O.

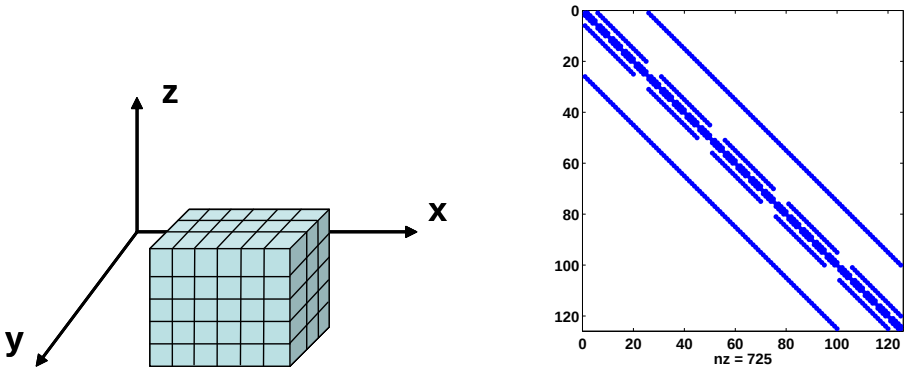
La combinación de estas tres características da lugar a diferentes formas de acceso a un fichero. Para cada una de ellas, MPI-2 I/O ofrece una rutina diferente, resumidas en la Figura 5-21.

Para las simulaciones cardiacas es necesario que el patrón de grabación de los datos escritos en disco sea independiente del número de procesadores que lo ha generado. Esta capacidad permite, por ejemplo, retomar una simulación con un número diferente de procesadores al que generó los ficheros de copia de seguridad. En realidad es una situación muy habitual, tanto en un cluster de PCs como en un despliegue Grid, que los recursos disponibles vayan variando a lo largo del tiempo debido al comportamiento y al envío de trabajos por parte de otros usuarios. Por lo tanto, resulta interesante permitir que una simulación, que en un momento dado estaba ejecutándose con un cierto número de procesadores, continue su ejecución en otro recurso con un número diferente de procesadores. Esta versatilidad permite adaptar la aplicación a la variabilidad del entorno de ejecución.

En este sentido, el simulador CAMAEC cumple esta característica y todos los ficheros que genera son independientes del número de procesadores. Este hecho provoca situaciones como la que se muestra en la Figura 5-22, donde un número de procesadores escribe información en un fichero y otro diferente la lee posteriormente.

Para guardar tanto el vector de potencial de membrana como el de corrientes iónicas, en la generación de los ficheros de copia de seguridad, se utiliza la rutina *MPI_File_write_ordered*. Esta rutina almacena en disco el contenido de un vector distribuido entre diferentes procesadores. Análogamente, se emplea la función *MPI_File_read_ordered* para la lectura de los datos de esos ficheros, obteniendo la distribución correcta en todos los procesadores.

La escritura del estado del tejido se lleva a cabo empleando la rutina *MPI_File_write_at* para guardar cada una de las células en un mismo fichero en disco. Esta rutina permite especificar el tipo de datos MPI empleado para describir la información a grabar, utilizando el generado previamente por la herramienta AutoMap, dependiendo del tipo de célula que a grabar. Análogamente, se utiliza la rutina *MPI_File_read_at* para poder leer de fichero el estado del tejido. El proceso de grabación de mallas y características celulares también utiliza la funcionalidad de MPI-2 I/O.



(a) Tejido 3D (b) Matriz de Coeficientes

Figura 5-23: Esquema de un tejido tridimensional con una geometría de paralelepípedo y la matriz de coeficientes del sistema de ecuaciones lineales asociado.

Como se puede observar el proceso de grabación de información combina la utilización de herramientas de generación automática de código con la funcionalidad avanzada de E/S paralela, ofrecida por el estándar MPI-2 I/O. Además, este procedimiento presenta una ventaja fundamental ya que al añadir un nuevo modelo celular al sistema no hay que implementar código especial de grabación de células puesto que únicamente se debe generar, de manera automática, un nuevo tipo de datos derivado MPI asociado al nuevo tipo de célula.

5.6. Extensión a Geometrías 3D

Aunque fuera de las líneas de investigación iniciales del proyecto CAMAEC, y con el objetivo de aumentar el abanico de problemas abordables, se ha extendido el sistema de simulación a geometrías tridimensionales sencillas basadas en un paralelepípedo, tal y como se puede observar en la Figura 5-23.a. Al añadir un nuevo eje de coordenadas (eje z), la complejidad del problema aumenta ya que así lo hace el número de células necesarias para abarcar un determinado volumen de tejido.

La distribución de datos entre procesadores sigue un esquema análogo al caso bidimensional, realizando una asignación de bloques de células consecutivas, donde la numeración se realiza siguiendo los ejes x, y y z, en el orden especificado. De esta manera y, al igual que en el caso bidimensional, se consigue distribuir todas las estructuras de datos entre los procesadores.

En un tejido tridimensional cambia el número de vecinos por célula a considerar por el esquema de discretización basado en diferencias finitas. En el tejido bidimensional se consideran 5 vecinos por células (los dos vecinos en cada uno de los ejes x e y más la propia célula), dando lugar a una matriz pentadiagonal. En un tejido tridimensional se consideran 7 vecinos por célula (de nuevo dos vecinos en cada uno de los ejes más la propia célula), dando lugar a una matriz heptadiagonal, tal y como se puede observar en la Figura 5-23.b. Esta matriz presenta unas propiedades similares, en cuanto

al buen condicionamiento, a las descritas para la matriz del tejido bidimensional. En este sentido, la utilización de un método de resolución del sistema de ecuaciones basado en Gradiente Conjugado sin precondicionar sigue siendo válida.

Sin embargo, el aumento en el número de vecinos provocado por la nueva discretización provoca un aumento en el número de las comunicaciones en la fase de resolución del sistema de ecuaciones. Por lo tanto, cabe esperar una ligera disminución en las prestaciones del sistema de simulación utilizando esta aproximación.

5.7. Evaluación de Prestaciones del Sistema de Simulación

En esta sección se realiza un análisis de las prestaciones obtenidas por el sistema de simulación cardiaca desarrollado. Las ejecuciones involucran el modelo celular Luo-Rudy Fase II, el método del Gradiente Conjugado sin precondicionado y un paso de tiempo de 8 μs, salvo que se indique lo contrario. Las características detalladas de los recursos computacionales empleados se pueden encontrar en el Anexo A.

Para el análisis se utilizan las métricas de incremento de velocidad y eficiencia, habituales en la evaluación de las prestaciones de un sistema paralelo [118]. El incremento de velocidad o *speedup* teórico mide la efectividad de la paralelización realizada, mediante la comparación entre el tiempo del mejor algoritmo secuencial y el tiempo de las ejecuciones con diferente número de procesadores. Dado que, generalmente, se desconoce el mejor algoritmo secuencial, se suele obtener un estimador calculado mediante la expresión 5.18,

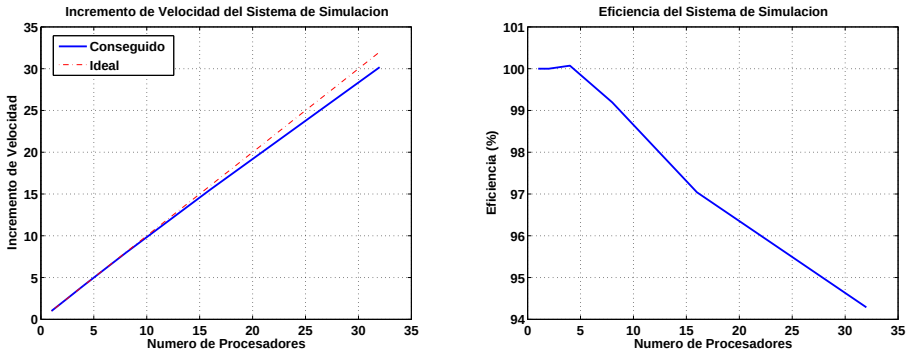
$$S_n = \frac{T_1}{T_n} \tag{5.18}$$

donde T_1 corresponde al tiempo de la ejecución realizada con 1 procesador y T_n es el tiempo de la ejecución con n procesadores. El máximo valor de incremento de velocidad alcanzable corresponde al número de procesadores (n), salvo si se obtiene un incremento de velocidad superlineal debido, por ejemplo, al efecto de las cachés en el tiempo de ejecución [119].

Análogamente, la eficiencia de un algoritmo paralelo es un indicador de la bondad de la paralelización que permite conocer el grado de utilización y aprovechamiento de los recursos computacionales de la máquina. Para ello, relaciona el incremento de velocidad alcanzado con el número de procesadores involucrado, mediante la expresión 5.19.

$$E_n = \frac{S_n}{n} \tag{5.19}$$

El máximo valor de eficiencia alcanzable es 1, correspondiente a un 100 %, indicando que todos los procesadores involucrados en la ejecución paralela están siendo plenamente utilizados. Obviamente,



(a) Incremento de Velocidad. (b) Eficiencia.

Figura 5-24: Incremento de velocidad y eficiencia del sistema de simulación. Tejido bidimensional de 1000x1000 células ejecutado en Kefren.

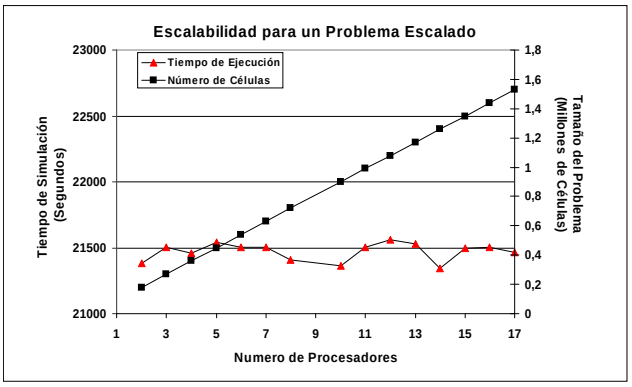


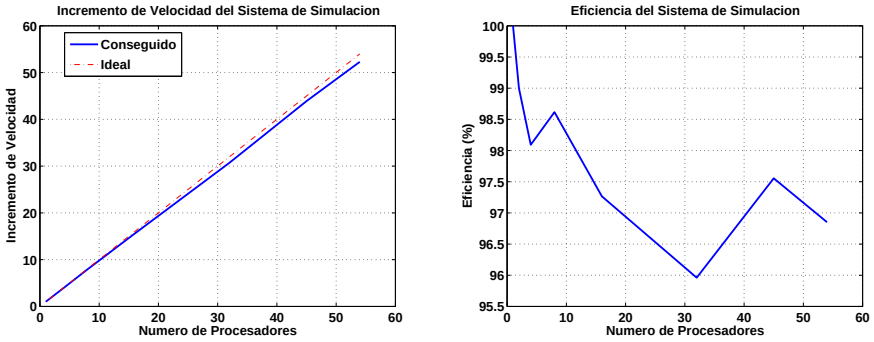
Figura 5-25: Escalabilidad del sistema de simulación. Tejido bidimensional de 1000x1000 células ejecutado en Kefren.

este valor no es generalmente alcanzado ya que existen comunicaciones y sincronizaciones entre procesos que impiden alcanzar tales cotas de eficiencia.

Se ha diferenciado entre las pruebas llevadas a cabo con tejidos bidimensionales y aquellas con tejidos tridimensionales.

5.7.1. Tejido Bidimensional

Las Figuras 5-24.a y 5-24.b muestran, respectivamente, el incremento de velocidad y la eficiencia del sistema de simulación, cuando se realiza una propagación del potencial de acción durante 10 ms, en un tejido ventricular de 1000x1000 células. Esta simulación da lugar a un sistema de 1 millón de ecuaciones lineales. Se han realizado ejecuciones hasta con 32 procesadores, empleando dos procesadores por nodo del cluster Kefren, y utilizando como librería de paso de mensajes ScaMPI. En la figura se observa que la aplicación alcanza un incremento de velocidad de 30.17 y una eficiencia del 94.2 % con 32 procesadores.



(a) Incremento de Velocidad. (b) Eficiencia.

Figura 5-26: Incremento de velocidad y eficiencia del sistema de simulación. Tejido bidimensional de 1000x1000 células ejecutado en Odin.

Un indicador adicional de la escalabilidad de una aplicación paralela consiste en la utilización de un problema escalado. Una aplicación eficientemente paralelizada debe requerir un tiempo de ejecución similar, cuando el tamaño del problema aumenta de manera proporcional al número de procesadores empleado. Para ello, la Figura 5-25 muestra el tiempo de ejecución, para diferente número de procesadores, cuando el tamaño del tejido, en número de células, aumenta de manera proporcional al número de procesadores. Para esta prueba se ha simulado un potencial de acción completo durante 250 ms. Se puede observar, por ejemplo, que una ejecución para un tejido de 1 millón de células, usando 11 procesadores, ha requerido aproximadamente 21500 segundos. Las simulaciones de esta prueba se han ejecutado utilizando únicamente un proceso por nodo del cluster Kefren, ya que, de esta manera, se obtiene resultados más rápidos que utilizando dos procesadores en cada nodo, principalmente debido a la compartición de recursos como la memoria por parte de los dos procesos del mismo nodo. Por esta razón, los resultados de la figura únicamente alcanzan hasta 17 procesadores.

Con el objetivo de evaluar las prestaciones del sistema de simulación con mayor número de nodos de computación, se han realizado ejecuciones sobre el cluster Odin. En este sentido, la Figura 5-26.a muestra los resultados de incremento de velocidad obtenidos, de la misma simulación cardiaca que en las pruebas realizadas en Kefren, empleando únicamente un procesador de cada nodo en el cluster Odin. Se puede observar que el sistema de simulación presenta una escalabilidad similar a la obtenida en las ejecuciones en Kefren. Estos resultados indican que, para tamaños de problema lo suficientemente grandes es de esperar que un aumento en un factor p del número de procesadores produzca una reducción en un factor muy parecido a p en el tiempo de ejecución.

La Figura 5-26.b muestra los resultados de eficiencia obtenidos en la ejecución. Los picos de eficiencia obtenidos para las ejecuciones de 8 y 45 procesadores están acentuados debido a la escala del eje Y pero parecen principalmente debidos a la utilización compartida del cluster por parte de

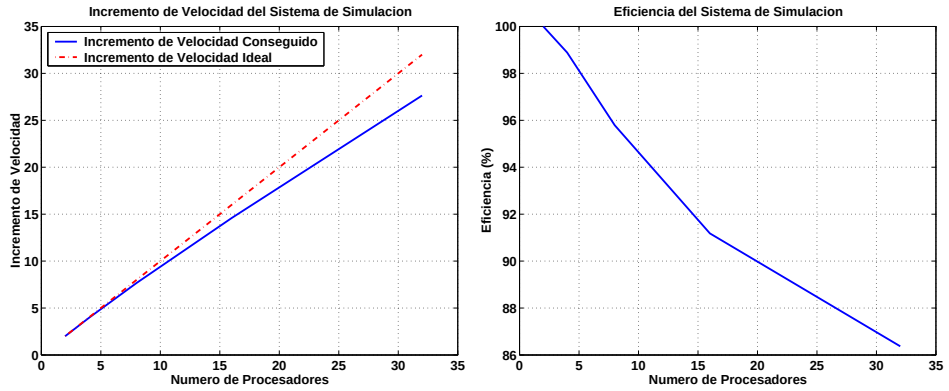


Figura 5-27: Incremento de velocidad y eficiencia del sistema de simulación. Tejido tridimensional de 100x100x100 células ejecutado en Kefren.

otros usuarios. En realidad, aunque los nodos de computación son exclusivos para una ejecución, no ocurre lo mismo con la red de comunicaciones, que es sensible a la variación de la carga del cluster en un momento dado.

Los resultados obtenidos son coherentes con la estrategia de paralelización empleada, reduciendo las comunicaciones y balanceando la carga entre todos los procesadores involucrados, dos de los aspectos más importantes para realizar una adecuada paralelización.

5.7.2. Tejido Tridimensional

La Figura 5-27 muestra el incremento de velocidad y la eficiencia del sistema cuando se simula la propagación del potencial de acción en un tejido cardiaco de 100x100x100 células, durante 250 ms, utilizando hasta 32 procesadores del cluster Kefren. Las simulaciones se han llevado a cabo ejecutando dos procesos en cada nodo del cluster. Se puede observar que el sistema de simulación escala de manera bastante lineal con respecto al número de procesadores.

Para una simulación de estas características, los tiempos de ejecución aparecen reflejados en la Tabla 5.4. La ejecución en un solo procesador requiere una cantidad de memoria tal que provoca el fenómeno de paginación al ejecutarse en un único nodo del cluster. Una ejecución en dos procesadores de diferentes nodos, requiere más de un día para su ejecución, mientras que, empleando 32 procesadores, se consigue reducir el tiempo a poco más de dos horas.

Esta es la verdadera ganancia para el usuario final, ya que acelerar el proceso de simulación cardiaca permite realizar mayor número de simulaciones por unidad de tiempo y, de esta manera, aumentar la productividad investigadora.

Como era de esperar, la escalabilidad del sistema en simulaciones tridimensionales es algo más reducida frente a las simulaciones bidimensionales. Estas diferencias son debidas al aumento en el número de comunicaciones del esquema de discretización basado en 7 vecinos. Sin embargo, los resultados obtenidos confirman la idoneidad del uso de una plataforma de computación basada en

Tabla 5.4: Tiempos de ejecución y resultados de escalabilidad para una simulación de potencial de acción en un tejido de 100x100x100 a lo largo de 250 ms (dt = 8 μs), usando hasta 32 procesadores del cluster Kefren.

Número de procesadores	Tiempo de Simulación (horas)	Incremento de Velocidad	Eficiencia
2	34.72		
4	17.55	3.95	98.88
8	9.06	7.66	95.79
16	4.75	14.58	91.18
32	2.51	27.63	86.36

clusters de PCs para la ejecución de simulaciones cardiacas.

5.8. Estado Actual del Sistema de Simulación: Áreas de Investigación

En la actualidad, el sistema de simulación CAMAEC se encuentra desplegado en producción en el cluster de PCs del Grupo de Bioelectrónica del Ci^2B. Además, hay abiertas varias líneas de investigación en las que se está utilizando este simulador como herramienta de simulación.

En primer lugar, Lucía Romero Pérez junto con el Prof. Dr. Jose María Ferrero de Loma Osorio están llevando a cabo estudios para evaluar el factor de seguridad (*Safety Factor, SF*) en tejidos bidimensionales. El parámetro SF resulta muy útil en el estudio de los bloqueos del frente de onda del potencial de acción. A su vez, estos bloqueos están íntimamente ligados a la generación y mantenimiento de arritmias potencialmente mortales como la taquicardia o la fibrilación ventricular. El uso del sistema de simulación CAMAEC permite llevar a cabo este tipo de simulaciones, facilitando así el estudio de este fenómeno. Esta línea ha dado lugar a la publicación de los resultados obtenidos [120, 121, 122, 123].

Una línea de investigación adicional está siendo llevada a cabo por la doctorando Karen Eliana Cardona Urrego, supervisada por el Prof. Dr. Javier Saiz Rodríguez. El objetivo principal es estudiar los efectos de la lidocaína en la actividad eléctrica cardiaca. La lidocaína es una droga antiarrítmica que permite inducir bloqueos en los canales de sodio. Esta droga afecta a la velocidad de conducción (Conductivity Velocity, CV) de la propagación eléctrica, así como al periodo refractario de las células. Los últimos resultados obtenidos [124] apuntan a que la lidocaína reduce la CV en un 40 % y aumenta el periodo refractario hasta un 14 % dependiendo de la cantidad suministrada. Otros artículos publicados en el marco de esta línea incluyen [125, 126].

Otra de las líneas de investigación está siendo impulsada por el doctorando Oscar Alberto Henao Gallo, supervisado por Jose María Ferrero. Aquí, se utiliza el simulador CAMAEC para investigar la

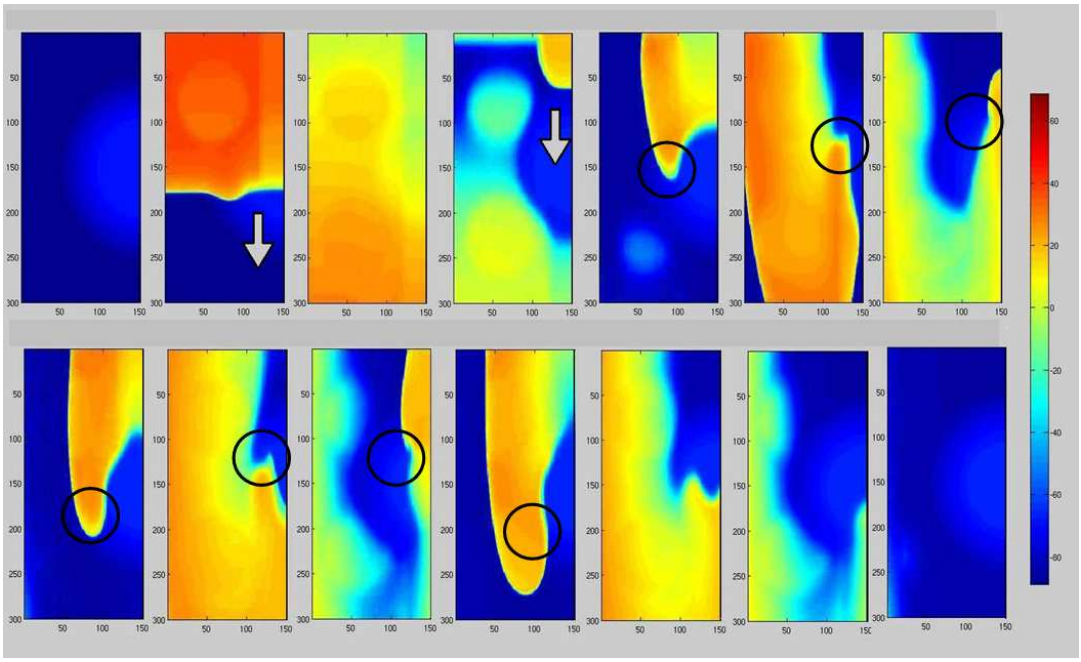


Figura 5-28: Reentrada transmural en una pared afectada por isquemia regional. Los círculos negros indican singularidades de fase (o pivotes) creados tras la rotura del frente de onda. (Cortesía de Oscar Alberto Henao).

influencia de lesiones isquémicas en pared transmural heterogénea sobre la desestabilización del frente de onda eléctrico, así como la formación de ondas reentrantes (ver Figura 5-28). La importancia de esta investigación radica en dilucidar el dilema de generación de fibrilación ventricular, que conlleva de manera irremediable a una muerte súbita. Este tipo de simulaciones suelen involucrar la simulación de tejidos con condiciones heterogéneas, de tamaño desde 100x100 hasta 600x600 células, llevadas a cabo con este sistema de simulación. Los primeros resultados obtenidos han sido publicados en [127].

Otra línea de investigación que utiliza el sistema de simulación CAMAEC está siendo llevada a cabo por Beatriz Trenor Gomis, doctora perteneciente al Ci^2B, bajo la responsabilidad de Jose María Ferrero y Javier Saiz. Esta linea se centra en la simulación y análisis de arritmias cardiacas, donde el sistema CAMAEC permite simular arritmias por reentrada generadas en los ventrículos, bajo distintas condiciones patológicas de isquemia en el tejido cardiaco [128]. En efecto, las inhomogeneidades electrofisiológicas que aparecen durante la isquemia miocárdica facilitan los bloqueos de la propagación del potencial de acción y la generación de reentradas [129, 130]. El programa CAMAEC permite controlar parámetros y condiciones difíciles de controlar experimentalmente, así como realizar análisis exhaustivos de las causas más íntimas que generan las distintas arritmias. Por ello, los resultados obtenidos enriquecen el conocimiento acerca de las arritmias cardiacas, complementando los resultados experimentales realizados por numerosos grupos de investigación, y permitiendo enfocar terapias para tratar dichas disfunciones.

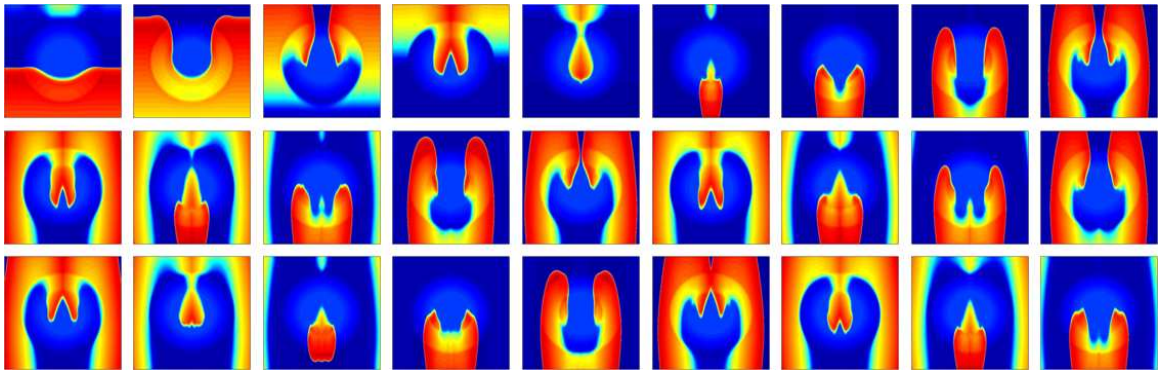


Figura 5-29: Propagación de potencial de acción sobre una zona central isquémica (Cortesía de Beatriz Trenor).

En la Figura 5-29, se muestra un ejemplo de arritmia cardiaca generada por la formación de una reentrada en un tejido isquémico. Cada fotograma representa la distribución de potenciales eléctricos en un instante determinado. Se puede observar cómo la propagación del estímulo eléctrico no es planar sino que sigue un circuito de reentrada debido a la presencia de células isquémicas en la parte central y a la aplicación de un estímulo prematuro.

En el campo de la simulación auricular, la doctorando Catalina Tobón Zuluaga, bajo la responsabilidad de Javier Saiz, está utilizando el sistema CAMAEC para investigar sobre los mecanismos que inician y perpetúan las arritmias auriculares y que ayudan a la aparición de fibrilación auricular (FA). Para ello, se investiga la incidencia de la geometría sobre el inicio de estas arritmias y el efecto de diferentes fármacos antiarrítmicos sobre los patrones de las taquicardias auriculares. Su importancia radica en que la fibrilación auricular es la arritmia cardiaca más frecuente en la práctica diaria, constituyendo uno de los principales problemas sanitarios actuales. Por este motivo, un adecuado diagnóstico de los mecanismos que generan y mantienen la FA puede permitir un tratamiento fármaco-quirúrgico selectivo que, aumentando su eficacia, provoque menores daños secundarios sobre el paciente. El desarrollo de un modelo de aurícula puede ser utilizado para evaluar los cambios estructurales y electrofisiológicos asociados a la FA, permitiendo el diseño de novedosas estrategias terapéuticas específicas para cada tipo de fibrilación.

Para las simulaciones se utiliza un modelo matemático de célula de aurícula humana [106] introducido en el programa CAMAEC. Actualmente se está investigando la aparición de reentradas debidas a la activación de focos ectópicos localizados alrededor de las venas pulmonares en la aurícula izquierda, como mecanismo de iniciación de la FA. Dicho tejido es anisotrópico e incluye orificios para las venas pulmonares. La estimulación de la aurícula se simula mediante la aplicación de estímulos periódicos y posteriormente es aplicado un segundo estímulo (foco ectópico) alrededor de los orificios de las venas pulmonares lo que, bajo determinadas condiciones, genera reentradas en el tejido simulado. Un ejemplo de este tipo de simulaciones se puede observar en la Figura 5-30.

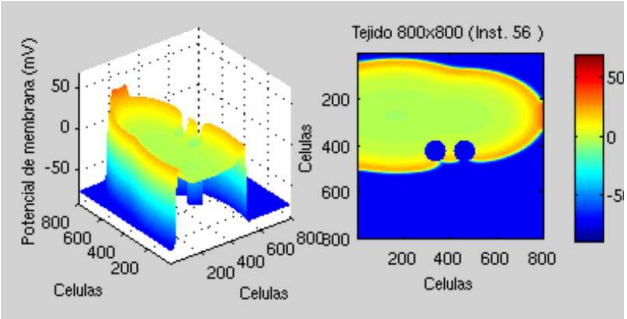


Figura 5-30: Propagación de potencial de acción sobre una zona con orificios que simulan las venas pulmonares (Cortesía de Catalina Tobón).

Posteriormente, mediante la simulación del efecto que tienen diferentes fármacos antiarrítmicos sobre las corrientes iónicas se estudia su eficacia para prevenir la aparición de circuitos reentrantes. Adicionalmente, se estudia el efecto de diferentes patrones de ablación sobre la conducción auricular mediante la simulación de líneas de ablación sobre el tejido simulado, con el fin de determinar un patrón para dicho tratamiento quirúrgico, más eficaz y de menores efectos secundarios. Los resultados obtenidos en esta línea han sido publicados en [131, 132].

5.9. Conclusiones

En este capítulo se ha descrito el sistema de simulación CAMAEC, que permite el estudio de la propagación del potencial de acción en tejidos cardiacos bidimensionales monodominio y su extensión a geometrías tridimensionales sencillas. En primer lugar se ha detallado su arquitectura y la estrategia de paralelización realizada, para aprovechar eficientemente las capacidades de cómputo de un cluster de PCs. En segundo lugar, se ha descrito la aproximación empleada para resolver el sistema de ecuaciones lineales involucrado, estudiando el uso de diferentes estrategias basadas en métodos iterativos y directos. Dado que la matriz de coeficientes es simétrica, definida positiva y bien condicionada, la aplicación del método iterativo del Gradiente Conjugado sin precondicionado ha resultado ser la opción más rápida para resolver el sistema de ecuaciones.

Posteriormente, se han descrito y evaluado los diferentes métodos de integración de ecuaciones diferenciales ordinarias empleados para permitir aumentar el paso de tiempo de simulación de manera estable. El método de Euler implícito, aplicado a la integración de las compuertas de los modelos celulares, combina la estabilidad ofrecida por los métodos implícitos con la sencillez y rapidez de cómputo de un método explícito. De esta manera, se ha conseguido aumentar de manera estable el paso de tiempo de simulación por encima de los límites impuestos por el método de Euler explícito, que suele ser el más utilizado en este tipo de simulaciones.

Además, se ha realizado una implementación eficiente del proceso de E/S para gestionar la

grabación de datos de la simulación. Para ello se han combinado técnicas de generación automática de código, para permitir una gran flexibilidad en la especificación de la información de grabación, con la utilización del estándar MPI-2 I/O de E/S paralela para conseguir un acceso colaborativo al disco entre los múltiples procesos involucrados en una simulación.

Luego, se han evaluado las prestaciones del sistema de simulación, empleando métricas de incremento de velocidad y eficiencia, obteniendo aceleraciones del orden del número de procesadores debido principalmente a la estrategia de paralelización empleada. Los resultados obtenidos garantizan dos de los principales objetivos iniciales, consistentes en acelerar el proceso de simulación cardiaca junto con la posibilidad de abordar problemas de mayor dimensión.

Finalmente, se han descrito las áreas de investigación en las que se está utilizando el sistema de simulación CAMAEC, tras su puesta en producción en los recursos computacionales del Ci^2B. Como conclusión, se ha desarrollado un sistema de simulación eficiente que permite acelerar el proceso de simulación cardiaca, permitiendo realizar mayor número de simulaciones por unidad de tiempo, lo que redunda en un aumento de la productividad investigadora. Además, la versatilidad del simulador permite su utilización en numerosas líneas de investigación. Por lo tanto, una herramienta de estas características supone una aportación importante al campo de la simulación cardiaca con el objetivo de permitir a los expertos biomédicos el estudio de las causas de determinadas patologías cardiacas.

Capítulo 6

La Optimización y Paralelización del Diseño de Proteínas

"Nothing is less productive than to make more efficient what should not be done at all". Peter Drucker, Management Leadership.

En este capítulo se describe la utilización de técnicas de Computación de Altas Prestaciones para la optimización de proteínas. En primer lugar se detalla el proceso actual de diseño de proteínas, que se pretende mejorar. Posteriormente, se aborda la implementación de una versión secuencial optimizada que gestiona de manera eficiente la información de la interacción energética entre rotámeros. A continuación, se describe la estrategia paralela de reducción dimensional, que permite el diseño de proteínas de mayor dimensión, mediante la distribución de la matriz de energías entre múltiples procesadores. Finalmente, el capítulo concluye con las principales ventajas de la aproximación llevada a cabo.

6.1. Introducción

El proceso de optimización combinatoria, llevado a cabo por el grupo del Prof. Jaramillo, modela una proteína como un conjunto de posiciones. En cada posición puede haber, en un momento dado, un solo rotámero del conjunto de rotámeros disponibles para dicha posición. Por ejemplo, la Figura 6-1 muestra una proteína con 3 posiciones, y un total de 90 rotámeros, donde los rotámeros [0,41] están disponibles para la posición 0, aquellos en [42,76] están disponibles para la posición 1 y los numerados en [77,89] corresponden a la posición 2. Los rotámeros se identifican de acuerdo a una numeración secuencial global que facilita la localización de su posición dentro de la proteína.

La Figura 6-2 muestra un pseudocódigo que resume las principales fases del diseño de proteínas.

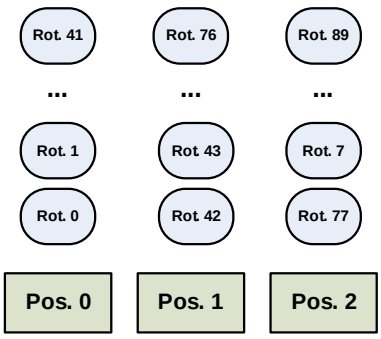


Figura 6-1: Esquema simplificado de la modelización de una proteína en el proceso de optimización combinatoria.

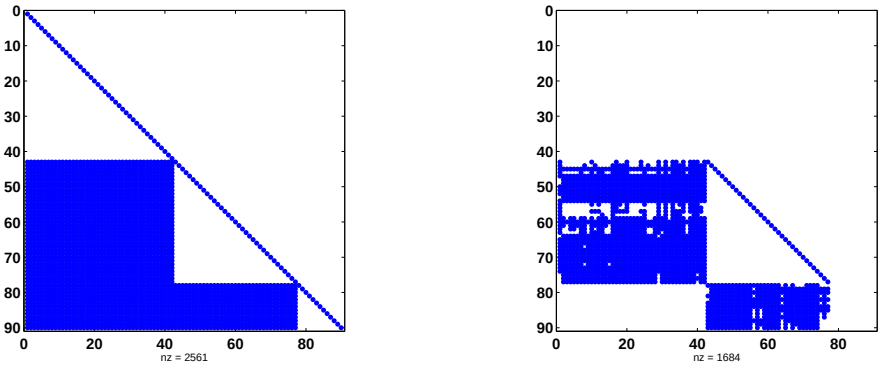
```
proteína = EstadoInicialProteina();
energiaActualObj1 = CalculaEnergiaObj1(proteína);
energiaActualObj2 = CalculaEnergiaObj2(proteína);
Para iter = 0 .. niter - 1 hacer
        Para pos = 0 .. npos - 1 hacer
                nuevoRot = MutaRotamero(proteína, pos);
                energiaNuevaObj1 = CalculaEnergiaObj1(proteína);
                energiaNuevaObj2 = CalculaEnergiaObj2(proteína);
                Si (MejorSolucion(energiaNuevaObj1, energiaNuevaObj2)) Entonces
                        aceptaMutacion(proteína, pos, nuevoRot);
                        energiaActualObj1 = energiaNuevaObj1;
                        energiaActualObj2 = energiaNuevaObj2;
                fSi
        fPara
fPara
```

Figura 6-2: Pseudocódigo del diseño de proteínas.

El código está basado en métodos de Monte Carlo y utiliza dos objetivos (plegado y enlace). Nótese que el código que se muestra ha sido simplificado, para poner de manifiesto únicamente las principales fases del proceso.

Inicialmente se elige una secuencia aleatoria de rotámeros para la proteína. A continuación, comienza un proceso iterativo que se repite un cierto número de iteraciones, indicado por el usuario. En cada iteración se realiza un recorrido por todas las posiciones de la proteína, evaluando la posibilidad de mutar el rotámero existente en esa posición por otro, del conjunto de rotámeros disponibles para esa posición. Para ello, se evalúa la energía que tendría la proteína en el caso de producirse el cambio. Si dicha energía es menor que la actual, entonces se acepta el cambio de rotámero.

De todas las etapas, la que mayor tiempo involucra es el cálculo de la energía de la proteína, ya que es necesario realizar múltiples accesos a la matriz de energías. En realidad, el proceso de optimización combinatoria es computacionalmente costoso, debido principalmente al elevado número de iteraciones a realizar y al acceso a esta matriz. Por lo tanto, resulta interesante conocer las características de las



(a) Objetivo 1 (Plegado, Folding)

(b) Objetivo 2 (Enlace, Binding)

Figura 6-3: Matrices de interacción energética entre rotámeros para ambos objetivos del proceso de optimización. Proteína de 3 posiciones y 90 rotámeros.

matrices de energía para desarrollar estructuras de datos que combinen un almacenamiento óptimo con un acceso eficiente a la información almacenada.

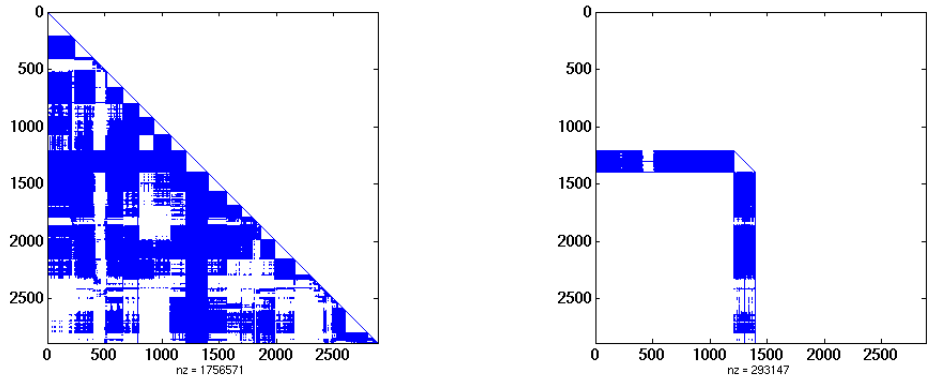
6.1.1. Matrices de Energía

Las matrices de energía se calculan en una fase previa al proceso de optimización y contienen la interacción energética entre dos rotámeros cualesquiera. Estas matrices tienen una dimensión igual al número total de rotámeros y, en ellas, se distinguen dos tipos de energía. En los elementos diagonales se almacenan las auto-energías (*single energies*), es decir, la interacción energética de un rotámero consigo mismo, mientras que el resto son las energías entre pares (*pair energies*).

Las matrices de energía son simétricas dado que la interacción energética entre dos rotámeros es una propiedad simétrica. Con las herramientas ya existentes, se realiza el almacenamiento de la parte triangular inferior, para reducir espacio en disco y en memoria. Otra característica peculiar de estas matrices es que un rotámero nunca puede interactuar con el resto de rotámeros asignados a su posición. En efecto, un estado concreto de la proteína incluye un rotámero diferente en cada posición y son éstos los que interactúan entre sí para calcular la energía de la proteína. Por lo tanto, nunca se puede dar una interacción entre dos rotámeros de la misma posición, de ahí que las correspondientes entradas de la matriz sean nulas.

La Figura 6-3 muestra la parte triangular inferior de las matrices de energía asociadas a la proteína de ejemplo, con 3 posiciones y 90 rotámeros, correspondiente a la Figura 6-1. La matriz de energías correspondiente al primer objetivo (plegado, Figura 6-3.a) presenta un nivel de llenado, con respecto a la matriz triangular inferior completa, del 31.61 %. La matriz de energías correspondiente al segundo objetivo (enlace, Figura 6-3.b) presenta un nivel de llenado del 20.71 %.

La Figura 6-4 muestra las matrices de energía asociadas a un ejemplo más realista, correspon-



(a) Objetivo 1 (Plegado)

(b) Objetivo 2 (Enlace)

Figura 6-4: Matrices de interacción energética entre rotámeros para ambos objetivos del proceso de optimización. Proteína de 101 posiciones y 2886 rotámeros.

diente a una proteína con 101 posiciones y 2886 rotámeros totales. La matriz para el objetivo 1 (Figura 6-4.a) presenta un índice de llenado del 42.17 % mientras que para el objetivo 2 es de tan solo un 7.03 %.

Las matrices de energía tienen un factor de dispersión bastante elevado, por lo que resulta imprescindible la utilización de técnicas de almacenamiento disperso que únicamente almacenen los elementos no nulos de la matriz. Además, no existe ningún patrón de acceso conocido a la matriz, ya que su información se utiliza para evaluar la interacción energética entre los rotámeros de la proteína, Por lo tanto, se debe permitir un acceso aleatorio eficiente a la matriz de energías.

6.2. Optimización Secuencial

Con el objetivo de combinar un bajo coste de almacenamiento y un acceso eficiente aleatorio a la matriz de energías, se ha optado por utilizar un esquema de almacenamiento basado en el formato CSR (Compressed Sparse Row, Fila Dispersa Comprimida) [80]. Para ello, se ha adaptado este conocido formato para que pueda ser utilizado en el almacenamiento de matrices simétricas, donde únicamente se considera la parte triangular inferior. Este nuevo formato se ha denominado LTCSR (Lower Triangular CSR).

El formato LTCSR permite almacenar una matriz dispersa simétrica, especificada a través de su parte triangular inferior, utilizando tres vectores. A continuación se describe el formato a través de un ejemplo. Consideremos la matriz cuadrada que se muestra a continuación, de dimensión $n = 6$ y con $nnz = 10$ elementos no nulos:

$$A = \begin{bmatrix} 11 & & & & & \\ 0 & 22 & & & & \\ 13 & 23 & 33 & & & \\ 0 & 24 & 0 & 44 & & \\ 0 & 0 & 35 & 0 & 55 & \\ 0 & 0 & 0 & 46 & 0 & 0 \end{bmatrix} \tag{6.1}$$

Utilizando el formato LTCSR la matriz puede representarse mediante los siguientes tres vectores:

$$AR = \begin{bmatrix} 11 & 22 & 13 & 23 & 33 & 24 & 44 & 35 & 55 & 46 \end{bmatrix} \qquad |AR| = nnz$$
$$IA = \begin{bmatrix} 0 & 1 & 2 & 5 & 7 & 9 & 10 \end{bmatrix} \qquad |IA| = n + 1$$
$$JA = \begin{bmatrix} 0 & 1 & 0 & 1 & 2 & 1 & 3 & 2 & 4 & 3 \end{bmatrix} \qquad |JA| = nnz$$

El vector AR contiene los elementos no nulos de la matriz triangular inferior, obtenidos tras un recorrido por filas de la matriz. El vector IA contiene el índice al vector AR donde comienzan los valores de cada fila. El vector JA contiene la columna donde está situado cada uno de los elementos del vector AR.

Mediante este formato, ya no es posible acceder en tiempo constante a $A(i,k)$, mediante la indexación de la matriz. Por el contrario, ahora resulta necesario utilizar la siguiente expresión:

$$AR[busca(JA, k, IA(i), IA(i+1) - 1)] \tag{6.2}$$

La función *busca* devuelve la posición global del elemento k en el subvector de JA comprendido en el intervalo de posiciones $[IA(i), IA(i+1)-1]$. Nótese que el mecanismo de construcción del vector JA garantiza que el subvector sobre el que se realiza la búsqueda está ordenado ascendentemente. Por lo tanto, es posible utilizar una estrategia de búsqueda binaria [133], que resulta más eficiente que la búsqueda tradicional en un vector desordenado.

La búsqueda binaria es una estrategia de búsqueda guiada que utiliza la información del elemento medio del vector para continuar la búsqueda por una mitad del vector o por la otra. De esta manera se consigue dividir a la mitad el tamaño del problema en cada paso de búsqueda. La búsqueda binaria presenta un coste logarítmico ($O(log_2(n))$ en el peor caso, siendo n el número de elementos del vector. Nótese que el peor caso ocurre cuando el elemento buscado no se encuentra en el vector. Esto supone una principal ventaja frente a la búsqueda secuencial que presenta un coste, en el peor de los casos, lineal con el número de elementos del vector ($O(n)$).

Como optimización adicional, nótese que si la diagonal de la matriz no contiene elementos nulos, es posible acceder de manera directa al elemento diagonal, para la obtención de las auto-energías en tiempo constante ($O(1)$) con la siguiente expresión:

Programa	Tiempo (Carga de Datos + Optimización)	Memoria (MBytes)
gBiObj	3.07 + 2.65 (5.73)	16
opt_biobj	19.24 + 4.98 (24.22)	16

Tabla 6.1: Comparativa de tiempos de ejecución y consumo de memoria para la optimización de una proteína de 101 posiciones y 2886 rotámeros (5000 iteraciones).

Programa	Tiempo (Carga de Datos + Optimización)	Memoria (MBytes)
gBiObj	116.04 + 2.19 (118.24)	654
opt_biobj	660.34 + 20.76 (681.11)	801

Tabla 6.2: Comparativa de tiempos de ejecución y consumo de memoria para la optimización de una proteína de 32 posiciones y 17388 rotámeros (5000 iteraciones).

$$AR[IA(i+1)-1] \tag{6.3}$$

La utilización de esquemas de almacenamiento de matrices dispersas implica guardar información sobre la posición de los elementos, involucrando un consumo de memoria adicional. En realidad, la ventaja de estas técnicas depende en gran medida del índice de llenado de la matriz dispersa. Para matrices suficientemente dispersas, la memoria utilizada en almacenar la información de las posiciones puede resultar despreciable frente al ahorro de almacenar únicamente las entradas no nulas de la matriz.

Además, este formato de almacenamiento impide el acceso aleatorio a la matriz en tiempo constante. Por el contrario, se deben realizar búsquedas en subvectores para poder indexar a la matriz. Afortunadamente, éstas búsquedas se aceleran en gran medida debido a la utilización de algoritmos eficientes y a la propia gestión de la memoria.

Una vez redefinida la principal estructura de datos, la validación del código desarrollado ha sido llevada a cabo mediante la ejecución de varios casos de estudio en los que se diseña una proteína. Al tratarse de métodos estocásticos es posible fijar la semilla de ambos generadores de números aleatorios a un mismo valor para obtener la misma secuencia de números aleatorios. De esta manera, es posible comprobar que ambos procesos iterativos obtienen los mismos resultados.

6.2.1. Comparativa de Prestaciones: Secuencial Optimizado

A continuación se muestra una comparativa de prestaciones entre el código original (*opt_biobj*) y la optimización secuencial llevada a cabo utilizando las nuevas estructuras de datos (*gBiObj*). El tiempo se ha desglosado en lectura de datos y en la parte de diseño de la proteína.

La Tabla 6.1 muestra los resultados de la comparativa para un diseño de una proteína de 101 posiciones, con 2886 rotámeros, durante 5000 iteraciones. La Tabla 6.2 muestra los resultados del diseño de una proteína de 32 posiciones, con 17388 rotámeros, durante 5000 iteraciones.

A la vista de los resultados se pueden extraer varias conclusiones. Para los casos de estudio pequeños, la reducción en el consumo de memoria obtenido por las técnicas de almacenamiento matricial disperso no tiene mucha influencia. En realidad *opt_biobj* implementa técnicas para tratar de reducir el consumo de memoria, guardando, para cada rotámero, una matriz con la interacción energética con los rotámeros de menor índice. Sin embargo, solo las técnicas de almacenamiento matricial disperso garantizan que únicamente se guardan las entradas no nulas de la matriz. Para un caso de estudio más grande (Tabla 6.2) se observa que la reducción del consumo de memoria es considerable, consiguiendo un 18 % menos de consumo de memoria.

La carga de datos de la matriz de energía también se ha acelerado en gran medida. Este tiempo incluye la lectura de datos y la creación de las estructuras de datos necesarias para la representación de la matriz de energías. La sencillez del formato LTCSR permite crear las estructuras de datos (vectores AR, JR e IA) de manera progresiva conforme se realiza la lectura de fichero. Esto contribuye a reducir el tiempo global de esta etapa. Esta aproximación ha permitido reducir la carga de datos en un factor en [5.69, 6.26], dependiendo del caso de estudio.

Por último, el proceso de optimización de proteínas también se ha reducido frente al código original, especialmente para el caso de estudio más grande. Esto es debido únicamente a la gestión de la memoria realizada en la matriz de energías. La utilización de estructuras de datos, que tienen en cuenta la jerarquía de la memoria de las arquitecturas modernas, permite disminuir el número de fallos de página y reducir así el tiempo de ejecución de las aplicaciones. En este sentido, la optimización realizada ha conseguido acelerar el proceso en un factor entre [1.87, 9.47], dependiendo del caso de estudio.

6.3. Aproximación Paralela: Reducción Dimensional

Una vez comprobada la bondad de una adecuada gestión de la memoria, en el almacenamiento y acceso a la matriz de energías, es importante permitir abordar problemas de mayor dimensión. Sin embargo, la dimensión de la matriz de energías depende del número de rotámeros, por lo que proteínas más grandes, con un mayor número de rotámeros, dan lugar a matrices de energías de mayor dimensión. Además, esta matriz de energías debe estar cargada en memoria para realizar el proceso de optimización, lo que supone un importante problema de consumo de memoria.

En realidad, la situación ideal pasa por realizar la optimización de una proteína, independientemente de la capacidad de memoria del recurso de computación. Para ello se utiliza una estrategia que combina una reducción dimensional del problema, con la utilización de múltiples procesadores para la optimización colaborativa de una proteína.

Una reducción dimensional implica que un elemento de proceso pueda optimizar parte del problema global. Para ello, cada proceso trabajará sobre todas las posiciones de la proteína pero únicamente

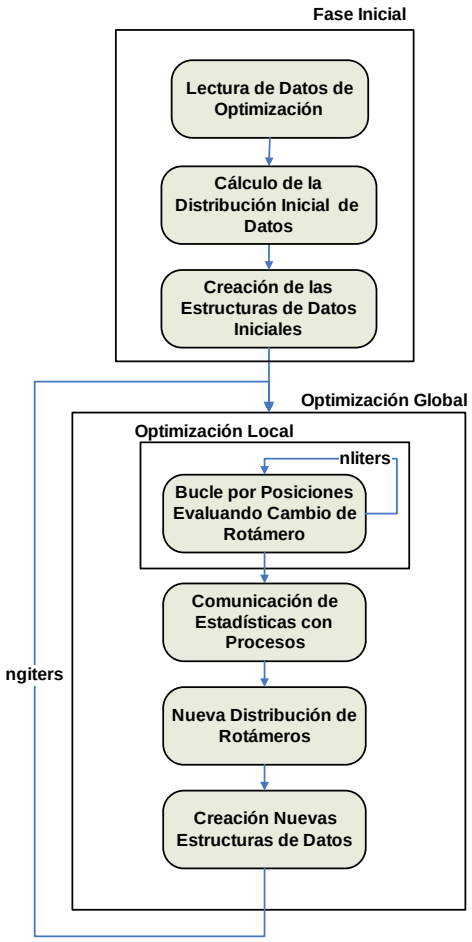


Figura 6-5: Principales fases del proceso de optimización de proteínas utilizando la aproximación paralela basada en reducción dimensional.

con un subconjunto de rotámeros, del conjunto total de rotámeros de cada posición. De esta manera, es posible distribuir la matriz de energías entre diferentes procesos encargados de las optimizaciones locales.

La Figura 6-5 resume las principales fases del proceso de optimización utilizando esta estrategia. En primer lugar existe una fase inicial, encargada de la lectura de los datos de la optimización a realizar. A continuación, cada proceso obtiene los rotámeros a utilizar en cada posición, en base a una distribución de probabilidad inicial equiprobable. Esta distribución de probabilidad determina los rotámeros que debe seleccionar cada procesador. Nótese que la componente probabilística permite introducir cierta aleatoriedad en dicho proceso de selección. Una vez conocidos los rotámeros, se debe cargar la parte correspondiente de la matriz de energías de disco y crear las estructuras de datos apropiadas.

Superada la fase inicial, comienza un proceso iterativo en el que cada iteración global consta de una serie de etapas. En primer lugar, cada procesador realiza una optimización local, sobre parte del

$$(A, P0) = \begin{bmatrix} 11 & & & & \\ 0 & 22 & & & \\ 0 & 24 & 0 & 44 & \\ 0 & 0 & 35 & 0 & 55 \end{bmatrix}$$

$$(A, P1) = \begin{bmatrix} 13 & 23 & 33 & & & \\ 0 & 0 & 0 & 46 & 0 & 0 \end{bmatrix}$$

Figura 6-6: Distribución de la matriz de energías en dos procesadores (P0 y P1).

problema global, utilizando los rotámeros asignados en cada posición de la proteína. Este proceso se repite un determinado número de iteraciones locales, permitiendo la obtención de estadísticas de aparición de rotámeros en las soluciones parciales. A continuación, esta información, obtenida en cada uno de los procesadores, se intercambia con el resto. Luego, a partir de la frecuencia de aparición de rotámeros en las soluciones parciales, se altera la distribución de probabilidad que permite la selección de los nuevos rotámeros. Esta nueva distribución se utilizará en la siguiente iteración global. Por último, conocidos los nuevos rotámeros, se debe cargar la información de la matriz de energías correspondiente a dichos rotámeros.

Este proceso se repite durante un determinado número de iteraciones globales o hasta que se detecta la convergencia entre la distribución de probabilidad de una etapa y la anterior. La detección de la convergencia puede realizarse, por ejemplo, en base a un criterio de error relativo, empleando alguna norma vectorial. Alcanzar la convergencia implica que las optimizaciones locales de cada proceso ya no aportan información nueva al proceso y, por lo tanto, se considera que se ha alcanzado la mejor solución. El resultado final del proceso es, por tanto, la secuencia de rotámeros que componen la proteína.

A continuación se describe la utilización de un nuevo formato de almacenamiento matricial distribuido para la matriz de energías y, posteriormente, se detallan las fases del proceso de optimización de proteínas distribuido basado en una reducción dimensional.

6.3.1. Formato de Matriz de Energías Distribuida

Para trabajar con la matriz de energías distribuida se ha realizado la adaptación del formato LTCSR para trabajar con múltiples procesadores. Para ello se ha creado el formato DLTCSR (Distributed LTCSR). Este formato permite almacenar una matriz dispersa de manera distribuida y replicada entre múltiples procesadores.

Partiendo de la matriz de energías mostrada en (6.1), supongamos que se debe distribuir entre dos procesadores P0 y P1 para que tengan la información de rotámeros diferentes. Siguiendo con el ejemplo, supongamos que P0 se encarga de las filas 0,1 y 3,4 mientras que P1 recibe las filas 2,5. Esta situación da lugar a las matrices que se muestran en la Figura 6-6.

En P0, los vectores que representan la información de la matriz, de acuerdo al formato LTCSR,

son los siguientes:

$$AR = \begin{bmatrix} 11 & 22 & 24 & 44 & 35 & 55 \end{bmatrix}$$
$$IA = \begin{bmatrix} 0 & 1 & 2 & 4 & 5 \end{bmatrix}$$
$$JA = \begin{bmatrix} 0 & 1 & 1 & 3 & 2 & 4 \end{bmatrix}$$

Mientras que en P1, los vectores que representan la submatriz correspondiente son:

$$AR = \begin{bmatrix} 13 & 23 & 33 & 46 \end{bmatrix}$$
$$IA = \begin{bmatrix} 0 & 3 & 4 \end{bmatrix}$$
$$JA = \begin{bmatrix} 0 & 1 & 2 & 3 \end{bmatrix}$$

El acceso a la información de la matriz, en el formato DLTCSR, únicamente difiere del necesario para LTCSR en la necesidad de traducir el acceso a una fila global a la fila local de la correspondiente matriz. En este sentido, la expresión para acceder al elemento $A(i, j)$ viene dada por la siguiente expresión:

$$AR[busca(JA, k, IA(g2l(i)), IA(g2l(i) + 1) - 1)] \tag{6.4}$$

En ella, $g2l$ es una función que traduce el identificador de fila global al identificador de fila local. Dado que cada procesador conoce los rotámeros (y por tanto las filas de la matriz) asignados a él, es posible realizar una búsqueda binaria de la fila global en ese vector ordenado. La posición en la que se encuentra el elemento se corresponde directamente con el identificador de fila local. Por lo tanto, es posible computar esa expresión con un coste $O(log_2(n))$, siendo n el número de elementos del vector de rotámeros asignados al procesador.

Nótese que este formato permite gestionar la existencia de filas replicadas en diferentes procesadores. Esta situación permite que, diferentes procesos, puedan trabajar con rotámeros comunes. En realidad, conforme avance el proceso de optimización cabe esperar que muchos rotámeros estén replicados en diferentes procesos, ya que éstos son considerados como los mejores rotámeros y tendrán una alta probabilidad de ser elegidos por la distribución de probabilidad, en múltiples procesadores.

6.4. Fases de la Aproximación Distribuida

En esta sección se detallan las principales fases de la aproximación distribuida para la optimización de proteínas, resumidas en la Figura 6-5.

6.4.1. Fase Inicial

Esta fase es la encargada de la lectura de los datos relacionados con la optimización. Actualmente, el usuario especifica el número de rotámeros diferentes con el que trabaja cada procesador, para cada una de las posiciones de la proteína. Dado que el número de rotámeros determina la submatriz de energía, esta aproximación permite limitar el consumo de memoria de los procesadores.

En un futuro, se plantea la posibilidad de calcular de manera automática el número de rotámeros a utilizar por cada procesador. Este valor podría ajustarse para aprovechar al máximo la memoria del sistema en la que se realiza la ejecución. De esta manera, el sistema podría adaptar su ejecución a las características del entorno para permitir que los procesos de optimización local dispusieran de la mayor información posible.

Esta fase también es la encargada de crear la distribución de probabilidad inicial, en cada proceso, encargada de la selección de los rotámeros. Como se ha comentado anteriormente, esta distribución inicial indica que todos los rotámeros son equiprobables, de acuerdo a la siguiente expresión:

$$P_{Gi} = \frac{1}{NR_i} \tag{6.5}$$

siendo P_{Gi} el vector de probabilidades globales correspondiente a la i-ésima posición de la proteína y NR_i el número de rotámeros en dicha posición. Por lo tanto, inicialmente se escogen tantos rotámeros aleatorios en cada posición, del conjunto total de rotámeros disponibles, como haya indicado el usuario.

Por último, esta fase realiza la lectura de datos y la creación de las estructuras asociadas. Es posible realizar la lectura de datos de forma eficiente, dado que cada procesador conoce los rotámeros con los que debe trabajar. Por lo tanto, únicamente se deben leer y procesar las correspondientes filas de la matriz de energías. Adicionalmente, para realizar una gestión eficiente de la memoria, se realiza un pre-procesado único de la matriz de energías en disco, para generar información sobre el número de entradas no nulas de la matriz, correspondientes a cada rotámero/fila. Esta información permite dimensionar los vectores (IA,JA y AR) con el tamaño de memoria exacto para albergar la información, lo que evita costosos aumentos de tamaño de las estructuras de datos.

6.4.2. Optimización Local

Esta fase realiza fundamentalmente el proceso de Monte Carlo descrito en la Figura 6-2. A lo largo de la optimización local se substituye, para cada una de las posiciones, el rotámero actual por otro de manera aleatoria. Si la nueva alternativa permite reducir la energía global de la proteína, se acepta el cambio de rotámero. Adicionalmente, cada procesador dispone de un vector de tamaño igual al número de rotámeros globales que almacena la frecuencia de aparición de cada rotámero en las soluciones parciales. En este sentido, cada vez que se acepta una mutación de un rotámero, o se

mantiene el rotámero que ya estaba, se incrementa en una unidad su contador de frecuencia. Esto permite la generación progresiva de estadísticas sobre la bondad de los rotámeros, determinada por su tendencia a aparecer en las soluciones parciales.

6.4.3. Comunicación de Estadísticas con Procesos

Tras el proceso de optimización local, llevado a cabo en cada procesador, se realiza una suma colectiva del vector de frecuencias de manera que el resultado global de la suma esté disponible en todos los procesadores, de acuerdo a la siguiente expresión:

$$F_{Gi} = \sum_{j=1}^{p} F_{Li}^{j} \tag{6.6}$$

siendo F_{Li}^{j} el vector local a cada procesador j que contiene la información de frecuencia de aparición de rotámeros, para cada posición i de la proteína, y F_{Gi} el vector de frecuencia global resultante, disponible en todos los procesadores. Para ello se utiliza la rutina *MPI_All_reduce* de MPI que, en la misma operación, realiza la comunicación y la suma distribuida. Esta es la única comunicación que se realiza entre los elementos de proceso.

Como se puede observar, en lugar de implementar una estrategia basada en maestro-esclavo, donde todos los procesadores envían la información a uno que calcula la suma y luego la distribuye, se utiliza una aproximación colaborativa. Además, nótese que la comunicación involucra un vector valores enteros, de tamaño igual al número de rotámeros globales, cuyo coste de almacenamiento y envío se reduce a la mitad, comparado con los obtenidos al utilizar valores en doble precisión.

Posteriormente, es necesario transformar el vector de frecuencia global para que refleje una distribución de probabilidad que capture el conocimiento adquirido durante el proceso de optimización local, mediante la siguiente expresión:

$$\begin{aligned} \sigma &= p \cdot nliters \cdot npos \\ P_{Ni} &= \frac{F_{Gi}}{\sigma} \end{aligned} \tag{6.7}$$

donde σ representa el número total de mutaciones de rotámeros evaluadas durante el proceso de optimización local y P_{Ni} representa la distribución de probabilidad de selección de rotámeros para la i-ésima posición de la proteína, resultante de la optimización local.

Finalmente, es necesario combinar esta información con la distribución de probabilidad de selección de rotámeros ya existente. De esta manera es posible incorporar nuevo conocimiento a la distribución de probabilidad, de forma progresiva, tras cada proceso de optimización local. Esto se consigue mediante la siguiente expresión:

$$P_{Gi}^{k} = \frac{P_{G_i}^{k-1} + P_{Ni}}{\sum_{i=1}^{NRi} (P_{G_i}^{k-1} + P_{Ni})} \tag{6.8}$$

donde $P_{G_i}^{k}$ representa la distribución de probabilidad global de selección de rotámeros, para cada posición i de la proteína, en la iteración k, obtenida a partir de la de la etapa anterior ($P_{G_i}^{k-1}$) y normalizada a 1, para cada una de las posiciones de la proteína.

Mediante esta aproximación, la realización de múltiples iteraciones globales permite refinar la distribución de probabilidad en base al conocimiento adquirido durante los procesos de optimizaciones locales.

6.4.4. Cálculo de la Nueva Distribución de Rotámeros

La distribución de probabilidad computada en la etapa anterior permite, a cada procesador, obtener los rotámeros con los trabajará en la siguiente iteración global. Para ello, en esta fase, cada procesador elige NRi rotámeros diferentes, en cada posición de la proteína, de acuerdo a la distribución de probabilidad. Esto permite que los procesadores tiendan a elegir buenos rotámeros pero, al mismo tiempo, la componente estocástica del proceso permite la posible selección de nuevos rotámeros que puedan mejorar a los ya existentes.

En este sentido, el principal problema de esta fase se reduce a obtener n muestras diferentes de una determinada población de tamaño N, donde $n <= N$. Para ello se utiliza una modificación del algoritmo DSS (Discrete Sequential Search) [134], disponible en la librería UNURAN [135], para permitir la obtención de muestras no repetidas.

El método DSS realiza las siguientes acciones:

1. Genera un número aleatorio entre 0 y 1 y lo multiplica por la suma de probabilidades de la distribución de probabilidad (1 para el caso que nos ocupa).

2. Se recorre el vector de probabilidades acumulando la suma en una variable, deteniendo el proceso en el momento en el que se supera dicho valor.

Para evitar obtener muestras duplicadas, una vez que se obtiene un valor se establece su probabilidad a 0. De esta manera, se evita que el método vuelva a escoger ese valor. Para verificar la bondad del método DSS, con la modificación dinámica de la distribución de probabilidad para la selección de valores diferentes, se ha realizado un experimento. Para ello, se ha considerado un vector de 500 componentes con una distribución Gaussiana de centro 250. Se han realizado 1000 iteraciones en las que se muestrean 10 valores diferentes de manera simultanea, utilizando el método DSS modificado. Además, se anota la frecuencia de aparición de los valores muestreados para verificar si la distribución de probabilidad de obtención de valores se asemeja a la distribución original.

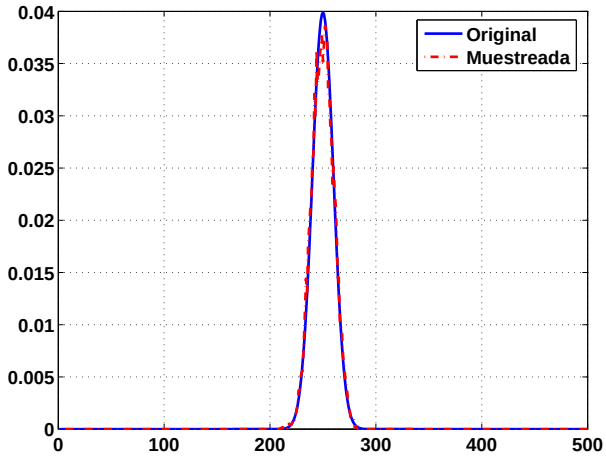


Figura 6-7: Comparativa de distribuciones de probabilidad para la evaluación de la bondad del método DSS modificado para la selección de elementos no repetidos. La gráfica muestra la frecuencia de aparición de cada uno de los elementos.

La Figura 6-7 compara gráficamente la distribución original con la distribución de probabilidad obtenida. Se puede observar que el método tiene un buen comportamiento. Obviamente, aumentar el número de muestras provoca que la distribución de probabilidad obtenida sea cada vez más diferente de la original. En efecto, dado que las muestras obtenidas no admiten repeticiones, el caso extremo de obtener 500 muestras diferentes de una población de 500 valores, origina una probabilidad del 100 % de obtener cada uno de los elementos.

Una vez escogidos los nuevos rotámeros que deben formar parte del nuevo proceso de optimización local, es necesario realizar la carga de la información energética de los nuevos rotámeros, lo que implica acceder al disco para crear las nuevas estructuras de datos. De esta manera, finaliza una iteración global del proceso de optimización de proteínas.

6.5. Validación Experimental

La validación del sistema se ha llevado a cabo por parte del grupo del Prof. Jaramillo, utilizando una proteína de tamaño razonable (101 posiciones y 4501 rotámeros). Para ello se fijó el número de iteraciones locales a 100000, un número apropiado para este sistema, y se verificó la obtención del mismo mínimo global (-461.78) con programas de optimización diferentes.

A continuación se probó a recuperar el mínimo global mediante reducciones dimensionales sucesivas. Para ello se utilizaron los siguientes parámetros: $NDIV$, factor de reducción dimensional empleado por los procesos, $MINR$, número mínimo de rotámeros por posición, para controlar el efecto de posibles fluctuaciones y $GITER$, el número de iteraciones globales. En este estudio úni-

NDIV	2	5	10	20	30
MINR	25	50	25	50	50
MÍNIMO	-461.364	-461.058	-457.350	-458.962	-460.156

Tabla 6.3: Cálculo del mínimo valor de energía, utilizando diferentes factores de reducción dimensional, con 10 iteraciones globales.

NDIV	2	5	10	15	30
MINR	1	25	25	25	50
MÍNIMO	-461.768	-461.768	-461.768	-461.364	-461.768

Tabla 6.4: Cálculo del mínimo valor de energía, utilizando diferentes factores de reducción dimensional, con 100 iteraciones globales.

camente se consideraron los valores $GITER = 10$ y $GITER = 100$ para evaluar el comportamiento del sistema. Las ejecuciones se han realizado empleando 10 procesadores.

La Tabla 6.3 muestra el mínimo global obtenido por los procesos de optimización llevados a cabo con diferentes factores de reducción dimensional, realizando 10 iteraciones globales. De la misma manera, la Tabla 6.4 realiza lo propio para las optimizaciones con 100 iteraciones globales.

A la vista de los resultados obtenidos, es posible extraer las siguientes conclusiones. En primer lugar, aunque con 10 iteraciones globales la solución obtenida se aproxima mucho al mínimo global, con 100 iteraciones globales se alcanza, prácticamente en todos los casos, dicho mínimo. El caso correspondiente a $NDIV = 2$, $MINR = 1$ y 100 iteraciones globales significa que es posible alcanzar el mínimo global trabajando con la mitad de la memoria RAM requerida por la proteína completa, lo que supone un importante logro. Esta aproximación permite, por lo tanto, optimizar proteínas cuya información no cabe en una sola plataforma secuencial. Nótese además que, es posible también alcanzar el mínimo global con reducciones dimensionales mayores, aunque limitando el mínimo número de rotámeros por posición.

6.6. Conclusiones

En este capítulo se ha descrito la creación de una herramienta basada en Computación de Altas Prestaciones para la optimización de proteínas de propósito específico. Para ello, se ha realizado, en primer lugar, una optimización del código secuencial mediante la utilización de estructuras de datos eficientes. El uso de técnicas de almacenamiento de matrices dispersas, combinado con métodos eficientes que consideran la influencia de la jerarquía de memoria, ha permitido obtener importantes ventajas. Por un lado, se ha conseguido reducir el tiempo de ejecución de optimizaciones de proteínas, hasta en un factor de 9.47, para el caso de proteínas de dimensión elevada. Por otra parte, se ha conseguido reducir el consumo de memoria de la herramienta, requiriendo hasta un 18 % menos de memoria, para casos de estudio de gran dimensión.

En segundo lugar, con el objetivo de abordar la optimización de proteínas de dimensión elevada, cuyos requisitos de memoria pueden superar los disponibles en una sola máquina, se ha optado por realizar una implementación distribuida. De esta manera, se permite la utilización concurrente de múltiples procesadores para la optimización colaborativa de una proteína. Para ello se ha recurrido a técnicas de almacenamiento matricial distribuido, y computación paralela, para introducir un proceso de reducción dimensional. De esta manera, se utilizan múltiples procesadores que realizan optimizaciones parciales de la proteína, trabajando con parte de los datos globales. Mediante el intercambio periódico de información de estas optimizaciones, es posible lograr la optimización global de la proteína. Esta aproximación permite realizar el proceso de optimización, aunque la plataforma computacional no disponga de suficiente memoria para la carga total de la matriz de energías.

En este sentido, la utilización de técnicas de Computación de Altas Prestaciones, ha permitido mejorar las herramientas existentes para la optimización de proteínas.

Capítulo 7

La Computación Basada en Grid

"What is the Grid? Well, there is a short answer, and then there is a very long answer". The Grid Café, CERN.

En este capítulo se realiza una introducción a la Computación en Grid. En primer lugar se analiza brevemente la historia de la computación científica que ha propiciado la aparición de esta tecnología. A continuación se describen los objetivos que pretende alcanzar esta estrategia de computación, detallando los principales *middleware* desarrollados para dar soporte al concepto de Grid. Posteriormente, se describen con mayor detalle los dos middleware Grid que se han empleado para dar soporte a las ejecuciones realizadas. Por un lado, el estándar *de facto* actual empleado para la construcción de Grids, Globus Toolkit. Por otro lado, el middleware LCG-2 que, desarrollado en el marco de los proyectos europeos LCG y EGEE, soporta el mayor despliegue computacional en Grid existente en la actualidad. Finalmente, se aborda la problemática de la seguridad en un entorno Grid, describiendo los mecanismos ofrecidos por Globus que han servido como base para la infraestructura de seguridad del sistema de Computación en Grid desarrollado.

7.1. Antecedentes y Precursores

La evolución de la ciencia siempre ha venido marcada por la necesidad de utilizar las máximas prestaciones de los recursos de computación existentes. De hecho, el incremento de la capacidad de cómputo de los sistemas ha permitido abordar problemas de mayor envergadura.

En este sentido, la evolución de la capacidad de cómputo ha venido marcada fundamentalmente por la Ley de Moore, que establece que el número de transistores de los circuitos integrados se dobla cada 18 meses aproximadamente. Por lo tanto, cada vez existen máquinas más potentes con mayor capacidad de proceso. No es de extrañar por tanto que un PC actual disponga de una potencia de cómputo comparable a la de un supercomputador de la década anterior.

La solución tradicional que se ha utilizado para la ejecución de las aplicaciones científicas ha sido una computación centralizada de altas prestaciones, basada en un servidor central, al que se conectan los usuarios para ejecutar sus aplicaciones. Estas máquinas multiprocesador de memoria compartida sufren problemas de falta de escalabilidad, siendo equipos muy costosos que la mayor parte del tiempo desaprovechan ciclos de reloj, ya que la demanda de potencia suele ser muy puntual. Una alternativa que surge a los sistemas multiprocesadores viene dada por los sistemas multicomputador, dando lugar a la computación basada en clusters de PCs, que ofrecen un buen ratio coste/prestaciones para la ejecución de aplicaciones paralelas.

Una extensión al concepto de computación basada en clusters de PCs es la llamada computación basada en Intranet, que permite el uso de los equipos de una organización para la ejecución de trabajos secuenciales o paralelos por medio de una herramienta de control de la carga. Estas herramientas permiten la ejecución de tareas en los recursos de una organización para aprovechar los ciclos ociosos de los procesadores. De esta manera, se consigue aumentar la productividad al realizar múltiples ejecuciones concurrentes en los recursos infrautilizados. Esta estrategia de computación aprovecha los recursos informáticos y facilita la escalabilidad al permitir añadir nuevas máquinas al sistema de forma sencilla. Existe bastante software dedicado a esta filosofía de computación, como Sun Grid Engine [136] de Sun Microsystems, Condor [137] de la Universidad de Wisconsin, InnerGrid [138] de GridSystems o Nimrod [61] de la Universidad de Monash.

Sin embargo, cuando el ámbito de las aplicaciones atraviesa las fronteras de una única organización, surgen una serie de problemas que no aparecen en la computación basada en Intranet. El hecho de trabajar con diferentes dominios de administración, que pueden estar sujetos a diversas políticas de seguridad, gestión y localización, supone un nuevo desafío para las estrategias de computación. Por lo tanto, la creación de una infraestructura de computación distribuida, que aglutine recursos de diferentes centros geográficamente dispersos, requiere tecnologías y herramientas avanzadas.

7.2. Objetivos Principales

Con los recientes incrementos en el ancho de banda de las redes de comunicación se impulsó la idea de enlazar distintos recursos, geográficamente distribuidos, para crear una infraestructura global de computación conocida como el Grid [139, 140]. El término Grid fue acuñado a mediados de los 90 para proponer una infraestructura de computación distribuida para la ingeniería y ciencia avanzada [141].

Inicialmente, los conceptos y tecnologías relacionados con el Grid fueron desarrollados para permitir la compartición de recursos en las colaboraciones científicas [142, 143]. La necesidad de este tipo de colaboraciones no era únicamente la compartición de datos experimentales sino también aplicaciones, recursos computacionales e instrumental específico como telescopios y microscopios [141].

Ante esta situación emergieron multitud de aplicaciones precisando computación y almacenamiento de datos distribuido para la ejecución de aplicaciones computacionalmente complejas, el acceso a bancos de datos distribuidos, la visualización colaborativa de datos y la compartición de instrumentos. Por lo tanto, existía una demanda de compartición de recursos coordinada para la resolución de problemas, en el marco de colaboraciones dinámicas, entre múltiples instituciones [144].

En realidad, el Grid puede ser considerado como un servicio para la compartición de potencia computacional y de capacidad de almacenamiento, al igual que el Web es un servicio para la compartición de información a través de Internet. Por lo tanto, la computación en Grid es una tecnología que permite la compartición de recursos entre máquinas pertenecientes a dominios de administración diferentes de manera transparente, eficiente y segura.

La principal idea que subyace a las tecnologías Grid es ofrecer una interfaz homogénea y estándar para poder acceder a los recursos. Estos recursos pueden ser máquinas de cálculo como supercomputadores, clusters de PCs o sistemas de almacenamiento, aunque también se pueden compartir fuentes de información como bases de datos o incluso instrumentos científicos.

Uno de los puntos importantes que ofrece esta tecnología es que permite la interconexión de recursos entre organizaciones diferentes respetando las políticas internas de seguridad, así como todo el software de gestión de recursos que ya disponga la organización. Además, las tecnologías Grid permiten el uso colaborativo de diferentes recursos hardware para crear un conjunto de recursos de computación formado por equipos heterogéneos y geográficamente distribuidos. Mediante el empleo de esta tecnología es posible superar los límites computacionales de una única organización, accediendo a nodos de cómputo de otras entidades.

Es bastante habitual, cuando se habla de las tecnologías Grid, establecer una analogía con la red eléctrica [145]. Cuando un usuario conecta cualquier aparato a la red eléctrica rara vez se preocupa de dónde proviene la energía que va a consumir, simplemente recibe un servicio: la energía eléctrica. Análogamente, si un usuario desea realizar algún cómputo intensivo podría acudir al Grid para solventar esa necesidad, mediante la utilización de un servicio de cómputo en el que la ejecución se realice de manera descentralizada. En este sentido, el usuario tendría acceso al servicio que necesita de manera transparente.

En realidad, tanto la red eléctrica como los objetivos fundamentales de la Computación en Grid cumplen cuatro características fundamentales. En primer lugar, por debajo existe una **infraestructura**. Así como en la red eléctrica se interconectan diferentes plantas generadoras de energía, en el Grid se interconectan recursos de computación como PCs, estaciones de trabajo o servidores de almacenamiento. En segundo lugar, en ambos casos se pretende establecer un acceso **transparente**, es decir, el usuario receptor del servicio no ha de preocuparse sobre qué planta suministra la energía eléctrica, o qué ordenador está realizando los cálculos, ya que el Grid se encargaría de resolver todos los detalles de implementación. En tercer lugar, ambas redes tratan de ser **accesibles**, es decir, que

el acceso a ellas sea posible desde la mayor parte del planeta. La red eléctrica, está muy extendida por la mayor parte de los países mientras que el Grid trata de conseguir que los recursos de computación sean accesibles desde diferentes plataformas, incluyendo PCs de sobremesa, PDAs, o teléfonos móviles. Finalmente, en ambos casos se obtiene un **servicio**. En el caso de la red eléctrica se obtiene electricidad mientras que en el Grid se obtiene un servicio de computación. Obviamente, al recibir un servicio es posible que se genere un cargo por parte de la entidad que ha ofrecido los recursos de computación, lo que se conoce en la bibliografía como economía de Grid [146].

Es importante destacar que, en la actualidad, esta visión idílica del Grid todavía no ha sido alcanzada plenamente. Sin embargo, se están llevando a cabo importantes labores de investigación para alcanzar los principales objetivos de las tecnologías Grid. Por lo tanto, es importante conocer la trayectoria de los principales proyectos de investigación relacionados y el estado actual de esta tecnología.

7.3. Estado del Arte en Computación en Grid

En la actualidad existen multitud de proyectos relacionados con las tecnologías Grid. Entre los proyectos desarrollados en EE.UU. cabe citar PPDG (Particle Physics Data Grid Collaboratory Pilot) que trata de desarrollar y poner en producción un despliegue Grid para integrar verticalmente aplicaciones específicas de experimentos nucleares y de física de altas energías. Otro proyecto importante, en cuanto a entidades involucradas y nivel de financiación recibida, es el GriPhyN (Grid Physics Network) encargado del desarrollo de tecnologías Grid para dar soporte a proyectos científicos y de ingeniería que se encargan de recolectar y analizar conjuntos de datos distribuidos del orden de petabytes. Ambos proyectos están enmarcados en el área de física de partículas pero existen otros dedicados a implantar esta tecnología en otros campos de la ingeniería. Este es el caso del proyecto NEESit (Enabling the Network for Earthquake Engineering Simulation), que interconecta una serie de centros a lo largo de EE.UU para colaborar en la simulación y en la predicción de terremotos. Por último, es interesante destacar también el proyecto TeraGrid, financiado por la National Science Foundation (NSF) y dedicado a la construcción y el despliegue de la mayor infraestructura distribuida para investigaciones científicas.

En Europa, la Unión Europea ha definido las tecnologías Grid como un área prioritaria para su desarrollo en el 6º y 7º Programa Marco, los instrumentos financieros de la investigación europea competitiva de mayor relevancia. Además, existen multitud de proyectos de investigación, tanto dedicados al desarrollo de middlewares Grid como en la aplicación del Grid en múltiples campos de la ingeniería.

El proyecto CrossGrid, finalizado en 2005, investigó en el uso de las tecnologías Grid en las aplicaciones interactivas, es decir, que requieren interactuar con el usuario durante su ejecución. Este

tipo de aplicaciones, que pueden ser además computacionalmente intensivas e involucrar enormes cantidades de datos, requieren un tiempo de respuesta acotado, lo que impone una problemática adicional.

En el año 2000 comenzó el proyecto EDG (European DataGrid) bajo el programa comunitario IST (Information Society Technologies) coordinado por el CERN (Centre Européen pour la Recherche Nucléaire), el laboratorio Europeo para el estudio de física de partículas. El principal objetivo de este proyecto fue la construcción de una infraestructura de computación de nueva generación que proporcionase computación intensiva, permitiendo el análisis de grandes bases de datos de manera distribuida a través de múltiples comunidades virtuales. Este proyecto finalizó satisfactoriamente a finales de Marzo 2004 y gran parte del desarrollo realizado en el mismo sirvió como punto de partida para el proyecto LCG [64].

El proyecto LCG (Large Hadron Collider Computing Grid Project) comenzó en 2004 con el objetivo de desarrollar toda la infraestructura de computación necesaria para la simulación, el procesado y el análisis de los datos proporcionados por el *Large Hadron Collider* (LHC), el acelerador de partículas más potente del mundo que actualmente está siendo construido en el CERN. Cuando el LHC esté completamente operativo, proporcionará el soporte para la realización de experimentos de más de 5000 científicos en 500 centros de investigación y universidades a lo largo del mundo. Se estima que el análisis de todos los datos generados, incluidas las comparaciones con las simulaciones teóricas, requerirá del orden de 100000 CPUs, teniendo en cuenta las capacidades de un PC actual. Obviamente, un solo centro no puede absorber tal capacidad de cómputo y, por lo tanto, el proyecto LCG trata de construir la infraestructura que permita distribuir todo el proceso de cálculo a través de múltiples centros geográficamente distribuidos. En este sentido, este proyecto centra sus esfuerzos en el desarrollo de un middleware Grid enfocado al área de la física de altas energías.

A la vista del potencial ofrecido por el Grid, surgió la necesidad de crear una infraestructura para la utilización del Grid en otros campos de la ingeniería como, por ejemplo, la biomedicina, geofísica, modelización climática, nanotecnología, astronomía, etc.

En este sentido el proyecto europeo EGEE[1] (Enabling Grids for E-sciencE) tomó como punto de partida todo el conocimiento generado en los proyectos EDG y LCG. Finalizado en 2006, su principal objetivo fue el desarrollo de una infraestructura Grid disponible para la comunidad científica las 24 horas del día. Este proyecto aglutinó a expertos de más de 27 países y proporcionó, a los investigadores en el ámbito académico y la industria, el acceso a múltiples recursos computacionales independientemente de su localización geográfica. Debido a los buenos resultados obtenidos el proyecto fue renovado (EGEE-2) durante dos años más. Actualmente aglutina a 90 instituciones de 32 países, convirtiendo a EGEE en el proyecto de Computación en Grid más ambicioso actualmente.

Uno de los principales objetivos del proyecto EGEE se centra en atraer el interés de nuevas

[1] Como curiosidad, inicialmente se denominó Enabling Grids for E-science in Europe, pero la posterior participación de socios extracomunitarios propició el cambio de denominación.

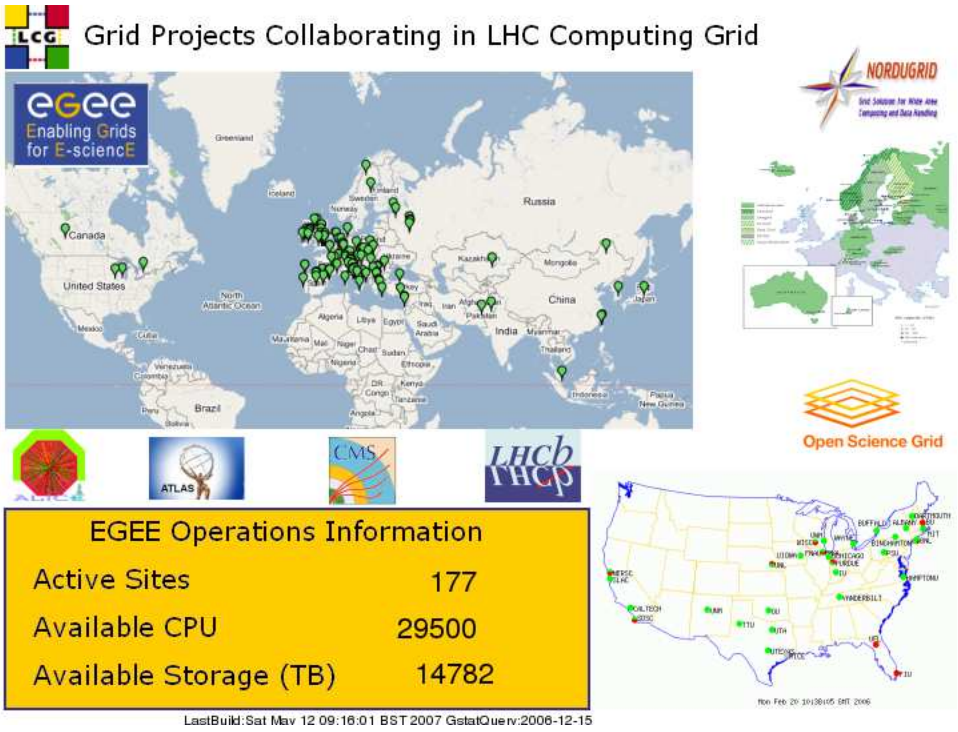


Figura 7-1: Distribución geográfica de los recursos computacionales disponibles en el marco de los proyectos LCG y EGEE (Mayo, 2007).

comunidades de usuarios hacia las tecnologías Grid. Este proyecto pretende concentrarse en tres áreas clave:

- Construir un Grid consistente, robusto y seguro que atraiga recursos de computación adicionales por parte de diferentes organizaciones.
- Mejorar y mantener de manera continuada el middleware Grid para ofrecer un servicio confiable a los usuarios.
- Atraer a nuevos usuarios, tanto industriales como científicos, y asegurarse de que reciben un entrenamiento y un soporte adecuado.

En la actualidad, el proyecto EGEE comparte la infraestructura de recursos computacionales de LCG. La Figura 7-1 muestra un mapa con la distribución actual de las máquinas disponibles en esta infraestructura Grid. Por lo general, al conjunto de recursos de computación de un Grid se le suele denominar *testbed*.

En este sentido, cada centro de investigación aporta un conjunto de recursos al testbed de EGEE y el acceso a los mismos se realiza de manera organizada. En realidad, el objetivo principal de las tecnologías Grid es integrar, virtualizar y gestionar los recursos y servicios dentro de *Organizaciones Virtuales* (VO) distribuidas, heterogéneas y dinámicas a través de diferentes organizaciones físicas.

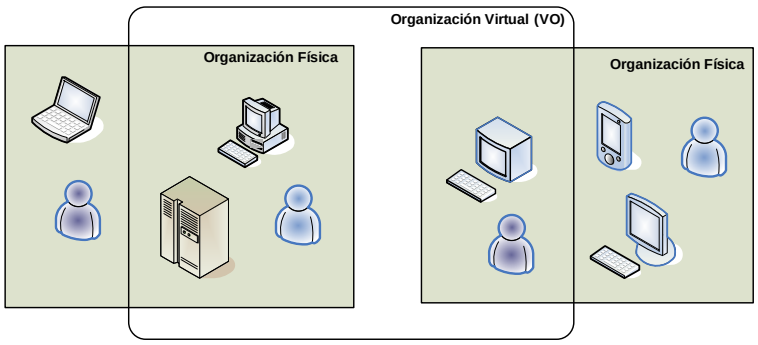


Figura 7-2: Estructura de una Organización Virtual que aglutina usuarios y recursos de diferentes organizaciones físicas.

Tal y como se muestra en la Figura 7-2, una organización virtual engloba un conjunto de individuos y recursos computacionales que permiten la resolución de problemas de manera colaborativa u otros propósitos. Las organizaciones virtuales proporcionan el contexto adecuado para vincular a los usuarios un conjunto de recursos. La compartición de recursos dentro de una VO es altamente controlada, donde los proveedores y consumidores de recursos están claramente diferenciados. En la actualidad existen diversas VOs en el marco de EGEE. La mayoría de ellas están relacionadas con la física de partículas, aunque existen otras nuevas para realizar colaboraciones en campos como la biomedicina, química computacional, sismología, etc.

En España, la iniciativa IRISGrid [147] surgió en el año 2002 con el objetivo de coordinar a nivel académico y científico a los grupos de investigación interesados en el desarrollo, implantación y aplicación de las tecnologías Grid. Además de las labores puramente de coordinación, IRISGrid tiene como objetivo crear la infraestructura Grid nacional que permita el uso de esta tecnología en aplicaciones diversas y el desarrollo e innovación en este campo. Actualmente, en la iniciativa IRISGrid participan 23 grupos, centros e institutos de investigación de España. Esta iniciativa pretende unir recursos distribuidos geográficamente para que los grupos involucrados tengan un testbed que soporte la investigación en cualquiera de las áreas de aplicación del Grid.

Por lo tanto, el campo de las tecnologías Grid es un campo muy activo que se encuentra de actualidad ya que su espectro de aplicación abarca múltiples áreas de la ciencia y la ingeniería. En ese sentido, existen múltiples proyectos relacionados con el desarrollo de middlewares Grid, como Polder [148], desarrollado en la Universidad de Amsterdam, o Condor [137], desarrollado en la Universidad de Wisconsin. Sin embargo, en los últimos años Globus Toolkit [149, 150] se ha convertido en el estándar de facto para el despliegue de Grids computacionales, principalmente por la adopción en sus últimas versiones de protocolos estándares y abiertos para el desarrollo de sus componentes.

7.4. La Alianza Globus y el Middleware Globus Toolkit

La alianza Globus [151] es una comunidad de organizaciones e individuos que desarrollan tecnologías fundamentales para el Grid. Para esta alianza, un Grid permite a los individuos la compartición de potencia computacional, bases de datos, instrumentos y otras herramientas on-line de manera segura, a través de las barreras institucionales o geográficas sin sacrificar la autonomía local. El consorcio Globus, formado tanto por académicos como por grandes empresas como HP, IBM e Intel, se encarga en gran parte del desarrollo de código abierto de Globus Toolkit (GT) junto con una gran comunidad de usuarios.

GT es una herramienta de libre distribución que permite la creación de Grids, facilitando la compartición de recursos computacionales de manera segura, a través de diferentes organizaciones, sin sacrificar la autonomía local de los diferentes dominios de administración que integren el Grid [144, 152]. Este paquete incluye servicios y librerías para la monitorización y gestión de recursos así como para la seguridad y gestión de ficheros. En un entorno de múltiples organizaciones aparece el problema de la heterogeneidad, donde existen ordenadores de arquitecturas diferentes y redes de datos diversas, que pueden dar lugar a problemas de incompatibilidad.

GT ha sido diseñado para eliminar los principales obstáculos que surgen de una colaboración entre organizaciones. Los principales servicios, interfaces y protocolos que ofrece Globus permiten a los usuarios el acceso a los recursos remotos como si se trataran de recursos locales, al mismo tiempo que preserva el control local sobre *quién* y *cuando* se accede a los recursos compartidos.

La versión 1.0 de Globus (GT1) [149] apareció en el año 1998. La versión 2.0 (GT2) estuvo disponible en el año 2002. Globus Toolkit 3.0 (GT3) [153] apareció en el año 2003 y, la más reciente, la versión 4.0 (GT4) [150] apareció en el año 2005. Inicialmente, tanto GT1 como GT2 empleaban interfaces propietarias para la comunicación entre los diferentes servicios aunque generalmente se implementaban utilizando protocolos estándares como LDAP (Lightweight Directory Access Protocol), HTTP (Hypertext Transfer Protocol) o FTP (File Transfer Protocol). La aparición de GT2 supuso la adopción de Globus Toolkit por parte de numerosos proyectos de investigación relacionados con el Grid, y se convirtió automáticamente en un estándar de facto para el despliegue de Grids computacionales. Con el lanzamiento de GT3 se realizó un acercamiento de este software a la tecnología de Servicios Web (Web Services, WS) [154] con el objetivo de exponer interfaces y comunicación basada en protocolos estándar para la interacción entre diferentes servicios. GT4 representó un importante avance en términos de funcionalidad, conformidad a estándares y usabilidad.

En este sentido, la aparición de GT4 ha marcado una clara diferencia entre los servicios existentes anteriormente en GT2 (conocidos ahora como pre-Web Services (pre-WS)) y los servicios ofrecidos por GT4 basados en interfaces estándar (Web Services). No obstante, dada la cantidad de herramientas desarrolladas que utilizan los servicios de GT2, todavía hoy se siguen manteniendo los servicios pre-WS por compatibilidad.

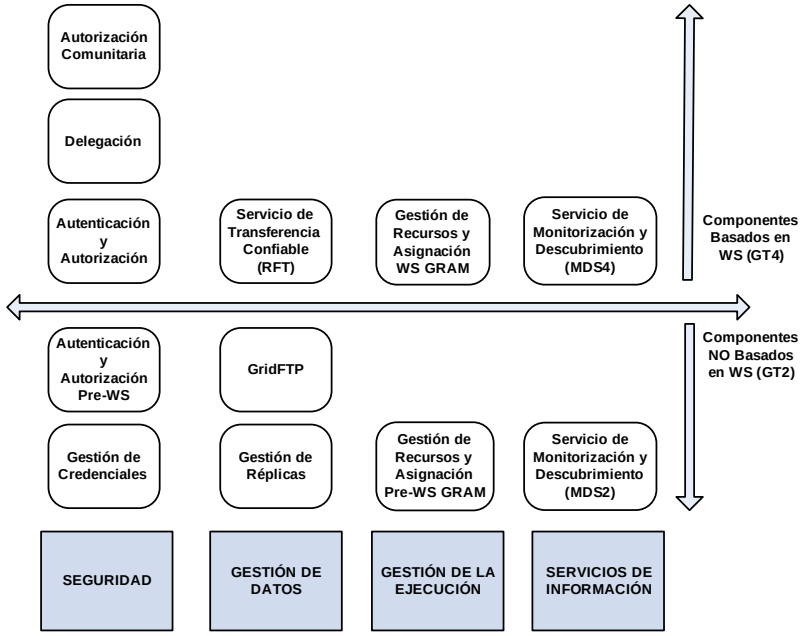


Figura 7-3: Diagrama de los principales componentes de GT4 en cada una de las cuatro áreas principales.

7.4.1. Componentes Principales de Globus Toolkit

La Figura 7-3 muestra los principales componentes ofrecidos por GT en cada una de las principales áreas involucradas en el Grid: Seguridad, Gestión de Datos, Gestión de la Ejecución y Servicios de Información. En ella se observa como se realiza una distinción entre los componentes heredados de GT2, que no están orientados a servicios, y los más recientes incluidos en GT4. A continuación se detalla brevemente la funcionalidad ofrecida en cada una de estas áreas:

Gestión de Recursos

El servicio GRAM (Globus Resource Allocation Manager) simplifica el uso de sistemas remotos proporcionando una interfaz estándar para la ejecución de tareas en ellos. GRAM reduce el número de mecanismos necesarios para el uso de recursos remotos ya que cada sistema puede emplear, a priori, multitud de mecanismos de gestión como planificadores locales, gestores de colas o sistemas de reservas diferentes (Local Resource Management System, LRMS). De esta manera, el usuario y desarrollador de aplicaciones únicamente ha de conocer los servicios que ofrece GRAM, en lugar de conocer los mecanismos particulares necesarios para interactuar con los LRMS para la ejecución de aplicaciones en cada una de las máquinas del Grid. Nótese en la Figura 7-3 que se distingue entre el servicio pre-WS GRAM, desarrollado originalmente en GT2, y el servicio WS GRAM, desarrollado en GT4.

Servicios de Información

Este apartado incluye al servicio MDS (Monitoring and Discovery Service) [155], encargado de proporcionar información sobre los recursos existentes en un Grid. Este servicio proporciona un mecanismo estándar para la publicación y el acceso a la información de los recursos computacionales. La implementación de MDS, como componente pre-WS (MDS2), involucra la existencia de una estructura jerárquica compuesta por dos elementos principales. Por un lado está el servicio GRIS (Grid Resource Information Service) [155], encargado de la recopilación de información en un recurso. Por otro lado, el servicio GIIS (Grid Index Information Service) [155] permite la creación de un directorio de información agregada que aglutina los datos recopilados por parte de múltiples servidores GRIS. El servicio MDS2 utiliza el protocolo LDAP para el acceso a la información publicada tanto por el recurso computacional como el servidor GIIS. En GT4, el servicio MDS4 es un servicio Grid que proporciona la información en formato XML a través de las interfaces requeridas empleando el lenguaje de consulta XPath (XML Path Language) [156].

La importancia del servicio MDS radica en que permite publicar el estado de los recursos de un Grid como puede ser la carga del procesador, la memoria disponible, el sistema operativo, así como información proporcionada por los LRMS instalados en el recurso.

Gestión de Datos

La gestión de datos incluye la habilidad de acceder y gestionar los datos dentro de un entorno Grid. Este módulo incluye el componente GridFTP, encargado de realizar transferencias de datos seguras y de altas prestaciones entre los recursos computacionales de un Grid. GridFTP es un protocolo de transferencia de datos basado en el conocido protocolo FTP, y optimizado para redes de área extensa (WANs) con grandes anchos de banda.

Adicionalmente, el Servicio de Transferencia Confiable (Reliable File Transfer, RFT) permite almacenar el estado de las transferencias en una base de datos para permitir cierta tolerancia a fallos y continuidad en el servicio.

7.4.2. Mecanismos de Seguridad Basados en Globus: GSI

Los aspectos de seguridad dentro de un Grid computacional son realmente importantes, dado que involucra el acceso a recursos de diferentes organizaciones. Para ello, en esta sección se repasa brevemente los principales conceptos de seguridad que posteriormente serán empleados por el sistema de Computación en Grid desarrollado.

En primer lugar, es importante realizar un proceso de *autenticación* que verifique la identidad de todos los participantes en el Grid, ya sean usuarios, recursos computacionales o servicios Grid. En segundo lugar, se debe garantizar que los recursos únicamente sean accedidos por usuarios con

ese privilegio y, por lo tanto, debe producirse un proceso de *autorización*. Nótese que la autorización está estrechamente ligada a la autenticación, puesto que antes de conceder un permiso de acceso es necesario verificar la identidad del cliente.

En tercer lugar, a menudo es imprescindible garantizar la *privacidad* en la comunicación entre un cliente y el servicio al que accede, de manera que ésta únicamente sea entendible por ambos. Si una tercera persona consigue interceptar un mensaje entre ambos no debe ser capaz de entender el contenido de la conversación. Finalmente, la propiedad de *integridad* garantiza que el mensaje transmitido llegue al destinatario tal y como fue enviado por el cliente. Por lo tanto, si una tercera persona consigue introducir alguna modificación en el mensaje, el destinatario debe poder detectar que ha sido modificado desde que fue enviado.

Para poder dar una respuesta a los problemas de seguridad que surgen en un Grid, GT incluye una capa de seguridad denominada GSI (Grid Security Infrastructure) [157, 158]. GSI utiliza los conceptos de criptografía de clave pública, también conocida como criptografía asimétrica. La funcionalidad ofrecida se puede resumir en tres puntos:

- Ofrecer servicios de comunicación segura, autenticada y confidencial, entre los elementos de un Grid computacional.
- Soportar esquemas de seguridad a través de las diferentes organizaciones involucradas en un Grid, descartando la creación de un sistema de seguridad gestionado de manera central.
- Permitir la identificación única para los usuarios del Grid, incluyendo la delegación de credenciales entre recursos.

Criptografía de Clave Publica

La criptografía es el arte o ciencia de cifrar y descifrar información utilizando técnicas matemáticas que hagan posible el intercambio de mensajes de manera que sólo puedan ser leídos por las personas a quienes van dirigidos. Específicamente, la criptografía de clave pública o criptografía asimétrica [159] está basada en la utilización de dos claves diferentes:

- Clave Privada. Esta clave debe ser conocida única y exclusivamente por su propietario.
- Clave Pública. Esta clave es conocida por cualquier usuario.

Si se utiliza cualquiera de ellas para cifrar un documento, la otra clave es la única que se puede utilizar para descifrar el contenido. El mecanismo de comunicación segura entre emisor y receptor se realiza empleando el esquema que se observa en la Figura 7-4. El emisor utiliza la clave pública del receptor, conocida por todos, para cifrar el mensaje. Dicho mensaje únicamente podrá ser descifrado empleando la clave privada del receptor y, por lo tanto, se garantiza que el receptor es el único que puede acceder al contenido del mensaje.

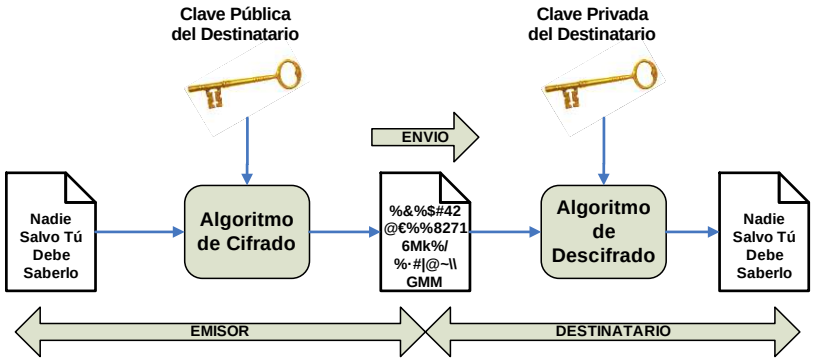


Figura 7-4: Utilización de las claves en criptografía de clave pública para garantizar la privacidad en la conversación.

Con el objetivo de poder garantizar la identidad de las partes involucradas en un Grid se utiliza el concepto de certificado digital.

Certificados Digitales

Un certificado digital es un documento electrónico que certifica el vínculo entre un usuario y su clave pública. Este documento viene firmado por una Entidad Certificadora (Certification Authority, CA) que da fe de la identidad del propietario de la clave.

Cualquier usuario del Grid debe disponer de un certificado de usuario que lo identifica de manera unívoca debido a la existencia de una entidad certificadora que así lo asegura. Es importante destacar que, en última instancia, se trata de un proceso en el que se debe confiar en una o varias entidades certificadoras que garantizan la identidad de los usuarios.

El certificado de usuario viene codificado en el formato estándar X.509 [160] que contiene, entre otros, los siguientes campos:

- Identidad: Es el nombre del usuario codificado como un Nombre Distinguido (Distinguished Name, DN) para que sea único. Está compuesto por una lista de valores separados por coma en los que se indica, entre otros, la Organización (O), la Unidad de Organización (Organizational Unit, OU), el Nombre Común (Common Name, CN) y el país (C).

- Clave pública del usuario: Incluye tanto la clave como la información acerca del algoritmo empleado para generarla.

- Identidad del emisor: Corresponde al DN de la CA que firmó el certificado.

- Firma digital: El certificado incluye una firma digital de toda la información del certificado para garantizar que la información incluida no ha sido modificada.

La Figura 7-5 muestra un ejemplo de certificado digital X.509 perteneciente al usuario con DN: "C=ES, O=UPV, OU=GRyCAP, CN=German Molto".

En realidad, no solo los usuarios disponen de un certificado sino también los recursos computacionales. De esta manera, es posible garantizar la autenticidad de los máquinas involucradas en un Grid.

En GSI, como medida de seguridad adicional, la clave privada va cifrada con una contraseña que únicamente debe ser conocida por su propietario. Esta situación implica que cada vez que se realizara una autenticación, sería imprescindible que el usuario teclease su contraseña para disponer de acceso a la clave privada. Como se puede observar, esta situación no es práctica en un entorno donde la autenticación es un mecanismo frecuente. Para poder solventar este problema, GSI implementa un mecanismo de delegación de credenciales a un certificado *proxy*.

Un proxy no es más que un certificado de usuario con un tiempo de validez límite cuya característica principal radica en que está firmado por las credenciales del usuario que lo crea. De esta manera, se consigue que el usuario delegue sus credenciales al proxy durante el tiempo de vida de éste último. Con esta estrategia se consigue que el usuario únicamente tenga que introducir la contraseña para acceder a la clave privada en la creación del proxy y no cada vez que se accede a un recurso, proceso que se conoce como identificación única (*single sign-on*).

La Figura 7-6 resume de manera esquemática los pasos a realizar para obtener un certificado digital de usuario, y el correspondiente proxy, para trabajar en un despliegue Grid basado en Globus. En primer lugar, el usuario genera una solicitud de certificado además de una clave privada que únicamente deberá ser conocida por él. A continuación presenta dicha solicitud ante una *Autoridad de Registro* (Registration Authority, RA) junto con un documento que garantice la identidad del usuario, como por ejemplo un DNI (Documento Nacional de Identidad). La RA valida la solicitud del usuario dando fe de que la petición ha sido sometida por el usuario que dice ser, y autoriza la emisión del certificado a la entidad certificadora. La CA firma la solicitud del certificado del usuario y le hace entrega del mismo. A partir de este momento, el usuario dispone de un certificado válido para trabajar en un Grid. Opcionalmente, el usuario puede solicitar su pertenencia como miembro de una Organización Virtual (VO) para poder disponer de acceso a los recursos que formen parte de dicha VO. Una vez confirmada el alta, el usuario procede a crear un proxy mediante el cual se realiza el acceso a los recursos computacionales, y que identifica al usuario durante el periodo de vida del proxy. Es posible encontrar una descripción más detallada de la funcionalidad ofrecida por GSI en [161].

Globus Toolkit no es el único middleware Grid existente en la actualidad. Sin embargo, ha sido el marco de referencia de desarrollos posteriores llevados a cabo por diferentes proyectos de investigación. Entre ellos, las infraestructuras Grid de producción tienen como objetivo el despliegue de recursos computacionales disponibles para la comunidad científica 24 horas al día. A continuación

```
Certificate:
  Data:
      Version: 3 (0x2)
      Serial Number:
          44:ec:4a:cd:00:01:00:00:00:cf
      Signature Algorithm: sha1WithRSAEncryption
      Issuer: C=ES, ST=Valencia, L=Valencia, O=UPV, OU=GRyCAP, CN=GRyCAP-UPV CA
      Validity
          Not Before: Jun 12 15:24:40 2006 GMT
          Not After : Jun 12 15:34:40 2007 GMT
      Subject: C=ES, O=UPV, OU=GRyCAP, CN=German Molto
      Subject Public Key Info:
          Public Key Algorithm: rsaEncryption
          RSA Public Key: (1024 bit)
              Modulus (1024 bit):
                  00:c0:35:17:e6:a6:7d:47:97:02:7a:66:c5:63:62:
                  ad:85:d4:ec:2f:d8:7d:10:8f:d5:ef:aa:a2:b9:e5:
                  24:83:0c:49:31:6f:f6:b6:7b:bd:49:62:60:54:a5:
                  e9:02:05:5a:a3:7d:a4:55:66:fd:fd:fb:97:cb:b5:
                  11:40:b0:1a:b4:51:d6:63:57:bc:32:10:3a:ca:0a:
                  bd:4f:2b:db:b5:60:61:a8:1e:4d:b4:74:bf:df:a6:
                  e7:33:14:e0:5c:58:36:6b:37:04:d7:11:09:93:c3:
                  d6:db:17:1d:b8:b9:08:6a:62:33:8b:22:82:8e:3c:
                  f8:18:5d:ed:3b:d8:5d:91:1f
              Exponent: 65537 (0x10001)
      X509v3 extensions:
          X509v3 Subject Key Identifier:
              96:EF:A8:08:08:B0:43:EA:CD:90:54:21:06:9E:7A:66:24:0C:2B:99
          X509v3 Authority Key Identifier:
              keyid:8C:54:A2:65:CA:D8:77:79:77:D9:1B:4A:E1:52:E3:90:36:FA:DC:B9
              DirName:/C=ES/ST=Valencia/L=Valencia/O=UPV/OU=GRyCAP/CN=GRyCAP-UPV CA
              serial:79:7D:53:98:BA:B2:F1:B0:4A:C0:6C:2F:41:D1:01:E6

          X509v3 CRL Distribution Points:
              URI:http://ra/CertEnroll/GRyCAP-UPV%20CA.crl
              URI:file://\\ra\CertEnroll\GRyCAP-UPV%20CA.crl

          Authority Information Access:
              CA Issuers - URI:http://ra/CertEnroll/ra_GRyCAP-UPV%20CA(1).crt
              CA Issuers - URI:file://\\ra\CertEnroll\ra_GRyCAP-UPV%20CA(1).crt

  Signature Algorithm: sha1WithRSAEncryption
      13:03:d8:db:82:38:14:75:69:23:7f:6a:01:da:b6:10:0a:24:
      89:ce:4f:ae:f6:7a:51:40:ea:0b:83:94:69:40:b0:5f:2f:a3:
      fa:18:13:65:10:58:39:70:6a:5b:2f:e0:27:cc:e5:33:78:99:
      51:e0:75:5e:0e:e3:4f:41:cc:ce
```

Figura 7-5: Ejemplo de certificado digital de usuario X.509.

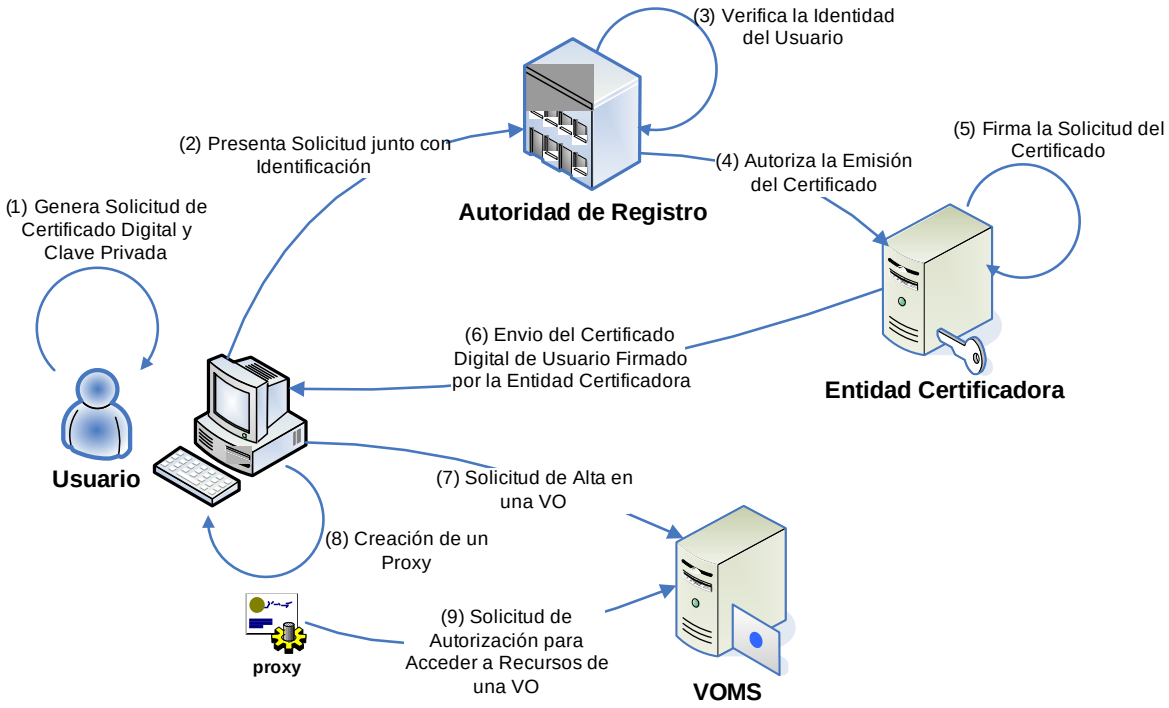


Figura 7-6: Diagrama de obtención de un certificado digital de usuario y posterior creación de un proxy.

se describe brevemente el middleware desarrollado en el marco del proyecto LCG, denominado LCG-2 y que actualmente constituye la infraestructura Grid de producción más grande del mundo.

7.5. Infraestructuras Grid de Producción: LCG-2

En la actualidad, el importante avance realizado en los principales proyectos europeos relacionados con la Computación en Grid (EDG, LCG y EGEE) ha permitido el despliegue y puesta en producción de una infraestructura Grid a gran escala. Los objetivos iniciales incluían el despliegue de componentes que pudieran procesar la ingente cantidad de información obtenida por el acelerador de partículas LHC. Posteriormente, el proyecto EGEE amplió el espectro a otros campos de la ciencia y de la ingeniería. Tanto LCG como EGEE comparten actualmente la misma infraestructura computacional y, por lo tanto, es común hablar indistintamente de recursos pertenecientes a LCG o a EGEE.

El middleware LCG-2 pretende dar solución a la problemática del despliegue y utilización de una infraestructura de tales características. Este software es el instalado en los centros que aportan recursos a esta infraestructura Grid que, actualmente, abarca más de 170 centros a lo largo del mundo, aunque principalmente situados en Europa.

Los usuarios de una infraestructura Grid basada en LCG-2 se agrupan en diferentes VOs. Ac-

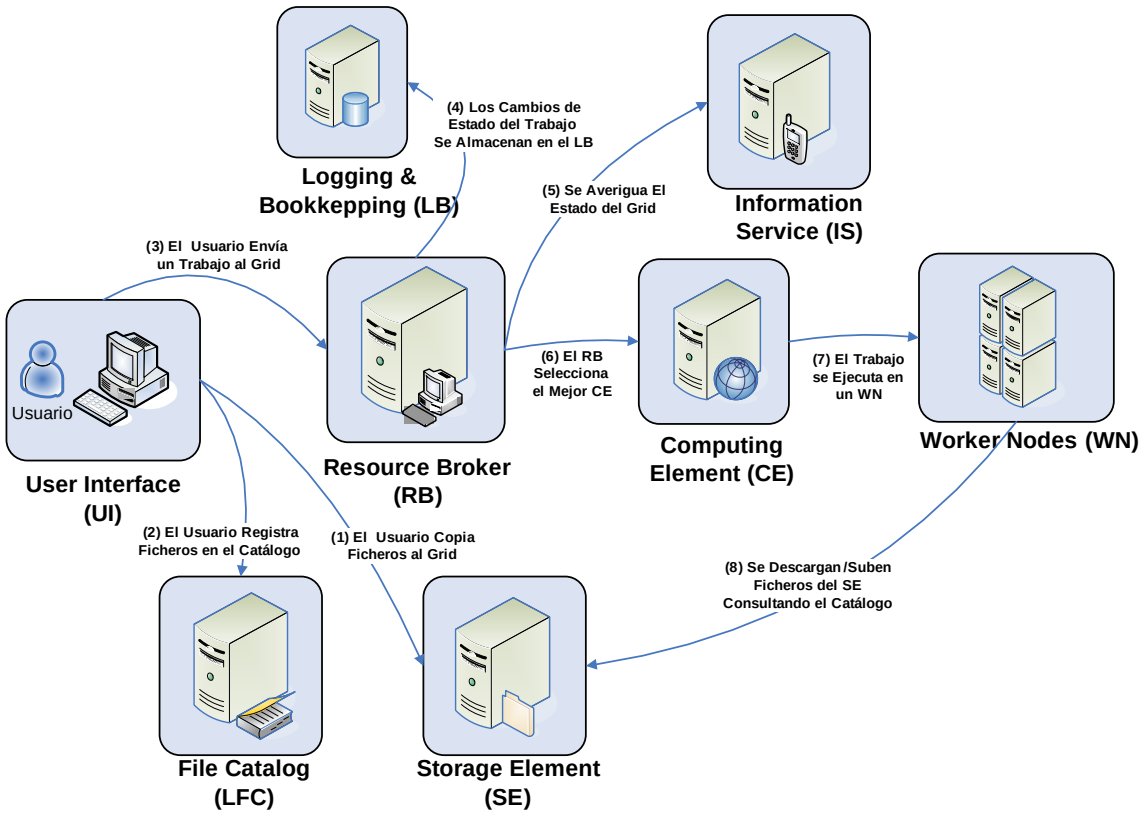


Figura 7-7: Arquitectura general del middleware LCG-2.

tualmente, las VOs mantienen una correspondencia directa con proyectos relacionados con los experimentos llevados a cabo con el LHC, como por ejemplo ALICE, ATLAS, CMS y LHCb. Además, en el marco del proyecto EGEE se han incluido algunas VOs adicionales, como *BIOMED*, que agrupa a la comunidad de usuarios que realizan investigaciones en el campo de la biomedicina.

La Figura 7-7 muestra un esquema de la arquitectura general del middleware LCG-2. A continuación se describen los principales componentes encontrados en un despliegue LCG-2. Como norma general, cada servicio corresponde a una máquina diferente, aunque es posible combinar varios servicios en una misma máquina:

- **User Interface (UI)**. Es el punto de entrada a un Grid basado en LCG-2 donde el usuario dispone tanto de su certificado personal como de su clave privada. Desde un UI el usuario puede ser autenticado y autorizado para acceder a los recursos de LCG-2. Las principales acciones que pueden realizarse desde un UI son: lanzamiento de trabajos para su ejecución; obtener un listado de máquinas disponibles para ejecutar un trabajo; averiguar el estado de los trabajos y cancelarlos; obtener el resultado de aquellos finalizados y gestionar ficheros y réplicas para ser almacenadas en el Grid.

- **Resource Broker (RB)**. Este servicio se encarga de recibir las peticiones de ejecución de

trabajos desde los UI y gestionar su ejecución en los diferentes elementos de computación (CE). Para ello debe consultar el estado del Grid al Sistema de Información (IS). Este servicio es, por tanto, el encargado de realizar la distribución de la carga computacional entre los diferentes recursos del Grid. En un despliegue LCG-2 pueden haber múltiples RBs.

- **Computing Element (CE)**. Este servicio representa a un elemento de computación encargado de recibir solicitudes de ejecución de trabajos. Un CE viene identificado por una cadena que referencia a una determinada cola de ejecución en la máquina. Un ejemplo de CE es el siguiente:

  ```
  ramses.dsic.upv.es:2119/jobmanager-pbs-biomedg
  ```

  Diferentes colas sobre una misma máquina son consideradas diferentes CEs. Por lo general, este servicio se instala en máquinas que constan de un gestor de trabajos (LRMS) para poder asignar las tareas recibidos a los diferentes nodos de computación (Worker Nodes). El ejemplo más claro consiste en instalar el CE en el nodo principal de un cluster, donde los nodos internos de cálculo son WN.

- **Worker Node (WN)**. Este servicio se encarga de gestionar la ejecución de los trabajos recibidos por un CE. Por lo general, los WN disponen de conectividad a la red, permitiendo el acceso a otros servicios.

- **Storage Element (SE)**. Este servicio proporciona un acceso uniforme a los recursos de almacenamiento. Un SE puede controlar desde simples servidores de disco, arrays de disco de tamaño medio e incluso dispositivos de almacenamiento masivo. Por lo general, cada organización que aporta recursos de computación a LCG-2 también comparte espacio de almacenamiento ofreciendo un SE para el almacenamiento temporal de ficheros.

- **Logging & Bookkeeping (LB)**. El LB permite almacenar la evolución del estado de los trabajos a lo largo del ciclo de vida. Por ello, las diferentes fases que atraviesa un trabajo son registradas por este servicio. Esta información puede ser accedida tanto por el usuario como por otros servicios para conocer en todo momento la evolución de las tareas.

- **LCG File Catalog (LFC)**. En LCG-2 los ficheros de datos pueden estar replicados en diferentes SE e, idealmente, no es necesario que los usuarios conozcan la localización de los mismos. LFC proporciona un catálogo que permite gestionar las replicas de los ficheros así como la traducción entre nombres lógicos (Logical File Names (LFN)), independientes de la localización y los nombres físicos (Storage Uniform Resource Locator (SURL)), que identifican el SE donde se almacena.

```
[
   Type = "job";
   StdOutput = "stdout.std";
   OutputSandBox = {
       "stdout.std",
       "stderr.std"
   };
   StdError = "stderr.std";
   Executable = "LCGExecutionWrapper.sh";
   Arguments = "-x 10 -y 10 -m lr95_lr00 ";
   InputSandBox = "/tmp/LCGExecutionWrapper.sh";
   VirtualOrganisation = "biomed";
]
```

Figura 7-8: Ejemplo reducido de especificación de un trabajo mediante JDL.

- **Information Service (IS)**. Este servicio se encarga de proporcionar información sobre los recursos LCG-2 y su estado. Dicha información es esencial para el buen funcionamiento de todo el Grid. Mediante esta información, los recursos de computación (CE) pueden ser encontrados y mediante la publicación de información sobre el estado del Grid es posible detectar posibles situaciones de fallo o analizar el uso de la infraestructura. En la actualidad, la información publicada en el IS sigue el esquema *GLUE* (Grid Laboratory for a Uniform Environment) [162]. Esta iniciativa trata de definir un esquema conceptual de la información para ser usada en la monitorización y descubrimiento de recursos. En LCG-2 se ha adoptado el servicio MDS como la base principal del servicio de información. Sin embargo, recientemente se está desplegando un nuevo tipo de servicio basado en una arquitectura relacional, denominado R-GMA (Relational Grid Monitoring Architecture). Adicionalmente, existe un componente llamado BDII (Berkeley Database Information Index) [163] que permite agregar la información de todos los recursos computacionales de una organización.

La Figura 7-7 recoge el esquema de interacción con los componentes de LCG-2 para la ejecución de tareas. Todas las operaciones llevadas a cabo por el usuario se realizan desde el UI. En primer lugar, el usuario copia y registra en el Grid los ficheros necesarios para la ejecución de la aplicación en uno o varios SE, dado que es posible disponer de diferentes replicas para un mismo fichero. Además, la localización de las réplicas debe almacenarse en el catálogo LFC junto con un identificador para, posteriormente, referenciar el fichero y tener acceso al mismo. A continuación el usuario proporciona la descripción del trabajo empleando el lenguaje JDL (Job Description Language) que especifica los principales atributos necesarios para su ejecución, tal y como se muestra en el ejemplo de la Figura 7-8.

En él se define el fichero ejecutable, la salida estándar y la de error, los argumentos de línea de comandos para el ejecutable y la VO a la que pertenece el trabajo que se pretende ejecutar. El atributo *InputSandbox* permite especificar los ficheros de los que depende la aplicación y que serán automáticamente transferidos al destino de la ejecución. Sin embargo, los límites actuales fijan un máximo de unos pocos MBytes para el tamaño del InputSandbox de un trabajo. Por lo tanto, ficheros

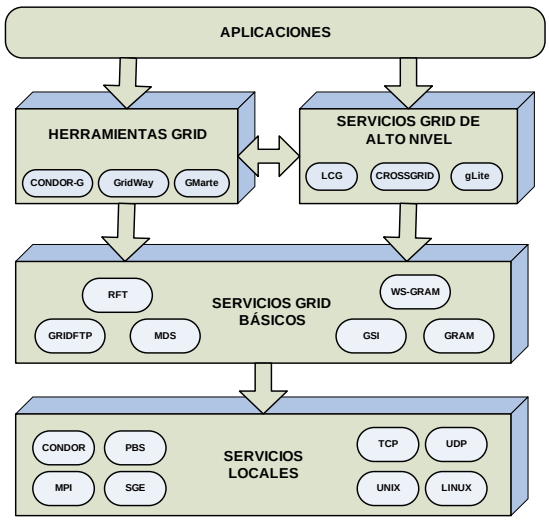


Figura 7-9: Diferentes capas de software en el uso de las tecnologías Grid.

grandes deben ser gestionados a través de los SE. El trabajo se envía al RB que es el encargado de gestionar su ejecución. Para ello busca el CE más apropiado a partir de la información sobre el estado del Grid obtenida por el IS y delega en el CE la ejecución del trabajo que, finalmente, se llevará a cabo en un WN. El RB monitoriza los cambios de estado del trabajo, que se registran en el LB, información que puede ser accedida por el usuario desde el UI.

Una vez que el trabajo ha finalizado, el usuario puede acceder a los resultados generados accediendo bien al OutputSandbox, que puede ser transferido a la máquina del usuario, o a los ficheros que hayan sido almacenados en algún SE.

7.6. Perspectiva de las Tecnologías Grid

La Figura 7-9 (versión modificada de la Fig. 4 en [164]) proporciona una perspectiva acerca las tecnologías Grid. En el nivel más bajo se encuentran los servicios locales proporcionados por las máquinas individuales. Entre ellos aparecen los servicios de red (TCP, UDP), los proporcionados por el sistema operativo (UNIX, LINUX) y aquellos que gestionan la ejecución de trabajos en la máquina (PBS, SGE, MPI, CONDOR). Por encima de ellos aparecen los servicio Grid básicos, como los proporcionados por Globus Toolkit (GRAM, GridFTP, GSI, MDS). Estos servicios básicos deben ser combinados adecuadamente para conseguir la ejecución remota de tareas. Es por ello que se han desarrollado servicios Grid de alto nivel que coordinan la utilización de los servicios básicos para ofrecer funcionalidad adicional. Además, existen herramientas Grid encargadas de simplificar la ejecución de tareas en Grids computacionales, como puede ser Condor-G [165], GridWay [166] y GMarte, la herramienta desarrollada en el marco de esta tesis doctoral.

La complejidad inherente al despliegue de una infraestructura Grid basada en servicios Grid

de alto nivel, como LCG-2, a menudo supone un freno para la utilización de estas tecnologías en entornos más reducidos. Esto es debido a los numerosos componentes existentes en LCG-2 y a la importante inversión en recursos necesaria para la instalación de los servicios en máquinas dedicadas, sin contar con el capital humano necesario para la instalación y mantenimiento del despliegue.

En efecto, existen numerosos ámbitos donde no es necesario recurrir a una infraestructura de producción a nivel mundial, con la complejidad que ello significa. Por lo tanto, resulta importante desarrollar herramientas Grid que simplifiquen la utilización de estas tecnologías para los usuarios. Es en este punto donde se centra el principal trabajo de la tesis, en el desarrollo de una herramienta Grid que gestione la ejecución de tareas en entornos Grid computacionales basados en GT, permitiendo la interoperabilidad con la infraestructura de producción de LCG-2.

7.7. Conclusiones

En este capítulo se ha realizado una introducción a la Computación en Grid enfocada a la ejecución de tareas científicas, que es la principal funcionalidad utilizada en el marco de la tesis. Para ello se han descrito los principales precursores que dieron lugar a la Computación en Grid, cuyo principal objetivo consiste en la creación una infraestructura de computación global distribuida entre múltiples centros sin sacrificar la autonomía local de cada uno de ellos.

A continuación se han descrito los principales proyectos relacionados con las tecnologías Grid. En primer lugar, se ha revisado la alianza Globus y el middleware Globus Toolkit, que representa el estándar de facto para el despliegue y utilización de Grids computacionales. Para ello, se han descrito los principales componentes y la infraestructura de seguridad ofrecida por esta herramienta. En segundo lugar, se han descrito los proyectos EDG, LCG y EGEE, destacando cómo el desarrollo de middlewares por encima de Globus ha permitido actualmente conseguir el despliegue de la mayor infraestructura Grid. En este sentido, se han descrito los principales componentes ofrecidos por el middleware LCG-2.

Las tecnologías Grid están teniendo un importante impacto en múltiples áreas de la ciencia y de la ingeniería y, con el objetivo de reducir la distancia entre estas tecnologías y los usuarios, resulta imprescindible el desarrollo de componentes de alto nivel que faciliten la interacción con el Grid para la ejecución remota de aplicaciones paralelas.

Capítulo 8

El Sistema de Computación en Grid GMarte

"If you talk about it, it's a dream. If you envision it, it's exciting. If you plan it, it's possible. But if you schedule it and do it, then it's real". Anthony Robbins, Coacher.

En este capítulo se describe GMarte (Grid Middleware to Abstract Remote Task Execution) [167, 168], el sistema de Computación en Grid desarrollado en esta tesis doctoral para soportar la ejecución de aplicaciones científicas sobre infraestructuras de computación distribuidas basadas en Globus Toolkit. En primer lugar se realiza una introducción y descripción de los principales objetivos del sistema. A continuación, se describe la arquitectura general de GMarte, detallando sus principales componentes. Posteriormente, se citan otras áreas en las que actualmente se está aplicando el sistema GMarte. Finalmente, se describen los trabajos relacionados en este campo, destacando las principales ventajas ofrecidas por el sistema desarrollado.

8.1. Introducción

En el capítulo 7 se puso de manifiesto que Globus Toolkit (GT) representa el estándar de facto actual para el despliegue de infraestructuras Grid. Sin embargo, GT únicamente proporciona un conjunto de servicios y herramientas básicas para el despliegue de Grids computacionales. En realidad, conseguir la ejecución remota de aplicaciones sobre una infraestructura Grid requiere una adecuada combinación de las herramientas proporcionadas por GT. Para ello se utilizan servicios que permitan la transferencia de ficheros, el acceso a la información de los recursos computacionales así como la ejecución de tareas, entre otros. De hecho, realizar ejecuciones completas de aplicaciones científicas en un entorno Grid requiere el uso de técnicas de metaplanificación [169] que proporcionen

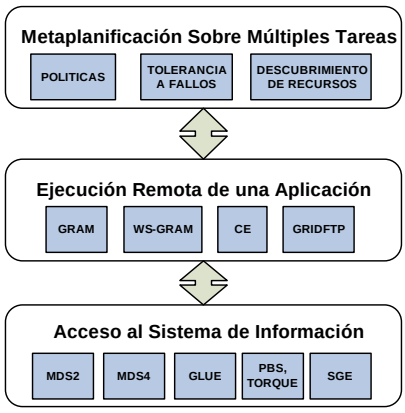


Figura 8-1: Capas de abstracción proporcionadas por GMarte y servicios relacionados.

la funcionalidad necesaria para la ejecución remota eficiente de tareas.

La metaplanificación consiste en la gestión eficiente de la ejecución de un conjunto de tareas sobre los recursos computacionales disponibles en las organizaciones involucradas en una infraestructura Grid. En realidad, el concepto de planificación de tareas se asocia generalmente a la gestión de la ejecución dentro de los límites de una organización. Por ello, dado que el Grid abarca múltiples organizaciones, se utiliza el concepto de metaplanificación para hablar de gestión de tareas sobre múltiples dominios de administración.

En este sentido, el principal objetivo de GMarte consiste en realizar labores de metaplanificación de tareas sobre recursos computacionales basados en GT. Para ello, se han desarrollado varias capas de abstracción por encima de GT para ofrecer un interfaz de alto nivel que oculte en gran medida la complejidad del middleware Grid subyacente. De esta manera, usuarios no expertos en Computación en Grid pueden obtener los beneficios de esta tecnología, mediante la ejecución de sus aplicaciones sobre un Grid computacional.

Además, uno de los pilares fundamentales de GMarte es la interoperabilidad con la infraestructura Grid de producción de LCG-2, permitiendo realizar ejecuciones en el mayor despliegue Grid existente en la actualidad. De esta manera se consigue aunar, en una misma plataforma, la sencillez de uso de una herramienta de alto nivel con la potencia obtenida de una infraestructura Grid de producción.

Es importante destacar que uno de los principales objetivos de diseño de GMarte consiste en obtener una herramienta genérica, que sea aplicable tanto a las dos aplicaciones biomédicas consideradas en el marco de la tesis como a otras muchas aplicaciones científicas. De esta manera, es posible aprovechar gran parte del trabajo desarrollado para su utilización en otros campos de la ciencia e ingeniería.

La Figura 8-1 muestra las tres principales capas de abstracción ofrecidas por GMarte. En el nivel más bajo proporciona un acceso uniforme a los diferentes sistemas de información de los recursos del Grid. Para ello, se proporciona una interfaz de alto nivel para acceder de manera transparente

a la información de los recursos (número de procesadores, memoria RAM disponible, etc.). Sobre dicha abstracción se ha desarrollado otra capa que permite la ejecución remota de tareas sobre recursos computacionales. Análogamente, el objetivo es ofrecer un interfaz común de alto nivel que oculte al usuario toda la complejidad de protocolos y diferencias técnicas que surgen dependiendo del middleware Grid utilizado. Finalmente, en el nivel más alto aparece la capa que ofrece abstracción sobre el proceso de metaplanificación. El principal objetivo de este nivel consiste en permitir al usuario la definición de las tareas a ejecutar, y dejar que el sistema realice de forma transparente todas las acciones necesarias para la ejecución eficiente de las mismas. Las conexiones entre cada una de las capas se hacen por medio de las interfaces de alto nivel definidas por cada una de ellas, lo que facilita la mantenibilidad del código y la reutilización de componentes.

Nótese que la realización de un diseño basado en capas permite la utilización de parte de la funcionalidad de GMarte. Por ejemplo, es posible utilizar la abstracción del acceso al sistema de información para desarrollar aplicaciones de monitorización del estado de los recursos, o incorporar la funcionalidad de ejecución de tareas a otros proyectos.

Resulta importante destacar que la introducción de componentes de alto nivel para la utilización de las tecnologías Grid supone un importante avance en la usabilidad de las mismas. En la actualidad, la utilización de las tecnologías Grid para la ejecución de aplicaciones científicas requiere de cierta habilidad y conocimiento de técnicas de programación utilizando diferentes herramientas, protocolos y tecnologías. Por lo tanto, ocultar la complejidad subyacente al usuario permite facilitar la utilización de la Computación en Grid para la ejecución de aplicaciones.

Actualmente, GMarte puede ser utilizado para tareas independientes, es decir, aquellas que pueden ser ejecutadas de manera concurrente dado que no existen dependencias entre ellas. Además, las tareas interactivas, es decir, aquellas que requieren la interacción del usuario durante la ejecución de la misma, no son soportadas en esta versión. Hay que destacar que existen multitud de aplicaciones basadas en tareas independientes, en especial las aplicaciones de barrido de parámetros o multiparamétricas cuyas ejecuciones requieren la utilización de la misma aplicación donde varía algún dato de entrada. Para este tipo de aplicaciones, basadas en el modelo de alta productividad, la utilización de una infraestructura Grid permite aportar la potencia computacional necesaria para su ejecución. Una de las principales ventajas de GMarte es que permite la ejecución de aplicaciones paralelas basadas en la librería de paso de mensajes MPI.

Se puede observar, por tanto, que GMarte se centra en la combinación de dos técnicas computacionales. Por un lado, la Computación de Altas Prestaciones permite la aceleración de una única tarea mediante la ejecución paralela en un cluster de PCs. Por otra parte, la Computación de Alta Productividad [170] pretende conseguir la gestión eficiente y el uso de todos los recursos de computación disponibles de una infraestructura distribuida. Por lo tanto, una integración de ambas técnicas que permita la realización concurrente de ejecuciones, que a su vez pueden aprovecharse del parale-

lismo de los nodos multiprocesadores de un Grid, parece ser una buena combinación para acelerar la productividad en la ejecución de múltiples tareas.

8.2. Generalidades y Componentes Principales

GMarte ha sido desarrollado empleando el lenguaje de programación Java. Esta decisión responde a varias razones. Por un lado, la portabilidad ofrecida por este lenguaje permite que la aplicación desarrollada pueda ejecutarse en cualquier plataforma y arquitectura en la que exista una implementación de la máquina virtual Java. Por otro lado, existe una tendencia actual en el uso de Java como lenguaje de desarrollo de aplicaciones Grid. Por ejemplo, el desarrollo de servicios Grid en GT4 se realiza en Java. En este sentido, la elección de dicho lenguaje facilita la integración de GMarte con otras herramientas Grid existentes.

La funcionalidad de GMarte se expone a través de un API (Application Programming Interface) de alto nivel orientado a objetos en lugar de ser una herramienta cerrada. El hecho de disponer de un API permite que otras aplicaciones incorporen parte de la funcionalidad de GMarte, bien para la interacción con los servicios de información o para la ejecución remota de aplicaciones. Esta aproximación facilita la integración entre diferentes componentes. Además, es posible la construcción de herramientas cerradas que, utilizando el API de GMarte, proporcionen la solución a un problema concreto.

Para el desarrollo de GMarte se han utilizado diversas herramientas. Por un lado, se utiliza el Java Commodity Grid Kit 1.2 [171], que proporciona mecanismos para utilizar la funcionalidad de GT2 desde el lenguaje Java. Esta herramienta permite utilizar el protocolo GridFTP para la transferencia de archivos, así como el servicio GRAM para la ejecución de trabajos en recursos computacionales basados en GT2. Por otro lado, se utilizan las librerías cliente de GT4 para el acceso tanto al servicio de información como al servicio de ejecución de trabajos de máquinas con GT4. Los mecanismos de seguridad empleados están basados en los proporcionados por la infraestructura de seguridad de Globus (GSI). Adicionalmente, se utiliza la herramienta XMLWrappers [111], modificada en el marco de la tesis, con el objetivo de proporcionar un entorno de enlace de datos para transformar de manera automática descripciones de objetos en lenguaje XML a objetos GMarte dentro de la máquina virtual Java. La conexión con LCG-2 se ha realizado empleando componentes del UIPnP (User Interface Plug & Play) [172], que permite desplegar un UI sin necesidad de privilegios de administración en cualquier máquina Linux.

GMarte está compuesto por unas 250 clases Java, que aglutinan cerca de 30000 líneas de código, organizadas en cuatro paquetes principales: Tareas, Recursos, Ejecución y Metaplanificación. Cada uno de ellos se divide a su vez en diferentes subpaquetes, que proporcionan el conjunto global de clases de las que consta el API de GMarte. Además, se ha realizado un importante esfuerzo de

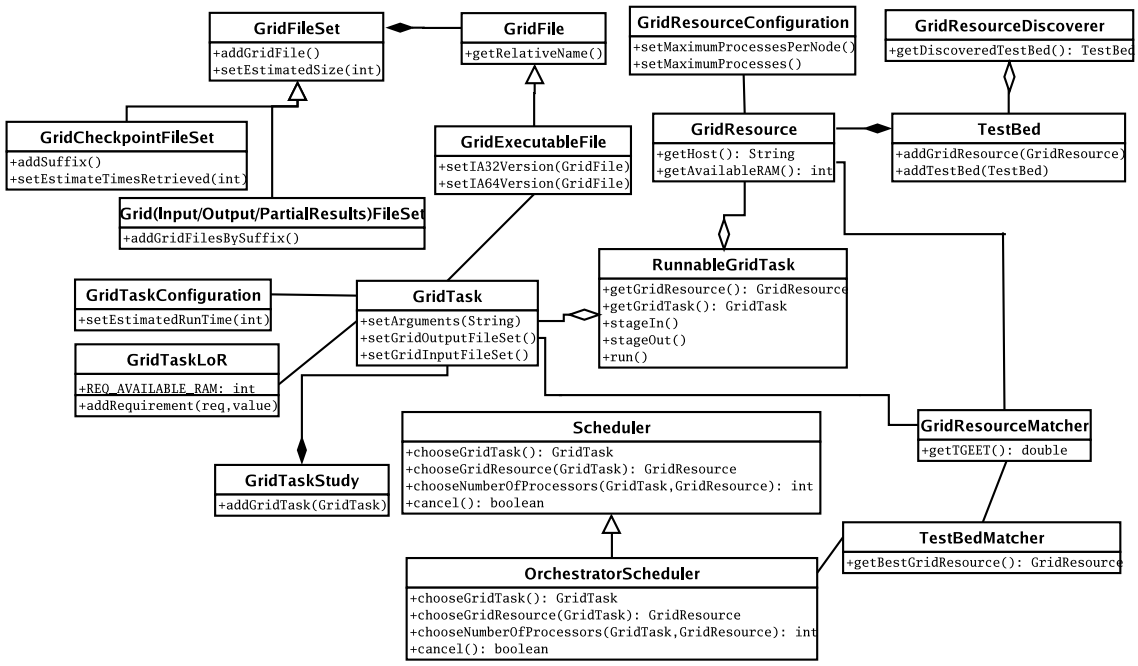


Figura 8-2: Diagrama de las principales clases de GMarte. La mayor parte de los métodos no han sido incluidos para simplificar el diagrama.

documentación del API, siguiendo el estándar JavaDoc para la descripción tanto de las clases como de cada uno de los métodos. La documentación incluye más de 13000 líneas y está accesible a través de Internet[1]. Dispone de referencias cruzadas que ayudan a reducir la curva de aprendizaje del uso del API.

La Figura 8-2 muestra, de manera esquemática, las principales clases de más alto nivel. En el diagrama, un objeto *GridTask* permite la definición de una tarea a ejecutar en el Grid, mientras que un objeto *GridResource* permite abstraer el concepto de recurso computacional. La clase *GridTaskStudy* representa la definición de un caso de estudio compuesto por múltiples tareas independientes, mientras que la clase *TestBed* aglutina un conjunto de recursos computacionales. La clase *Scheduler* define la interfaz que deben cumplir todos los planificadores de tareas implementados, que proporcionan la funcionalidad de asignación de tareas a la infraestructura Grid definida.

El API de alto nivel, proporcionado por GMarte, oculta al usuario los servicios de bajo nivel del middleware Grid subyacente. En este sentido, ofrece un conjunto de abstracciones de alto nivel que permite al usuario inexperto en las tecnologías Grid centrar su atención principalmente en la definición de tareas, sin tener que lidiar con la utilización de diferentes protocolos y servicios para conseguir realizar ejecución remota y eficiente de tareas. Además, el API expuesto es neutral a la tecnología lo que permite realizar cambios en la interacción con el middleware subyacente sin que ello afecte a la interfaz utilizada por el usuario, facilitando así la extensibilidad y actualización del

[1]Documentación del API de GMarte - http://www.grycap.upv.es/~gmolto/gmarte/internal/doc-devel

código.

En este sentido, el desarrollo de una capa de abstracción de estas características supone reducir la complejidad del Grid, acercándolo al usuario. La existencia de herramientas fáciles de utilizar por la comunidad de usuarios del Grid, supone un importante avance en la rápida adopción de estas tecnologías por la comunidad científica.

En las siguientes secciones se detalla la funcionalidad ofrecida por cada una de las capas de abstracción que forman parte de GMarte.

8.2.1. Abstracción del Acceso a los Sistemas de Información

Los recursos computacionales de un Grid publican información sobre su estado. Esta información incluye, por un lado, datos de carácter general, como el sistema operativo instalado, la arquitectura del procesador o la cantidad total de memoria RAM. Por otro lado, los recursos también indican su estado actual, es decir, la cantidad de memoria RAM disponible, el número de procesadores libres en caso de tratarse de una máquina multiprocesador o cluster de PCs, la carga del sistema en el último minuto, y demás información de carácter dinámico.

Este hecho permite que otros servicios puedan tomar decisiones considerando la información aportada por cada uno de los recursos. En realidad, la información es especialmente valiosa para un proceso de selección de recursos, dentro de un proceso de metaplanificación, con el objetivo de decidir el recurso computacional que recibirá una determinada ejecución en base al estado de los mismos. Por ejemplo, es de esperar que aquellas máquinas que dispongan de menos capacidades computacionales reciban menos carga de trabajo que aquellas más potentes.

GT ofrece funcionalidad de servicios de información por medio del servicio MDS. Este servicio está diseñado para proporcionar un mecanismo estándar para la publicación de las capacidades y el estado de un recurso. El servicio MDS trata de simplificar los servicios de información del Grid proporcionando un único esquema y una interfaz común para los diferentes tipos de información que habitualmente se utilizan dentro de una organización virtual.

Sin embargo, en el caso de los recursos computacionales, especialmente para las máquinas multiprocesador y los clusters de PCs, existen diferencias en la información proporcionada por el servicio MDS referente al número de procesadores del recurso, que depende del gestor de trabajos instalado (i.e., Portable Batch System (PBS), Torque o Sun Grid Engine (SGE), entre otros). Estas diferencias son mucho más manifiestas en el caso del middleware LCG-2 que introduce cambios en el esquema, el principal componente que define la estructura de la información proporcionada por el servicio MDS.

Además, los protocolos subyacentes para el acceso a MDS2 (el sistema de información utilizado por GT2 y LCG-2), y MDS4 (ofrecido por GT4), son diferentes. Por una parte, el servicio MDS2 ofrece la información a través de un servicio en el propio recurso computacional, enlazado al puerto 2135, que publica la información vía interfaces LDAP. Por otra parte, MDS4 expone la información

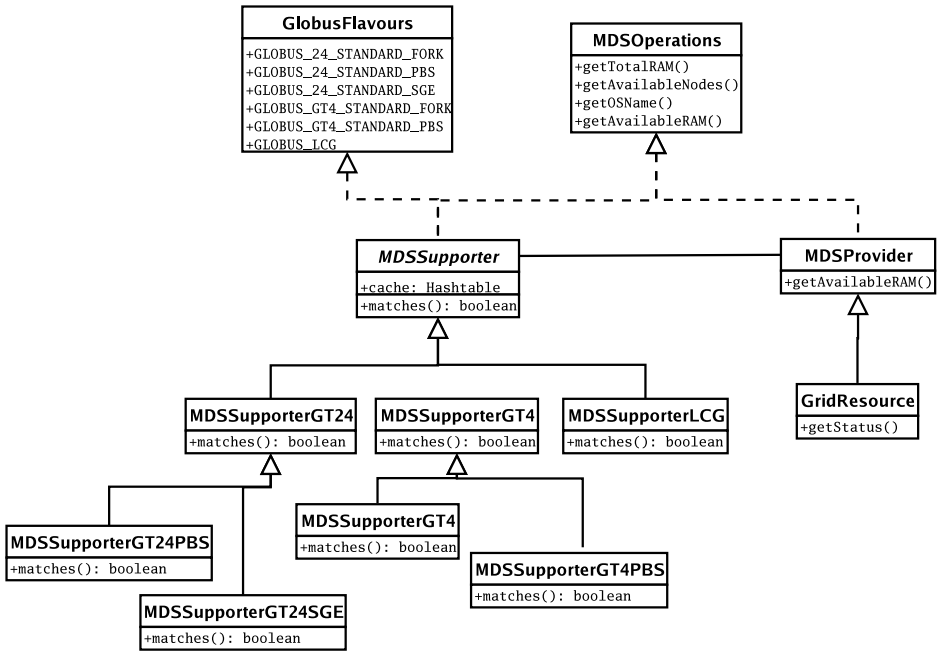


Figura 8-3: Diagrama de clases para la gestión de los servicios de información de Globus.

a través de un servicio Grid denominado *DefaultIndexService* que publica información en formato XML y suele ser accedido a través de mensajes SOAP (Simple Object Access Protocol) [173], que es el mecanismo de intercambio de mensajes entre servicios Web. Las consultas, además, se realizan empleando el lenguaje XPath.

Para lidiar con este abanico de diferencias de esquemas, lenguajes y protocolos, ofreciendo al usuario un API común para el acceso a la información de cualquier recurso computacional, se ha implementado una estrategia basada en *Supporters* y *Providers*. Esta aproximación proporciona un entorno extensible basado en la reutilización de componentes ya desarrollados en GMarte, que permite incorporar de manera sencilla aquellas futuras divergencias en la información que proporcionen los diferentes servicios de información. De esta manera, los detalles de bajo nivel quedan ocultos tras un API unificado para el acceso a la información de las máquinas.

La Figura 8-3 describe las principales clases que utiliza GMarte para la gestión de los servicios de información de Globus. Por motivos de brevedad, únicamente se muestran los métodos y atributos más relevantes. La interfaz *MDSOperations* especifica la signatura de los métodos que ofrecen la información de un recurso del Grid, es decir, aquellos métodos de alto nivel que utiliza el usuario para acceder a la información del recurso. La interfaz *GlobusFlavours* especifica las diferentes variantes de Globus soportadas por GMarte.

La clase *MDSSupporter* implementa la interfaz MDSOperations y, por definición, está obligada a implementar las operaciones definidas en la interfaz. Sin embargo, esta clase únicamente proporciona aquellos métodos comunes para todas las variantes de Globus. Por lo tanto, es una clase abstracta

que no puede ser instanciada. Son sus subclases las encargadas de proporcionar la implementación del resto de métodos. Por ejemplo, la clase *MDSSupporterPBS* implementa los métodos que acceden a la información del servidor MDS de una recurso con Globus Toolkit 2.4 y el gestor de colas PBS instalado. La clase *MDSSupporterLCG* proporciona la implementación de los métodos que acceden a la información ofrecida por el servicio MDS de un Computing Element basado en LCG-2.

El método abstracto *matches*, definido en la clase MDSSupporter, se implementa en las clases descendientes para indicar si un determinado recurso es del tipo definido en el MDSSupporter. Mediante un sencillo análisis de la información ofrecida por el servicio MDS es posible determinar qué MDSSupporter es compatible con el recurso, el cual se utilizará para los accesos posteriores a la información del recurso.

Consideremos ahora la clase *MDSProvider*. Dado que también implementa la interfaz MDSOperations, debe implementar las operaciones allí definidas. Sin embargo, lo importante es que, en este caso, la implementación se realiza por medio de la delegación en una instancia de una de las subclases de MDSSupporter, que realmente realizará la consulta al servicio MDS del recurso. En este sentido, cuando se crea un nuevo objeto de tipo GridResource, la máquina destino se investiga para averiguar el MDSSupporter apropiado (aquel cuyo método *matches* devuelve true). Posteriormente, el usuario interactúa con el recurso computacional a través de la abstracción GridResource que, por herencia, expone los métodos definidos en la interfaz MDSOperations. Por ejemplo, si el usuario quiere averiguar la cantidad de memoria RAM disponible para un recurso, únicamente deberá invocar el método *getAvailableRAM* del correspondiente GridResource para obtener la información de manera automática.

Resulta fácil observar que esta capa de abstracción en el acceso al sistema de información permite homogeneizar el proceso de recopilación de información sobre recursos computacionales de un Grid, sin necesidad de que el usuario conozca ni utilice, en ningún momento, los numerosos protocolos e interfaces utilizados internamente. Además, el desarrollo realizado supone la creación de un marco extensible que permite añadir, de manera sencilla, el soporte para otros tipos de sistemas de información sin más que añadir un nuevo Supporter.

Para poder comparar la facilidad de uso del API de GMarte en el acceso al sistema de información de los recursos computacionales, se muestra a continuación como se realizaría una consulta del número de procesadores libres de una máquina llamada pilos.itaca.upv.es:

Si la máquina fuera un Computing Element de LCG-2, se utilizaría una consulta LDAP siguiendo el esquema de información GLUE:

```
ldapsearch -x -h pilos.itaca.upv.es -p 2135 -b "mds-vo-name=local,o=grid" \
 '(objectClass=GlueCEState)' GlueCEStateFreeCPUs
```

Si la máquina tuviera instalado GT4, se realizaría una consulta XPath al servicio de información:

```
wsrf-query -a -z none -s \
https://pilos.itaca.upv.es:8443/wsrf/services/DefaultIndexService \
"count(//*[local-name()='GLUECE']/glue:ComputingElement/ glue:State/@glue:FreeCPUs)"
```

Utilizando el API de GMarte se accedería a la información con el siguiente código, independientemente del middleware Grid subyacente:

```
GridResource gr = new GridResource(‘‘pilos.itaca.upv.es’’);
gr.getAvailableNodes();
```

Es fácil comprobar que la utilización de un API de alto nivel simplifica en gran medida las consultas sobre el estado de los recursos. Además, esta funcionalidad puede ser utilizada independientemente de las labores de metaplanificación ofrecidas por GMarte, para ser incorporada en otros proyectos. Por ejemplo, se han desarrollado componentes gráficos para la monitorización del estado de los recursos empleando para ello el API de GMarte para el acceso a la información de los recursos computacionales. Actualmente, GMarte es capaz de interactuar con los sistemas de información de GT2, GT4 y el de los CE de LCG-2.

Acelerando el Acceso al Sistema de Información

GMarte implementa un componente que permite acelerar el acceso a la información de los recursos mediante un sistema de *cache*. El proceso de extracción de información de un recurso involucra una conexión al servicio MDS de la máquina destino para someter una consulta. Este acceso al sistema de información es bastante frecuente, especialmente si se desea utilizar en un proceso de metaplanificación. En estas circunstancias, la selección de recursos implica obtener información sobre los recursos computacionales para tomar una decisión del mejor recurso para ejecutar una tarea. Durante esta fase, se produce una sobrecarga temporal y un uso intensivo de la red que puede ser reducido utilizando técnicas de caching.

Para ello, se clasifican los atributos de un recurso en dos tipos:

1. Estáticos: Todos aquellos atributos que no varían en el tiempo a lo largo de un proceso de metaplanificación, como por ejemplo, el Sistema Operativo, el tipo de procesador y la cantidad total de memoria RAM.

2. Información Dinámica: Todos aquellos atributos que pueden variar con el tiempo a lo largo de un proceso de metaplanificación, como por ejemplo, el número de procesadores libres, el nivel de carga de la CPU y la memoria RAM disponible.

Para reducir las consultas innecesarias, GMarte implementa el componente de cache mediante una tabla hash para cada objeto GridResource. Una tabla hash es una estructura de datos que permite

realizar la búsqueda de objetos por nombre en tiempo constante, siempre que se implemente de manera eficiente. Dicha tabla almacena el resultado de la primera consulta sobre cualquier atributo estático del recurso. De esta manera, las consultas posteriores sobre el mismo atributo estático del recurso serán resueltas de manera local en tiempo despreciable y sin consumo de red. En caso de solicitar un atributo dinámico, como el número de procesadores disponibles, GMarte dirige la consulta directamente al recurso para obtener la información más reciente.

Adicionalmente, se ha implementado un mecanismo de tolerancia a fallos transparente al usuario por el cual una consulta fallida se reintenta hasta un máximo de 3 veces (valor configurable) antes de indicar que el recurso ha fallado. De esta manera es posible tolerar ciertos fallos en la red que puedan ocasionar problemas al realizar consultas al servicio de información de una máquina remota.

Descubrimiento de Recursos

GMarte implementa mecanismos para el descubrimiento de recursos tanto en infraestructuras basadas en Globus, a partir de un GIIS, como en infraestructuras basadas en LCG-2, a partir de un BDII. Estos componentes permiten agregar la información computacional de los recursos de una organización y, de esta manera, es posible acceder a un listado de recursos computacionales. Para ello se proporciona una interfaz común que internamente utiliza los protocolos y esquemas apropiados para la obtención de un listado de recursos computacionales.

La utilización de mecanismos de descubrimiento de recursos permite al usuario delegar en GMarte la ejecución de las tareas en los recursos que estén accesibles. A continuación se muestra un ejemplo del código necesario para el descubrimiento de recursos en el testbed de EGEE pertenecientes a la VO biomed.

```
GridResourceDiscoverer grd = new BDIIResourceDiscoverer(‘lcgbdii02.ifae.es’’);
DiscoveryOptions do = new DiscoveryOptions();
do.setVirtualOrganization(‘‘Biomed’’);
grd.setDiscoveryOptions(do);
TestBed tb = grd.getDiscoveredTestBed();
```

8.2.2. Abstracción en la Definición de Tareas

En GMarte la definición de una tarea se realiza a través de un objeto GridTask. Un conjunto de tareas ya definidas se agrupan en un objeto GridTaskStudy.

Un GridTask representa la unidad de ejecución en el Grid y lleva asociados varios componentes que tienen una representación como objetos GMarte. Los principales componentes son los siguientes:

- GridExecutableFile. Representa el fichero ejecutable de la aplicación. Se permite la especificación de diferentes versiones del ejecutable para las diferentes plataformas computacionales

que se pueden encontrar en la infraestructura computacional. Por ejemplo, dicha clase permite definir un ejecutable para las plataformas Intel de 32 bits y otro para las de 64 bits, con el objetivo de aprovechar al máximo las prestaciones de las arquitecturas más recientes.

- GridInputFileSet. Permite especificar los ficheros de entrada que requiere la aplicación para su ejecución. Se permite especificar tanto una ruta completa como un conjunto de ficheros de un directorio que cumplan un determinado patrón de nombrado.

- GridOutputFileSet. Permite especificar los ficheros resultado que genera la aplicación y que se desea tener disponibles en la máquina local tras finalizar la ejecución de la tarea. Típicamente engloba los ficheros generados por la aplicación, así como los ficheros de salida estándar y salida estándar de error. Dado que los ficheros de salida pueden presentar un esquema de nombrado que puede ser desconocido a-priori, esta clase permite la utilización de comodines para especificar los ficheros a transferir.

- GridCheckpointFileSet (Opcional). Permite definir los ficheros de copia de seguridad que genera una aplicación de manera periódica para almacenar su estado. A menudo, aplicaciones computacionalmente intensivas, basadas en un proceso iterativo, permiten la generación de estos ficheros de respaldo para continuar la simulación desde el último punto de restauración.

- GridPartialResultsFileSet (Opcional). Permite la definición de ficheros de resultados parciales que va generando la aplicación a lo largo de su ejecución. Esta funcionalidad permite solapar la ejecución remota de la aplicación con la transferencia de los resultados parciales a la máquina cliente, lo que reduce en gran medida el tiempo de ejecución global para las aplicaciones que soporten dicha funcionalidad.

Dado que GMarte se utiliza con aplicaciones científicas computacionalmente intensivas, se han implementado mecanismos que permitan al usuario especificar los requisitos computacionales de una tarea. Esta información se utiliza para descartar aquellas máquinas que no cumplen los requisitos de ejecución mínimos impuestos por la tarea. Tratar de ejecutar una aplicación que precisa una serie de recursos en una máquina que no los dispone, provocará un incremento del tiempo de ejecución de la tarea o incluso un fallo de la aplicación durante su ejecución.

Para evitar ese problema, un objeto GridTask puede especificar una lista de requisitos que debe cumplir un recurso computacional para poder realizar la ejecución de la tarea. Para ello, se emplea un objeto de la clase *GridTaskLoR*, que permite especificar requisitos como la memoria RAM disponible, el juego de instrucciones de la plataforma remota o el mínimo número de procesadores libres que debe disponer un recurso multiprocesador. Adicionalmente, también es posible limitar el número máximo de procesadores involucrados en una ejecución paralela para una determinada tarea. De esta manera, el usuario puede indicar al sistema que su aplicación no escala adecuadamente a partir

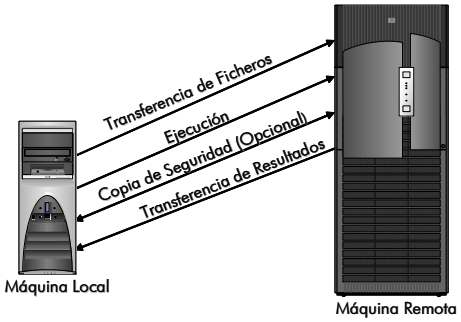


Figura 8-4: Fases a realizar para la ejecución remota de una aplicación.

de un cierto número de procesadores y, por lo tanto, no es necesario que se realice una ejecución paralela con un mayor número de procesadores.

Es importante destacar que una adecuada configuración de estos valores requiere un conocimiento básico sobre el comportamiento de la aplicación. Para ello, existen diversas técnicas de instrumentación y análisis post-morten [174] que representan herramientas adecuadas para obtener unos buenos estimadores sobre el consumo de memoria de la aplicación así como, por ejemplo, el número óptimo de procesadores involucrados en una ejecución paralela.

8.2.3. Abstracción de la Ejecución Remota de una Aplicación

Actualmente, GMarte permite la ejecución de tareas en recursos computacionales basados en GT2, GT4 y CE de LCG-2. Adicionalmente, permite delegar la ejecución de tareas en los metaplanificadores de LCG-2 (RB). Por lo tanto, se distingue entre las acciones llevadas a cabo para gestionar la ejecución de una aplicación en una máquina remota, y las necesarias para delegar la ejecución de la misma en otro sistema de asignación de tareas.

Para conseguir la correcta ejecución de una tarea (GridTask) en un recurso computacional (GridResource) se deben realizar una serie de fases que tienen una correspondencia directa, en el entorno GMarte, con un conjunto de métodos de la clase *RunnableGridTask*.

La Figura 8-4 resume las principales fases que involucra la ejecución remota de una aplicación en un recurso computacional. En primer lugar se realiza la fase de transferencia de archivos iniciales, o *stage-in*. En esta fase, todos los ficheros de entrada de los que depende la aplicación (GridInputFileSet) así como el propio fichero ejecutable del que depende la aplicación, son transferidos a la máquina destino. Para ello se emplea el protocolo GridFTP que ofrece transferencia de datos eficiente y empleando los mecanismos de seguridad de GSI. Los atributos de ejecución de los ficheros no se mantienen al realizar la transferencia por GridFTP. Por ello, una solución posible consiste en aprovechar el hecho de que las implementaciones del servidor GridFTP implementan una serie de comandos conocidos como *site-specific*, de manera que se puede enviar el comando *CHMOD* para cambiar los permisos de acceso a un fichero a través de la conexión FTP.

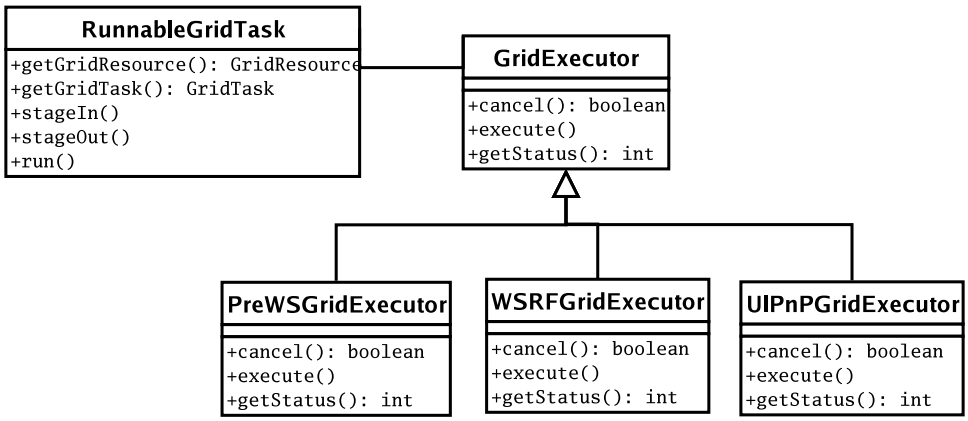


Figura 8-5: Diagrama de clases para la gestión de la ejecución.

Una vez transferidos los ficheros requeridos a la máquina destino se realiza la fase de ejecución. En ella es necesario transformar la definición de la tarea al formato apropiado utilizado por la máquina destino. En caso de ser un recurso basado en GT2 o un CE de LCG-2 se utiliza el servicio GRAM y la especificación de trabajos en el lenguaje RSL (Resource Specification Language). En el caso de ser un recurso basado en GT4 se debe utilizar el servicio WS-GRAM y la especificación de trabajos en el lenguaje XML-RSL.

En este sentido, se han desarrollado traductores que, de manera totalmente transparente al usuario, permiten la generación automática de la descripción de la tarea en el lenguaje requerido por el servicio de ejecución de la máquina remota.

En GMarte no se realiza la ejecución directa de la aplicación sino que se crea de manera automática un shell-script (*wrapper*) que se encarga de realizar acciones previas y posteriores a la ejecución de la aplicación. Si la máquina destino es un CE de LCG-2, y no hay un sistema de ficheros compartidos entre el CE y el WN que recibe la ejecución, se produce una transferencia de los ficheros necesarios del CE al WN previo a la ejecución de la aplicación y una transferencia posterior de los ficheros de resultados del WN al CE. En cualquiera de los casos el wrapper realiza la ejecución de la aplicación capturando el código de salida de la misma para, posteriormente, ayudar a la detección de fallos en la ejecución.

La Figura 8-5 muestra la estrategia empleada para poder acoplar múltiples adaptadores de ejecución. La clase GridExecutor define una serie de métodos genéricos que permiten la interacción con los recursos Grid. Cada subclase implementa el adaptador concreto a emplear para la interacción con el middleware específico del recurso, tanto para la ejecución del trabajo como para la consulta de su estado.

Una vez comenzada la ejecución de la aplicación, esta se integra de manera automática en el gestor de trabajos de la máquina remota (LRMS). De esta manera, se respetan las políticas de ejecución de las organizaciones, ya que los trabajos recibidos a través del Grid están sujetos al sistema de colas

del recurso, sin interferir con el resto de trabajos de otros usuarios.

Durante la ejecución de la aplicación se monitoriza el estado de la misma para detectar fallos a través de los mecanismos de notificación de errores ofrecidos por GRAM. Además, en esta fase es posible que se produzca una interacción con el recurso para la recogida tanto de resultados parciales como de ficheros de copia de seguridad, generados periódicamente por la aplicación.

Una vez acabada la ejecución, se realiza la fase de transferencia de resultados o *stage-out*. Para ello, se han implementado métodos recursivos de descarga de datos que permiten transferir, mediante GridFTP, los ficheros y directorios especificados en el objeto GridOutputFileSet vinculado a la tarea.

Finalmente, se realiza la fase de eliminación de ficheros. Esta fase se encarga de borrar todos aquellos ficheros que hayan sido generados durante la ejecución en la máquina remota, con el objetivo de ahorrar espacio en disco en el recurso destino. Para ello se han implementado métodos de eliminación recursiva de directorios remotos, funcionalidad no existente en el cliente GridFTP utilizado.

Tolerancia a Fallos

Dentro de esta capa se ha implementado un esquema de tolerancia a fallos en las transferencias de datos, que permite a GMarte recuperarse ante fallos producidos en la red. Para ello, se ha ampliado el comportamiento del cliente GridFTP, ofrecido por el Java CoG, para permitir realizar transferencias tolerantes a fallos. Para ello, una transferencia fallida se reintenta hasta 3 veces antes de comunicar el error a las capas superiores. Además, se utiliza un mecanismo basado en marcadores de prestaciones que permite detectar una transferencia que, aun no siendo fallida, no progresa de la manera esperada. De esta manera se consigue abortar la transferencia y notificar el error sin necesidad de esperar un tiempo innecesario.

Por otra parte, esta capa incluye gestión de errores producidos durante la ejecución de la tarea en el recurso remoto. Para ello se utilizan los mecanismos de detección de cambio de estado ofrecidos por las librerías cliente de GT. Sin embargo, dado que es frecuente en GT2 que una ejecución fallida sea detectada simplemente como una tarea finalizada, es importante disponer de mecanismos adicionales para detectar la ejecución satisfactoria de la tarea. Para ello GMarte incorpora dos mecanismos:

- Palabra Clave en Salida Estándar. El shell-script encargado de ejecutar la aplicación se encarga de escribir por la salida estándar el código de finalización de la aplicación junto con una palabra clave. Cuando el fichero de salida estándar se transfiere a la máquina cliente, GMarte procesa automáticamente el fichero para detectar la existencia de ambos marcadores.

- Inexistencia de Salida Estándar de Error. Opcionalmente, se puede comprobar que el fichero de salida estándar de error esté vacío. Esta situación generalmente indica que la aplicación se ha ejecutado de forma satisfactoria.

Estos mecanismos permiten que la capa superior, dedicada al proceso de metaplanificación, pueda decidir si una ejecución ha finalizado de forma inesperada y, por lo tanto, debe ser replanificada.

8.2.4. Ejecución en Recursos LCG-2 vía Delegación

Aunque GMarte es capaz de realizar ejecuciones en los CE de un despliegue LCG-2, bien es cierto que esta infraestructura de producción dispone de sus propios gestores de carga encargados de la distribución de trabajos a los CE. Por lo tanto, es una buena práctica delegar la ejecución de las tareas a los RB de LCG-2. Esta funcionalidad permite ampliar de manera automática el conjunto de recursos computacionales a todos aquellos disponibles en el marco de los proyectos LCG y EGEE. Además, GMarte es capaz de utilizar de manera simultánea recursos computacionales de diferentes tipos, permitiendo la utilización de infraestructuras Grid heterogéneas.

Para ello se utiliza la funcionalidad ofrecida por el UIPnP, que permite la instalación de un componente UI de LCG-2 sin necesidad de disponer de privilegios de administración ni una máquina dedicada. Simplemente con las credenciales de usuario apropiadas para el acceso a la infraestructura de LCG-2, expone una serie de comandos para el envío de trabajos, consulta del estado, transferencia de ficheros a los SE, y gestión del catálogo LFC.

En este sentido se ha desarrollado un API en Java para acceder a la funcionalidad del UIPnP. Para ello se ha creado en primer lugar un componente que permite la ejecución de shell-scripts con cierta tolerancia a fallos, es decir, que reintenta las operaciones fallidas un cierto número de veces. Sobre este componente se ha construido una pasarela que permite acceder a la funcionalidad del UIPnP desde Java. GMarte utiliza este último componente para poder interactuar con LCG-2.

La ejecución de una tarea delegada es ligeramente diferente al proceso anteriormente descrito. Sin embargo, se ha incorporado a GMarte de manera que el proceso resulte totalmente transparente al usuario.

En primer lugar se produce la fase de stage-in, donde los ficheros dependientes de la aplicación y el propio fichero ejecutable son transferidos a un SE y registrados en un catálogo LFC. A continuación, el componente *UIPnPGridExecutor* de la Figura 8-5 transforma la definición de la tarea al lenguaje JDL apropiado para la especificación de trabajos en LCG-2.

Posteriormente, GMarte genera de manera automática un wrapper, encargado de la ejecución de la aplicación en un WN, que realiza las siguientes acciones:

1. Descarga de los ficheros de la aplicación del SE al WN en el que se producirá la ejecución.

2. Ejecución de la aplicación que provocará la generación de los resultados.

3. Transferencia de los ficheros generados por la aplicación desde el WN al SE y registro en el catálogo.

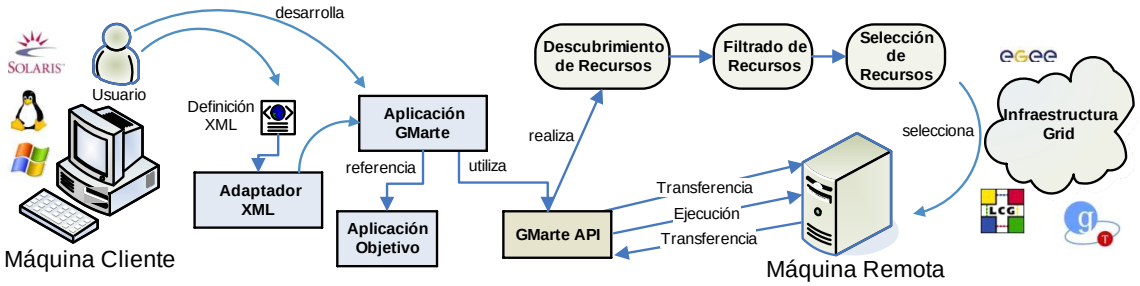


Figura 8-6: Diagrama general de la funcionalidad de metaplanificación ofrecida por GMarte.

Estas tareas son llevadas a cabo desde la máquina remota (el WN). La evolución del estado de las tareas se realiza por medio de las consultas al servicio LB por lo que se ha tenido que implementar un mapeo del diagrama de transición entre estados de una tarea en LCG-2 al propio en GMarte. De esta manera es posible integrar en GMarte la ejecución de tareas delegadas de manera transparente al usuario.

Una vez finalizada la ejecución de la tarea se produce la fase de stage-out, donde los ficheros que han sido transferidos al SE deben ser recuperados para tenerlos disponibles en la máquina cliente para su procesamiento.

La interoperabilidad de GMarte con una infraestructura de producción como LCG-2 supone un importante valor añadido para su utilización como proveedor de potencia computacional a las aplicaciones, permitiendo la utilización coordinada de recursos basados en GT y delegación de ejecución de tareas sobre LCG-2.

8.2.5. Abstracción del Proceso de Metaplanificación

El proceso de metaplanificación se encarga de realizar la ejecución de un conjunto de tareas sobre un conjunto de recursos computacionales. Para ello, GMarte realiza los pasos que se muestran en la Figura 8-6.

El proceso de planificación requiere una definición previa del conjunto de tareas (GridTaskStudy) y del conjunto de recursos a emplear para su ejecución (TestBed). Para ello, el usuario puede construir una aplicación Java que utilice el API de GMarte para definir estos objetos. Alternativamente, se han desarrollado interfaces en XML que evitan la necesidad de construir una aplicación en Java para acceder a la funcionalidad principal de GMarte (descritas en el capítulo 9). En este caso, se realiza un proceso de construcción automática de objetos en el entorno GMarte a partir de la descripción XML tanto de las tareas como de los recursos. Una vez especificados dichos componentes, es responsabilidad de GMarte realizar todas las labores necesarias para conseguir la ejecución eficiente de las tareas sobre la infraestructura Grid.

En primer lugar, si el usuario así lo especifica, se debe llevar a cabo un descubrimiento de recursos

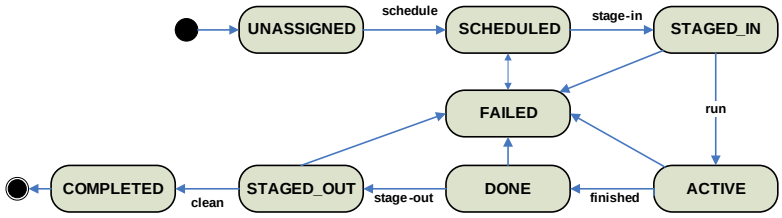


Figura 8-7: Diagrama simplificado de transición entre estados a lo largo del ciclo de vida de una tarea.

para obtener una listado de máquinas potenciales para realizar la ejecución. En segundo lugar se realiza una fase de filtrado de recursos, que se encarga de descartar aquellas máquinas que no son accesibles con las credenciales ofrecidas por el usuario.

Una vez que acaban estas dos fases, que únicamente se realizan una vez por cada sesión de metaplanificación, comienza un proceso iterativo para ejecutar las tareas pendientes en la infraestructura Grid. Para cada una de las tareas se realiza un proceso de selección de recursos, de manera que se elige el recurso mejor considerado en cada momento para realizar la ejecución de la tarea. Posteriormente se realizan las fases de transferencia de archivos, ejecución y transferencia de resultados, antes de dar por concluida la ejecución de la tarea.

Para poder realizar de manera eficiente este proceso y aplicarlo a un entorno multi-hilo, donde se puedan planificar concurrentemente múltiples tareas, se realiza un proceso de metaplanificación basada en estados.

Metaplanificación Basada en Estados

Para poder realizar la metaplanificación, en primer lugar se han definido un conjunto de estados por los que atraviesa una tarea a lo largo de su ciclo de vida. De esta manera se reduce el problema de la metaplanificación a un problema de transición entre estados. La Figura 8-7 resume los principales estados por los que pasa una tarea durante todo el proceso. Existen algunos estados intermedios que no se muestran en la Figura y que responden a detalles de implementación.

La definición de los estados se muestra a continuación:

- UNASSIGNED. Es el estado inicial de una tarea, que todavía no ha sido seleccionada por el metaplanificador.
- SCHEDULED. El metaplanificador ha seleccionado el mejor recurso para la tarea pero todavía no ha comenzado la transferencia de datos al recurso destino.
- STAGED_IN. La fase de transferencia inicial de archivos ha sido completada satisfactoriamente y en la máquina destino se encuentra tanto la aplicación como los archivos dependientes.
- ACTIVE. La tarea está en ejecución en el recurso remoto.

- DONE. La tarea ha finalizado su ejecución en la máquina destino. Como se ha comentado anteriormente, es posible que una tarea en este estado realmente haya sufrido un fallo en su ejecución, por lo que se utilizarán los mecanismos de detección de errores implementados por GMarte para detectar los posibles errores en la ejecución.
- STAGED_OUT. Los archivos que ha generado la aplicación han sido transferidos a la máquina local.
- COMPLETED. La tarea ha acabado satisfactoriamente, los ficheros creados en la máquina destino han sido eliminados y no se debe realizar ninguna acción adicional con la tarea. Por lo tanto, este es el último estado que alcanza una tarea.
- FAILED. La tarea ha fallado, bien porque no pudo ser ejecutada por la máquina remota o porque se produjo un fallo durante la ejecución.

Inicialmente, la tarea parte del estado UNASSIGNED, donde únicamente puede ser promocionada al estado SCHEDULED mediante la selección, por parte del metaplanificador, del recurso en el que se debe ejecutar. Posteriormente, una vez realizada la fase de transferencia de archivos iniciales, la tarea queda en la fase de STAGED_IN. Una vez que comienza la ejecución de la tarea en la máquina destino, ésta pasa a estado ACTIVE. Cuando la ejecución finaliza la tarea alcanza automáticamente el estado DONE o quizás el estado FAILED en caso de algún error durante la ejecución. Una vez realizada la transferencia de resultados se alcanza el estado STAGED_OUT, donde se comprueba el código de salida de la aplicación y la palabra clave para verificar que la tarea acabó satisfactoriamente. En caso afirmativo se alcanza el estado COMPLETED, tras borrar todos los ficheros generados en la máquina destino, o el estado FAILED si se descubre que la tarea falló. Nótese que, adicionalmente, los errores irrecuperables ocurridos durante las fases de stage-in y stage-out también conducen al estado FAILED.

Mediante esta estrategia, una tarea que está en un determinado estado únicamente puede ser promocionada al siguiente estado, siguiendo su ciclo de vida de acuerdo al diagrama de transición entre estados de la Figura 8-7.

8.3. El metaplanificador OrchestratorScheduler

La clase OrchestratorScheduler proporciona la funcionalidad de un metaplanificador de tareas multi-hilo. Dado que implementa la interfaz Scheduler, esta clase proporciona código a los métodos que definen la política para seleccionar la tarea a ejecutar (*chooseGridTask*), para seleccionar el mejor recurso una vez seleccionada la tarea (*chooseGridResource*) y para elegir el número de procesadores involucrado en la ejecución una vez elegida la tarea y el recurso donde ejecutarla (*chooseNumberOf-Processors*).

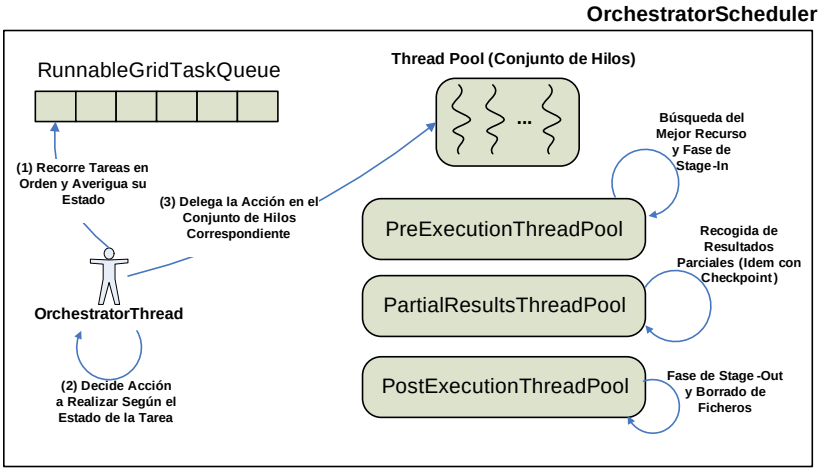


Figura 8-8: Esquema general de los componentes de OrchestratorScheduler.

La Figura 8-8 muestra un esquema general de los componentes que integran este metaplanificador. Se observa que incluye una cola de tareas (objetos RunnableGridTask) y un hilo de ejecución (OrchestratorThread) que se encarga de recorrer circularmente la cola de tareas averiguando el estado de cada una de ellas para decidir la acción a llevar a cabo. La cola de tareas sigue un modelo de cola de prioridad, donde las tareas están ordenadas de forma lógica de acuerdo a varios niveles diferentes de prioridad que son especificados por el usuario. La prioridad impone un ordenación sobre el conjunto de tareas, de manera que aquellas más prioritarias son atendidas antes que las de menor prioridad. El orden entre las tareas de idéntica prioridad es arbitrario.

El OrchestratorScheduler dispone de tres conjuntos de hilos de ejecución (*thread pool*) que permiten realizar concurrentemente las principales fases en la ejecución de tareas. Para ello se utiliza la funcionalidad, recientemente incluida en Java 1.5, para la creación de aplicaciones concurrentes, permitiendo la gestión y coordinación de grupos de hilos de ejecución. El grupo de hilos *PreExecutionThreadPool* permite realizar las tareas de pre-ejecución de las tareas, es decir, selección del mejor recurso y transferencia de datos para completar la fase de stage-in. Este conjunto de hilos, por lo tanto, permite que se puedan ejecutar concurrentemente las fases de stage-in para diferentes tareas.

Es fácil imaginarse que, al realizar concurrentemente varias fases de selección de recursos, es probable que la selección no sea óptima puesto que, en principio, es posible elegir el mismo recurso varias veces en base al número de procesadores libres, lo que, obviamente provocará que varias ejecuciones vayan destinadas a la misma máquina. Para paliar este problema, GMarte implementa técnicas de reserva de procesadores a nivel de cliente. Por lo tanto, nada más realizar una selección de recurso, se marcan una serie de nodos como reservados. Esta información es utilizada por las selecciones de recurso concurrentes para llevar a cabo su decisión, teniendo constancia de la existencia de menos procesadores disponibles de los que reporta el servicio de información.

El grupo de hilos *PartialResultsThreadPool* permite realizar concurrentemente la recogida de resultados parciales de diferentes ejecuciones, definidos en la clase *GridPartialResultsFileSet*. Esta funcionalidad resulta de utilidad para aplicaciones que involucran un proceso iterativo, donde se genera cada cierto tiempo un conjunto de datos. Mediante esta característica no es necesario esperar al final de la ejecución para procesar los datos obtenidos, sino que pueden ser transferidos al cliente cada cierto tiempo para solapar el post-proceso con la ejecución de la aplicación. Para la recogida periódica de ficheros de copia de seguridad se utiliza una estrategia análoga.

Finalmente el *PostExecutionThreadPool* aglutina un conjunto de hilos de ejecución que permiten realizar las labores de post-ejecución para diferentes tareas. Esto implica realizar la transferencia de datos de la máquina remota al cliente, así como el borrado de datos en la máquina destino.

Los conjuntos de hilos han sido diseñados para que inicialmente estén vacíos y puedan alcanzar un límite máximo de número de hilos en cada conjunto, cuyo valor es configurable por el usuario. De esta manera, es posible gestionar progresivamente la carga de trabajo del metaplanificador aumentando de manera dinámica el número de hilos de ejecución de cada conjunto, hasta alcanzar los límites impuestos por el usuario.

El uso de conjuntos de hilos, para implementar el metaplanificador, permite acelerar en gran medida el proceso de planificación frente a las alternativas secuenciales tradicionales. Además, el número de hilos de ejecución empleados no depende del número de tareas a ejecutar, lo que proporciona una alternativa escalable para el proceso de metaplanificación. Por lo general, las pruebas realizadas sugieren utilizar un valor de 4 hilos de ejecución para cada una de las fases. De esta manera es posible gestionar de manera concurrente hasta 4 tareas, lo que permite acelerar el proceso de metaplanificación sin introducir una sobrecarga excesiva en la máquina que ejecuta el metaplanificador. Lógicamente, estos valores dependen de la máquina en la que se ejecuta el metaplanificador.

8.3.1. Selección de Recursos

Por lo general, la selección del mejor recurso responde a la utilización de una función que permita valorar la idoneidad de una máquina para la ejecución de una aplicación. En GMarte, dicha función está basada en un modelo de prestaciones que trata de obtener un estimador del tiempo de ejecución global de la aplicación en cada uno de los recursos computacionales disponibles. Para ello se considera el ancho de banda del enlace de conexión de red y características dinámicas del recurso como el número de procesadores libres. Para las máquinas secuenciales se utiliza la información de la carga computacional del recurso, para decidir si es capaz de ejecutar una tarea con una cierta calidad de servicio.

Por lo tanto, el modelo de prestaciones ayuda al metaplanificador a decidir el mejor recurso para ejecutar la tarea a planificar en cada momento, escogiendo a aquel que proporcione el mínimo tiempo de ejecución estimado.

El modelo de prestaciones

El modelo de prestaciones utiliza información proporcionada por el usuario e información computada por GMarte para calcular el Tiempo Total de Ejecución Estimada en Grid (TTEEG), que representa el intervalo de tiempo desde que el metaplanificador decide asignar la tarea a un recurso hasta que la transferencia de resultados de la tarea finaliza.

Cada objeto GridTask (Figura 8-2) especifica el Tiempo de Ejecución Estimado (TEE) a través de la clase GridTaskConfiguration; el Tamaño Estimado de los Ficheros de Stage-In (TEFSI); el Tamaño Estimado de los Ficheros de Stage-Out (TEFSO), por medio de la clase GridOutputFileSet; el Tamaño Estimado de los Ficheros de Checkpoint (TEFC) así como el número de copias de seguridad que se van a generar (k_c), a través de la clase GridCheckpointFileSet; el Tamaño Estimado de los Ficheros de Resultados Parciales (TEFRP) así como el número de resultados parciales que se van a generar (k_{rp}). El valor de TEFSI es calculado automáticamente por GMarte, dado que los ficheros de stage-in son inicialmente conocidos antes del comienzo del proceso de metaplanificación.

El TTEEG, descrito por la Ecuación 8.1, incluye tanto el Tiempo Estimado de Ejecución en el Recurso Remoto (TEERR) como el Tiempo Estimado de Transferencias (TET).

$$TTEEG = TEERR + TET \tag{8.1}$$

Para estimar el TEERR se utiliza un modelo sencillo, descrito por la Ecuación 8.2, que favorece la elección de máquinas multiprocesador donde se pueda realizar una ejecución paralela, siendo más valorada aquella máquina con mayor número de procesadores libres (p). Para ello, se asume un modelo de paralelización perfecta donde el tiempo de ejecución se dividirá en un factor p al realizar una ejecución con tantos procesadores.

$$TEERR = \begin{cases} \frac{TEE}{p} & \text{Si la aplicación es paralela.} \\ TEE & \text{Si la aplicación es secuencial.} \end{cases} \tag{8.2}$$

Por otra parte, la estimación del TET se lleva a cabo de acuerdo a la Ecuación 8.3 teniendo en cuenta el ancho de banda (BW) entre la máquina cliente y el recurso remoto. Nótese que en el cálculo de esta expresión se está introduciendo de manera implícita el concepto de proximidad de recursos, priorizando la selección de recursos que permitan una alta tasa de transferencia de datos, por lo general, aquellos que se encuentren en un entorno cercano a la máquina cliente. GMarte tiene en cuenta los cambios en las condiciones de la red, ya que el valor de BW se recalcula despues de cada transferencia de datos y se asigna al recurso. Esto permite tener disponible un estimador dinámico del ancho de banda entre el cliente y la máquina remota. Esta característica permite detectar congestiones en la red, donde el tiempo involucrado en una transferencia de datos masiva podría superar al tiempo de ejecución.

$$TET = \frac{TEFSI + k_c \cdot TEFC + k_{rp} \cdot TEFRP + TEFSO}{BW} \tag{8.3}$$

Una vez seleccionado el recurso computacional, hay que decidir el número de procesadores involucrados en la ejecución. Para el caso de las aplicaciones secuenciales la decisión es trivial (1 procesador) pero no es así para las aplicaciones paralelas. La decisión involucra 3 criterios ortogonales. En primer lugar, el usuario suele preferir que la ejecución sea lo más rápida posible, por lo que tenderá de forma natural a preferir un elevado número de procesadores. Por otra parte, es conocido que un mayor número de procesadores no implica necesariamente una reducción en el tiempo de ejecución, ya que depende de la escalabilidad de la aplicación paralela. En realidad, es habitual que la eficiencia de una aplicación paralela, que mide la utilización de los recursos computacionales, disminuya conforme aumenta el número de procesadores. Finalmente, los propietarios de los recursos computacionales están interesados en maximizar su utilización, aprovechando la capacidad de cómputo de las máquinas para satisfacer las peticiones de múltiples clientes.

La estrategia adoptada es la siguiente: El usuario puede definir un límite mínimo y máximo de procesadores a utilizar en la ejecución paralela. El límite mínimo supone definir una cierta calidad de servicio para su aplicación. Aquellas máquinas que no dispongan del número mínimo de procesadores serán descartadas por el metaplanificador. El límite máximo restringe el número de procesadores a utilizar, y es conveniente fijar ese valor al número de procesadores por encima del cual la eficiencia de la aplicación es notablemente inferior. Este valor es completamente dependiente de la aplicación y es responsabilidad del usuario su cálculo aproximado, dado que el metaplanificador, al ser un componente genérico para múltiples aplicaciones, no debe disponer de mecanismos para verificar su validez.

Para aquellas tareas que no definen una estrategia de selección de número de procesadores, se utiliza una política amigable con los recursos computacionales que supone restringir las ejecuciones paralelas para que no ocupen la totalidad de los procesadores libres. Para ello se ha optado por una estrategia sencilla, donde se fija un umbral de 8 procesadores disponibles en el recurso, por encima del cual las ejecuciones serán llevadas a cabo con la mitad de los procesadores disponibles. Obviamente, siempre respetando los límites impuestos por el usuario. Esta sencilla política evita ocupar la totalidad de una máquina remota con una única ejecución paralela, permitiendo albergar otras ejecuciones en el recurso, bien del mismo usuario o de otros.

En la actualidad, el campo de las tecnologías Grid dedicadas a la metaplanificación de tareas incluye numerosos trabajos dedicados únicamente a la política de selección de tareas. En realidad, es conocido que no existe una estrategia de asignación de tareas óptima para cualquier tipo de aplicación. Por eso, en lugar de optar por el desarrollo de un único metaplanificador se ha optado por una arquitectura abierta que permita al usuario incorporar diferentes estrategias de metaplanificación. Por ello, la clase Scheduler (Figura 8-2) es abstracta y únicamente define las operaciones que debe

implementar cualquier metaplanificador.

Cualquier usuario puede modificar la funcionalidad del metaplanificador de manera sencilla mediante la extensión de la clase OrchestratorScheduler y la redefinición de cualquier de los métodos para alterar la política de selección de tareas, selección de recursos y selección de número de procesadores.

Alternativamente, si únicamente se desea cambiar la política de selección de recursos y mantener la implementación del metaplanificador, es posible conseguirlo sin más que realizar una subclase de GridResourceMatcher (Figura 8-2) y sobrescribir el método *getTGEET* (Total Grid Estimated Execution Time, TGEET) para puntuar de manera diferente a los recursos.

En este sentido, GMarte proporciona un marco fácilmente extensible para la introducción de nuevas estrategias, tanto de metaplanificación como de selección de recursos, que pueden ser adaptadas para su utilización en aplicaciones concretas que precisen unos requisitos de ejecución determinados.

8.3.2. Tolerancia a Fallos

La tolerancia a fallos que implementa el metaplanificador se apoya en los mecanismos de detección de errores implementados en las capas inferiores, tanto de acceso al sistema de información como la de ejecución de tareas.

Si se produce un fallo en la ejecución de una tarea, debe replanificarse de acuerdo al diagrama de transición de estados descrito en la Figura 8-7. Esta acción implica desencadenar un nuevo proceso de selección de recurso para poder ejecutar nuevamente la aplicación. Para tratar de reducir el impacto en el tiempo de ejecución de una aplicación fallida, GMarte soporta la especificación de tareas que permiten salvar periódicamente su estado. Para ello, la tarea debe generar una serie de ficheros a partir de los cuales se pueda retomar la ejecución desde el último instante en el que se guardó el estado.

En este sentido, una GridTask puede especificar el conjunto de ficheros de copia de seguridad que generará la aplicación cada cierto tiempo. Asimismo, el metaplanificador puede ser configurado para que periódicamente transfiera esos ficheros a la máquina local. Tener disponible esos ficheros en la máquina local permite que, en caso de fallo en un recurso, se pueda continuar la ejecución a partir de la última copia de seguridad. En realidad, cuando se selecciona una tarea para su ejecución y comienza la fase de transferencia de archivos, se investiga si existe información de copia de seguridad. En ese caso, será transferida antes del comienzo de la ejecución para evitar comenzar la ejecución desde el principio.

En el caso de que el fallo se produzca en un recurso computacional, por ejemplo si se detecta un fallo en el acceso al servicio de información, entonces las acciones a realizar cambian. En este caso se debe marcar el recurso como no disponible, y todas las tareas que se estaban ejecutando en el mismo deben ser replanificadas forzosamente en otros recursos que hayan disponibles. Existe un

componente llamado *GridResourceResurrector* que se encarga de investigar periódicamente el estado de los recursos del TestBed, para volver a recuperarlos e incorporarlos como recursos disponibles para la ejecución de tareas. Esta funcionalidad resulta muy útil para incorporar de manera dinámica recursos al proceso de metaplanificación.

En la actualidad, GMarte es capaz de guardar una copia de su estado en memoria secundaria, empleando los mecanismos de serialización de Java. De esta manera, en caso de fallo en el propio metaplanificador no se pierde la definición de tareas realizada y es posible retomar nuevamente la sesión de planificación. Esto implica que, para aquellas tareas que no hayan finalizado, se produce una reconexión para continuar su monitorización. De esta manera, un fallo en GMarte no afecta a la ejecución remota de las tareas.

Este esquema de tolerancia a fallos multinivel es capaz de gestionar un porcentaje bastante elevado de errores, que se pueden producir en cada uno de los niveles, permitiendo un comportamiento relativamente robusto del sistema.

8.4. Ejemplos Prácticos de Utilización de GMarte

Una vez revisada la funcionalidad ofrecida por GMarte, en esta sección se presentan dos ejemplos de utilización del API de alto nivel para comparar la facilidad de uso de GMarte frente a los mecanismos tradicionales ofrecidos por Globus. El primero tiene como objetivo realizar la ejecución remota de una tarea en un recurso de nuestra elección. El segundo utiliza la funcionalidad de metaplanificación para realizar la ejecución remota de tareas de manera desatendida, sin que el usuario decida los recursos específicos a utilizar.

8.4.1. Ejemplo de Ejecución Remota Simple

Se desea ejecutar, empleando 4 procesadores de una máquina remota llamada *ramses.dsic.upv.es*, una aplicación denominada *app3D* que reside en el directorio */tmp* de la máquina cliente. Esta tarea requiere un conjunto de ficheros de entrada, residentes en el mismo directorio, cuyo sufijo común es *.dat* y unos argumentos de linea de comandos (*-x 10*). Cuando la ejecución finalice, se desea que todos los ficheros generados durante la ejecución de la aplicación cuya extensión sea *.out*, sean transferidos automáticamente a la máquina local. Además, la tarea requiere que el recurso disponga como mínimo de 128 MBytes de RAM disponibles o, en caso contrario, se impedirá la ejecución sobre el recurso. Se asume que los aspectos relativos a seguridad, como credenciales válidas, están bien establecidos entre ambas máquinas. Bajo estas condiciones el extracto de aplicación GMarte mostrado en la Figura 8-9, ejecutado en la máquina cliente, consigue los objetivos.

Utilizando exclusivamente las herramientas ofrecidas por Globus Toolkit (2.4.3), el usuario debería haber realizado las siguientes acciones (los paréntesis indican el comando utilizado para realizar

```
GridTask gt =  new GridTask();
GridExecutableFile gef = new GridExecutableFile("/tmp/app3D");
gt.setGridExecutableFile(gef);
GridInputFileSet gifs =  new GridInputFileSet();
gifs.addGridFilesBySuffix("/tmp",".dat");
gt.setGridInputFileSet(gifs);
GridOutputFileSet gofs = new GridOutputFileSet();
gofs.addGridFilesBySuffix(".out");
gt.setGridOutputFileSet(gofs);
gt.setArguments("-x 10");
GridTaskLoR gtl = new GridTaskLoR();
gtl.addRequirement(AVAILABLE_RAM, 128);
gt.addGridTaskLoR(gtl);
GridResource gr = new GridResource("ramses.dsic.upv.es");
RunnableGridTask rgt = new RunnableGridTask(gt,gr);
rgt.setNumberOfProcessors(4);
rgt.stageIn(); rgt.run(); rgt.stageOut();
```

Figura 8-9: Ejemplo de aplicación GMarte para la ejecución remota de una tarea.

la acción):

1. Investigar si la máquina está encendida (*ping*), si el servicio de información MDS se está ejecutando (*grid-info-host-search*), si se están usando credenciales válidas para acceder al recurso, y si el servicio GRAM de la máquina destino se está ejecutando correctamente (*globusrun*).

2. Asegurarse que el recurso tiene al menos 128 MBytes de memoria RAM disponible, mediante la realización de una consulta LDAP (*grid-info-host-search*), que únicamente devuelva el valor del atributo *Mds-Memory-Ram-freeMB* obtenido del servidor MDS. Además, se debe construir otra consulta LDAP que devuelva el valor del atributo *Mds-Computer-Total-Free-nodeCount*. Si existen diversas colas de ejecución en el recurso, se debe investigar los valores de cada una de ellas para decidir si existen al menos 4 procesadores disponibles.

3. Decidir en qué punto del sistema de archivos de la máquina remota van a residir todos los ficheros y transferirlos a través de GridFTP (*globus-url-copy*). Como problema adicional, el protocolo GridFTP no mantiene los atributos de los ficheros y, por lo tanto, el fichero ejecutable en el recurso remoto tendrá su *flag* de ejecución desactivado, lo que previene su ejecución.

4. Decidir si la ejecución va a ser interactiva o en segundo plano, y comenzar la ejecución (*globusrun*) con el número de procesadores especificado. Si la ejecución es en segundo plano, es preciso consultar periódicamente el estado del trabajo (*globus-job-status*) hasta que el trabajo alcanza el estado DONE o FAILED.

5. Averiguar donde se habrán generado los ficheros resultado escritos por la aplicación, y transferirlos a la máquina local (*globus-url-copy*). Además, como medida de cortesía se deben borrar

```
GridTaskStudy gts = new MiCasoDeEstudio();
String hosts = {"machine1","machine2"};
TestBed tb = new TestBed(hosts);
Scheduler sched = new OrchestratorScheduler(tb, gts);
SchedulerConfiguration sconf = new SchedulerConfiguration();
sconf.setThreadCheckpointingPeriod(1800);
sched.setSchedulerConfiguration(sconf);
sched.start();
sched.waitUntilFinished();
```

Figura 8-10: Ejemplo de uso del API de GMarte para la metaplanificación de tareas.

todos los ficheros de datos generados en el recurso remoto, para que no consuman espacio en disco de manera innecesaria.

6. Realizar detección de errores a lo largo de todas las fases anteriores, para tomar acciones apropiadas que permitan, como mínimo avisar al usuario y, en el mejor de los casos, recuperarse ante los errores ocurridos.

Además, la descripción de acciones realizada asume que el recurso remoto es GT2. En caso de ser GT4, se deberían emplear protocolos y herramientas diferentes para la realización de las tareas descritas, lo que incrementa en gran medida la complejidad involucrada. Es fácil observar que, empleando los mecanismos tradicionales ofrecidos por GT, un usuario se enfrenta a un nivel de complejidad lo suficientemente elevado como para dificultar la adopción de la tecnología Grid para la ejecución de sus aplicaciones.

La secuencia de instrucciones de la aplicación GMarte (Figura 8-9) involucra la utilización de algunos servicios de GT como GridFTP, para transferencia de ficheros, MDS, para el acceso a la información del recurso y GRAM para la ejecución del trabajo. Sin embargo, se observa que el usuario queda alejado de toda esa complejidad, ya que únicamente interacciona con un API que le permite expresar de manera declarativa *qué* ejecutar, en lugar de centrarse en los aspectos prácticos sobre *cómo* ejecutar en un Grid.

8.4.2. Ejemplo de Metaplanificación

En este ejemplo se utilizan las capacidades de GMarte para realizar la ejecución de un conjunto de tareas. En este sentido, se muestra la funcionalidad de metaplanificación de tareas, proporcionado por la clase *OrchestratorScheduler*, para la ejecución de tareas sobre un conjunto de recursos computacionales formado por dos máquinas.

En este ejemplo, la clase *MiCasoDeEstudio* es una subclase de GridTaskStudy que debe ser declarada e implementada por el usuario, y que contiene la definición de todas las tareas (GridTasks) que van a ser ejecutadas en el Grid. Para cada tarea, el usuario especifica los ficheros de entrada,

los argumentos de línea de comandos, los ficheros resultado y el resto de parámetros configurables para las tareas, de la misma manera que se realizó en el primer ejemplo. Para las aplicaciones de barrido de parámetros es posible definir todos los atributos de una tarea, y clonarla posteriormente en múltiples instancias. En cada copia de tarea se modificaría el parámetro correspondiente. En este ejemplo particular se asume que la aplicación del usuario genera periódicamente unos ficheros de copia de seguridad, y el planificador se configura para transferir esos ficheros de la máquina remota a la local cada 30 minutos.

A partir de este ejemplo se puede observar que es bastante simple utilizar la funcionalidad de planificador de tareas de GMarte. Únicamente se debe definir el conjunto de tareas (GridTaskStudy), el conjunto de recursos computacionales (TestBed) y, opcionalmente, modificar la configuración del metaplanificador. Adicionalmente, se puede utilizar la funcionalidad de descubrimiento de recursos (GridResourceDiscoverer) para obtener un TestBed a partir de los recursos publicados en un GIIS o en un BDII.

Dado que GT no proporciona funcionalidad de metaplanificación, no es posible mostrar ninguna comparación con GMarte. Sin embargo, es fácil observar que toda la complejidad subyacente de la interacción con el Grid queda oculta tras el API de GMarte.

8.5. Áreas de Aplicación de GMarte

En la actualidad, el sistema de Computación en Grid GMarte está siendo aplicado en otros campos diferentes a los cubiertos por esta tesis.

En primer lugar, se está utilizando en el área del análisis dinámico de estructuras, que es una de las fases que mayor tiempo precisan en el ciclo de diseño de un edificio. Los elevados requisitos computacionales y de memoria, para gestionar de manera eficiente simulaciones estructurales tridimensionales, precisan la utilización de técnicas de computación de altas prestaciones. Además, existen leyes nacionales como la NCSE-02 (Norma de Construcción Sismoresistente) que obligan el estudio de diferentes alternativas estructurales bajo la influencia de múltiples terremotos.

En el marco de los proyectos de cálculo de estructuras llevados a cabo en el GRyCAP, se ha desarrollado un servicio Grid orientado al cálculo dinámico de estructuras, accesible a través de Internet por medio de herramientas gráficas. En este sentido, se ha incorporado la funcionalidad de GMarte para dotar al servicio Grid de un metaplanificador para la ejecución eficiente de simulaciones estructurales sobre una infraestructura Grid [175].

En segundo lugar, se ha establecido una colaboración con el Área de Fotónica del Grupo de Modelización Interdisciplinar (Intertech), de la Universidad Politécnica de Valencia, en el campo de la simulación del guiado de la luz en fibras de cristal fotónico. Las especiales características de guiado de la luz en fibras de cristal fotónico tienen potenciales aplicaciones en una gran variedad de cam-

pos que van desde las comunicaciones ópticas hasta la fabricación de dispositivos optoelectrónicos. Las propiedades de propagación del campo electromagnético en estos sistemas vienen determinadas por la configuración particular de la estructura dieléctrica transversal. A partir de un modelo que representa la propagación no lineal de un campo monocromático en un medio inhomogéneo, se trata de analizar la evolución no lineal del campo eléctrico a lo largo de la fibra. Esto supone un alto coste computacional, que puede acarrear semanas o incluso meses de ejecución de múltiples simulaciones, independientes entre sí, en las cuales varían diferentes parámetros de configuración. Para ello, se está realizando la ejecución concurrente de las diferentes simulaciones paramétricas, en una infraestructura Grid, mediante la herramienta GMarte.

En tercer lugar, se ha iniciado una colaboración con el Laboratorio de Fisiología Computacional (FisioComp) de la Universidad Federal Juiz de Fora, en Brasil. En esta colaboración se plantea estimar una serie de parámetros de una simulación cardiaca, a partir de los resultados de la misma en forma de electrocardiograma. Este proceso implica la ejecución de múltiples simulaciones cardiacas y la utilización de algoritmos genéticos, encargados de lanzar de dichas simulaciones. En este sentido, se plantea la utilización de un servicio Grid de metaplanificación basado en GMarte que permita ejecutar dichas simulaciones, en una infraestructura Grid, de manera transparente para el usuario. Ello supondrá la integración de la aplicación de algoritmos genéticos desarrollada en FisioComp con el servicio Grid de metaplanificación basado en GMarte.

Por último, recientemente se ha iniciado otra colaboración con el Departamento de Informática de la Universidade do Minho, en Portugal, para la integración de un entorno de programación basado en esqueletos llamado JaSkel [176]. Esta herramienta permite ayudar al programador a construir aplicaciones paralelas bien estructuradas, proporcionando un conjunto de patrones comunes de paralelización llamados esqueletos. En JaSkel, un programador construye una aplicación paralela seleccionando y componiendo los patrones paralelos que implementan su estructura. Posteriormente, los esqueletos se completan con el código dependiente de la aplicación. Esto permite a los programadores escribir código que puede ser ejecutado de manera eficiente en máquinas de memoria compartida y clusters de PCs. En esta colaboración se ha realizado una primera integración de GMarte y JaSkel para la creación de un entorno que permite la ejecución de aplicaciones paralelas sobre clusters y entornos Grid. Para ello, se utiliza la funcionalidad de GMarte para la ejecución transparente de esqueletos JaSkel en plataformas Grid.

8.6. Trabajos Relacionados

Actualmente, existen varias lineas de investigación que tratan de proporcionar un entorno para simplificar el uso de los recursos distribuidos. El Grid Application Framework for Java (GAF4J) [177] es un sencillo entorno de clases que se encarga de abstraer la problemática esencial de interactuar con

una infraestructura Grid, que se asume que es Globus Toolkit 2.0. Javelin [178] es una infraestructura basada en Java para la computación paralela en Internet, que proporciona un entorno diseñado *ad hoc* para conseguir la ejecución remota mediante la utilización de RMI (Remote Method Invocation) para las comunicaciones. También, ProActive [179, 180] es una librería Java para la computación paralela, distribuida y concurrente, mediante la creación de objetos activos que pueden ser distribuidos y comunicarse por medio de las interfaces proporcionadas.

Todas estas herramientas tienen como objetivo simplificar el proceso de migrar o desarrollar una aplicación Java, para que sea ejecutada en un entorno de computación distribuido. Sin embargo, GMarte es un middleware desarrollado en Java que permite la ejecución remota de aplicaciones que pueden haber sido escritas en cualquier lenguaje de programación, no únicamente Java. Por supuesto, la aplicación debe poder ejecutarse en la máquina remota.

También existen proyectos cuyo objetivo fundamental consiste en la ejecución sobre recursos distribuidos, proporcionando, de esta manera, funcionalidad de metaplanificación. Legion [181] es un middleware basado en objetos que persigue la creación de un único supercomputador virtual, a partir de un conjunto de recursos distribuidos. Legion proporciona un API para la gestión de recursos, así como la gestión de las ejecuciones, pero está basado en su propio entorno, con lo que su uso queda restringido a las infraestructuras computacionales basadas en Legion. Condor [182] es un software que permite la utilización efectiva de la potencia computacional de un conjunto de recursos. Sin embargo, aunque Condor proporciona una pasarela para su uso con Globus (Condor-G [165]), actualmente solo ofrece una interfaz de línea de comandos y no un API específico que provea mayor nivel de expresividad.

Quizá el proyecto más relacionado con el trabajo desarrollado es GridWay [166]. GridWay es un metaplanificador de código abierto que permite la ejecución desatendida, confiable y eficiente de trabajos sobre recursos computacionales basados en Pre-WS GRAM y WS-GRAM. El metaplanificador ofrece funcionalidades avanzadas de migración de tareas en base a detección de pérdidas de prestaciones. Sin embargo, GridWay únicamente está disponible para plataformas UNIX. GMarte, por el contrario, es multiplataforma pudiendo ser ejecutado en multitud de arquitecturas y sistemas operativos.

El entorno GMarte combina varios aspectos importantes. En primer lugar, proporciona un marco fácil de utilizar para los usuarios que quieran utilizar las ventajas de un Grid computacional para la ejecución de aplicaciones de alta productividad. En segundo lugar, GMarte permite la interacción con Grids basados en Globus Toolkit y LCG-2. Esto permite utilizar su funcionalidad en los numerosos Grids computacionales basados en GT y delegar la ejecución de tareas sobre una infraestructura Grid de producción. En tercer lugar, GMarte proporciona interfaces de alto nivel, en la forma de un API orientado a objetos y mediante documentos XML. De esta manera se obtiene una alta versatilidad para utilizar la herramienta como parte de otras aplicaciones. En cuarto lugar,

la capacidad multiplataforma le convierte en un componente genérico capaz de ser desplegado en diferentes ámbitos y máquinas.

8.7. Conclusiones

En este capítulo se ha descrito GMarte, el sistema de Computación en Grid desarrollado para realizar metaplanificación de tareas sobre Grids computacionales. Se ha detallado la arquitectura del sistema, compuesta por diferentes capas de abstracción sobre Globus Toolkit, detallando cada una de las principales áreas: Abstracción del acceso al sistema de información, abstracción del proceso de ejecución de una tarea y abstracción del proceso de metaplanificación. Todas ellas se encargan de ocultar la complejidad del Grid, utilizando los protocolos adecuados y exponiendo al usuario un conjunto de interfaces de alto nivel.

El desarrollo de una herramienta de estas características permite que usuarios no expertos en Computación en Grid utilicen la funcionalidad de GMarte para la ejecución de aplicaciones científicas. A continuación se resumen las principales características:

- Facilidad de Uso. Es fácil de utilizar por usuarios que desconozcan los servicios de bajo nivel ofrecidos por GT. Para ello, ofrece un API de alto nivel que permite utilizar su funcionalidad desde un lenguaje de programación. Además se puede interactuar con GMarte mediante la especificación de tareas y recursos computacionales a través de XML.

- Eficiencia. Implementa estrategias enfocadas a conseguir una ejecución eficiente de las tareas. Este punto implica tanto la definición de políticas de selección de recursos computacionales, como la gestión eficiente del proceso de metaplanificación, para reducir la sobrecarga introducida por el propio proceso de asignación de tareas.

- Multiplataforma. Es accesible desde múltiples arquitecturas y sistemas operativos. Aunque actualmente los sistemas Grid están prácticamente enfocados a su uso en entornos UNIX, no se puede obviar la existencia de numerosas máquinas Windows dentro de la comunidad científica donde reside un interés en obtener los beneficios del Grid.

- Extensibilidad. La utilización de un marco de programación orientado a objetos permite modificar el sistema de forma sencilla, para introducir nuevas políticas de selección de recursos e incluso nuevos metaplanificadores, aprovechando la infraestructura ya implementada.

- Soporte para Computación Paralela. Permite la ejecución de aplicaciones paralelas basadas en el estándar MPI, en los clusters de PCs de la infraestructura Grid.

- Tolerancia a Fallos. En una infraestructura distribuida como un Grid, esta característica resulta imprescindible para garantiza un nivel de robustez que permita tolerar la mayor parte de

los fallos que se puedan producir a lo largo del proceso de planificación de tareas. GMarte implementa técnicas de detección y recuperación de errores, a varios niveles, tanto en la ejecución de las aplicaciones como en las transferencias de datos.

- Interoperabilidad con Infraestructuras de Producción: Permite delegar la ejecución de tareas a los RB de LCG-2. Esta funcionalidad permite utilizar de manera combinada recursos de una infraestructura Grid local y recursos del mayor despliegue computacional existente en la actualidad.

El desarrollo de componentes genéricos, que faciliten la utilización de las tecnologías Grid, permite acelerar la adopción de las mismas por parte de comunidades científicas interesadas en la utilización del Grid. En este sentido, el desarrollo de GMarte supone la aportación de un grano de arena al ecosistema de herramientas Grid.

Capítulo 9

El Servicio Grid de Metaplanificación GMarteGS

"We are stuck with technology when what we really want is just stuff that works". Douglas Adams, British author.

En este capítulo se describe la adaptación de GMarte a una arquitectura orientada a servicio, con el objetivo de desarrollar un metaplanificador multiusuario, accesible a través de Internet, que ofrezca un servicio genérico de ejecución remota de aplicaciones sobre Grids computacionales. En primer lugar se describen brevemente los conceptos subyacentes a una Arquitectura Orientada a Servicio. A continuación se instancian dichos conceptos a las herramientas ofrecidas por Globus Toolkit 4, citando los estándares adoptados. Posteriormente, se muestra la arquitectura del servicio Grid desarrollado, describiendo el flujo de interacción entre el cliente y el servicio. Luego, se aborda la descripción de un cliente gráfico empleado para interactuar con el servicio Grid, detallando el uso de la tecnología empleada para crear un componente genérico, accesible desde Internet, que requiere una mínima configuración por parte del usuario. Finalmente, se describen los trabajos relacionados, destacando las principales ventajas del sistema desarrollado.

9.1. Introducción

La Arquitectura Orientada a Servicio (Service-Oriented Architecture, SOA) [183], tal y como es definida por el grupo de trabajo de Modelos de Referencia de OASIS [184], es un paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios de administración. Para ello, proporciona herramientas comunes para ofrecer, descubrir y utilizar dichas capacidades.

SOA permite la creación de aplicaciones que combinan servicios débilmente acoplados e interope-

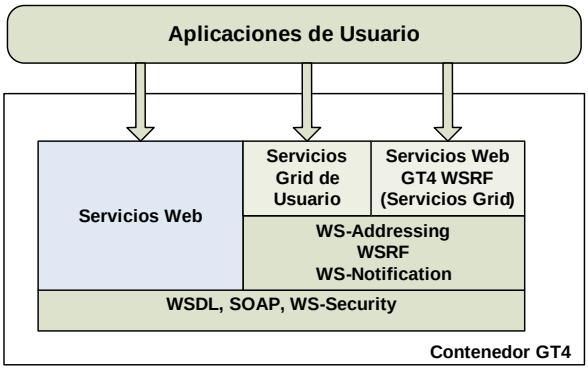


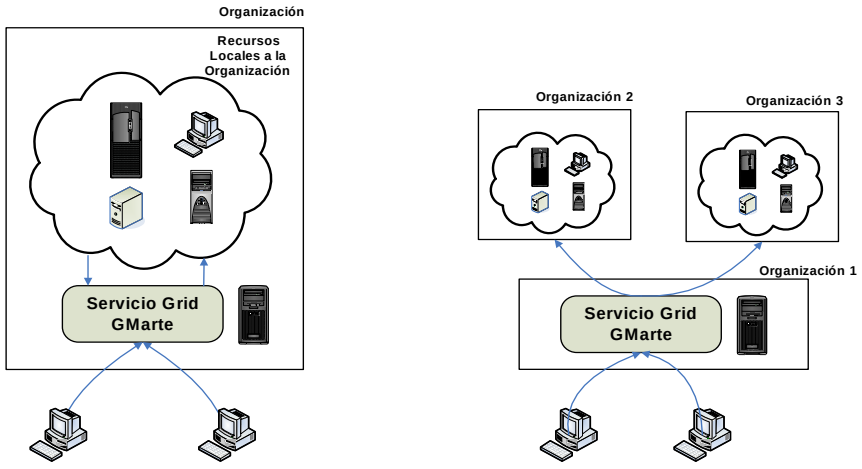
Figura 9-1: El contenedor de aplicaciones proporcionado por GT4.

rables. Estos servicios interactúan entre ellos en base a una definición formal de sus interfaces, que es independiente tanto de la plataforma subyacente como del lenguaje de programación. La definición del interfaz oculta la implementación del servicio, que puede realizarse en cualquier lenguaje. Esto permite que diferentes servicios, implementados con diferentes lenguajes, puedan interactuar entre ellos por medio de las interfaces definidas. Por lo general, SOA se basa en un conjunto de estándares de Servicios Web [185]. Con el objetivo de introducir los conceptos utilizados en el desarrollo del servicio Grid de metaplanificación, se describe brevemente a continuación la utilización de SOA, por parte de Globus Toolkit, aplicada a la Computación en Grid.

GT4 proporciona la implementación de un conjunto de servicios Grid que se adhieren a la especificación OGSA (Open Grid Services Architecture) [186]. Esta especificación ha sido desarrollada por el OGF (Open Grid Forum) [187] y su principal objetivo es la definición de una arquitectura abierta, común y estándar para todos los servicios que pueden encontrarse en un sistema Grid (gestión de trabajos, gestión de recursos, servicios de seguridad, etc). OGSA define la arquitectura subyacente en base a unos servicios Web especiales, denominados servicios Grid, que son capaces de mantener su estado de una invocación a otra, característica que no soportan los servicios Web tradicionales.

En este sentido, la especificación WSRF (Web Services Resource Framework) [188] se encarga de definir cómo son capaces de almacenar el estado los servicios Grid, empleando para ello el concepto de recurso (*WS-Resource* [189]). Básicamente, el estado no se guarda en el servicio Grid sino en un contenedor de información llamado recurso. GT4 incluye una implementación de WSRF, así como un conjunto de servicios Grid, desarrollados sobre esta tecnología, que cumplen las especificaciones de OGSA. Además, se facilita el desarrollo de nuevos servicios Grid, basados en esta tecnología, que sean interoperables y utilizables por la comunidad de usuarios del Grid.

La Figura 9-1 muestra un resumen del contenedor de aplicaciones proporcionado por GT4. En el nivel más bajo se encuentran el lenguaje WSDL (Web Services Definition Language) [190], que permite la definición de las interfaces de los servicios, el protocolo SOAP, que permite la comunicación



(a) Acceso a un Grid de Organización

(b) Acceso a múltiples Grids

Figura 9-2: Diferentes escenarios en los que se puede aplicar el servicio Grid de metaplanificación ofrecido por GMarte.

entre servicios Web, y los mecanismos de seguridad proporcionados por WS-Security [191]. Por encima, se implementa WSRF, junto con otras especificaciones, para permitir el desarrollo de los servicios Grid estándar, así como la creación de servicios Grid por parte de los usuarios.

En este campo se incluye una de las aportaciones de esta tesis, mediante la creación de un servicio multiusuario genérico e interoperable con otros servicios, para la ejecución de aplicaciones científicas genéricas sobre Grids computacionales.

El desarrollo de una herramienta de estas características implica varias ventajas. En primer lugar, la utilización de una aproximación orientada a servicio permite separar las funciones de especificación de tareas y gestión de la ejecución, en diferentes máquinas. Esto permite la creación de un sistema multiusuario que pueda gestionar concurrentemente las sesiones de metaplanificación de diferentes usuarios. Además, la utilización de un esquema orientado a servicio simplifica la funcionalidad del cliente, permitiendo la creación de componentes ligeros encargados de la interacción con el servicio.

Por otra parte, este escenario abre nuevas posibilidades, como las mostradas en la Figura 9-2. Por un lado, es posible desplegar el servicio de metaplanificación dentro de una organización, sirviendo de punto de entrada a sus recursos computacionales. De esta manera, se permite facilitar el acceso a un Grid de organización (Figura 9-2.a). Esta situación permite a una entidad proporcionar acceso seguro y controlado a múltiples clientes que desean ejecutar sus tareas en los recursos de la organización. Por otro lado, es posible desplegar el servicio dentro de una organización para realizar ejecuciones en recursos de otras instituciones (Figura 9-2.b). De esta manera, el servicio proporciona una pasarela entre un conjunto de recursos computacionales distribuidos, posiblemente de diferentes infraestructuras Grid, y los clientes que desean el acceso a esa infraestructura.

Como se puede observar, utilizar una aproximación orientada a servicio permite nuevas formas de

computación que ya no están limitadas a la mera ejecución de tareas, sobre un Grid computacional, desde una máquina cliente. Un componente de estas características puede encargarse de la gestión de tareas, tanto dentro de una organización como en múltiples organizaciones, aprovechando las características de metaplanificación implementadas en GMarte.

9.2. Funcionalidad e Interacción

El servicio Grid basado en GMarte (en adelante GMarteGS) proporciona servicios de metaplanificación a múltiples clientes. Para ello, se introduce el concepto de *sesión de metaplanificación*, y el servicio ofrece una serie de operaciones para su definición y manipulación. Se define una sesión de metaplanificación como el conjunto de tareas que desea ejecutar un cliente. Adicionalmente, puede incluir información sobre los recursos computacionales a utilizar y la configuración del metaplanificador. De esta manera, se permite que el usuario pueda especificar los recursos a utilizar para la ejecución de sus tareas o bien utilizar aquellos que estén accesibles desde el servicio Grid. La sesión de metaplanificación tiene una correspondencia directa con un objeto GMarte, denominado *MetaschedulingSession*, que aglutina la siguiente información:

- Definición de las tareas a ejecutar. Esta información tiene correspondencia directa con el objeto de tipo GridTaskStudy, disponible en el entorno de abstracción proporcionado por GMarte.
- Definición de las recursos sobre los que ejecutar. Esta información está representada por un objeto de tipo TestBed.
- Configuración del proceso de metaplanificación. El usuario puede especificar parte de la configuración del metaplanificador, como el número de hilos de ejecución empleados para las principales fases, la política de planificación empleada, etc. Esta información se especifica mediante un objeto de tipo SchedulerConfiguration.

Todo intercambio de información entre el cliente y el servicio se realiza mediante documentos XML. La principal razón es que, al ser un formato en modo texto, puede obtenerse como resultado de la invocación de un método, facilitando así la comunicación entre el cliente y el servicio.

Para ello, se permite la definición de los principales objetos GMarte en lenguaje XML. Además, se ha implementado un sistema de enlace de datos basado la herramienta XmlWrappers [111], desarrollada por el autor, que permite obtener un objeto GMarte a partir de su descripción en formato XML y viceversa. De esta manera se facilita la transformación de una representación textual a una representación como objeto Java. Esta aproximación permite que el cliente defina las tareas, los recursos y la configuración del metaplanificador, en formato XML. Estas definiciones serán utilizadas por GMarteGS para obtener los correspondientes objetos GMarte, en Java, necesarios para comenzar el proceso de metaplanificación.

```
<?xml version=‘‘1.0’’?>
<GridTaskStudy>
 <GridTask>
  <ExecFile> app </ExecFile>
  <GridInputFileSet>
    <GridFile> datai </GridFile>
  </GridInputFileSet>
  <GridOutputFileSet>
   <WildCard> *.out </WildCard>
  </GridOutputFileSet>
  <Arguments> -x 10 </Arguments>
 </GridTask>
</GridTaskStudy>
```

```
<?xml version=‘‘1.0‘‘?>
<TestBed>
 <GridResource>
  <Name>
      pilos.itaca.upv.es
  </Name>
 </GridResource>
 <GridResource>
  <Name>
     muro.itaca.upv.es
  </Name>
 </GridResource>
</TestBed>
```

```
<?xml version=‘‘1.0‘‘?>
<SchedulerConfiguration>
 <ClassName>
   OrchestratorScheduler
 <ClassName>
</SchedulerConfiguration>
```

Figura 9-3: Ejemplo de documentos XML simplificados para la especificación de las tareas, recursos y configuración del metaplanificador.

La Figura 9-3 muestra un ejemplo de documentos XML, simplificados por razones de espacio, para la descripción de las tareas, de los recursos computacionales y la configuración del metaplanificador. En la actualidad, los documentos XML permiten una expresividad ligeramente inferior a los correspondientes objetos GMarte.

A continuación, se describen las principales operaciones soportadas por el servicio:

- Creación de una Sesión. Permite la creación de una nueva sesión de metaplanificación.
- Inicialización de la Sesión. Esta operación permite especificar los documentos XML que describen las tareas y, opcionalmente, las máquinas y la configuración del metaplanificador. Así mismo, las credenciales del usuario son delegadas al servicio para que la autenticación con los recursos del Grid sea realizada con la identidad del usuario.
- Obtención de Información de la Sesión. Obtiene información como el nombre de la sesión, la carpeta en la máquina del servicio donde se guardan los datos, la información del tiempo invertido en su ejecución, etc.
- Obtención del Estado de la Sesión. Esta operación permite obtener un resumen del estado de la sesión. En él se indica, para cada una de las tareas, la máquina a la que ha sido asignada, el número de procesadores involucrados en la ejecución y el estado en el que se encuentra, dentro del ciclo de vida de las tareas (tal y como se mostró en la Figura 8-7).
- Obtención del Estado de la Infraestructura Grid. Obtiene un resumen del estado de los recursos computacionales. En él se indica el nombre de la máquina, junto con información computacional básica como el número de procesadores disponibles frente a los totales o la memoria RAM disponible.
- Cancelación de la Sesión. Permite detener todas las ejecuciones de la sesión. Esta acción implica

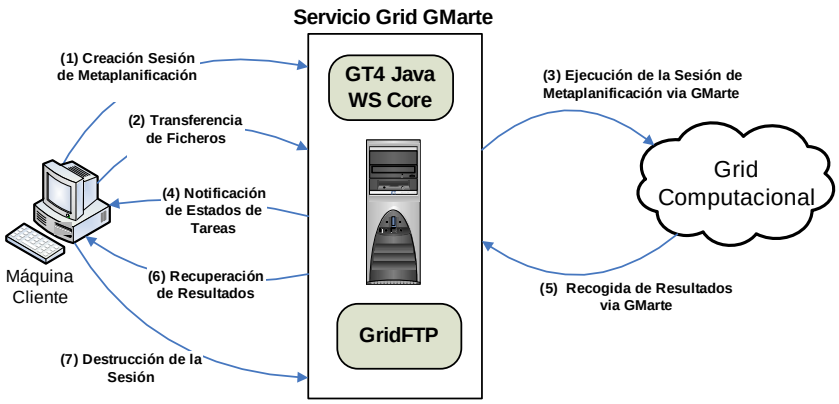


Figura 9-4: Funcionalidad general proporcionada por el servicio de metaplanificación.

cancelar tanto los trabajos que se encuentran activos en los recursos computacionales como aquellos que todavía no han sido ejecutados.

- Destrucción de la Sesión. Permite cancelar una sesión y eliminar toda la información vinculada a la misma, es decir, los ficheros almacenados en la máquina del servicio y los recursos asociados. Esta operación impide la ejecución de cualquier operación posterior sobre esta sesión.

Con respecto a la delegación de credenciales del usuario, cabe destacar que es igualmente posible que el servicio Grid utilice unas determinadas credenciales, independientemente de las suministradas por el cliente. De esta manera, es posible que múltiples clientes puedan tener acceso al Grid a través de una única identidad. Además, utilizando un conjunto de credenciales, es posible proporcionar una pasarela computacional entre diferentes infraestructuras Grid que precisen credenciales diferentes para ser accedidas.

La Figura 9-4 resume de manera esquemática la principal funcionalidad ofrecida por el servicio Grid desarrollado. En primer lugar, el cliente crea una nueva sesión de metaplanificación, con el objetivo de ejecutar un conjunto de tareas sobre una infraestructura Grid. Posteriormente, el cliente realiza la transferencia de los datos necesarios para la ejecución de las tareas a la máquina que aloja el servicio. Asimismo, proporciona la descripción de las tareas y, opcionalmente, la lista de recursos y la configuración del metaplanificador que debe ser utilizada en la sesión.

A continuación, el servicio Grid realiza la ejecución de la sesión, utilizando la funcionalidad proporcionada por GMarte. Las tareas se ejecutan en la infraestructura Grid y los resultados generados son almacenados en la máquina del servicio. Al mismo tiempo, el usuario es notificado de los cambios de estado de las tareas. Una vez finalizada la sesión, el cliente puede recuperar los resultados de salida, a su máquina, para comenzar el post-proceso de los mismos. Nótese que también es posible obtener los resultados de salida de aquellas tareas individuales que ya hubieran finalizado.

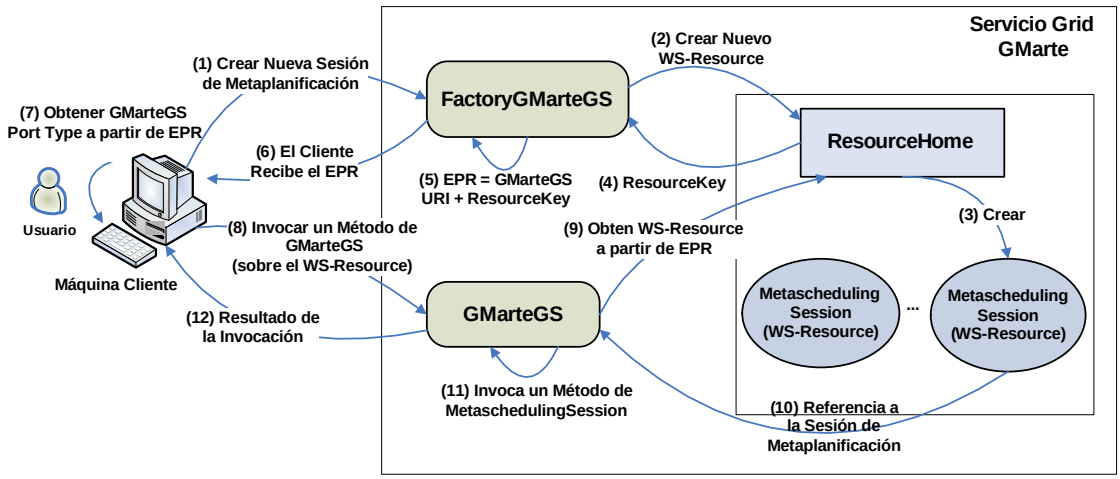


Figura 9-5: Arquitectura del servicio Grid de metaplanificación así como el flujo de información en la interacción cliente-servicio.

9.3. Arquitectura e Implementación

El servicio GMarteGS utiliza las especificaciones WSRF, implementadas en GT4, para proporcionar un servicio Grid genérico para la metaplanificación de tareas. El desarrollo de este servicio consta actualmente de unas 60 clases Java que aglutinan casi 10000 lineas de código.

La Figura 9-5 muestra la arquitectura principal del servicio Grid desarrollado. Está compuesto a su vez de dos componentes, el servicio factoría *FactoryGMarteGS* y el servicio instancia *GMarteGS*, siguiendo las recomendaciones de [161]. Por una parte, el servicio factoría se encarga de crear nuevos WS-Resource, es decir, nuevas sesiones de metaplanificación, o lo que es lo mismo, un nuevo objeto MetaschedulingSession. Por otra parte, el servicio instancia representa el núcleo del servicio Grid desarrollado, que implementa las operaciones que afectan a una sesión de metaplanificación concreta.

La separación de los servicios de factoría e instancia permite introducir nuevos escenarios. Por ejemplo se podría disponer de un conjunto de servicios Grid de metaplanificación, desplegados en diferentes máquinas, y con acceso a diferentes infraestructuras Grid. El servicio factoría sería el encargado de decidir el metaplanificador más adecuado para satisfacer cierta petición de ejecución, por ejemplo en base a la capacidad de cómputo proporcionada por cada despliegue Grid o cualquier otro parámetro. Esta estructura daría lugar a un proceso de metaplanificación de metaplanificadores que, aún siendo una estrategia interesante, queda fuera de los objetivos marcados por esta tesis doctoral.

La interacción entre la máquina cliente y el servicio Grid de metaplanificación, en el nivel más bajo, también se muestra en la Figura 9-5. En primer lugar, el cliente debe conocer la URI (Uniform Resource Identifier) del servicio FactoryGMarteGS, que representa una dirección única que lo identifica, por ejemplo, *http://pilos.itaca.upv.es:8080/wsrf/services/FactoryGMarteGS*.

Posteriormente, se crea una nueva sesión de metaplanificación (paso 1). Esto provoca (paso 2)

```
<ns1:GMarteGSResourceReference xsi:type="ns2:EndpointReferenceType"
  xmlns:ns1="http://www.grycap.upv.es/GMarte/GMarteGS"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/03/addressing">
 <ns2:Address xsi:type="ns2:AttributedURI">http://pilos.itaca.upv.es:8080/wsrf/services/GMarteGS</ns2:Address>
 <ns2:ReferenceProperties xsi:type="ns2:ReferencePropertiesType">
  <ns1:GMarteGSResourceKey>76922650-8c30-11db-9fa2-b46f07141c39</ns1:GMarteGSResourceKey>
 </ns2:ReferenceProperties>
 <ns2:ReferenceParameters xsi:type="ns2:ReferenceParametersType"/>
</ns1:GMarteGSResourceReference>
```

Figura 9-6: Ejemplo de Endpoint Reference (EPR).

que el componente *ResourceHome* cree un nuevo WS-Resource (paso 3), que contiene la sesión de metaplanificación (objeto MetaschedulingSession), devolviendo un identificador único del nuevo WS-Resource creado (paso 4). Usando la clave asociada a este recurso más la URI del servicio GMarteGS se crea un EPR (EndPoint Reference) que es devuelto al cliente (pasos 5 y 6). En este contexto, un EPR identifica de forma única a la sesión de metaplanificación (WS-Resource) y al servicio GMarteGS. La Figura 9-6 muestra un ejemplo de EPR, que consta básicamente de una referencia al servicio y a la clave que identifica al WS-Resource.

Para poder invocar un método del servicio GMarteGS se debe obtener el PortType a partir del EPR (paso 7). El cliente, usando el PortType, invoca alguno de los métodos publicados por el servicio GMarteGS (paso 8). Posteriormente, el servicio es capaz de localizar automáticamente, usando el componente ResourceHome, el WS-Resource sobre el que se trabajará (paso 9). De esta manera se obtiene una referencia a la sesión de metaplanificación (paso 10), y la operación es ejecutada (paso 11), afectando únicamente a dicha sesión de metaplanificación. Finalmente, el resultado obtenido por la invocación del método se devuelve al cliente (paso 12).

9.3.1. Especificaciones WSRF

El servicio Grid desarrollado utiliza diversas especificaciones de WSRF, dado que la utilización de estándares facilita la creación de herramientas interoperables. A continuación se describe la utilización de cada una de las especificaciones:

En primer lugar, la especificación WS-Resource [189] describe la relación entre un servicio Web y un recurso, así como los mecanismos de acceso a los recursos a través de interfaces basadas en servicios Web. En GMarteGS, cada sesión de metaplanificación corresponde a un WS-Resource y, por lo tanto, permite aislar sesiones de metaplanificación de diferentes usuarios. El hecho de que cada WS-Resource tenga un identificador único permite que, dado un EPR, sea posible determinar de manera automática la sesión sobre la que se debe ejecutar una cierta operación.

En segundo lugar, la especificación WS-ResourceProperties [192] estandariza el mecanismo para definir las propiedades de un recurso, que representan su estado, y como éstas pueden ser declaradas como parte de la interfaz del servicio. En GMarteGS, las propiedades de cada recurso incluyen un

documento XML con el estado de la sesión de metaplanificación, capturando, de esta manera, el estado del WS-Resource. Este estado puede ser accedido por medio de herramientas estándar.

En tercer lugar, la especificación WS-ResourceLifeTime [193] define mecanismos para destruir un WS-Resource. Concretamente, determina dos estrategias diferentes: destrucción inmediata y destrucción basada en tiempo. GMarteGS permite la destrucción de sesiones de metaplanificación en cualquier momento, deteniendo la ejecución de las tareas pendientes y liberando todos los recursos asociados a la misma. Además, implementa una política de liberación automática de sesiones, de manera que una sesión de metaplanificación finalizada, que no haya sido liberada en un determinado periodo de tiempo (por defecto 15 días), es automáticamente destruida. Esta aproximación permite utilizar el servicio Grid como un repositorio temporal de datos para albergar los resultados de salida de las ejecuciones.

Por ultimo, también se han utilizado especificaciones de WS-Notifications, compuesto por WS-Topics [194], WS-BaseNotification [195] y WS-BrokeredNotification [196]. Aunque no forman parte de WSRF, están disponibles en GT4. Estas especificaciones permiten que una parte de un sistema sea notificada de la ocurrencia de un evento en otro punto de él. Para ello, se plantea un sistema de publicación y subscripción, basado en *temas*, donde un cliente se subscribe a un tema para recibir las notificaciones asociadas. Esta característica ha sido incorporada al servicio Grid, de manera que los cambios en el estado de la sesión de metaplanificación, son notificados al cliente mediante un documento XML que resume el estado de la sesión de metaplanificación. Dado que las fases iniciales y finales del proceso de metaplanificación provocan numerosos cambios de estado, se han implementado mecanismos básicos de congestión para agregar múltiples notificaciones que ocurren en el mismo intervalo de tiempo, con el objetivo de reducir la sobrecarga en la comunicación entre el cliente y el servicio.

Sin la utilización del sistema de notificaciones, los múltiples usuarios del servicio Grid deben consultar de manera periódica el estado de la sesión de metaplanificación para comprobar si las ejecuciones han finalizado. Esto supone una elevada carga en el servicio ya que cada petición involucra la creación de un documento XML que resume el estado de cada una de las tareas de la sesión, junto con su correspondiente envío al cliente. Con la aproximación basada en notificaciones es posible reducir esta carga, limitando la comunicación de cambios de estado entre el cliente y el servicio al momento en el que acontecen, lo que reduce en gran medida el número de comunicaciones realizadas.

Como contrapartida, la utilización de un esquema basado en notificaciones implica la existencia de una instalación de GT4, al menos el Java WS Core, en la máquina cliente para recibir la información de los cambios de estado. Esto supone un problema en la creación de clientes ligeros para la interacción con el servicio, principalmente debido a la complejidad de la instalación de GT4. Además, la configuración del cortafuegos del cliente debe permitir la recepción de las notificaciones.

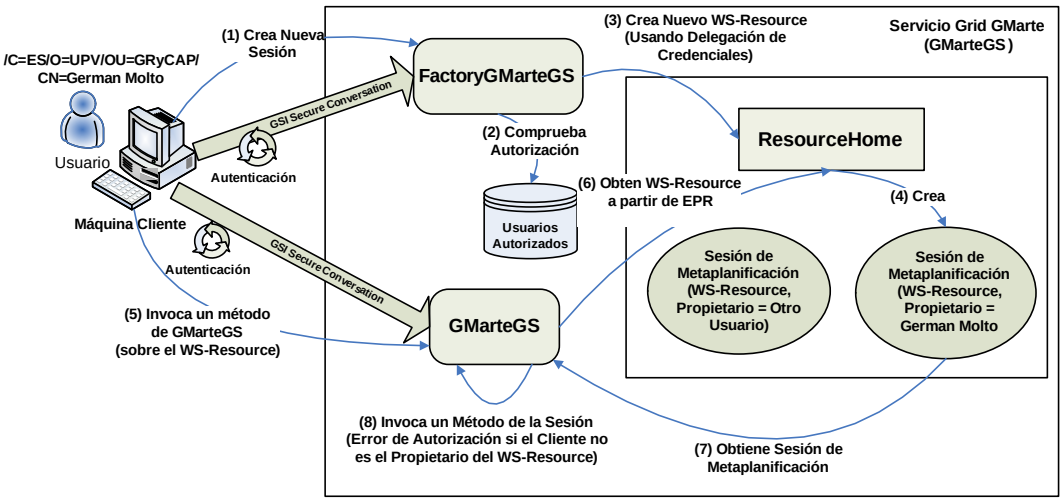


Figura 9-7: Diagrama de la interacción entre cliente y servicio con los mecanismos de seguridad empleados.

9.4. Infraestructura y Esquema de Seguridad Empleado

La seguridad en un entorno multiusuario, accesible desde Internet, como el ofrecido por GMarteGS, es crucial debido al elevado número de amenazas posibles. Por lo tanto, resulta imprescindible la implementación de mecanismos que garanticen las principales características de una comunicación segura [197]: privacidad, integridad, autenticación y autorización. Para dar solución a estos problemas, se han utilizado mecanismos estándar basados en GSI y proporcionados por GT4.

La Figura 9-7 muestra un diagrama de interacción entre el cliente y el servicio Grid detallando los aspectos de seguridad involucrados. El cliente dispone de un certificado de usuario firmado por una CA en la que confía el servicio Grid. De no ser así, no es posible establecer ningún tipo de comunicación entre el cliente y el servicio, protegiendo de esta manera la integridad del servicio.

Para crear una nueva sesión de metaplanificación, el usuario contacta con el servicio FactoryGMarteGS, que ha sido configurado para requerir la utilización de GSI Secure Conversation [157] para entablar el diálogo. Este mecanismo de seguridad, obliga a realizar automáticamente el proceso de autenticación mutua, por el que ambas partes se aseguran de estar dialogando con el receptor apropiado. Además, el canal de conexión entre cliente y servicio se configura para incluir privacidad e integridad de los datos. Adicionalmente, el servicio FactoryGMarteGS dispone de acceso a una base de datos de usuarios registrados, de manera que un usuario que no haya sido dado de alta previamente en el servicio, no podrá crear una sesión de metaplanificación.

Asimismo, el cliente utiliza un mecanismo de delegación de credenciales que permite al servicio crear un nuevo WS-Resource (sesión de metaplanificación) asignándole como propietario las credenciales del usuario. Esta característica es de importancia fundamental, puesto que los posteriores accesos que se realicen sobre un WS-Resource únicamente podrán ser realizados por el usuario crea-

dor del mismo. Hay que destacar que esta aproximación de seguridad es más completa y eficaz que un simple control de acceso a usuarios autorizados a la entrada del servicio, ya que las políticas de acceso a la información (WS-Resources) residen en el propio recurso y se comprueban al tratar de acceder al mismo.

Se puede observar que los mecanismos de seguridad implementados permiten conseguir los siguientes objetivos:

- Todas las comunicaciones realizadas entre el cliente y el servicio son privadas, y no pueden ser alteradas por un tercer agente sin que el receptor se percate.
- Un usuario que no ha sido dado de alta en el sistema no puede crear una sesión de metaplanificación en el servicio.
- Solo el creador de una sesión puede realizar una operación sobre la misma. Por lo tanto, otro usuario dado de alta en el sistema no tiene acceso a dicha sesión.

Nótese que, adicionalmente, la máquina del servicio debe disponer de las credenciales de las CAs que firmaron los certificados de los recursos que se pretende utilizar durante el proceso de metaplanificación.

La implementación de mecanismos de seguridad estándar basados en GSI permite el despliegue de servicios Grid seguros, aptos para su utilización en un entorno multiusuario accesible a través de Internet, donde pueden estar expuestos a numerosos problemas de seguridad.

9.5. Cliente del Servicio Grid de Metaplanificación

El servicio Grid GMarteGS utiliza protocolos estándares como SOAP y XML para el intercambio de datos, además de WSDL para la especificación de la interfaz. Por lo tanto, cualquier cliente que utilice dichos protocolos puede interactuar de manera programática con el servicio desarrollado, sin importar el lenguaje de programación empleado.

Adicionalmente, se ha desarrollado una librería cliente que proporciona un API en Java para facilitar la interacción con GMarteGS. En primer lugar, esta librería implementa rutinas para la gestión de datos entre el cliente y el servicio, aprovechando la implementación tolerante a fallos basada en GridFTP disponible en GMarte. De esta manera, se proporcionan métodos para la transferencia automática de la información de las tareas a partir de su descripción en XML. Básicamente, la especificación XML de las tareas es procesada para detectar los ficheros necesarios para la ejecución de los trabajos del usuario. Esto permite ocultar la complejidad de la transferencia de datos al usuario, que únicamente debe proporcionar la definición de las tareas. En segundo lugar, la librería cliente permite la interacción con el servicio GMarteGS mediante métodos de alto nivel, en Java, que ocultan la complejidad de la utilización de los protocolos subyacentes.

El desarrollo de esta librería cliente de alto nivel, permite que otras aplicaciones puedan acceder a la funcionalidad del servicio para delegar la ejecución de tareas sobre un Grid computacional. Esto permite una sencilla integración entre las aplicaciones de los usuarios y GMarteGS, de manera programática mediante la utilización del API desarrollado.

Alternativamente, se ha desarrollado un cliente gráficos. El objetivo de esta herramienta es dar soporte a aquellos usuarios que desean utilizar fácilmente una infraestructura Grid para la ejecución de su aplicación. Este tipo de usuarios no requiere utilizar un API sino que precisa de herramientas gráficas, fáciles de utilizar, que oculten la complejidad de la interacción con el Grid para la ejecución de tareas.

Para este tipo de usuarios se ha desarrollado una herramienta final, que permite la interacción con GMarteGS, y que soporta la siguiente funcionalidad:

1. Realiza transferencias de datos tolerante a fallos entre el cliente y el servicio, tanto para los ficheros de entrada como para los resultados de salida, utilizando la funcionalidad de la librería cliente.

2. Permite al usuario especificar la información XML sobre las tareas y, opcionalmente, sobre los recursos computacionales y la configuración del proceso de metaplanificación, requeridos para crear una nueva sesión de metaplanificación.

3. Muestra el estado actual del proceso de metaplanificación, a través de una tabla que indica el estado de cada una de las tareas.

4. Muestra el estado del conjunto de recursos, mediante una tabla que indica el estado de cada uno de los recursos involucrados.

Con el objetivo de crear una herramienta fácilmente accesible por parte de usuarios no expertos en computación, se ha optado por la utilización de la tecnología Java Web Start [198]. Esto ha permitido la creación de un componente desplegable en Internet que únicamente precisa de un navegador web, con soporte para Java, para ser accedido.

9.5.1. Java Web Start: Desarrollo de Componentes Ubicuos

La tecnología Java Web Start (JWS), desarrollada a través de Java Community Process, implementada por Sun Microsystems e incluida en el Java Runtime Environment, permite desplegar un aplicación Java completa accesible a través de la red.

La principal ventaja de esta tecnología es que el usuario únicamente precisa un navegador Web con soporte para Java para acceder a la aplicación, sin realizar ninguna configuración o instalación especial de la aplicación en la máquina cliente. Además, no se necesitan privilegios de administración

en la máquina cliente para poder ejecutar la aplicación de la red. Esta característica es muy importante ya que permite a los clientes acceder a la aplicación desde máquinas públicas, con cuentas de usuario limitadas.

La utilización de JWS para el despliegue de una aplicación involucra, principalmente, los siguientes pasos:

1. El desarrollador publica la aplicación, así como todas las librerías de las que depende, en un servidor Web. Este proceso involucra un conjunto de archivos JAR (Java ARchive) y un fichero JNLP (Java Network Launch Protocol) que describe y referencia los ficheros de la aplicación.

2. El cliente, con un navegador Web, accede al fichero JNLP. Esto provoca la activación automática, en la máquina cliente, del soporte JWS para descargar todos los ficheros necesarios, especificados en el JNLP.

3. Una vez que los ficheros están disponibles en la máquina cliente, la aplicación comienza a ejecutarse fuera del ámbito del navegador. En este punto, se considera que la aplicación ha sido satisfactoriamente desplegada en la máquina del cliente.

JWS implementa un sistema de cache que evita la descarga innecesaria de ficheros que ya han sido previamente obtenidos. Esta estrategia permite, por un lado, reducir el consumo de red en el cliente, con la consiguiente reducción del tiempo de acceso a la aplicación. Además, el control de versiones asegura que el cliente siempre utiliza la última versión de la aplicación, que será automáticamente descargada durante el acceso. Esta funcionalidad permite al administrador del servicio mantener un control total sobre la herramienta cliente, impidiendo que se puedan utilizar versiones antiguas del cliente que puedan resultar incompatibles con el servicio.

En importante destacar que, aunque la aplicación se ejecute fuera del navegador, no significa que pueda tener acceso total a la máquina del usuario. En realidad, la aplicación se ejecuta dentro de la caja de arena (*sandbox*) proporcionada por la máquina virtual Java, estando así restringida de acuerdo a sus políticas de seguridad.

9.5.2. Utilización del Cliente Gráfico

Antes de poder hacer uso del cliente gráfico desarrollado, es imprescindible realizar una fase de configuración de la seguridad basada en GSI para que el cliente pueda interactuar de manera segura con el servicio y, posteriormente, pueda ser autentificado y autorizado en los recursos computacionales del Grid.

Para ello el usuario debe tener disponible en la máquina cliente tanto su certificado de usuario como su clave privada. Adicionalmente, se precisan las credenciales de la CA que firmó el certificado de la máquina que aloja el servicio Grid. Hay que destacar que la configuración de seguridad en

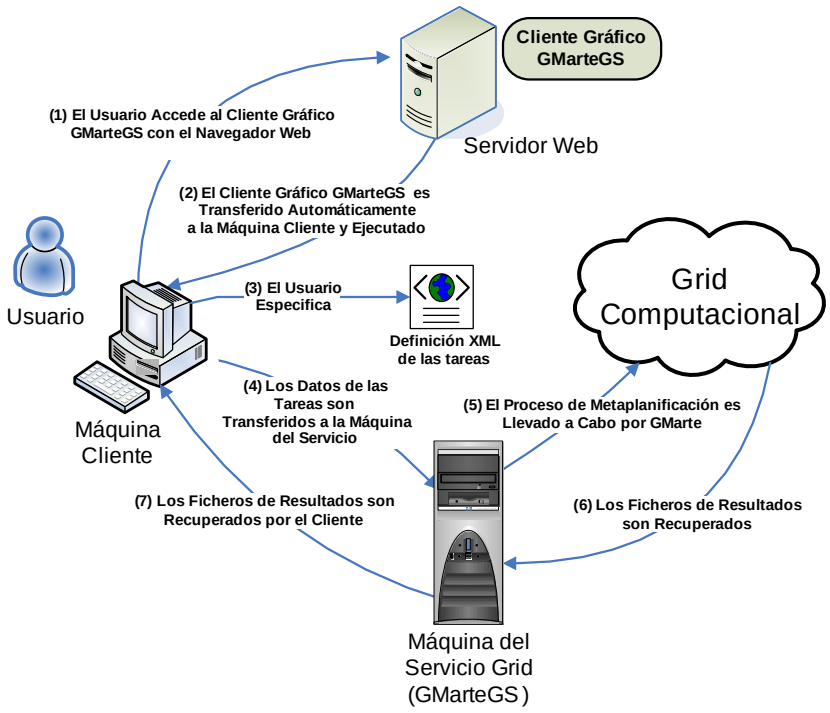


Figura 9-8: Diagrama de los principales pasos involucrados en el acceso a la funcionalidad de metaplanificación de GMarte empleando una aproximación orientada a servicio.

la máquina cliente es un proceso que se ha de llevar a cabo una sola vez, salvo que el certificado de usuario haya caducado. Para facilitar este procedimiento, se ha desplegado mediante JWS la herramienta de configuración del Java Commodity Grid, que ofrece una herramienta paso a paso para configurar la seguridad basada en GSI de la máquina cliente.

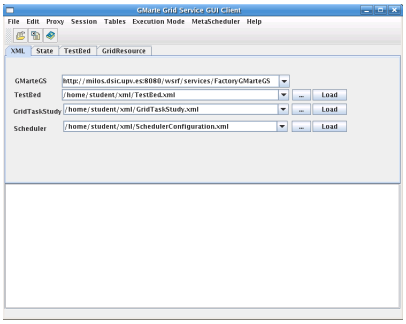
Tras finalizar esta configuración, el acceso al Cliente Gráfico GMarteGS implica los pasos que se pueden observar en la Figura 9-8. En primer lugar, el usuario accede al servidor Web que permite lanzar el cliente gráfico a través del protocolo JNLP, que automáticamente activa el soporte JWS en el cliente para descargar el componente solicitado. Esto implica descargar el cliente gráfico así como todo un conjunto de librerías (ficheros JAR) que, a grandes rasgos, cubren las siguientes áreas:

1. Librerías de GMarte empleadas para las transferencias de datos confiables, acceso a los documentos XML, etc.

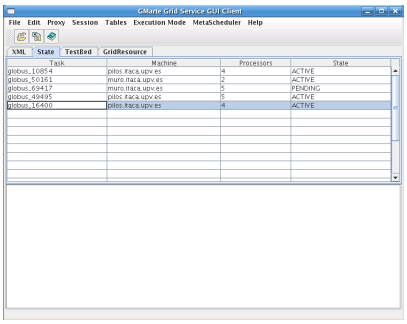
2. Librerías criptográficas requeridas para las comunicaciones GSI.

3. Componentes de traceo (logging) que permiten analizar el estado del proceso de ejecución de los componentes de GMarte.

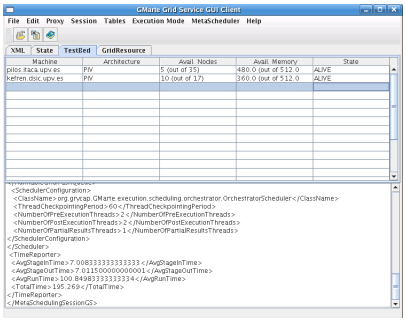
4. Librerías cliente de GT4 que permiten acceder a servicios Grid basados en WSRF, lo que incluye componentes como módulos Axis para acceder a servicios Web y definición WSDL.



(a) Pestaña Principal



(b) Estado de las Tareas



(c) Estado de las Máquinas

Figura 9-9: Aspecto del cliente gráfico desarrollado para interactuar con el servicio Grid de metaplanificación.

Esta información requiere un total de 8 MBytes que deben ser descargados en la máquina cliente. Considerando el sistema de cache que implementa JWS, se puede afirmar que se proporciona un acceso ligero al cliente del servicio de metaplanificación.

Antes de que la aplicación comience a ejecutarse en la máquina cliente se notifica al usuario que ésta solicita un acceso sin restricciones a su máquina (para leer certificados, realizar transferencias por GridFTP, crear directorios, etc.), mostrando la información del certificado con el que se firmó la aplicación. Una vez aceptada la petición, el componente gráfico se ejecuta automáticamente, permitiendo al usuario la interacción con el servicio Grid para realizar la ejecución de las aplicaciones.

La Figura 9-9 muestra el aspecto de esta herramienta, formada por tres pestañas principales. La primera de ellas permite especificar, como mínimo, el fichero XML con la descripción de las tareas. La segunda pestaña permite visualizar el estado de las tareas a lo largo del proceso de metaplanificación, actualizándose de manera automática ante la recepción de notificaciones de cambio de estado. La tercera pestaña muestra el estado de los recursos computacionales. Actualmente, esta tabla se actualiza de manera periódica y ante la petición del usuario.

La consecución de una herramienta gráfica para la interacción con una arquitectura orientada a servicio, que permite la ejecución transparente de aplicaciones sobre infraestructuras Grid, supone la culminación del proceso de abstracción de la ejecución de tareas en Grid. La utilización de una herramienta de estas características permite aprovechar la potencia computacional de un Grid de

manera sencilla por los usuarios.

9.6. Tolerancia a Fallos

Adoptar una arquitectura orientada a servicio, donde el cliente y el servidor residen en máquinas diferentes, da lugar a nuevas problemáticas que deben ser convenientemente gestionadas.

Por una parte, los fallos producidos en el cliente no deben afectar a las ejecuciones en los recursos computacionales remotos. La estrategia implementada en GMarteGS, para tolerar los fallos en el cliente, se basa en el conocimiento de que un EPR identifica unívocamente tanto al servicio como al WS-Resource que representa la sesión de metaplanificación.

Por ello, cuando se crea una nueva sesión de metaplanificación, el EPR se almacena automáticamente en un directorio por defecto en la máquina cliente. Adicionalmente, el usuario puede guardar la sesión (almacenar el EPR) en cualquier otro fichero. En realidad, este identificador es la única información necesaria para que un nuevo cliente pueda acceder a la información de la sesión correspondiente, en el servicio Grid. Por lo tanto, un fallo en el cliente puede ser solventado mediante la ejecución de un nuevo cliente, que utilice el EPR de la sesión. Esta funcionalidad también permite desacoplar el cliente y el servicio de manera que el usuario puede crear una sesión de metaplanificación, apagar su máquina local y, posteriormente, conectarse de nuevo al servicio para consultar el estado de sus ejecuciones.

Los fallos ocurridos durante el proceso de metaplanificación, tanto en la ejecución de las tareas como en los recursos computacionales, son gestionados por las políticas de tolerancia a fallos implementadas en GMarte (descritas en la sección 8.3.2). De esta manera, el servicio Grid queda aislado de este tipo de fallos, teniendo constancia, únicamente, de los cambios de estado de las tareas durante el proceso de metaplanificación.

Por otra parte, resulta apropiado gestionar los problemas ocurridos en el propio servicio. Por defecto, los WS-Resources permanecen siempre en memoria dentro del servicio Grid. Es por esto que, ante un fallo en el contenedor de aplicaciones de GT4 o en la propia máquina del servicio, los recursos activos en memoria desaparecen y, con ellos, las sesiones de metaplanificación asociadas. En este sentido se ha implementado el concepto de persistencia de recursos para poder almacenar los WS-Resources (la sesión de metaplanificación) en disco.

Para ello, cuando se inicializa una sesión, se utiliza el mecanismo de serialización, ofrecido por Java, para guardar una copia del estado de la sesión de metaplanificación en el disco de la máquina que alberga el servicio. En caso de producirse algún fallo, es posible reiniciar el contenedor de aplicaciones o la máquina del servicio de manera que cualquier operación ejecutada sobre una sesión provoca una carga automática de la misma para poder relanzar la sesión de asignación de tareas, únicamente para aquellas tareas que no hubieran finalizado previamente. Utilizando los mecanismos

de persistencia implementados en GMarte, es posible realizar una reconexión a las tareas activas, para continuar su monitorización.

9.7. Trabajos Relacionados

En la literatura es posible encontrar diversas herramientas Grid orientadas a servicio. OGCE (Open Grid Computing Environments) [199] está dedicado a la construcción de unos componentes denominados *portlets*, que permiten el desarrollo de portales accesibles vía Web para la interacción con recursos Grid. Esta aproximación permite que los usuarios puedan fácilmente interactuar con recursos Grid a través de una navegador, pero dificulta la integración de una aplicación con un portal Grid de esas características.

En el marco del proyecto eNANOS se ha implementado un metaplanificador orientado a servicio denominado eNANOS Resource Broker [200], que pretende la creación de una arquitectura coordinada a 3 capas para la ejecución de aplicaciones paralelas, desde el nivel de Grid hasta el nivel de procesador. El sistema ha sido desarrollado empleando GT3 y admite compatibilidad con GT2. Aunque se argumenta que el sistema es fácilmente extensible para soportar nuevas versiones de GT, no parece existir todavía una versión basada en GT4.

Hasta el momento, existen pocas referencias relacionadas con la utilización de WSRF para ofrecer servicios Grid de metaplanificación. En [201] se describe el diseño de un metaplanificador basado en WSRF, junto con una aplicación cliente, que permite la asignación de tareas individuales a recursos. El servicio Grid recibe peticiones de ejecución de tareas que ocasionan un proceso de descubrimiento de recursos para cada ejecución. Esta aproximación puede resultar más costosa para las aplicaciones multiparamétricas, en las que resulta preferible realizar el descubrimiento al comienzo de la sesión de metaplanificación. Además, en el trabajo no se detallan los mecanismos de seguridad empleados.

En esta línea, CSF4 (Community Scheduler Framework) [202] es un metaplanificador orientado a servicios Grid que también utiliza la especificación WSRF para su implementación. Mientras GMarteGS permite la gestión de sesiones de metaplanificación que pueden involucrar conjuntos de recursos diferentes, CSF4 recibe trabajos individuales especificados en lenguaje RSL en los que se especifica el recurso en el que realizar la ejecución. Sin embargo, CSF4 no coordina la transferencia de datos dependientes de la aplicación. Además, GMarteGS utiliza mecanismos de seguridad para la protección entre diferentes sesiones de metaplanificación, mientras que no existe documentación al respecto en CSF4.

Recientemente, la versión 3.0 de GridBus Service Broker [203, 204] ha incluido una interfaz WSRF de servicio Grid para su metaplanificador implementado con GT4. Sin embargo, al igual que ocurre con el resto de trabajos, no existe información de la infraestructura de seguridad utilizada para la protección de trabajos entre diferentes usuarios.

9.8. Conclusiones

En este capítulo se ha descrito la creación de un servicio Grid de metaplanificación genérico, con soporte para múltiples usuarios, y accesible a través de Internet, para la utilización transparente de infraestructuras Grid. El servicio Grid utiliza las especificaciones WSRF para la construcción de servicios Grid interoperables. Además, utiliza una infraestructura de seguridad basada en GSI para la protección y privacidad entre diferentes usuarios.

Además, se ha descrito la creación de un cliente gráfico para facilitar la labor de interacción con el servicio de metaplanificación. La herramienta gráfica permite al usuario especificar los ficheros XML necesarios para la configuración de la sesión de metaplanificación y observar la evolución de las tareas durante la ejecución de la sesión.

Para proporcionar un acceso ubicuo al cliente gráfico, se ha desplegado la aplicación utilizando la tecnología Java Web Start. De esta manera, el cliente únicamente precisa un navegador Web con soporte para Java, y una mínima configuración de seguridad para interactuar con una infraestructura Grid computacional.

En este sentido, la creación de componentes genéricos para el Grid, accesibles por medio de interfaces estándar, supone una importante aportación al ecosistema de herramientas Grid existentes. La interoperabilidad con otras herramientas permite incorporar su funcionalidad como parte de otras aplicaciones. Además, es importante destacar que el sistema desarrollado es suficientemente general como para permitir su aplicación en otros campos de la ciencia y de la ingeniería en la que sea necesario la utilización de Computación en Grid para la ejecución de tareas.

Capítulo 10

Aplicación de las Tecnologías Grid a la Simulación Cardiaca

"I do not fear computers. I fear the lack of them". Isaac Asimov, Writer and Professor of Biochemistry.

En este capítulo se describe la utilización del sistema GMarte para la ejecución de casos de estudio cardiacos sobre infraestructuras Grid. En primer lugar, se aborda el problema de la portabilidad de las aplicaciones complejas, detallando algunas soluciones para permitir su ejecución en un Grid. A continuación, se describe la ejecución de casos de estudio cardiacos en un Grid local, en un Grid regional y en un Grid global, realizando un análisis de los resultados obtenidos. Finalmente, se detalla la utilización del servicio Grid GMarteGS como pasarela computacional para la ejecución de casos de estudio cardiacos, en infraestructuras Grid combinadas, por parte de ingenieros biomédicos.

10.1. Adaptación del Simulador a Entornos Grid

Una infraestructura Grid está formada por recursos computacionales heterogéneos, de diferentes arquitecturas (Intel, AMD, Sparc, etc.) y diferentes sistemas operativos (Fedora, Debian, Solaris, etc.).

En esta situación, la ejecución eficiente de una aplicación en una máquina remota requiere la migración del código fuente a la máquina destino, para generar un ejecutable apropiado. Para ello, es posible utilizar los mecanismos ofrecidos por GT para la transferencia de ficheros (GridFTP) y la ejecución de tareas (GRAM), realizando el proceso de compilación y enlace del ejecutable antes de su ejecución.

Sin embargo, las aplicaciones que dependen de otras librerías científicas no pueden beneficiarse de esta estrategia, ya que se debería migrar a la máquina remota todas esas librerías. Dado el acceso

controlado a los recursos de un Grid, esta aproximación resulta poco viable.

Este es el caso del sistema de simulación cardiaca CAMAEC, que depende de todo un conjunto de librerías (PETSc, BLAS, LAPACK, MUMPS, etc.) cuya existencia en la máquina destino no puede ser asumida. Por lo tanto, aquellas aplicaciones que dependen de librerías no existentes en la máquina destino requieren mecanismos adicionales para poder ejecutarlas.

En realidad, la utilización de las tecnologías Grid no debería implicar diseñar nuevamente una aplicación existente. Sería un esfuerzo baldío adaptar una aplicación para su ejecución en un Grid, cuando, actualmente, estas tecnologías están en constante evolución. Por lo tanto, resulta importante huir de estrategias que requieran la modificación del código fuente de una aplicación para permitir su ejecución en Grid, salvo en casos donde la mejora de productividad obtenida compense la inversión en el desarrollo de la nueva aplicación.

En este sentido, el simulador CAMAEC no ha precisado modificación para ser ejecutado sobre una plataforma Grid. Sin embargo, se han realizado una serie de adaptaciones para dotar de cierta portabilidad a la aplicación.

10.1.1. Simulador Autocontenido

Conseguir la ejecución del simulador en una amplia variedad de máquinas requiere generar un ejecutable autocontenido, sin dependencias de librerías externas. Para ello, es importante conocer el mecanismo de librerías empleado en el sistema Linux, la plataforma más utilizada en los recursos computacionales de un Grid.

En Linux existen tres clases de librerías: estáticas, dinámicas y compartidas. Las bibliotecas estáticas son colecciones de ficheros objeto, es decir, módulos ya compilados. Estos módulos objeto, típicamente ficheros con la extensión *.o*, se agrupan dentro de una librería estática, típicamente con la extensión *.a*, empleando utilidades del sistema como *ar* y *ranlib*. Cuando el enlazador (linker) se encuentra con un fichero con la extensión ".a" realiza la búsqueda de módulos en el mismo, seleccionando y añadiendo al programa aquellos que resuelven las referencias que todavía no han sido satisfechas.

Por otra parte, las librerías dinámicas son objetos reubicables marcados con un código especial que los identifica como librerías dinámicas. El enlazador no añade al código del programa los módulos, sino que selecciona como resueltos los identificadores aportados por la librería. El enlazador reconoce una librería dinámica si su nombre termina con la extensión ".so". En el momento de la ejecución de una aplicación con este tipo de enlace, se produce el enlace dinámico entre el código ejecutable y las librerías de las que depende.

La Figura 10-1 muestra la lista de librerías dinámicas de las que depende el sistema de simulación CAMAEC, obtenida con el comando *ldd* de Linux. Se observa que la aplicación depende, entre otras, de librerías dinámicas de PETSc y de las librerías de paso de mensajes MPI. El tamaño de este fichero

```
libpetscgsolver.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscgsolver.so (0x40013000)}
libpetscgrid.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscgrid.so (0x40223000)}
libpetscmesh.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscmesh.so (0x40333000)}
libpetscts.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscts.so (0x403bd000)}
libpetscsnes.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscsnes.so (0x403f4000)}
libpetscsles.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscsles.so (0x40448000)}
libpetscdm.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscdm.so (0x407b9000)}
libpetscmat.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscmat.so (0x40851000)}
libpetscvec.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetscvec.so (0x41172000)}
libpetsc.so => /usr/local/soft/petsc/2.1.6/lib/libg/linux/libpetsc.so (0x4122f000)}
libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x41352000)
libstdc++.so.5 => /usr/lib/libstdc++.so.5 (0x41430000)
libpthread.so.0 => /lib/i686/libpthread.so.0 (0x414e2000)
libfmpi.so => /opt/scali/lib/libfmpi.so (0x41513000)
libmpi.so => /opt/scali/lib/libmpi.so (0x4151f000)
libdl.so.2 => /lib/libdl.so.2 (0x41664000)
libg2c.so.0 => /usr/lib/libg2c.so.0 (0x41667000)
libc.so.6 => /lib/i686/libc.so.6 (0x42000000)
libm.so.6 => /lib/i686/libm.so.6 (0x41685000)
libxml2.so.2 => /usr/lib/libxml2.so.2 (0x416a8000)
libgcc\_s.so.1 => /lib/libgcc\_s.so.1 (0x41750000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x41758000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
libz.so.1 => /usr/lib/libz.so.1 (0x41785000)
```

Figura 10-1: Librerías dinámicas de las que depende el sistema de simulación.

ejecutable es de 1.9 Mbytes. Como se ha comentado, una aplicación enlazada dinámicamente debe localizar en tiempo de ejecución las librerías de las que depende.

La alternativa al enlace dinámico es el enlace estático, donde el código de las librerías se incluye dentro del fichero ejecutable. La mayoría de compiladores soportan la opción de linea de comandos *-static,* que indica al compilador que trate, en la medida de lo posible, de no enlazar con las librerías dinámicas. Para ello, deben existir versiones estáticas de las librerías de las que depende la aplicación.

En el caso del simulador CAMAEC, el enlace estático genera un ejecutable autocontenido de 14 Mbytes, que puede ser reducido a 6.6 Mbytes descartando los símbolos del ejecutable mediante el comando *strip* de Linux. Este ejecutable autocontenido también incluye las librerías de MPI, con el propósito de aislar al simulador paralelo de la implementación MPI existente en la máquina remota. Para ello se ha empleado como librería de paso de mensajes una versión de MPICH [68] especialmente configurada para deshabilitar la comunicación mediante memoria compartida entre procesos del mismo nodo. Se ha comprobado que dicha comunicación puede causar problemas de mala gestión de memoria, debido a que la implementación utiliza las primitivas IPC (InterProcess Communication) de Unix System V [69].

Básicamente, el problema ocurre cuando los procesos de una ejecución paralela se comunican mediante memoria compartida y se produce un fallo en alguno de ellos. En ese caso, puede ocurrir que la página de memoria que estaban empleando no se libere. Si sucede varias veces, es posible que se agote la memoria asignada al usuario para realizar la comunicación entre sus procesos. En caso de tener acceso a la máquina, el problema se puede resolver de manera sencilla mediante la liberación manual de esas páginas de memoria. Sin embargo, en un entorno Grid, donde no se dispone de acceso directo a los recursos, resulta más apropiado prevenir el problema. Para ello se deshabilita el empleo

de memoria compartida para la comunicación entre procesos. Esto implica que, aunque dos procesos estén ejecutándose en el mismo nodo, su comunicación se llevará a cabo por medio de *sockets*.

10.1.2. Optimizaciones Dependientes de la Arquitectura

Los compiladores suelen ofrecer una serie de opciones para generar código para una arquitectura concreta. Si un ejecutable que contiene instrucciones específicas de una arquitectura es ejecutado sobre otra diferente, fallará. Por lo tanto, mantener cierto nivel de portabilidad requiere deshabilitar cualquier tipo de optimización dependiente de la arquitectura. El compilador gcc ofrece las opciones *-march* y *-mcpu* que permiten la generación de código específico para la arquitectura que se indique. Por lo tanto, esas opciones no deben ser empleadas salvo que la arquitectura de la máquina destino coincida con la de la máquina en la que se compiló la aplicación.

Además, existen versiones optimizadas, dependientes de la arquitectura, para los núcleos computacionales básicos BLAS y LAPACK, como las proporcionadas por la Intel MKL. Tal y como se ha comentado anteriormente, si la aplicación se enlaza con las librerías optimizadas para una plataforma, no será posible ejecutarla de manera portable en otras. Por último, destacar que las opciones tradicionales de compilación como *-O3* no suponen riesgo alguno ya que no introducen ninguna optimización dependiente de la plataforma. Tan solo realizan procesos de reordenación de código y optimizaciones seguras.

10.1.3. Diversificando para Diferentes Arquitecturas

En la actualidad, resulta importante aprovechar de manera eficiente la potencia computacional de las arquitecturas de 64 bits, como la ofrecida por el procesador Intel Itanium II. Este procesador ofrece compatibilidad binaria con las aplicaciones de 32 bits existentes. Esto implica, por ejemplo, que una aplicación compilada en una máquina con un procesador Xeon puede ser ejecutada en una máquina Itanium, siempre que las dependencias estén correctamente resueltas en la máquina destino. Obviamente, una ejecución correcta no implica una ejecución eficiente, por lo que es de esperar una reducción de las prestaciones, frente a las obtenidas si se hubiera compilado de manera nativa la aplicación en la máquina Itanium.

La Tabla 10.1 muestra los tiempos de ejecución, expresados en segundos, de una simulación sobre un tejido tridimensional de 30x30x30 células, durante 2 ms, con un ejecutable generado en la máquina Kefren (camaec3D_ia32) y con un ejecutable generado en la máquina Bastet (camaec3D_ia64). Los prefijos indican la arquitectura de la máquina en la que se generaron los ejecutables, donde IA32 corresponde a una arquitectura de 32 bits (Intel Xeon) e IA64 corresponde a una arquitectura de 64 bits (Intel Itanium II). La descripción de los recursos se puede encontrar en el Anexo A.

En la tabla se observa que la ejecución en modo compatibilidad en el procesador Itanium II, es decir, el ejecutable camaec3D_ia32 en la máquina Bastet, obtiene unas pobres prestaciones. El

	Kefren (Xeon)	Bastet (Itanium II)
camaec3D_ia32	55.12	288.79
camaec3D_ia64	-	40.51

Tabla 10.1: Comparativa de tiempos de ejecución, en segundos, en plataformas de 32 y 62 bits para ejecutables generados en ambas arquitecturas.

resultado indica que la simulación es 5.24 veces más lenta que ejecutando en la plataforma de 32 bits donde se generó el ejecutable. Sin embargo, la ejecución en la máquina Bastet del simulador generado en ella, resulta un 36 % más rápida comparado con la compilación y ejecución en la arquitectura de 32 bits. Obviamente, no es posible ejecutar en una plataforma de 32 bits una aplicación compilada en una máquina de 64 bits.

A la vista de los resultados obtenidos, resulta importante disponer de una versión del simulador para la arquitectura Intel Itanium II, para mejorar las prestaciones obtenidas en ella. De hecho, se podría generalizar esta aproximación para abarcar más arquitecturas, generando un ejecutable para Pentium III, otro para Pentium IV, otro más para Intel Itanium II, etc, llegando incluso a realizar combinaciones con las diferentes versiones de las librerías optimizadas. Sin embargo, se ha optado por una estrategia más factible, donde se dispone de dos versiones del simulador autocontenido, una válida para las arquitecturas Intel de 32 bits y otra válida para las arquitecturas Intel de 64 bits. De esta manera, se consiguen aprovechar las capacidades ofrecidas por los nodos Itanium de un infraestructura Grid.

10.1.4. Prestaciones del Simulador Adaptado

Una vez realizadas las adaptaciones necesarias al simulador para su ejecución en un entorno Grid, obteniendo un simulador autocontenido y relativamente portable, resulta importante cuantificar la pérdida de prestaciones que se produce frente al simulador original optimizado.

Para ello se han realizado una serie de ejecuciones en los clusters Ramses y Kefren (ver Anexo A). Se llevó a cabo una simulación de propagación de potencial de acción sobre un tejido de 60x60x60 células durante 2 ms, empleando un paso de tiempo de 10 μs, dando lugar a 200 pasos de tiempo. La simulación precisó un total de 170 Mbytes de memoria RAM y, por lo tanto, ninguna de las ejecuciones superó la memoria RAM de la máquina. Además, se realizaron 2 ejecuciones de cada tipo, obteniendo un promedio del tiempo de ejecución.

Para la comparativa de resultados se han utilizado dos versiones del simulador. En el caso del simulador denominado *SimOpt*, se emplean las librerías BLAS y LAPACK de la MKL para la arquitectura donde se ejecuta. El simulador denominado *SimGrid* utiliza las correspondientes librerías ofrecidas por el sistema operativo, sin optimizaciones dependientes de la arquitectura. Además, SimGrid contiene las adaptaciones que se han comentado en puntos anteriores, es decir, es un simu-

	Ramses		Kefren	
	SimGrid	SimOpt	SimGrid	SimOpt
Ejecución Modelo	3.17	3.24	1.29	1.27
Sistema de Ecuaciones	0.99	0.86	0.19	0.18
Tiempo Total	864.21	851.88	306.23	296.91

Tabla 10.2: Tiempo promedio de ejecución de un paso de simulación, expresado en segundos, empleando 1 procesador. Además, se muestra el tiempo global de la simulación.

	Ramses		Kefren	
	Grid	Optimizado	Grid	Optimizado
Ejecución Modelo	0.405	0.412	0.173	0.1689
Sistema de Ecuaciones	0.174	0.156	0.087	0.05
Tiempo Total	126.88	125.102	69.96	49.85

Tabla 10.3: Tiempo promedio de ejecución de un paso de simulación, expresado en segundos, empleando 8 procesadores. Además, se muestra el tiempo global de la simulación.

lador autocontenido y relativamente portable. Ambas aplicaciones han sido compiladas empleando la directiva de optimización -*O3*. Así mismo, en ambos casos se ha enlazado con la librería PETSc compilada con la opción de optimización (-*O*).

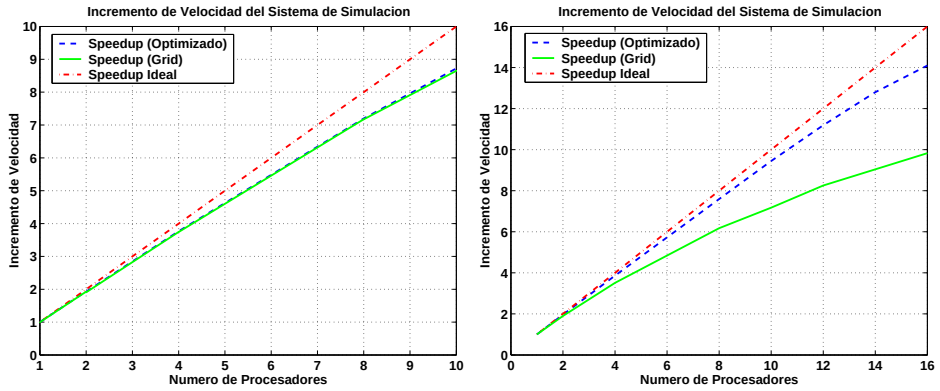
Con respecto a la comunicación entre procesos, en Ramses se utiliza la red Gigabit Ethernet. En Kefren, sólo si el ejecutable utiliza ScaMPI, la versión de MPI proporcionada por Scali, se emplea la red SCI optimizada para dicha comunicación. En otro caso, se emplea una red Fast Ethernet que también se utiliza para tareas de gestión entre los nodos.

La Tabla 10.2 muestra los tiempos de simulación, para ambos simuladores, empleando únicamente un solo procesador. De la misma manera, la Tabla 10.3 presenta los resultados obtenidos para una ejecución con 8 procesadores. En ambos casos se muestra el tiempo promedio de un paso de simulación, desglosado en las dos fases más costosas. Por un lado, la ejecución del modelo celular, que únicamente involucra cálculo y, por otro lado, la resolución del sistema de ecuaciones, que combina el cálculo y las comunicaciones entre procesadores.

A la vista de los resultados, se pueden extraer diversas conclusiones. En primer lugar, se observa que, para el cluster Ramses, existe menos variabilidad temporal entre SimGrid y SimOpt, tanto en secuencial como en paralelo. Sin embargo, para el cluster Kefren, existe una mayor variabilidad en el tiempo de la ejecución paralela, especialmente en la fase de resolución del sistema de ecuaciones.

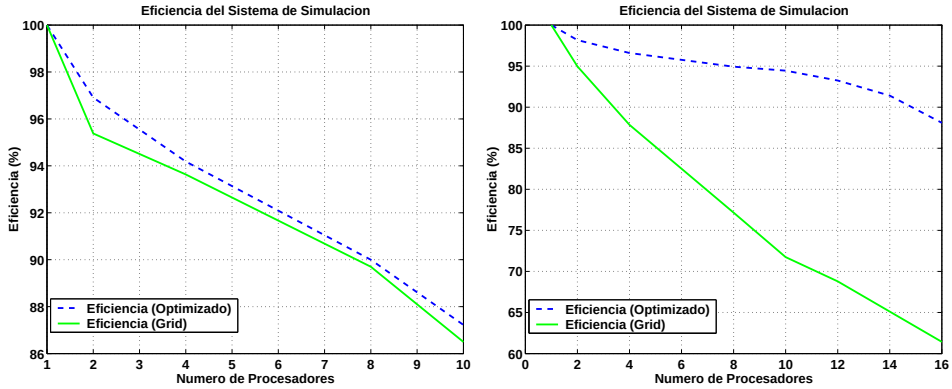
Hay que destacar que, en el cluster Kefren, el simulador adaptado a Grid utiliza la red Fast Ethernet, en lugar de la red SCI. Esta primera red se utiliza tanto para la comunicación entre procesos como para la gestión entre los diferentes nodos del cluster. De esta manera, las comunicaciones son más lentas y, por lo tanto, es de esperar una caída en las prestaciones en las fases que precisen de comunicaciones, como la de resolución del sistema de ecuaciones.

La Figura 10-2 muestra una comparativa de los incrementos de velocidad obtenidos en cada uno



(a) Ejecuciones en Ramses (b) Ejecuciones en Kefren

Figura 10-2: Comparativa del incremento de velocidad obtenido para el simulador adaptado a Grid y para el simulador optimizado.



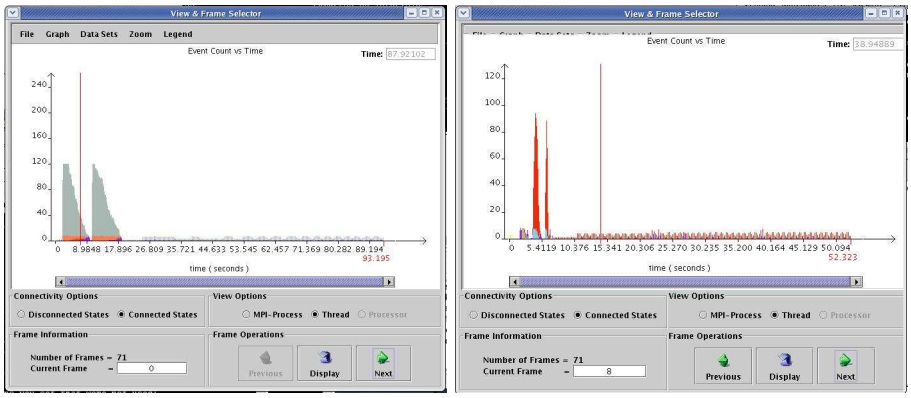
(a) Ejecuciones en Ramses (b) Ejecuciones en Kefren

Figura 10-3: Comparativa de la eficiencia obtenida para el simulador adaptado a Grid y para el simulador optimizado.

de los clusters para ambos simuladores. Análogamente, la Figura 10-3 muestra una comparativa de las eficiencias obtenidas. En las gráficas se observa que las ejecuciones llevadas a cabo en el cluster Ramses presentan una ligera diferencia entre SimOpt y SimGrid, siendo la ventaja a favor del simulador optimizado. Sin embargo, esta diferencia es mucho más acusada para las ejecuciones llevadas a cabo en Kefren. En este caso, los resultados de escalabilidad, conforme aumenta el número de procesadores, resultan peores para el simulador Grid que para el simulador optimizado.

10.1.5. Análisis Post-Mortem

Para poder investigar con mayor nivel de detalle la pérdida de prestaciones obtenida en el simulador adaptado a Grid, en el cluster Kefren, se ha optado por emplear una herramienta de análisis *post-mortem* de programas paralelos.



(a) SimGrid (b) SimOpt

Figura 10-4: Comportamiento de la aplicación, analizado con la herramienta Jumpshot-3.

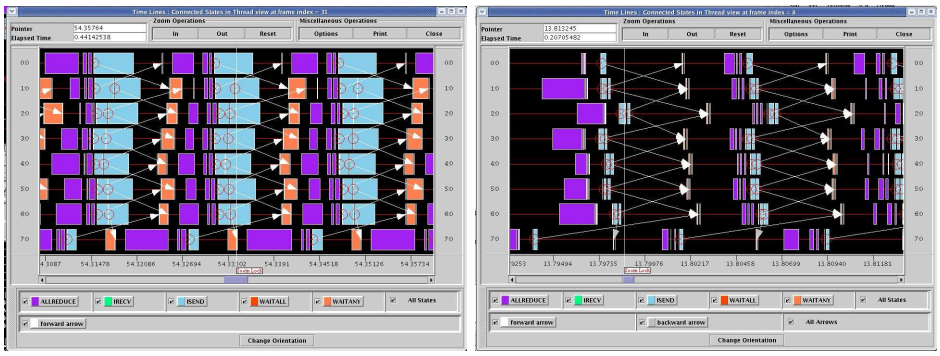
La librería MPE (Multi-Processing Environment) [205] utiliza las características de MPI para proporcionar herramientas de análisis de prestaciones, así como la depuración y representación gráfica de las comunicaciones entre procesos. Estas herramientas se denominan de análisis post-mortem porque recopilan información durante la ejecución del programa, que será procesada tras la ejecución.

En ese sentido, se han instrumentado las ejecuciones en el cluster Kefren, con 8 procesadores, para analizar y justificar la pérdida de prestaciones producida. Los datos recopilados se pueden visualizar mediante la herramienta *Jumpshot-3*, incluida dentro de la librería MPE.

La Figura 10-4 muestra la información que ofrece esta herramienta. El eje horizontal indica el tiempo de la simulación, mientras que el eje vertical contabiliza el número de eventos registrados dentro de una determinada ventana temporal. Se puede observar que la aplicación presenta un comportamiento en base a patrones periódicos durante toda su ejecución, salvo en el instante inicial. En efecto, al comienzo existe un proceso de generación de las principales estructuras de datos (tejido cardiaco, matrices del sistema de ecuaciones, etc.). Posteriormente, el simulador entra en un proceso iterativo donde cada paso de simulación realiza siempre las mismas fases. La figura muestra como la fase de creación de matrices y estructuras de datos iniciales requiere más tiempo para SimGrid, que para SimOpt.

En este sentido, analizar el comportamiento de una iteración, para ambos simuladores, permite averiguar la fase que requiere la mayor parte del tiempo. La Figura 10-5 muestra un detalle correspondiente a un paso de simulación. El eje horizontal corresponde al tiempo de simulación y el eje vertical muestra el identificador de cada proceso. Las flechas indican comunicaciones entre procesos, permitiendo visualizar el patrón de comunicaciones durante la ejecución. Los bloques indican las operaciones realizadas por cada proceso.

Los bloques de color azul claro representan la operación MPI_ISEND, un envío de datos no bloqueante entre procesadores. Se observa que, para SimGrid, estos envíos requieren más tiempo



(a) SimGrid (b) SimOpt

Figura 10-5: Detalle del comportamiento de un paso de simulación.

comparado con los respectivos en SimOpt. En efecto, tal y como se apuntaba anteriormente, esta situación es debida a la utilización de la red SCI. En el caso de SimOpt sí que se emplea esta red, dedicada exclusivamente a la comunicación entre procesadores. Dado que ofrece una latencia muy baja, permite operaciones de comunicación muy rápidas. Por el contrario, SimGrid en el cluster Kefren utiliza la red de comunicaciones por defecto, una Fast Ethernet que, además, no está dedicada en exclusiva para la comunicación entre procesos. Esto provoca operaciones de comunicación mucho más lentas, que afectan directamente a la eficiencia del algoritmo paralelo provocando una ejecución más costosa temporalmente.

Como conclusión, resulta importante destacar que las técnicas de enlace estático, combinadas con la deshabilitación de las optimizaciones dependientes de la arquitectura, permiten la obtención de aplicaciones autocontenidas que pueden ser ejecutadas en un amplio espectro de máquinas Linux, la principal plataforma en los despliegues Grid computacionales. El impacto que estas técnicas tienen sobre el tiempo de ejecución es bastante reducido. Sin embargo, el hecho de aislar la aplicación paralela de la implementación MPI de la máquina remota, impide la utilización de redes específicas, optimizadas para la comunicación entre procesos. Esto provoca una reducción de las prestaciones paralelas. No obstante, la posibilidad de acceder a un elevado número de recursos mediante las tecnologías Grid, compensa sobradamente la pérdida de prestaciones provocada por la utilización de estas técnicas.

Por otra parte, recientemente se está impulsando la idea de trabajar con máquinas virtuales aplicadas al Grid para la creación de espacios de trabajo virtuales (virtual workspaces) [206], con el objetivo de solventar la problemática de heterogeneidad de los recursos de computación. Mediante esta aproximación, la máquina virtual contiene el entorno de ejecución apropiado y puede ser desplegada sobre una máquina física. Esta reciente línea de trabajo está pendiente de ser incorporada como parte del proyecto Globus.

10.2. Ejecución de Casos de Estudio Cardiacos en Grid

En esta sección se evalúa el uso de GMarte para la ejecución de varios casos de estudio cardiacos sobre infraestructuras Grid. El primero de ellos permite encontrar la ventana vulnerable de un tejido bajo condiciones de isquemia, realizando las ejecuciones sobre un Grid local compuesto por máquinas del GRyCAP. El segundo caso de estudio investiga los efectos en la propagación eléctrica, de la isquemia de miocardio, realizando las ejecuciones sobre un Grid regional formado por recursos de dos centros de investigación nacionales. Finalmente, se aborda el mismo estudio planteado en el segundo caso, pero ejecutado en un Grid global compuesto por máquinas de diferentes centros de investigación a lo largo del mundo. En todos los casos se evalúa la ventaja de la utilización del Grid frente a otras alternativas computacionales, como el uso exclusivo de Computación de Altas Prestaciones o la ejecución secuencial sobre un PC.

El objetivo de esta sección consiste en evaluar las ventajas del sistema de Computación en Grid desarrollado, en términos de productividad. En este sentido y, con el objetivo de simplificar la exposición, únicamente se aporta un estudio detallado de las implicaciones biomédicas para el primer caso de estudio. En el resto de estudios la exposición se centra principalmente en los resultados de ejecución.

10.2.1. Ventana Vulnerable: Ejecución en Grid Local

La fibrilación ventricular es una arritmia muy grave que representa una de las principales causas de muerte súbita [207, 208]. Esta arritmia suele ser provocada por la generación de una reentrada, es decir, un frente de onda eléctrico que circula alrededor de un obstáculo anatómico o funcional [209]. En el caso de las reentradas funcionales, una parte del tejido cardiaco que se encuentre en su periodo refractario puede suponer un obstáculo para la propagación del frente de onda. De este modo, la propagación sigue una vía alternativa hasta que el lugar de bloqueo recupera su excitabilidad y el frente de onda reentra, excitando de nuevo el lugar de origen [210].

Por lo general, se suelen considerar dos factores para analizar las reentradas. Por un lado, el instante de aplicación de la estimulación prematura determina la posibilidad de generación de reentradas. Por otro lado, las condiciones electrofisiológicas del tejido son responsables del mantenimiento de la misma. De hecho, en primer lugar se debe aplicar un estímulo para conseguir una zona en periodo refractario. Sin embargo, el frente de onda únicamente puede conseguirse si un segundo estímulo (estímulo prematuro) no se aplica demasiado pronto. De esta manera, el instante de tiempo del estímulo prematuro es un factor importante en la iniciación de la reentrada.

Como se ha comentado anteriormente, las condiciones electrofisiológicas del tejido son responsables del mantenimiento de la reentrada. Por ejemplo, debido a la obstrucción de una arteria coronaria durante la fase aguda de isquemia de miocardio, los cambios electrofisiológicos, como las alteraciones

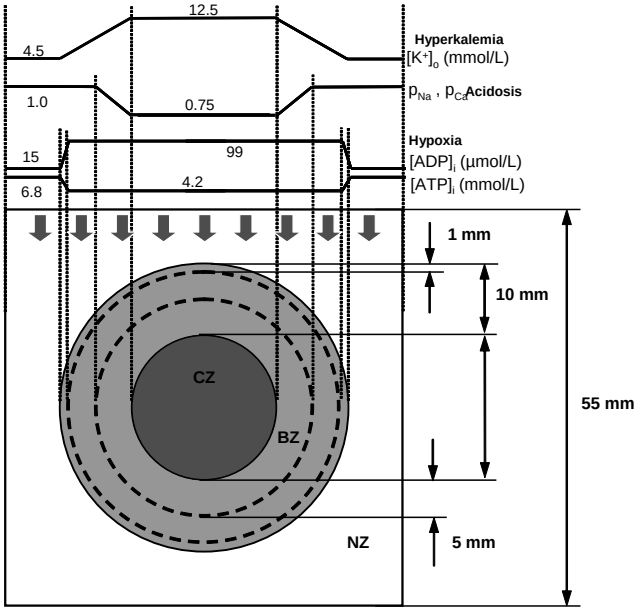


Figura 10-6: Tejido isquémico de 55x55 mm con una zona normal (NZ), zona de borde (BZ) y una zona central isquémica (CZ).

en el nivel de oxígeno celular, de la concentración de potasio extracelular, y de la acidosis [211, 212], predisponen al corazón a la ocurrencia de arritmias reentrantes [129]. En efecto, la isquemia de miocardio es una condición en la que la ausencia de oxígeno en el músculo del corazón viene acompañada de una inadecuada retirada de los resultados del metabolismo debida a la reducción de flujo sanguíneo

El estudio de este fenómeno puede llevarse a cabo de manera efectiva mediante simulaciones basadas en computador, que analicen las causas de las reentradas. Sin embargo, en la literatura científica no hay prácticamente constancia de simulaciones de reentradas en situaciones patológicas de isquemia. En este caso de estudio, se analizan los efectos de la estimulación prematura en la probabilidad de generación de reentradas sobre un tejido isquémico y, por lo tanto, la vulnerabilidad del tejido a las reentradas.

El aspecto novedoso de este estudio biomédico teórico radica en el análisis de los mecanismos que dan lugar a las reentradas, en forma de figura de ocho, durante la fase aguda de isquemia de miocardio. De hecho, las propiedades de los tejidos virtuales pueden controlarse de manera detallada a nivel electrofisiológico, algo extremadamente difícil de conseguir por medios únicamente experimentales. En este estudio computacional, además de analizar los efectos de diferentes factores, se simula un proceso realista de isquemia para encontrar las causas de iniciación de la reentrada.

Descripción del Tejido Isquémico

Para las simulaciones se ha utilizado el modelo celular Luo-Rudy Fase II [10, 213] de propagación de potencial de acción. Para incluir los efectos de la isquemia provocados por la corriente de potasio sensible a ATP, se ha introducido en el modelo la formulación de Ferrero de dicha corriente [214]. En las simulaciones, se ha considerado un tejido bidimensional [215, 216]. Por ello, se ha dividido un cuadrado de 55x55 mm en elementos de 100x100 μm tal y como se muestra en la Figura 10-6. En ella se representa un tejido anisotrópico ventricular, afectado por una zona de isquemia regional aguda. Se pueden distinguir tres zonas con diferentes condiciones electrofisiológicas: zona normal (NZ), zona de borde (BZ) y zona central isquémica (CZ).

Para simular la isquemia de miocardio, se han aumentado diversos parámetros electrofisiológicos en la zona central isquémica. En primer lugar, se ha elevado la concentración de potasio extracelular ($[K^+]_o$). En segundo lugar, la falta de oxígeno en la célula se representa mediante un descenso en la concentración intracelular de ATP ($[ATP]_i$) y un incremento en la concentración intracelular de ADP ($[ADP]_i$), que es la responsable de la activación de los canales de potasio sensibles a ATP [217, 218]. Finalmente, se ha considerado en la CZ un bloqueo parcial de canales de sodio y calcio provocados por la acidosis [219, 220]. Los valores de estos parámetros se muestran también en la Figura 10-6. Con respecto a la zona de borde (BZ), se ha considerado un gradiente espacial de variación para cada parámetro, desde su valor normal al valor isquémico, de acuerdo a estudios experimentales [28]. Además, se ha tomado un ratio de anisotropía de 3:1, mediante los correspondientes valores de resistividad longitudinal y transversal.

Protocolo de Estimulación

Para analizar la vulnerabilidad a reentradas del tejido isquémico se ha definido una ventana vulnerable. Este intervalo de tiempo incluye los instantes de estimulación prematura que dan lugar a una reentrada. Cuanto más grande sea la ventana, más vulnerable es el tejido cardiaco para desarrollar una reentrada mediante estimulación prematura.

En primer lugar, se aplicó un estímulo básico en el que la parte superior del tejido fue estimulada mediante un pulso de corriente de 90 nA en amplitud (correspondiente a 1.5 veces el estímulo umbral) y 2 ms de duración. Este estímulo se retrasó 50 ms, tal y como se muestra en la Figura 10-7, con el objetivo de permitir la estabilización de las variables del modelo celular.

En segundo lugar, se aplica un segundo y prematuro estímulo, en el instante t_2, en el mismo lugar en el tejido y con la misma amplitud. Posteriormente, se ejecutan 20 simulaciones variando el instante de estimulación prematura ($t_2 = t_i$). De esta manera, el tejido fue estimulado con diferentes intervalos de acoplamiento. Se tomaron $t_i s$ cada 5 ms con el objetivo de estimular entre el instante 200 ms y 300 ms. Nos referimos a este método como el protocolo de estimulación S_1-S_2, donde S_1 es el estímulo básico y S_2 es el prematuro.

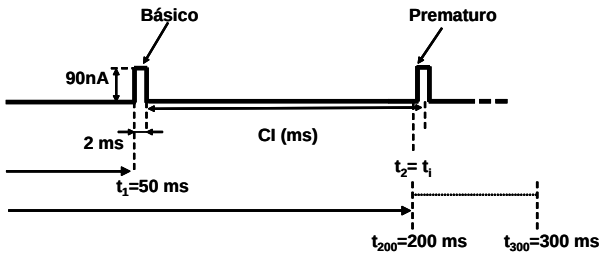


Figura 10-7: Protocolo de estimulación.

Tabla 10.4: Características principales de los recursos computacionales.

Máquina	Procesadores	Memoria
Kefren	20 dual Intel Xeon 2.0 Ghz	1 GByte
Ramses	12 dual Intel Pentium III 866 Mhz	512 MBytes
Odin	55 dual Intel Xeon 2.8 Ghz	2 GBytes

Infraestructura Grid Empleada

Para realizar la ejecución del caso de estudio, se ha empleado una infraestructura Grid compuesta por varios clusters de PCs del GRyCAP, cuyas características principales se muestran en la Tabla 10.4. Todas las máquinas llevan instalado Globus Toolkit 2.4.3 como soporte Grid y disponen de credenciales válidas, firmadas por la entidad certificadora del GRyCAP. Estos clusters no están únicamente dedicados para realizar ejecuciones vía Grid y, por lo tanto, el número inicial de nodos de computación disponibles antes de comenzar el proceso de metaplanificación era de 17 (Kefren), 2 (Ramses) y 55 (Odin).

Descripción Computacional del Caso de Estudio

Para realizar la ejecución eficiente del caso de estudio se decidió realizar una primera simulación para obtener un checkpoint en el instante 200 ms, a partir del cual continuaron las 20 simulaciones a ejecutar. Las simulaciones abarcaron un periodo de 600 ms, tiempo suficiente para detectar una reentrada en el tejido cardiaco. El paso de simulación fue de 0.024 ms, que previamente se estimó que era lo suficientemente pequeño para capturar la dinámica de las corrientes iónicas. El potencial de membrana de todas las células del tejido se guardó cada 4 ms, dando lugar a un total de 229 MBytes de resultados por simulación. Esta información permite la generación de un vídeo, como parte del postproceso, para que el experto biomédico determine si se ha producido una reentrada. Es importante destacar que las ejecuciones mostradas permitieron obtener la ventana vulnerable con una resolución de 5 ms. Por lo tanto, se precisaron realizar simulaciones adicionales para obtener la ventana vulnerable con una precisión de 1 ms.

Tabla 10.5: Resumen del proceso de asignación de tareas en la infraestructura Grid local para cada una de las máquinas.

Máquina	**Simulaciones**
Kefren	1 (8 p.), 1 (4 p.), 1 (5 p.)
Ramses	1 (4 p.)
Odin	15 (8 p.), 1 (4 p.)

La descripción del caso de estudio consideró como parámetro de variación el retraso en la aplicación del segundo estímulo. Para este caso de estudio, se configuró el metaplanificador para usar 4 hilos de ejecución para la fase de stage-in y otros 4 para la fase de stage-out. Esta política permitió planificar hasta 4 simulaciones de manera concurrente.

Además, se limitó a 8 el número máximo de procesadores empleados en cada ejecución paralela, independientemente del número de procesadores disponible en el recurso computacional seleccionado. Aunque la elección de ese umbral puede ser relativamente arbitraria, en el caso que nos ocupa corresponde al número de procesadores por encima del cual la aplicación obtuvo resultados de eficiencia por debajo del 80 %, en el caso más desfavorable, en el análisis post-mortem. Fijar un umbral de eficiencia permite escoger un número de procesadores máximo apropiado para mantener una adecuada tasa de utilización de los recursos computacionales de la infraestructura. Además, un efecto secundario de esta limitación es que representa una política amistosa que permite asignar múltiples ejecuciones a un recurso, en lugar de ocupar toda su capacidad con una única simulación. Adicionalmente, se consigue que varias simulaciones puedan ser ejecutadas concurrentemente sobre el mismo cluster de PCs.

Resultados de Ejecución

La Tabla 10.5 resume el proceso de asignación de tareas. En primer lugar, un total de 16 simulaciones fueron asignadas dinámicamente a Odin, con 8 y con 4 procesadores. En Kefren, 3 simulaciones fueron ejecutadas con diferente número de procesadores. En Ramses, solamente se realizó una ejecución, con 4 procesadores.

Se puede observar como el metaplanificador distribuyó de manera dinámica la carga entre los clusters de PCs, de acuerdo a su capacidad computacional, basada en el número de procesadores disponibles. Además, aunque Ramses inicialmente tenía solo dos procesadores libres, la posterior liberación de recursos permitió al metaplanificador considerar la ejecución de una simulación con 4 procesadores. GMarte permite que, ante cada simulación, se desencadene un proceso de selección de recurso que considera el estado dinámico del Grid. Esto permite cambiar la decisión de mejor recurso en cada asignación de tarea, pudiendo adaptarse a los cambios de disponibilidad en el Grid.

La ejecución del caso de estudio precisó 336 minutos (5.6 horas), desde el inicio del proceso de

metaplanificación hasta la finalización de la transferencia de datos de la última tarea a la máquina cliente. Una aproximación secuencial tradicional, ejecutando una simulación tras otra en un solo nodo de Odin, precisó 17468 minutos (más de 12 días).

Consideremos ahora la utilización exclusiva de la Computación Paralela en un cluster formado por 16 procesadores. La elección de ese número responde a un tamaño promedio de cluster de un centro de investigación de tamaño medio o un pequeño laboratorio de investigación biomédica. Emplear esta aproximación computacional, realizando dos ejecuciones simultaneas de 8 procesadores, en el cluster Odin, precisó un total de 2812 minutos. Por lo tanto, la aproximación basada en Computación en Grid fue casi 52 veces más rápida que la ejecución secuencial, y 8.3 veces más rápida que la ejecución utilizando exclusivamente Computación Paralela.

Resultados del Caso de Estudio

En este caso de estudio se ha utilizado el modelo Luo-Rudy Fase II para simular la propagación eléctrica en un tejido con isquemia regional aguda. Los resultados obtenidos indican que las condiciones de isquemia predisponen al corazón a la reentrada y que su generación depende en gran medida del instante de estimulación prematura.

Para ello, se han analizado los patrones de excitación del tejido simulado de acuerdo al protocolo S_1-S_2. Como consecuencia de la inyección del estímulo básico S_1, el potencial de acción se propagó por todo el tejido. Sin embargo, después de la inyección de S_2, los patrones de excitación dependen del intervalo de acoplamiento empleado. Por un parte, si S_2 se inyecta antes del instante 243 ms, cuando las células estimuladas todavía no se han recuperado del periodo refractario, no se genera un potencial de acción. Por otra parte, después del instante 267 ms, la zona superior del tejido ya ha recobrado su excitabilidad, al igual que el resto de células cuando les llega el potencial de acción, por lo que éste se propaga nuevamente por todo el tejido. Sin embargo, cuando S_2 se envía entre esos dos instantes de tiempo, se obtienen patrones de reentrada en el tejido simulado. Por lo tanto, la ventana vulnerable abarca el intervalo temporal [243,267] ms.

La Figura 10-8 muestra las diferentes etapas de la actividad eléctrica, aplicando S_2 en el instante 250 ms, por lo tanto dentro de la ventana vulnerable. Cada fotograma representa el potencial de membrana en el tejido en un instante dado de tiempo, codificado mediante una escala de colores. Como se puede apreciar en la figura, en la primera fase de activación del tejido justo después de aplicar el estímulo S_2, la parte superior del tejido ha recuperado su excitabilidad y se produce un frente de propagación hacia abajo. La onda de excitación fue planar en NZ pero se curva conforme llega a BZ y CZ. Esta curvatura se explica por las diferencias en la velocidad de conducción longitudinal. De hecho, existe suficiente evidencia experimental que la propagación de potencial de acción se retrasa en las zonas isquémicas [221], mientras que la zona de borde se caracteriza por una conductividad eléctrica por encima de lo normal [219].

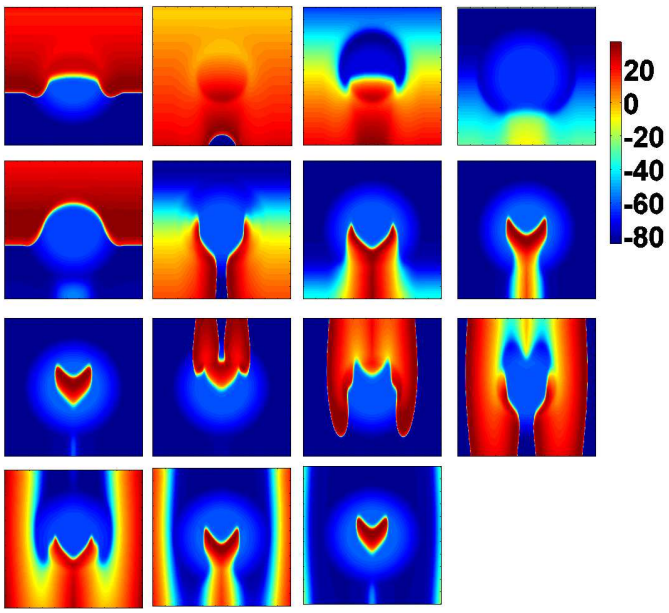


Figura 10-8: Fotogramas del potencial de membrana del tejido despues de la estimulación mediante S_2. El color rojo indica voltaje depolarizado y el color azul indica repolarizado. Los fotogramas están separados por un espaciado de 50 ms, comenzando desde el instante 100 milisegundos, hasta 800 ms. La primera fila corresponde al efecto del estímulo S_1. S_2 fue aplicado en el instante 250 ms a la parte superior del tejido. Inicialmente, la propagación eléctrica ocurre de arriba a bajo.

El patrón de reentrada consistió en dos circuitos paralelos de reentrada que se asemejan a la reentrada en forma de figura de ocho obtenida por Janse en un corazón isquémico de cerdo [129]. En sus experimentos, bloquearon la arteria coronaria descendente anterior izquierda del corazón de un cerdo. Unos minutos después, al igual que en estas simulaciones, se obtuvieron patrones de excitación consistentes en reentradas en forma de figura de ocho, inmediatamente anteriores a la aparición de fibrilación ventricular.

Las simulaciones llevadas a cabo en este caso de estudio muestran la vulnerabilidad a la reentrada para tejidos isquémicos regionales, proporcionando pistas sobre las corrientes responsables de las arritmias reentrantes. Es conocido que, bajo condiciones patofisiológicas, una serie de latidos anormales puede estimular prematuramente el tejido dando lugar a circuitos reentrantes [221], tal y como confirman los resultados aquí obtenidos. Sin embargo, es difícil analizar experimentalmente los mecanismos que subyacen a los patrones de activación obtenidos y, por ello, la simulación proporciona una herramienta muy importante.

Los resultados obtenidos sugieren que diferentes grados de isquemia alterarían la vulnerabilidad a las reentradas. De hecho, no solo se deberían considerar drogas antiarrítmicas sino drogas vasodilatadores, como apertores de canales de potasio, que pueden suponer una novedosa aproximación farmacológica en el tratamiento de las arritmias reentrantes. Este caso de estudio ha sido aceptado para publicación en la revista IEEE Transactions on Information Technology in Biomedicine [222].

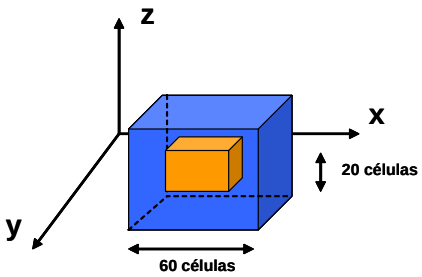


Figura 10-9: Tejido 3D de 60x60x60 células con una zona central de 20x20x20 células bajo condiciones de isquemia.

10.2.2. Isquemia de Miocardio: Ejecución en Grid Regional

En este caso de estudio se planteó estudiar los efectos de la isquemia de miocardio en la propagación del potencial de acción. Los mecanismos de generación de las taquicardias ventriculares y la fibrilación ventricular, la más mortal de las arritmias, pueden ser estudiados usando un modelo de tejido cardiaco en el que cierta parte del substrato es isquémico, resultante de la oclusión de una arteria coronaria, mientras que el resto del tejido adyacente permanece en condiciones normales [28].

En la zona afectada por el proceso de isquemia, los valores de varios parámetros electrofisiológicos sufren variaciones a lo largo del tiempo conforme la isquemia progresa. La concentración de potasio extracelular ($[K^+]_o$), en primer lugar, cambia desde su valor normal, 5.4 mmol/L, hasta un valor de 12.5 mmol/L, durante los primeros 5 minutos de patología, alcanzando un estacionario durante los siguientes 5 minutos de isquemia [218]. En segundo lugar, la concentración intracelular de ATP ($[ATP]_i$) decrece de manera lineal con el tiempo, desde un valor normal de 6.8 mmol/L a 4.6 mmol/L, durante los primeros 10 minutos de isquemia. Simultaneamente, la concentración intracelular de ADP ($[ADP]_i$) se incrementa de 15 μmol/L hasta 100 μmol/L en el mismo intervalo de tiempo [214]. Finalmente, la acidosis reduce la conductancia máxima de las corrientes de sodio (I_{Na}) y calcio ($I_{Ca(L)}$) a un 75 % de sus valores normales entre el minuto 5 y 10 del episodio de isquemia [223].

Por lo tanto, los parámetros mencionados, presentes en el modelo de propagación de potencial de acción cardiaco, cambian de manera diferente con el tiempo durante los primeros 10 minutos de la isquemia de miocardio. En las simulaciones que aquí se presentan se evaluó el comportamiento eléctrico a corto plazo del tejido, partiendo de diferentes instantes de tiempo desde la aparición de la isquemia. Por lo tanto, el tiempo desde el inicio de la isquemia es el principal parámetro que varía en el conjunto de las simulaciones.

Para ello se simuló un tejido 3D que comprende una zona central isquémica, formada por un cubo de 20x20x20 células, en el que las condiciones de $[K+]_o$, $[ATP]_i$, $[ADP]_i$, I_{Na} e $I_{Ca(L)}$ varían con el tiempo. Tal y como se puede apreciar en la Figura 10-9, este tejido está inmerso en un cubo de

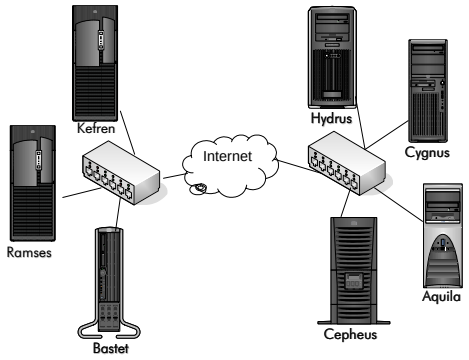


Figura 10-10: Infraestructura Grid empleada. A la izquierda las máquinas del GRyCAP y a la derecha las máquinas del ASDS.

60x60x60 células. Los parámetros electrofisiológicos del tejido que rodea a la zona central isquémica se mantienen en sus valores normales.

De esta manera, se analizó la influencia en la propagación del potencial de acción de diferentes grados de isquemia que ocurren entre los 0 y los 10 minutos desde la aparición de la patología, utilizando un incremento de tiempo de 0.5 minutos. Esto da lugar a un total de 21 simulaciones paramétricas.

Cada ejecución simuló un periodo de 80 ms con un paso de tiempo de 10 μs. Se aplicó un estímulo supra-umbral a todas las células del plano inferior durante el intervalo de simulación [50, 52] ms. Además, se guardó una instantánea del potencial de membrana de las células del tejido cada 1 ms durante el intervalo [40, 80] ms, dando lugar a un total de 68 MBytes de información binaria. Una simulación de estas características, ejecutada en secuencial en el cluster Kefren, precisó 3.8 horas con un consumo de 180 MBytes de memoria RAM.

Para poder ejecutar las simulaciones resultantes de los casos de estudio cardiacos, se utilizó una infraestructura Grid formada por recursos computacionales de dos entidades diferentes. La Figura 10-10 muestra las máquinas de las que constó el banco de pruebas. En primer lugar se utilizaron 2 clusters de PCs, de 10 nodos biprocesadores disponibles de cómputo cada uno de ellos, y una estación de trabajo Intel Itanium 2, del GRyCAP. En segundo lugar se utilizaron recursos del Grupo de Arquitectura y Seguridad de Sistemas Distribuidos (ASDS), de la Universidad Complutense de Madrid, formado por 4 PCs de sobremesa. La unión de ambas infraestructuras computacionales se realizó mediante la confianza mutua en las entidades certificadoras del GRyCAP y del ASDS.

Resultados de Ejecución

La Tabla 10.6 resume la distribución de tareas a los nodos del Grid, indicando el total de ejecuciones que se realizaron en cada una de las máquinas. Para cada una de ellas se indica, entre paréntesis, el número de procesadores empleado. Durante la metaplanificación, las máquinas Bastet

Tabla 10.6: Resumen de la asignación de tareas en el Grid regional.

Máquina	Simulaciones
Kefren	4 (4 p.), 3 (3 p.), 2 (2 p.) 2 (1 p.)
Ramses	1 (6 p.), 3 (5 p.), 1 (4 p.), 1 (1 p.)
Hydrus	2 (1 p.)
Cygnus	2 (1 p.)

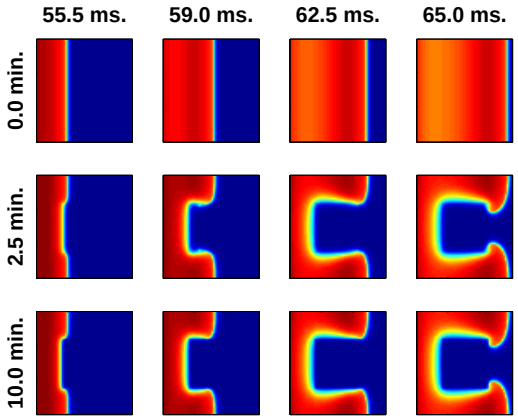
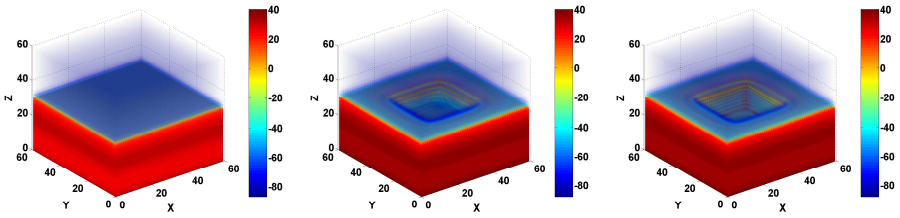


Figura 10-11: Potencial de membrana en cuatro instantes de tiempo bajo tres grados de isquemia. La propagación está aconteciendo de izquierda a derecha.

y Cepheus presentaron una alta carga computacional, por lo que GMarte decidió no asignarles ninguna simulación. Adicionalmente, durante el proceso de asignación de tareas, la máquina Aquila no ofreció suficiente memoria RAM disponible para permitir la ejecución de una simulación y, por lo tanto, nunca fue escogida por el metaplanificador. Es importante recordar que esta información es publicada por el servicio de información de los recursos y utilizada por GMarte para la estrategia de selección de recursos.

Se observa que, empleando la Computación en Grid, se pudieron ejecutar de manera transparente 4 simulaciones en las máquinas del ASDS, de 1 procesador cada una de ellas. La ejecución del caso de estudio utilizando la Computación en Grid precisó 10.27 horas, mientras que una ejecución secuencial, en uno de los nodos del cluster Kefren, requirió 81.1 horas. Finalmente, utilizando una aproximación basada en Computación Paralela, ejecutando cada simulación con 4 procesadores en el mismo cluster, permitiendo de esta manera dos ejecuciones simultáneas, requirió un total de 11.9 horas. Por lo tanto, a la vista de los resultados obtenidos, la aproximación Grid resultó ser 7.9 veces más rápida que una ejecución secuencial y 1.15 veces más rápido que la ejecución basada exclusivamente en Computación Paralela.



(a) Isquemia tras 0 min. (b) Isquemia tras 2.5 min. (c) Isquemia tras 10 min.

Figura 10-12: Representación tridimensional del potencial de membrana de las células del tejido en el instante de simulación 59 ms.

Resultados del Caso de Estudio

La Figura 10-11 resume los resultados obtenidos tras la ejecución del caso de estudio. En ella se representa la evolución del potencial de membrana de las células de un corte vertical, en el centro del cubo, para tres diferentes condiciones de isquemia, en cuatro instantes de tiempo distintos. La barra de colores es la misma que aparece en la Figura 10-12.

Las condiciones de isquemia a los 0 minutos no introducen ninguna interferencia en la propagación del potencial de acción, mientras que niveles más severos de isquemia provocan una ralentización en la propagación del potencial de acción dentro del área de tejido afectada. Este fenómeno se puede apreciar claramente mediante la representación tridimensional mostrada en la Figura 10-12, donde la propagación del potencial de acción avanza de abajo a arriba. En ella se muestra una representación tridimensional del potencial de membrana para todas las células del tejido en el instante de simulación 59, para tres grados diferentes de isquemia. Nuevamente, se puede observar como la zona isquémica reduce la velocidad de propagación del potencial de acción, ralentizándose en mayor medida conforme aumenta el grado de isquemia [221].

Los resultados obtenidos en este caso de estudio fueron publicados en un artículo [224] que se presentó en el congreso VecPar 2004. El artículo fue galardonado con el "Best Student Paper Award" y seleccionado para una edición especial de *Lecture Notes in Computer Science* [225].

10.2.3. Isquemia de Miocardio: Ejecución en Grid Global

Para esta prueba se planteó la ejecución del caso de estudio anterior sobre un Grid global, geográficamente distribuido, formado por recursos computacionales de diferentes centros de investigación. El único cambio introducido radica en que las simulaciones se llevaron a cabo durante 20 ms, en lugar de 80 ms, ya que el principal objetivo es analizar los resultados temporales de ejecución. Cada simulación generó un total de 33 MBytes de información binaria. Para este estudio, se optó por realizar la ejecución en el despliegue Grid de EGEE basado en el middleware LCG-2. Para ello, se utilizó GMarte como metaplanificador directo sobre los CE de LCG-2.

En este sentido, se decidió seleccionar algunos centros que ofrecen recursos a la VO biomed con

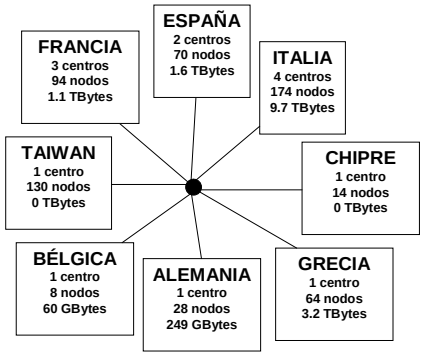


Figura 10-13: Centros elegidos de la VO *Biomed* para la ejecución del caso de estudio.

el objetivo de realizar la ejecución del caso de estudio en parte de dicha infraestructura. Para ello se escogieron previamente varios centros geográficamente dispersos. La Figura 10-13 resume los centros elegidos de esta VO, dentro del testbed de EGEE. Cada cuadrado indica el nombre del país, el número de participantes que contribuyen recursos a esta VO, el número de nodos de computación disponibles para ejecución y la capacidad total de almacenamiento de todas las máquinas, en el momento de realizar las ejecuciones.

En el diagrama se observa que los centros elegidos aglutinan casi 600 nodos de computación, geográficamente distribuidos entre 8 países, ofreciendo una capacidad de almacenamiento total del orden de TBytes.

Recursos Computacionales Empleados

Además de los recursos anteriormente mencionados, se incluyó en la infraestructura un cluster de PCs del GRyCAP. Hay que destacar que, dado que el Grid es una entidad dinámica, parte de los recursos utilizados no estaban disponibles para ejecución en el momento de comenzar el proceso de metaplanificación. La Figura 10-14 resume el estado de los centros disponibles para la ejecución antes de empezar la ejecución de tareas. Cada cuadrado indica el nombre del centro, el país en el que está situado, y el número de nodos de computación que estaban disponibles justo antes del comienzo de la metaplanificación.

Resulta importante destacar que, cuando se realizaron las ejecuciones, la ejecución de tareas paralelas basadas en MPI no estaba soportada por LCG-2. Por lo tanto, aunque GMarte permite la ejecución de aplicaciones paralelas a través de Globus Toolkit, las ejecuciones en máquinas de EGEE se ejecutaron en un solo procesador. El recurso computacional del GRyCAP no pertenece al testbed de EGEE, y ha sido configurado para permitir la ejecución de aplicaciones paralelas basadas en MPI.

Nótese que, en el momento de la ejecución del caso de estudio, la máquina en Alemania, la de Grecia, y los dos centros del testbed de EGEE en España no estaban disponibles para la ejecución.

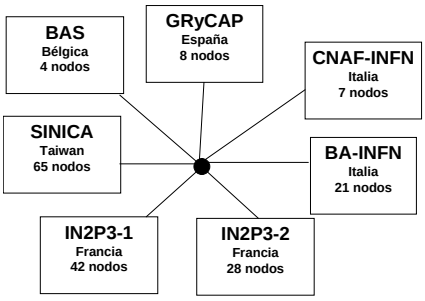


Figura 10-14: Estado de la infraestructura Grid utilizada justo antes del comienzo del proceso de metaplanificación.

Tabla 10.7: Distribución de las simulaciones en la infraestructura Grid global, para cada máquina. El número entre paréntesis indica el número de procesadores involucrado en la ejecución.

Máquina	**País**	**Simulaciones**
GRyCAP	España	2 (7 p.), 1 (2 p.)
CNAF-INFN	Italia	7 (1 p.)
IN2P3-1	Francia	2 (1 p.)
IN2P3-2	Francia	8 (1 p.)
BA-INFN	Italia	1 (1 p.)
SINICA	Taiwan	-

El acceso al recurso computacional en Chipre produjo un fallo de autorización con las credenciales suministradas, debido probablemente a algún problema transitorio. En efecto, el uso de infraestructuras distribuidas está sujeto a problemática diversa, como la falta de disponibilidad puntual, que debe ser gestionada por el sistema Grid empleado.

Resultados de Ejecución

La Tabla 10.7 resume la asignación de tareas final al despliegue Grid. Dos de las simulaciones, inicialmente asignadas a IN2P3-2, resultaron prematuramente fallidas (los registros indicaron un problema en el GASS caché, probablemente algún problema transitorio), pero los mecanismos de tolerancia a fallos en GMarte replanificaron las ejecuciones a otros recursos disponibles. Se puede observar como la mayoría de las simulaciones fueron asignadas en el entorno geográfico.

Nótese que la máquina en Taiwan nunca fue seleccionada, aunque tenía el mayor número de nodos de computación disponibles durante el proceso de metaplanificación. De hecho, si el recurso computacional no permite la ejecución paralela, no importa el número de procesadores disponibles, puesto que únicamente se llevarán a cabo ejecuciones secuenciales. Además, el coste de la transferencia de datos entre la máquina local y el recurso en Taiwan se estimó del orden de 7 veces mayor que comparado con uno de los recursos en Italia. Por lo tanto, el selector de recursos consideró la información de la proximidad de recursos para reducir el tiempo global de ejecución del caso de

estudio.

El tiempo total de ejecución del caso de estudio, desde el comienzo de la metaplanificación hasta la transferencia de datos de la última tarea pendiente, fue de 102 minutos. Una ejecución secuencial del caso de estudio, una simulación tras otra sobre un Pentium IV, una plataforma habitual en el testbed de EGEE, precisó un total de 1569 minutos. Empleando un cluster de PCs, realizando una simulación tras otra con 8 procesadores, precisó un total de 221 minutos.

A la vista de los resultados, se ha obtenido un incremento de velocidad de 15.38 al ejecutar 21 simulaciones empleando la estrategia basada en Grid. Esto representa una reducción del tiempo de ejecución total del caso de estudio de más de un día a poco más de hora y media.

Los resultados de este caso de estudio se presentaron en el congreso "Computer-Based Medical Systems 2005" (CBMS 2005) [226].

10.3. Discusión Sobre los Resultados

Los casos de estudio cardiacos involucran tareas paramétricas independientes. Esto ha permitido ejecutar de manera concurrente las diferentes simulaciones sobre una infraestructura Grid. Además, en el caso de los clusters de PCs, cada simulación ha podido aprovechar un nivel adicional de concurrencia, permitiendo la ejecución paralela en múltiples procesadores de dicha máquina. En este sentido, es importante realizar una discusión sobre los resultados obtenidos, extrapolando las conclusiones a otros ámbitos.

En primer lugar, la utilización de una política de selección dinámica de recursos ha permitido que, ante cada simulación, se desencadene un proceso de selección del mejor recurso para ejecutar la tarea. Esta aproximación permite, por un lado, considerar el carácter dinámico de una infraestructura Grid y evitar asignar una simulación a una máquina que no dispone de los suficientes recursos para su ejecución. Por otra parte, es posible generalizar esta aproximación realizando un proceso de descubrimiento de recursos ante cada planificación de una tarea. De esta manera, es posible encontrar algún recurso más apropiado que los ya disponibles. Sin embargo, por lo general, es factible asumir que la composición de la infraestructura Grid no varía durante el proceso de metaplanificación, aunque sí lo hace su estado.

En segundo lugar, la capacidad de la infraestructura Grid, en número de máquinas y prestaciones de las mismas, tiene un impacto directo en el tiempo de ejecución de los casos de estudio. Por un lado, la selección de recursos en GMarte requiere un tiempo lineal con el número de máquinas, para investigar su estado antes de tomar una decisión. Por otra parte, dicha capacidad afecta al nivel de concurrencia en la ejecución del caso de estudio cardiaco. En el caso límite, considerando una infraestructura Grid compuesta por una sola máquina, con un solo procesador, el tiempo de ejecución global del caso de estudio corresponde a su tiempo de ejecución secuencial, una simulación

tras otra, más la sobrecarga introducida por el proceso de metaplanificación y las transferencias de datos de cada tarea. Si el número de máquinas de la infraestructura es mayor que el número de tareas, permitiendo alcanzar el máximo nivel de concurrencia, entonces el tiempo de ejecución del caso de estudio se corresponde con el tiempo de la ejecución más lenta, además de, nuevamente, la sobrecarga de la metaplanificación y las transferencias de datos. Nótese que la sobrecarga de la metaplanificación incluye, principalmente, el tiempo dedicado a la selección de recursos y a la interacción con las máquinas remotas.

En tercer lugar, la utilización de infraestructuras Grid geográficamente distribuidas supone un aumento del coste temporal en la interacción con los recursos computacionales, principalmente para la transferencia de datos. En efecto, la utilización de Internet como red de comunicaciones supone una reducción del ancho de banda y un aumento de la latencia en las comunicaciones entre máquinas. Esta nueva situación implica un cambio en la estrategia de selección de recursos. De hecho, una máquina con numerosos nodos de computación libres no debe ser escogida si la transferencia de datos con ella supone un coste tan elevado que no compensa su ejecución remota. En este sentido, es necesario utilizar mecanismos de selección de recursos que incluyan el coste de las transferencias de datos, tal y como utiliza GMarte, mediante modelos de prestaciones y actualización dinámica de estado del ancho de banda entre máquinas, tras cada transferencia de datos. Nótese que, adicionalmente, es posible realizar un proceso inicial de investigación de características de recursos, previo a la ejecución, para disponer de una estimación del coste de las transferencias de datos antes de tomar una decisión que pueda resultar errónea.

Por último, la ocurrencia de fallos en un entorno Grid es bastante habitual y, en este sentido, es importante disponer de políticas de replanificación de tareas que puedan gestionar la ejecución automática de una tarea fallida en un nuevo recurso computacional. La utilización de estas estrategias, combinadas con la generación de ficheros de copia de seguridad dependientes de la aplicación, para evitar una ejecución desde el principio, permite reducir el impacto de los fallos en el tiempo de ejecución de los casos de estudio.

Pese a la problemática de la utilización de infraestructuras Grid, los resultados obtenidos tras la ejecución de los casos de estudio cardiacos permiten obtener varias conclusiones. En primer lugar, la Computación en Grid ha permitido utilizar de manera transparente recursos geográficamente distribuidos. Esta situación permite acceder a un conjunto de recursos computacionales remotos, como si fueran locales, para la ejecución de aplicaciones científicas. En segundo lugar, la utilización combinada de la Computación en Grid con la Computación de Altas Prestaciones ha permitido ejecutar un mayor número de simulaciones por unidad de tiempo, aprovechando el paralelismo de los nodos multiprocesadores de un Grid. Esto redunda en un aumento de la productividad investigadora, permitiendo la obtención de más resultados en menos tiempo.

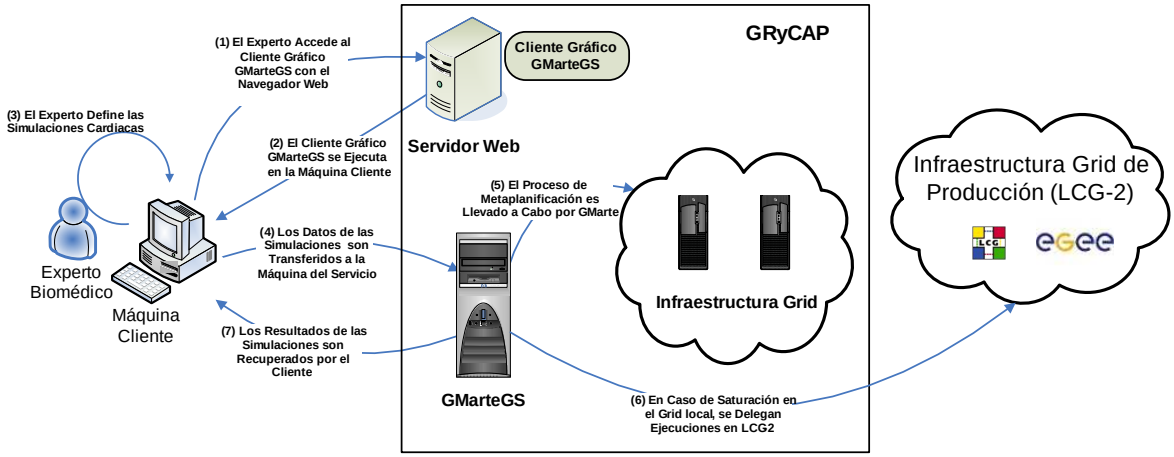


Figura 10-15: Arquitectura de la pasarela computacional para la ejecución de simulaciones cardiacas por parte de expertos biomédicos.

10.4. Utilización Transparente del Grid

Con el objetivo de permitir a los expertos biomédicos la utilización transparente del Grid, se ha desplegado el servicio GMarteGS para la ejecución de simulaciones cardiacas en recursos computacionales del GRyCAP.

La Figura 10-15 muestra la arquitectura del sistema desplegado. El experto biomédico utiliza la funcionalidad de GMarteGS a partir de su interfaz gráfica. Esta herramienta le permite la creación de sesiones de metaplanificación para la ejecución de simulaciones cardiacas sobre una infraestructura Grid desplegada en los recursos del GRyCAP. Actualmente, dicha infraestructura consta de dos clusters de PCs pero puede ser aumentada de manera transparente al usuario, ya que la herramienta cliente está desacoplada de los recursos computacionales.

GMarteGS actúa de pasarela computacional proporcionando acceso seguro y eficiente para la ejecución de simulaciones cardiacas. Además, la posibilidad de conexión de GMarte con la infraestructura Grid de producción LCG-2 permite ofrecer servicios de computación a dos niveles. Inicialmente, las simulaciones recibidas se ejecutan en las máquinas del GRyCAP. Cuando los recursos computacionales locales se agotan, es posible delegar la ejecución de las simulaciones cardiacas sobre LCG-2.

Utilizando esta aproximación es posible satisfacer los elevados requisitos computacionales de las simulaciones cardiacas. Además, los expertos biomédicos utilizan herramientas gráficas de alto nivel que les permiten utilizar las tecnologías Grid de manera sencilla. Finalmente, nótese que esta arquitectura soporta múltiples usuarios, con sus correspondientes sesiones de metaplanificación, constituyendo un entorno multi-usuario para la aceleración de las simulaciones cardiacas.

En la actualidad, esta arquitectura está siendo utilizada en producción por los investigadores del

Ci^2B, para la ejecución de simulaciones cardiacas, en áreas tan diversas como las descritas en la sección 5.8.

10.5. Conclusiones

En este capítulo se ha descrito la utilización de las tecnologías Grid en la simulación de la actividad eléctrica cardiaca, mediante la aplicación del sistema de Computación en Grid GMarte al sistema de simulación de propagación de potencial de acción CAMAEC.

En primer lugar, se ha descrito la problemática de la portabilidad de aplicaciones en un entorno heterogéneo. Para ello, se ha mostrado que las técnicas de enlace estático, combinadas con la utilización de optimizaciones independientes de la arquitectura, permiten la generación de ejecutables válidos para plataformas similares. Además, se han medido las prestaciones del simulador adaptado, confirmando que la ligera perdida de prestaciones obtenida se compensa con la posibilidad de ampliar el número de recursos computacionales disponibles para la ejecución.

Posteriormente, se han ejecutado diferentes casos de estudio en infraestructuras Grid para evaluar los beneficios obtenidos. Para ello, se ha considerado una infraestructura local, otra regional y otra global, como bancos de pruebas para la ejecución concurrente de simulaciones cardiacas.

Las pruebas realizadas utilizan un esquema de ejecución basado en el modelo de alta productividad, muy apropiado para las simulaciones cardiacas multiparamétricas. La independencia entre las simulaciones permite su ejecución concurrente para reducir el tiempo de ejecución de los casos de estudio. Para ello, se ha utilizado un esquema basado en dos niveles de paralelismo que combina Computación de Altas Prestaciones y Computación en Grid, para realizar de manera concurrente ejecuciones que, a su vez, explotan el paralelismo dentro de un cluster de PCs.

Por último, se ha desplegado el servicio GMarteGS como pasarela computacional para la ejecución de simulaciones cardiacas en la infraestructura Grid del GRyCAP, permitiendo la posibilidad de delegar las ejecuciones en los recursos de LCG-2. Esta aproximación permite que los expertos biomédicos utilicen de forma transparente el Grid para la ejecución de sus tareas.

A la vista de los resultados obtenidos, se ha conseguido reducir en gran medida el tiempo de ejecución de los casos de estudio cardiacos, permitiendo la realización de mayor número de simulaciones por unidad de tiempo. Además, la utilización de infraestructuras Grid de producción globales, como la proporcionada por EGEE, permite aumentar el número de recursos computacionales en varios en órdenes de magnitud. Esta situación permite a los científicos abordar nuevos desafíos que requieran una potencia computacional que supere los recursos de una única organización.

Capítulo 11

Aplicación de las Tecnologías Grid al Diseño de Proteínas

"The human mind treats a new idea the way the body treats a strange protein; it rejects it". P. B. Medawar, Biologist.

En este capítulo se describe la aplicación de las tecnologías Grid, utilizando GMarte, para la optimización de proteínas. A diferencia del capítulo anterior, donde GMarte se utiliza como meta-planificador de tareas, en este capítulo se aborda una aproximación diferente. En este caso se utiliza el sistema de computación en Grid desarrollado como capa de abstracción en el acceso a los recursos de EGEE, mediante la utilización y combinación de los servicios de dicha infraestructura de manera transparente al usuario. Para ello, en primer lugar se aborda la problemática del despliegue de aplicaciones paralelas en máquinas remotas. A continuación, se describe la nueva funcionalidad implementada en GMarte para satisfacer los requisitos de la aplicación de optimización de proteínas. Finalmente, el capítulo concluye con la ejecución de un caso de estudio de diseño de proteínas sobre la infraestructura computacional de la organización virtual Biomed de EGEE.

11.1. Introducción

Generalmente, las aplicaciones determinan parte de los requisitos de diseño de los sistemas de computación. En este caso, la herramienta de optimización de proteínas gBiObj precisa de cierta funcionalidad adicional del sistema GMarte, para su ejecución eficiente en un despliegue Grid.

La herramienta gBiObj, descrita en el capítulo 6, ha sido desarrollada en lenguaje C y, salvo las librerías MPI, únicamente depende de librerías estándar del sistema que pueden encontrarse en cualquier máquina Linux. Por lo tanto, para la ejecución en Grid de esta aplicación, es preferible

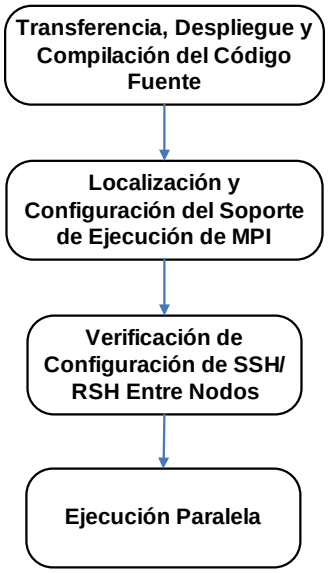


Figura 11-1: Diagrama de las principales fases para la ejecución paralela de aplicaciones instalables.

realizar previamente una migración y compilación del código fuente en la máquina remota. De esta manera, es posible garantizar un mayor nivel de portabilidad, frente a la utilización de ejecutables estáticos autocontenidos.

Adicionalmente, las múltiples tareas que involucra un caso de estudio de proteínas precisan acceder a las mismas matrices de energía, relacionadas con los objetivos de optimización. Dado que los ficheros que albergan estas matrices tienen un tamaño elevado, del orden de cientos de MBytes, y son compartidos entre todas las tareas, resulta importante desarrollar mecanismos eficientes para su gestión.

11.2. Requisitos de Ejecución Eficiente de gBiObj

En esta sección, se describe la ampliación de la funcionalidad de GMarte para soportar los requisitos de ejecución eficiente de esta aplicación.

11.2.1. El Despliegue y Ejecución Remota de Aplicaciones Paralelas

La Figura 11-1 resume las principales fases necesarias para realizar la ejecución de una aplicación cuyo código fuente se desea migrar a una máquina remota. En primer lugar, se debe transferir el código fuente de la aplicación a la máquina remota. El procedimiento más sencillo consiste en empaquetar todo el código en un fichero comprimido, que es posteriormente descomprimido en la máquina remota. Para ello, se ha incorporado en GMarte la posibilidad de especificar una configuración de despliegue (a través de un objeto *DeploymentConfiguration*) para los ficheros. Esta información incluye los comandos a ejecutar para instalar un fichero concreto, como por ejemplo, la descompresión

de un fichero. Estas acciones son llevadas a cabo por el shell-script encargado de la ejecución de la aplicación.

Una vez que se ha descomprimido el código fuente, se debe compilar la aplicación. Este proceso involucra la selección de las herramientas apropiadas, como el compilador o los ficheros de inclusión necesarios. Actualmente existen utilidades que tratan de resolver este problema, como Automake [227] o Autoconf [228], en plataformas Unix. Sin embargo, estas herramientas no permiten la selección apropiada del entorno MPI disponible en la máquina remota.

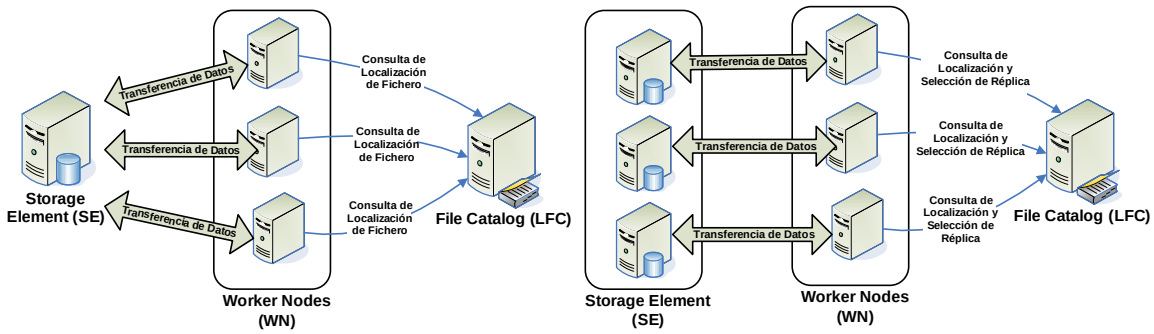
En efecto, en una máquina pueden coexistir diferentes implementaciones de MPI. Sin embargo, existen diferencias en el proceso de compilación y en el de ejecución, que dificulta la utilización transparente de diferentes versiones. Actualmente, una de las implementaciones de MPI más extendida es MPICH, tanto en los nodos de LCG-2 como en máquinas paralelas con Globus Toolkit instalado. Por lo tanto, la estrategia en el proceso de compilación de la aplicación consiste en buscar una distribución de MPICH en la máquina remota. Una vez localizada, se delega la compilación en el script *mpicc*, que permite realizar el enlace con las librerías apropiadas.

Una solución sencilla y eficiente a la problemática de la diferencia entre versiones de librerías MPI consistiría en que las máquinas paralelas de una infraestructura Grid definieran una serie de variables de entorno. Estas variables permitirían especificar las herramientas necesarias, en el caso de querer utilizar una versión optimizada de MPI para esa máquina o, por el contrario, la librería MPICH. Sin embargo, esta aproximación requiere adoptar un estándar de facto por parte de la comunidad de usuarios y desarrolladores del Grid. Por el momento, esta situación todavía no se ha producido en LCG.

Una vez compilada la aplicación, como medida de precaución adicional, resulta apropiado verificar que la configuración de SSH (Secure SHell) o RSH (Remote SHell) en el cluster de PCs permite la ejecución de comandos en los nodos internos sin necesidad de solicitar contraseña. Este requisito es imprescindible para la ejecución paralela de la aplicación, utilizando el script *mpirun*.

Es importante destacar que, recientemente, ha aparecido MPICH2 [229], una nueva implementación del estándar de MPI, que incluye todas las extensiones de MPI-2 a MPI (gestión dinámica de procesos, operaciones unilaterales, etc.). En esta nueva versión, el mecanismo de puesta en marcha de procesos difiere del existente en MPICH. Mientras que en MPICH los procesos comienzan su ejecución a partir de comandos SSH o RSH, en esta nueva versión se utilizan una serie de demonios, denominados *mpd's*, que establecen comunicación entre las máquinas antes del comienzo de la aplicación paralela. De esta manera, se proporciona un mecanismo rápido y escalable para la puesta en marcha de aplicaciones paralelas. Sin embargo, dado que esta nueva versión todavía no está ampliamente implantada, ni siquiera en los recursos de LCG-2, su utilización queda fuera de los objetivos de esta tesis.

La última fase se encarga de ejecutar la aplicación paralela, empleando el comando mpirun, donde



(a) Acceso a datos sin replicas

(b) Acceso a datos con 3 réplicas

Figura 11-2: Optimización del acceso a datos compartidos mediante la gestión de réplicas en múltiples SE.

es preciso especificar tanto el número de procesadores a emplear como la lista de máquinas en las que realizar la ejecución. Para ello, dada la integración tanto de GT como de los CE de LCG-2 con el LRMS instalado, por ejemplo PBS, es posible acceder a la lista de máquinas seleccionadas por el sistema de colas.

En este sentido, las fases descritas han sido incorporadas al wrapper que se encarga de gestionar la ejecución de la aplicación en la máquina remota. De esta manera, resulta posible realizar la ejecución remota de una aplicación cuyo código fuente se compila de forma dinámica antes de la ejecución paralela de la misma.

11.2.2. Gestión de Ficheros Compartidos

Se ha implementado en GMarte la posibilidad de especificar ficheros compartidos para las tareas. Un fichero compartido se asume que va a ser utilizado por múltiples tareas dentro de un caso de estudio. Por lo tanto, se implementan técnicas para su gestión eficiente.

En primer lugar, con el objetivo de ahorrar espacio en disco en las máquinas remotas, y reducir el consumo de red, se ha implementado un control de ficheros compartidos que impide transferir el mismo archivo a una misma máquina, si ésta dispone de un sistema de archivos compartido. En este caso, la transferencia se realiza una sola vez y, cada tarea, dispondrá de un enlace simbólico al fichero. Por ejemplo, para un cluster de PCs que reciba la ejecución de 10 optimizaciones de proteínas, cuyos datos de entrada requieran 1 GByte, en lugar de transferir y almacenar 10 copias del mismo fichero, únicamente se realiza una sola vez. Por lo tanto, el consumo de espacio en disco, así como la transferencia de ficheros por red, se reduce, en este sencillo ejemplo, en un factor de 10.

Además, las ejecuciones a través LCG-2 presentan una problemática adicional. Los ficheros de gran dimensión se deben almacenar en un SE, que será posteriormente accedido desde el WN para la descarga de los ficheros necesarios para la ejecución. Esta situación puede provocar un cuello de botella en el acceso al SE, por parte de las múltiples tareas de un caso de estudio, tal y como se

muestra en la Figura 11-2.a. En ella se observa como el ancho de banda del SE debe repartirse entre las múltiples tareas que acceden de manera concurrente para la descarga de datos.

Con el objetivo de reducir este cuello de botella se utiliza la aproximación mostrada en la Figura 11-2.b. En ella, se utilizan los mecanismos de réplica de ficheros y gestión de catálogo de LCG-2 para permitir diferentes copias de un mismo fichero, almacenadas en diferentes SE. Posteriormente, antes de la ejecución de la tarea, el wrapper se encarga de seleccionar una réplica del fichero, actualmente de manera aleatoria, para la descarga de datos al WN, previa a la ejecución. Esta aproximación permite distribuir la carga en el acceso a datos compartidos.

11.3. Caso de Estudio: Ejecución Delegada en EGEE

En esta sección se evalúan las ventajas de utilizar la computación en Grid para la optimización de proteínas, por medio de la funcionalidad de ejecución delegada en los recursos de EGEE, ofrecida por GMarte.

Para ello se plantea la optimización de una proteína de 101 posiciones y 7453 rotámeros mediante una reducción dimensional de factor 4 y realizando ejecuciones con 4 procesadores. Esta aproximación permite que cuatro procesadores puedan optimizar de manera simultanea la misma proteína. Cada procesador trabajará, en cada posición de la proteína, con un cuarto del número total de rotámeros por posición. Esta aproximación permite reducir a un 25 % el consumo de memoria de la ejecución en cada procesador, con respecto a la memoria necesaria para la optimización de la proteína completa. En este caso concreto, el consumo de memoria RAM para una optimización, empleando todos los rotámeros por posicion, precisa 111 MBytes. Nótese que, para el caso de proteínas grandes, la utilización de reducción dimensional permite abordar problemas que, en una plataforma secuencial, no pueden ser resueltos.

Con el objetivo de aumentar la exploración del espacio de búsqueda, dentro del proceso de optimización, se han ejecutado 20 optimizaciones independientes. Cada una de ellas parte de una semilla aleatoria, permitiendo diferentes ramificaciones iniciales del proceso de búsqueda. Cada optimización realiza 15 iteraciones globales y 30000 iteraciones locales. La información obtenida en cada ejecución consiste en las diferentes conformaciones de la proteína, mostradas por la salida estándar, conforme avanza el proceso de optimización.

Las matrices de energía asociadas a esta proteína ocupan 210 MBytes en disco, y pueden ser comprimidas en un fichero de 59 MBytes. También es posible comprimir el código fuente de la aplicación en un fichero de 56 KBytes. Esta es, fundamentalmente, la información necesaria para realizar la optimización. Con el objetivo de reducir la sobrecarga en el acceso a los datos compartidos, ambos ficheros son replicados, en 4 SE diferentes, antes del comienzo de la metaplanificación delegada.

El Anexo B incluye la descripción, utilizando el API de GMarte, del caso de estudio que aquí se

Tabla 11.1: Características de los recursos computacionales del caso de estudio proteómico.

Máquina	Nodos Totales	Nodos Disponibles
ce01.athena.hellasgrid.gr	226	180
gridce.sns.it	6	0
marseillece01.mrs.grid.cnrs.fr	204	195
ramses.dsic.upv.es	20	20

presenta.

11.3.1. Infraestructura Grid

Para este caso de estudio se ha utilizado la infraestructura de recursos de la VO Biomed de EGEE, bajo el middleware LCG-2. De acuerdo a la información de monitorización proporcionada por GridIce[1], actualmente consta de 1987 colas sobre las que se puede lanzar de manera simultánea hasta un total de 10224 trabajos secuenciales. Además, dispone de una capacidad de almacenamiento del orden de 2.5 Petabytes. Esta infraestructura de producción soporta entre 15000 y 40000 trabajos mensuales, relacionados con aplicaciones biomédicas.

Sin embargo, LCG-2 no soporta de manera oficial las ejecuciones paralelas basadas en MPI. En realidad, la actividad relacionada con la combinación de computación paralela y Grid en esta infraestructura de producción es bastante escasa. Actualmente, solo parte de esa gran infraestructura permite la ejecución paralela de aplicaciones. Además, la mayor parte de esos recursos presentan problemas de configuración, generalmente relacionados con la configuración de los WN para soportar las conexiones sin contraseña, requisito indispensable para la utilización de MPICH.

Ante esta situación, resulta imprescindible la utilización de listas blancas que permita dirigir las ejecuciones paralelas a un subconjunto de la infraestructura que, previamente, se ha verificado que cumple todos los requisitos para permitir este tipo de ejecuciones. Nótese que este detalle no implica pérdida de genericidad de la herramienta desarrollada. En efecto, conforme los administradores de recursos del Grid adopten las estrategias adecuadas para soportar ejecuciones paralelas, el uso de listas blancas será cada vez menos generalizado entre los usuarios.

Para las ejecuciones aquí descritas, se ha utilizado como lista blanca los recursos descritos en la Tabla 11.1. La Figura **??** muestra un ejemplo de fichero JDL generado de manera automática por GMarte, a partir de la especificación de la tarea, para delegar la ejecución de la tarea en uno de los RB de LCG-2. Se puede observar que la ejecución queda restringida a las máquinas pertenecientes a la lista blanca.

Es importante destacar que, en este caso de estudio, GMarte no actúa como metaplanificador de tareas, sino que únicamente proporciona una pasarela de alto nivel para la ejecución de tareas en un

[1] http://gridice2.cnaf.infn.it:50080

```
[
    Type = "job";
    NodeNumber = 4;
    JobType = "mpich";
    StdOutput = "stdout.std";
    OutputSandBox = {
        "stdout.std",
        "stderr.std"
    };
    StdError = "stderr.std";
    Executable = "LCGExecutionWrapper.sh";
    InputSandBox = "/home/gmolto/grid/gBiObj/out/tesis/gmolto_13989/LCGExecutionWrapper.sh";
    VirtualOrganisation = "biomed";
    LRMS_Type = "pbs";
    Requirements = ((other.GlueCEInfoLRMSType=="PBS")&&(((RegExp("ramses.dsic.upv.es",other.GlueCEUniqueID)|| ...)
    rank = other.GlueCEStateFreeCPUs
]
```

Figura 11-3: Ejemplo de JDL generado por GMarte para delegar la ejecución de la tarea en un RB de LCG-2.

Grid computacional. La planificación de tareas es llevada a cabo por el RB de LCG-2.

11.3.2. Resultados de Ejecución

La sesión de ejecución duró un total de 102 minutos, desde el comienzo de la copia de datos a los SE, hasta la transferencia de resultados a la máquina cliente. Todas las ejecuciones fueron asignadas a la misma máquina, *ce01.athena.hellasgrid.gr*. En efecto, la política de selección de recursos empleada por el RB consiste en elegir la máquina con mayor número de procesadores libres.

Se precisaron casi 5 minutos para la distribución de las réplicas en los SE y su registro en el catálogo LFC. La ejecución de cada optimización precisó un promedio de 73 minutos, de los cuales entre 2 y 11 minutos fueron empleados para la selección de la réplica y la descarga de datos del SE al WN. Esta variabilidad en los tiempos se debe al cuello de botella provocado por múltiples accesos al mismo SE. Dado que las ejecuciones se van enviando de forma progresiva, utilizando una aproximación multi-hilo, que puede lanzar hasta 3 ejecuciones de manera simultánea, las tareas comienzan su ejecución en instantes de tiempo ligeramente diferentes. De ahí que las primeras tareas en ejecutarse todavía no padezcan la sobrecarga en el acceso a los datos. Nótese que, para reducir dicha sobrecarga resulta importante encontrar un adecuado balance entre el número de réplicas, con el consiguiente desperdicio de espacio de almacenamiento, y un acceso eficiente a las mismas.

Una ejecución secuencial de cada una de las tareas paralelas, una detrás de otra, precisó 28 horas sobre una plataforma Pentium Xeon a 2.8 GHz. Por lo tanto la aproximación Grid fue 16.47 veces más rápida que una aproximación secuencial. Esta aproximación ha permitido obtener mayor número de resultados por unidad de tiempo.

Es importante destacar, en este punto, que la ejecución de aplicaciones paralelas en LCG-2 todavía está sujeta a numerosos problemas. No solo hay pocos recursos que soporten ejecución paralela, sino que, aquellos que así lo indican, a menudo sufren problemas de configuración. Además,

se ha dado el caso de problemas en la ejecución en recursos que previamente habían sido verificados que permitían una ejecución paralela, dado que los WN utilizados pueden ser diferentes.

En este sentido, cabe esperar que, con la introducción de nuevas aplicaciones paralelas al catálogo de aplicaciones de EGEE, se realice un esfuerzo mayor en dotar a esta infraestructura de producción de estabilidad para la ejecución de aplicaciones paralelas basadas en MPI.

Por otra parte, la utilización de GMarte para interactuar con LCG-2 permite al usuario utilizar la misma definición de caso de estudio para su ejecución en infraestructuras locales y en infraestructuras globales de producción. De esta manera, la transición entre ambos despliegues se convierte en un proceso transparente al usuario.

11.4. Conclusiones

En este capítulo se ha descrito la utilización de la Computación en Grid para la ejecución paramétrica del proceso de optimización de proteínas. Para ello, se ha incorporado en GMarte la funcionalidad específica para esta aplicación. Por un lado se ha implementado soporte para la migración y despliegue de aplicaciones a máquinas remotas del Grid. Por otro lado, se ha añadido soporte para la gestión de ficheros compartidos, permitiendo, en el caso de interactuar con LCG-2, la distribución automática de diferentes réplicas en varios SE para reducir el cuello de botella en el acceso a los datos compartidos.

A continuación, se ha ejecutado un caso de estudio paramétrico en la infraestructura de recursos de la VO Biomed dentro del banco de pruebas de EGEE. Esta aproximación ha permitido un aumento de la productividad al realizar mayor número de ejecuciones por unidad de tiempo.

Resulta importante destacar que, en este caso, la utilización de GMarte para delegar la ejecución en LCG-2 proporciona al usuario una capa de abstracción alto nivel. De esta manera, el usuario no debe gestionar de manera manual las transferencias de datos, la gestión de réplicas, la monitorización de las tareas, la interacción con el catálogo y el resto de servicios requeridos para la ejecución de tareas en LCG-2.

Capítulo 12

Conclusiones y Trabajos Futuros

"To succeed, jump as quickly at opportunities as you do at conclusions". Benjamin Franklin, Politician, Scientist and Inventor.

En este capítulo se presentan las conclusiones generales de esta tesis. En primer lugar, se resumen las principales aportaciones de este trabajo. Posteriormente, se describe la producción científica generada y los proyectos que han soportado las líneas de investigación de esta tesis. Finalmente, el capítulo concluye con los trabajos futuros en cada una de estas áreas.

12.1. Resumen y Contribuciones Principales

En esta tesis se ha planteado la combinación de dos estrategias de ejecución: la Computación de Altas Prestaciones y la Computación en Grid, con el objetivo de acelerar la ejecución de aplicaciones científicas y resolver problemas de mayor dimensión.

Para ello, se ha desarrollado GMarte, un sistema que permite la ejecución de aplicaciones científicas genéricas en infraestructuras Grid basadas en el estándar de facto Globus Toolkit, y en la mayor infraestructura de producción Grid existente en la actualidad, el testbed de EGEE/LCG. En este sentido, se han implementado interfaces de alto nivel para la interacción con este componente, basadas en lenguaje XML, para usuarios finales y basadas en Java, para incorporar su funcionalidad como parte de otras aplicaciones.

Posteriormente, se ha desarrollado GMarteGS, una arquitectura multi-usuario y segura, orientada a servicios Grid, que permite la metaplanificación de tareas sobre este tipo de infraestructuras. Para ello se han utilizado tecnologías estándar, permitiendo la creación de componentes interoperables. De esta manera, es posible incorporar su funcionalidad en otros servicios Grid y en aplicaciones de usuario, para delegar la ejecución en Grid de aplicaciones científicas. En este sentido, se han desarrollado interfaces gráficas para la interacción con GMarteGS, de manera que se permite la

utilización transparente de una infraestructura Grid para la ejecución de tareas por parte de usuarios no expertos en estas tecnologías. Además, esta plataforma incorpora mecanismos de seguridad para prevenir accesos no autorizados a la misma, además de la protección e integridad de datos entre los diferentes clientes del servicio.

El desarrollo de una herramienta de estas características permite reducir la distancia entre el Grid y el usuario, proporcionando una completa abstracción en la utilización de dichas tecnologías. Por lo tanto, esta parte del trabajo supone una aportación importante al ecosistema de aplicaciones Grid existentes para la ejecución eficiente de aplicaciones científicas, tanto secuenciales como paralelas.

Esta estrategia de computación combinada se ha utilizado, principalmente, para la ejecución de tareas multiparamétricas independientes entre sí. De esta manera, es posible ejecutar de manera concurrente cada tarea en las diferentes máquinas de un despliegue Grid. Al mismo tiempo, cada ejecución aprovecha el paralelismo de las máquinas multiprocesador de un Grid. Esta aproximación ha sido utilizada para la ejecución en Grid de dos aplicaciones biomédicas, la simulación de la actividad eléctrica cardiaca y la optimización de proteínas de propósito específico.

Por una parte, se ha desarrollado un sistema basado en Computación de Altas Prestaciones que permite la simulación eficiente de la propagación de potencial de acción en tejidos cardiacos bidimensionales y estructuras tridimensionales paralelepípedas. El simulador implementa técnicas eficientes de integración numérica de ODEs, que permiten la utilización de pasos de tiempo elevados. Además, utiliza E/S paralela y técnicas de generación automática de código para gestionar de manera colaborativa y eficiente el proceso de grabación de datos entre todos los procesos. La aplicación obtiene buenas prestaciones paralelas sobre un cluster de PCs, alcanzando, por ejemplo, un incremento de velocidad del 94.2 %, con 32 procesadores, para una simulación sobre un tejido de 1000x1000 células. Esto ha conseguido reducir el tiempo de las simulaciones cardiacas, además de permitir la simulación de tejidos de mayor dimensión. Por ejemplo, se ha podido simular un tejido de 100x100x100 células en 2.51 horas, que precisaba 34.72 horas sobre 2 procesadores de un cluster de PCs. En una plataforma secuencial, con 1 GByte de memoria, ni siquiera era posible su resolución debido a problemas derivados de la falta de memoria.

Actualmente, el sistema de simulación cardiaca CAMAEC se está utilizando en producción, por parte de investigadores del Ci^2B, para el estudio de la propagación de potencial de acción bajo diferentes condiciones (descritas en la sección 5.8). El desarrollo de una herramienta de estas características supone una aportación importante al campo de la electrofisiología cardiaca ya que se trata de un sistema de simulación completo, paralelo y eficiente, para la simulación de la propagación del potencial de acción. De esta manera, este tipo de herramientas computacionales facilitan la labor de los expertos biomédicos, permitiendo reducir el tiempo una simulación mediante la utilización de un cluster de PCs.

Por otra parte, se ha desarrollado una aplicación eficiente para la optimización de proteínas de

propósito específico. Para ello se han utilizado métodos eficientes de almacenamiento de matrices dispersas distribuidas y replicadas que, realizando una gestión adecuada de la memoria, ha permitido reducir hasta casi un factor de 10 el tiempo de ejecución de las optimizaciones. Además, se ha utilizado una reducción dimensional que permite la optimización colaborativa de una misma proteína por múltiples procesadores. Esta aproximación consigue la optimización de proteínas en máquinas cuya capacidad de memoria no permite albergar todos los datos necesarios. En este sentido, la aportación al campo del diseño de proteínas ha consistido en el desarrollo de una herramienta que puede llegar a optimizar proteínas que, actualmente, no pueden ser abordadas debido a los requisitos espaciales de las mismas.

Para evaluar los beneficios de la estrategia computacional propuesta, se ha utilizado GMarte para la ejecución de casos de estudio cardiacos y de diseño de proteínas sobre infraestructuras Grid. Esta aproximación ha permitido reducir en gran medida el tiempo invertido en la ejecución de estos casos de estudio. Por ejemplo, se ha conseguido reducir el tiempo de ejecución de un caso de estudio cardiaco, de análisis de isquemia, que tarda 12 días sobre una plataforma secuencial, a 5.6 horas, sobre una infraestructura Grid.

Por lo tanto, la utilización de una estrategia combinada, que aprovecha la potencia de un Grid Computacional para la ejecución de aplicaciones de alta productividad, cuyas tareas utilizan además Computación de Altas Prestaciones, permite reducir el tiempo de ejecución de los casos de estudio. Esta situación consigue realizar mayor número de simulaciones por unidad de tiempo y, de esta manera, se posibilita aumentar la productividad de los investigadores.

Resulta importante destacar que la genericidad adoptada en el desarrollo de las herramientas de Computación en Grid ha permitido su utilización en otras aplicaciones, fuera del marco de esta tesis, estableciendo colaboraciones con otros grupos de investigación. Por ejemplo, se ha utilizado GMarte para la ejecución en Grid de análisis de estructuras de edificación. También, para la creación de un entorno integrado de esqueletos de programación paralela cuyas tareas puedan ser ejecutadas en un Grid. Además, el servicio GMarteGS se ha utilizado como soporte computacional para la simulación del guiado de la luz en fibras de cristal fotónico.

En este sentido, la creación de herramientas genéricas e interoperables supone una aportación al ámbito científico, permitiendo su aplicación futura en campos de la ciencia y de la ingeniería que puedan precisar de servicios de computación avanzada mediante las tecnologías Grid.

12.2. Proyectos y Producción Científica

La presente tesis doctoral ha sido desarrollada en el marco de diversos proyectos de investigación. A continuación se proporciona una breve descripción de cada uno de ellos.

La investigación llevada a cabo en el campo de la simulación cardiaca ha estado soportada por

varios proyectos. En primer lugar aparece el proyecto "Computación Avanzada en la Modelización de la Actividad Eléctrica Cardiaca. Simulación y Visualización de las Alteraciones Producidas en Arritmias Ventriculares por la Acción de Fármacos Antiarrítmicos y la Aplicación de Campos Eléctricos" (CAMAEC, TIC2001-2686). El objetivo principal de este proyecto fue el desarrollo de modelos bidimensionales de propagación de potencial de acción y la creación de un sistema de computación eficiente que permitiera simular y visualizar la propagación eléctrica en tejidos cardiacos, para permitir el estudio de los mecanismos involucrados en la generación, mantenimiento y terminación de diferentes patologías cardiacas. Este proyecto perteneció a la convocatoria del Programa Nacional de Tecnologías de la Información y las Comunicaciones, dentro del Plan Nacional de I+D de la CICYT y finalizó satisfactoriamente en 2004. En segundo lugar se encuentra el proyecto interdisciplinar "Computación Avanzada en la Modelización, Simulación y Visualización de Arritmias Ventriculares" (CAMAV, 200110691). Este proyecto, financiado por la Universidad Politécnica de Valencia, complementó las lineas de actuación del proyecto CAMAEC. En tercer lugar, aparece el proyecto "Computación Avanzada en la Modelización de la Actividad Eléctrica del Corazón. Simulación de Ritmos Complejos Asociados a Arritmias Cardiacas en Estructuras Anatómicamente Realistas" (CAMAEC-II, TIN2004-03602), acabado en 2006. Por último, aparece el proyecto "Modelos Teóricos y Computación Avanzada en el Estudio de Señales Bioeléctricas en Células y Tejidos. Implicaciones en el Análisis de Arritmias Cardiacas, Electroestimulación y Ablación por Radiofrecuencia (CAMAEC-III, TEC2005-04199) que acaba a finales de 2008.

La parte dedicada a la optimización del proceso de proteínas viene enmarcada en el proyecto titulado "Componentes de Nueva Generación para la Explotación Eficiente de Infraestructuras Grid en e-Ciencia" (ngGrid, TIN2006-12890), financiado por el Ministerio de Educación y Ciencia. Sus objetivos principales son el desarrollo de una serie de componentes y aplicaciones de alto nivel que puedan ser utilizadas por los expertos en diferentes áreas científicas, siendo una de ellas la optimización de proteínas de propósito específico.

Referente a la parte de Computación en Grid, parte del contenido de esta tesis está dentro del ámbito del proyecto titulado "Investigación, Desarrollo e Innovación de Servicios GRID: Aplicación a Modelos Cliente-Servidor, Colaborativos y de Alta Productividad" (GRID-IT, TIC2003-01318). Este proyecto pertenece al Plan Nacional de Investigación Científica, Desarrollo en Innovación Tecnológica 2000-2003 del Ministerio de Ciencia y Tecnología. El objetivo principal del proyecto GRID-IT consiste en el despliegue de una serie de aplicaciones en un entorno Grid cuyo análisis permita desarrollar un conjunto de servicios Grid que proporcionen un nivel de abstracción adecuado y una solución estándar para diversos problemas comunes. Otro objetivo adicional consiste en acercar las tecnologías GRID a usuarios en diferentes áreas de aplicación, como puede ser la medicina, el cálculo de estructuras, la imagen sintética, etc., de modo que esta tecnología no quede restringida a ámbitos de aplicación puramente científicos.

Además, la tesis también está enmarcada en una de las lineas de actuación del proyecto titulado "Enabling Grids for E-sciencE" (EGEE, IST-2003-508833). El Grupo de Redes y Computación de Altas Prestaciones (GRyCAP), grupo de investigación del doctorando, participa en este proyecto dentro del área de trabajo de identificación de aplicaciones y soporte. Esta vinculación ha permitido disponer de acceso al banco de pruebas de EGEE/LCG.

12.2.1. Publicaciones de Investigación

La realización de esta tesis doctoral ha dado lugar a la publicación de una serie de artículos de investigación, detallados a continuación:

- Publicaciones relacionadas con el desarrollo del sistema de simulación cardiaca basado en Computación de Altas Prestaciones:

  [1] G. Moltó, "Integración en PETSc de librerías externas para la resolución de sistemas de ecuaciones lineales. Instalación y evaluación de prestaciones," Tech. Rep. DSIC-II/06/04, Departamento de Sistemas Informáticos y Computación, 2003.

  [2] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Simulación de la actividad eléctrica cardiaca: Una implementación basada en computación de altas prestaciones," in *Libro de Actas de las XIV Jornadas de Paralelismo, 15-17 de Septiembre de 2003, Leganés-Madrid*, pp. 33–38, 2003.

  [3] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "High performance cardiac tissue electrical activity simulation on a parallel environment," in *Proceedings of the First European HealthGrid Conference*, pp. 84–91, 2003.

  [4] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Empleo de técnicas computacionales avanzadas para la aceleración de las simulaciones cardiacas," in *Libro de Actas del XXII Congreso Anual de la Sociedad Española de Ingeniería Biomédica*, pp. 267–270, 2004.

  [5] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Computer simulation of action potential propagation on cardiac tissues: An efficient and scalable parallel approach," *Parallel Computing: Software Technology, Algorithms, Architectures and Applications, included in series: Advances in Parallel Computing*, vol. 13, pp. 339–346, 2004.

- Publicaciones de resultados biomédicos obtenidos mediante la colaboración con el Grupo de Bioelectrónica, la mayoría de ellos usando el sistema de simulación cardiaca desarrollado:

[1] J. M. Ferrero (Jr.), B. Trénor, F. Montilla, J. Saiz, J. M. Alonso, and G. Moltó, "Vulnerability to reentry during the acute phase of myocardial ischemia: A simulation study," in *Computers in Cardiology 2003*, pp. 425–428, 2003.

[2] B. Trenor, J. M. Ferrero, J. Saiz, A. Ferrero, J. Chorro, M. Monserrat, V. Hernández, and G. Moltó, "Effects of pinacidil on refractoriness in acutely ischemic tissue: Insights from experiments and simulations," in *Computers in Cardiology 2004*, pp. 529–532, 2004.

[3] L. Romero, J. M. Ferrero (Jr.), F. J. Saiz, B. Trénor, M. Monserrat, J. M. Alonso, G. Moltó, and F. Montilla, "Effects of acute ischemia on the restitution curves of myocardial tissue: A simulation study," in *Computers in Cardiology 2004*, pp. 525–528, 2004.

[4] L. Romero, J. M. Ferrero (Jr.), F. J. Saiz, B. Trénor, M. Monserrat, J. M. Alonso, G. Moltó, and F. Montilla, "Efecto de la isquemia aguda sobre las curvas de restitución del tejido ventricular," in *Libro de Actas del XXII Congreso Anual de la Sociedad Española de Ingeniería Biomédica*, pp. 261–264, 2004.

[5] K. Cardona, J. Saiz, M. Monserrat, J. M. Ferrero (Jr.), and G. Moltó, "Effects of the antiarrythmic drug lidocaine on ventricular electrical activity: A computer modelling study," in *32nd Annual International Conference on Computers in Cardiology 2005*, pp. 893–896, 2005.

[6] C. Tobón, F. J. Saiz, J. M. Ferrero (Jr.), G. Moltó, J. Gomis-Tena, and F. Hornero, "Simulación de la propagación del potencial de acción en tejido auricular humano," in *XXIII Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB2005)*, pp. 393–396, 2005.

[7] K. Cardona, J. Saiz, J. M. Ferrero, M. Martínez, G. Moltó, and V. Hernández, "The effect of lidocaine on reentrant ventricular circuits in acute ischemic situations: A computer modelling study," in *Computers in Cardiology*, vol. 33, pp. 209 – 212, 2006.

[8] L. Romero, B. Trenor, J. M. Ferrero Jr., J. Saiz, G. Moltó, and J. M. Alonso, "Safety factor in simulated 2D cardiac tissue: Influence of altered membrane excitability," in *Computers in Cardiology*, vol. 33, pp. 217–220, 2006.

[9] C. Tobón, J. Saiz, F. Hornero, C. Ruiz, V. Hernández, and G. Moltó, "Contribution of electrophysiological remodeling to generation of re-entries around the right pulmonary veins in the human atrium: A simulation study," in *Computers in Cardiology*, vol. 33, pp. 773–776, 2006.

[10] O. A. Henao, J. M. Ferrero (Jr.), J. Saiz, L. Romero, G. Moltó, and J. M. Alonso, "Effect of regional ischemia in arrhythmia vulnerability for heterogeneous transmural cardiac wall: A simulation study," in *Computers in Cardiology*, vol. 33, pp. 777–780, 2006.

[11] K. Cardona, J. Saiz, J. M. Ferrero (Jr.), M. A. Martínez, G. Moltó, and V. Hernández, "Efecto de la lidocaína sobre la refractariedad y la vulnerabilidad a reentradas," in *Proceedings of the XXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2006)*, pp. 201–204, 2006.

[12] L. Romero, B. Trenor, J. Ferrero (Jr.), J. Saiz, G. Moltó, and J. M. Alonso, "Estudio de la propagación y bloqueo en un tejido cardíaco virtual mediante el factor de seguridad," in *Proceedings of XXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2006)*, pp. 205–208, 2006.

[13] C. Tobón, J. Saiz, F. Hornero, and G. Moltó, "Remodelado eléctrico y sus efectos en un modelo de tejido auricular humano," in *Proceedings of the XXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2006).*, pp. 213–216, 2006.

[14] O. A. Henao, K. Cardona, J. M. Ferrero, J. Saiz, and G. Moltó, "Influencia de hiperkalemia y acidosis en la formación de rotores en pared transmural ventricular: una simulación teórica.," in *Proceedings of the XXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2006)*, pp. 101–104, 2006.

[15] B. Trenor, L. Romero, J. M. Ferrero (Jr.), J. Saiz, G. Moltó, and J. M. Alonso, "Vulnerability to reentry in a regionally ischemic tissue. a simulation study," *Annals of Biomedical Engineering*, 2007. To appear.

- Publicaciones relacionadas con el desarrollo de GMarte y su extensión a una arquitectura orientada a servicios:

[1] J. M. Alonso, V. Hernández, and G. Moltó, "Enabling high level access to grid computing services," *ERCIM News. Special: Grids: The Next Generation*, vol. 59, pp. 42–43, 2004.

[2] J. M. Alonso, V. Hernández, and G. Moltó, "An object-oriented view of grid computing technologies to abstract remote task execution," in *Proceedings of the Euromicro 2005 International Conference*, pp. 235–242, 2005.

[3] J. M. Alonso, V. Hernández, and G. Moltó, "GMarte: Grid middleware to abstract remote task execution," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 15, pp. 2021–2036, 2006.

[4] J. M. Alonso, V. Hernández, and G. Moltó, "Towards on-demand metascheduling on computational grids," in *Proceedings of the 15th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP 2007)*, pp. 84–88, IEEE, 2007.

[5] G. Moltó, V. Hernández, and J. M. Alonso, "A service-oriented WSRF-based architecture for metascheduling on computational grids," *Future Generation Computer Systems (International Journal of Grid Computing)*, 2007. To appear.

- Publicaciones relacionadas con la aplicación de las tecnologías Grid a la simulación de la actividad eléctrica.

[1] J. M. Alonso, V. Hernández, and G. Moltó, "Globus-based grid computing simulations of action potential propagation on cardiac tissues," *Lecture Notes in Computer Science*, vol. 3149, pp. 444–451, 2004.

[2] J. M. Alonso, V. Hernández, and G. Moltó, "Grid computing based simulations of the electrical activity of the heart," *Lecture Notes in Computer Science*, vol. 3036. Part I, pp. 482–485, 2004.

[3] J. M. Alonso, V. Hernández, and G. Moltó, "Grid computing based simulations of action potential propagation on cardiac tissues," tech. rep., Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, 2004.

[4] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Three-dimensional cardiac electrical activity simulation on cluster and grid platforms," in *Proceedings of the Vecpar 2004 International Conference*, pp. 485–498, 2004.

[5] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Computación de altas prestaciones sobre entornos grid en la simulación eléctrica de tejidos cardiacos," in *Proceedings of the XV Jornadas de Paralelismo, Almeria (Spain)*, pp. 271–276, 2004.

[6] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Three-dimensional cardiac electrical activity simulation on cluster and grid platforms," *Lecture Notes in Computer Science*, vol. 3402, pp. 219–232, 2005.

[7] J. M. Alonso, V. Hernández, and G. Moltó, "Experiences on a large scale grid deployment with a computationally intensive biomedical application," in *18th IEEE International Symposium on Computer-Based Medical Systems* (I. C. Society, ed.), pp. 567–569, 2005.

[8] J. M. Alonso, V. Hernández, and G. Moltó, "Biomedical and civil engineering experiences using grid computing technologies," in *Parallel Computing (ParCo) 2005* (G. Joubert, W. Nagel, F. Peters, O. Plata, P. Tirado, and E. Zapata, eds.), vol. 33 of *Parallel Computing: Current & Future Issues of High-End Computing*, pp. 647–654, NIC-Directors (John von Neumann Institute for Computing), 2005.

[9] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, and J. Saiz, "A WSRF-based computational gateway to the EGEE infrastructure for the simulation of cardiac electrical activity," in *Proceedings of the 2nd EGEE User Forum*, 2007. To appear.

[10] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, J. Saiz, and B. Trenor, "A grid computing-based approach for the acceleration of simulations in cardiology," *IEEE Transactions on Information Technology in Biomedicine*, 2007. To appear.

- Publicaciones en las que se aplica el sistema GMarte en otras líneas de investigación. Actualmente, se incluye la ejecución en Grid de análisis de estructuras de edificación, la integración con el entorno de programación JasKel, basado en esqueletos, y el desarrollo de un sistema de simulación de metaplanificación.

[1] J. M. Alonso, C. de Alfonso, V. Hernández, and G. Moltó, "A grid-based application for the three-dimensional dynamic analysis of high-rise buildings," in *Tenth International Conference on Civil, Structural and Environmental Engineering Computing* (B. Topping, ed.), pp. 1–16, Civil-Comp Press, 2005.

[2] J. M. Alonso, V. Hernández, and G. Moltó, "Aplicación de la computación paralela y las tecnologías grid en el cálculo dinámico de estructuras de edificación," in *Libro de Actas del 12th International Congress on Computer Science Research*, pp. 3–14, 2005.

[3] J. M. Alonso, V. Hernández, R. López, and G. Moltó, "A service oriented system for on demand dynamic structural analysis over computational grids," in *VECPAR'06: Seventh International Meeting on High Performance Computing for Computational Science*, 2006.

[4] J. M. Alonso, V. Hernández, R. López, and G. Moltó, "A grid service development for 3d structural analysis," in *CST2006: The Eighth International Conference on Computational Structures Technology*, 2006. To appear.

[5] J. M. Alonso, V. Hernández, R. López, and G. Moltó, "Experiences using grid services in structural dynamics," in *Proceedings of ECCPM 2006: e-Business and e-Work in Architecture, Engineering and Construction* (M. . Scherer, ed.), pp. 359–366, Taylor & Francis Group, London, 2006.

[6] J. M. Alonso, V. Hernández, and G. Moltó, "A high-throughput application for the dynamic analysis of structures on a grid environment," *Advances in Engineering Software*, 2006. To appear.

[7] J. M. Alonso, V. Hernández, R. López, and G. Moltó, "Una aproximación orientada a servicios grid para el análisis estático y dinámico de estructuras de edificación," in *Proceedings of the XXIII Conferencia Latinoamericana de Informática (CLEI 2006)*, 2006.

[8] J. M. Alonso, V. Hernández, R. López, and G. Moltó, "A service oriented system for on demand dynamic structural analysis over computational grids," *Lecture Notes in Computer Science*, vol. 4395, pp. 13–26, 2007. Included in a Special Edition for VecPar 2006 Conference.

[9] J. M. Alonso, V. Hernández, G. Moltó, A. Proença, and J. L. Sobral, "Grid enabled JasKel skeletons with GMarte," in *Proceedings of the Iberian Grid Infrastructure Conference*, 2007. To appear.

[10] J. M. Alonso, V. Hernández, and G. Moltó, "A virtual simulation-oriented framework to evaluate metascheduling policies in computational grids," in *Proceedings of the International Technology, Education and Development Conference (INTED 2007)* (IATED, ed.), p. 53, 2007.

[11] J. M. Alonso, V. Hernández, R. López, and G. Moltó, "Grid4build: A high performance grid framework for the 3d analysis and visualisation," in *Proceedings of the Iberian Grid Infrastructure*, 2007. To appear.

Cabe citar, en esta sección, el premio "Best Student Paper Award" otorgado al doctorando por el artículo:

[1] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Three-dimensional cardiac electrical activity simulation on cluster and grid platforms," in *Proceedings of the Vecpar 2004 International Conference*, pp. 485–498, 2004.

12.3. Trabajos Futuros

Aunque las herramientas desarrolladas en esta tesis se están utilizado en producción por parte de otros investigadores, sería prematuro hablar de versiones finales. Existe todavía una importante labor de investigación y desarrollo para ampliar la funcionalidad de cada una de ellas. En realidad, aunque pueda parecer un tópico, es quizá mayor el trabajo pendiente que el ya realizado.

En el campo de la simulación cardiaca, resulta importante utilizar modelos geométricos detallados que, combinados con modelos celulares avanzados, permitan una simulación realista de la propagación eléctrica. Además, la incorporación del modelo de propagación bidominio, que considera el medio intracelular y el extracelular, puede suponer una importante ventaja para el estudio de otros fenómenos cardiacos. Por otra parte, existen muchos modelos celulares, dependientes del tejido a estudiar, que no han sido incorporados al sistema CAMAEC. Integrar futuras características a este sistema aumentará la funcionalidad del mismo, permitiendo a los expertos biomédicos avanzar en el conocimiento de este campo.

En el campo de la Computación en Grid, son numerosas las líneas de actuación futuras. En primer lugar, se plantea la interoperabilidad de GMarte con otras infraestructuras Grid existentes, aparte de LCG-2. Con la aparición de gLite, la nueva versión del middleware de EGEE, nuevas interfaces aparecen que deberían ser integradas en GMarte para poder utilizar de forma transparente recursos computacionales basados en este nuevo sistema. Además, resulta interesante incorporar nuevas estrategias de metaplanificación y de selección de recursos que puedan resultar eficientes para otros tipos de aplicaciones. Adicionalmente, la tolerancia a fallos es una característica importante en este tipo de sistemas. En este sentido, aumentar el abanico de fallos gestionado redundará en la robustez del sistema frente a situaciones imprevistas.

Por otra parte, la utilización de una arquitectura orientada a servicios permite idear nuevos escenarios. Por ejemplo, GMarte podría delegar la ejecución de tareas en un servicio GMarteGS permitiendo, de esta manera, la existencia de diferentes servicios Grid de metaplanificación conectados a diferentes Grid. De esta manera, se utilizaría una aproximación recursiva a la interconexión de infraestructuras Grid.

En el campo de la optimización de proteínas, también existen muchas líneas de actuación. En primer lugar, es posible incorporar un criterio de parada al proceso iterativo de optimización, basado en técnicas como la información mutua. Por otra parte, resultaría de interés implementar un módulo que, dependiendo de las características del entorno de ejecución, decidiera la distribución de datos óptima. Además, es necesario realizar un análisis más exhaustivo de las implicaciones de la reducción dimensional, en diferentes casos de estudio. No obstante, los resultados preliminares obtenidos invitan a ser optimistas en este aspecto.

Apéndice A

Descripción de los Recursos Computacionales Utilizados

En este anexo se describen los recursos computacionales que se han utilizado en las ejecuciones descritas en la tesis.

A.1. Máquinas del GRyCAP

Esta sección resume las principales características de los recursos del GRyCAP que han sido utilizados.

A.1.1. El Cluster Ramses

El cluster de PCs Ramses está formado por 12 nodos IBM xSeries 330 de los cuales uno se utiliza como nodo principal y los 11 restantes están dedicados exclusivamente a cálculo. Cada nodo está formado por 2 procesadores Pentium III a 866 Mhz y 512 MBytes de memoria RAM.

Los nodos están interconectados mediante dos redes independientes: una *Fast Ethernet* y una *Gigabit Ethernet*. La red Fast Ethernet se utiliza para tareas administrativas como por ejemplo, realizar la comunicación entre nodos para mantener un sistema de archivos compartido entre todos ellos, a través de NFS. La red Gigabit Ethernet se utiliza exclusivamente para la comunicación entre procesos. Por lo tanto, las labores administrativas no interfieren con la comunicación entre procesos.

Cada nodo lleva instalado el Sistema Operativo Scientific Linux SL Release 3.0.4. El gestor de trabajos instalado es Torque 1.0.1p5. En este cluster está instalado el middleware Grid LCG 2.6.

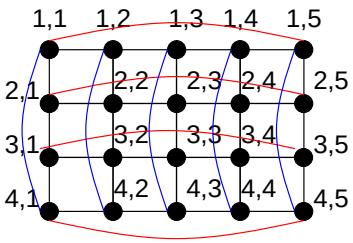


Figura A-1: Esquema de interconexión de la red SCI de Kefren.

A.1.2. El Cluster Kefren

El cluster de PCs Kefren consta de 20 nodos biprocesadores Pentium Xeon a 2 GHz y 1 GByte de memoria RAM. Todos los nodos están dedicados a cálculo científico.

Los 20 nodos están interconectados mediante una red SCI de baja latencia en forma de Toro 2D dispuestos en una matriz de 4x5 nodos, tal y como se puede observar en la Figura A-1. En primer lugar se crea una matriz de 4 filas y 5 columnas de nodos, interconectando cada nodo con su vecino tanto en horizontal como en vertical. Además, existe también una conexión que cierra cada una de las filas y cada una de las columnas. Esta red se reconfigura automáticamente en caso de fallo de algún nodo para encontrar rutas alternativas entre dos nodos cualesquiera del cluster.

La red SCI ofrece una latencia del orden de 2.7 μs. Esta baja latencia permite que la red ofrezca una buenas prestaciones para el envío de mensajes entre procesadores. Adicionalmente, el cluster consta de una red Fast Ethernet a la que están conectados todos los nodos de computación y que se utiliza, fundamentalmente, para labores administrativas.

Cada nodo lleva el Sistema Operativo Red Hat 8.0. El gestor de trabajos instalado es ScaOPBS (Scali Packaging of Open PBS), que como su propio nombre indica no es más que una variante del gestor Open PBS. El cluster dispone de una version optimizada de MPI, llamada ScaMPI, que permite aprovechar las prestaciones de la red SCI.

A.1.3. El Cluster Odin

El cluster de PCs Odin consta de 55 nodos biprocesadores Intel Pentium Xeon a 2.8 GHz. Uno de los nodos actúa de nodo principal mientras que los 54 restantes sirven para cálculo científico. Cada nodo consta de 2 Gigabytes de RAM. Los nodos están interconectados mediante una red SCI con topología de Toro 2D en malla de 11x5

Cada nodo lleva instalado el Sistema Operativo Red Hat Enterprise Linux ES release 3 (Taroon Update 2). El gestor de trabajos instalado es ScaTorque (Scali Packaging of Torque). Al igual que Kefren, este cluster dispone de ScaMPI.

Tabla A.1: Características principales de los recursos empleados del grupo ASDS.

Máquina	**Procesadores**	**Memoria**
Hydrus	Pentium IV 2.53 Ghz	512 MBytes
Cygnus	Pentium IV 2.53 Ghz	512 MBytes
Aquila	Pentium III 666 Mhz	128 MBytes
Cepheus	Pentium III 666 Mhz	256 MBytes

A.1.4. La Estación de Trabajo Bastet

La estación de trabajo Bastet consta de 2 procesadores Intel Itanium 2 a 900 Mhz con un total de 4 GBytes de memoria RAM. El Sistema Operativo instalado es Red Hat Linux Advanced Server Workstation release 2.1 AW (Derry).

El middleware Grid instalado es Globus Toolkit 2.4.3 y no hay ningún gestor de colas actualmente instalado en la máquina.

A.2. Máquinas del Grupo ASDS

La Tabla A.1 resumen las principales características, en el momento de las ejecuciones, de los recursos computacionales ofrecidos por el Grupo de Arquitectura y Seguridad de los Sistemas Distribuidos, de la Universidad Complutense de Madrid. Todos los equipos llevan el Sistema Operativo Red Hat 9.0 y el middleware Grid instalado es Globus Toolkit 2.4.

Apéndice B

Descripción de un Caso de Estudio con el API de GMarte

En este capítulo se proporciona la descripción del caso de estudio de diseño de proteínas, ejecutado en el capítulo 11, utilizando el API de GMarte. Nótese que el API de alto nivel permite al usuario centrarse en la definición de las tareas de ejecución y delegar en GMarte las labores de ejecución.

```
package org.grycap.GMarte.test.protein;

import org.grycap.GMarte.tasks.*;
import org.grycap.GMarte.tasks.files.*;
import org.grycap.GMarte.resources.*;
import org.grycap.GMarte.execution.gateway.egee.uipnp.*
import org.grycap.GMarte.execution.scheduling.orchestrator.OrchestratorScheduler;
import org.apache.log4j.PropertyConfigurator;
import org.grycap.GMarte.execution.scheduling.Scheduler;

public class TesisProteinCaseStudy extends GridTaskStudy {

  public TesisProteinCaseStudy(int ntasks) {
    GridTask gridTask = createGridTask();
    addGridTask(gridTask);
    GridTask gridTask2;
    int i;

    for (i = 0; i < ntasks - 1; i++) {
      gridTask2 = gridTask.duplicate();
```

```
    gridTask2.setName("Protein Optimization #" + i);
    addGridTask(gridTask2);
  }
}

/**
 * Create a GridTask and define all the appropriate parameters.
 * @return The newly created GridTask.
 */
protected GridTask createGridTask() {
  GridTask gt;
  GridTaskConfiguration gtc;
  DeploymentConfiguration dc;
  GridInputFileSet gifs;

  gt = new GridTask();
  GridExecutableFile gef = new GridExecutableFile("gBiObj");
  gef.setIsRemote(true);
  gt.setLocalDataContainer("/home/gmolto/grid/gBiObj/out/tesis");
  gt.setGridExecutableFile(gef);
  String arguments =
      "--gra1 2trx.39lig/2trx.39lig.fold.ugrat "  +
      "--pairs1 2trx.39lig/2trx.39lig.fold.list " +
      "--nri1 2trx.39lig/quarterdesign.nri " +
      "--gra2 2trx.39lig/2trx.39lig.bind.ugrat "  +
      "--pairs2 2trx.39lig/2trx.39lig.bind.list " +
      "--nri2 2trx.39lig/quarterdesign.nri " +
      "--type ugrat --ngiters 15 --nliters 30000 -t 10 " +
      "--lambda_min 0 --lambda_max 1 --lambda_delta 0.5";
  gt.setArguments(arguments);

  gifs = new GridInputFileSet();
  GridFile gf1 = new GridFile("/home/gmolto/grid/gBiObj/gBiObj.tgz");
  gf1.setIsShared(true);
  gifs.addGridFile(gf1);

  GridFile gf = new GridFile("/home/gmolto/grid/CaseStudies/2trx.39lig.tgz");
  gf.setIsShared(true);
  gf.setNumberOfReplicas(4);
```

```
    dc = new DeploymentConfiguration();
    dc.addDeployCommand("tar zxvf 2trx.39lig.tgz");
    dc.addUndeployCommand("rm 2trx.39lig.tgz");
    dc.addUndeployCommand("rm -rf 2trx.39lig");
    gf.setDeploymentConfiguration(dc);

    gifs.addGridFile(gf);
    gt.setGridInputFileSet(gifs);

    dc = new DeploymentConfiguration();
    dc.addDeployCommand("tar zxvf gBiObj.tgz");
    dc.addDeployCommand("make");
    dc.addUndeployCommand("rm gBiObj.tgz");
    dc.addUndeployCommand("rm -rf src");

    gtc = gt.getGridTaskConfiguration();
    gtc.setDeploymentConfiguration(dc);
    gt.setGridTaskConfiguration(gtc);

    GridOutputFileSet gofs = new GridOutputFileSet();
    gofs.addWildCard("*.std");
    gt.setGridOutputFileSet(gofs);

    GridTaskLoR gtl = new GridTaskLoR();
    gtl.addRequirement(GridTaskLoR.REQ_MAX_PROCS, 4);

    gt.setGridTaskLoR(gtl);

    return gt;
  }


  public static void main(String args[]){
    Scheduler sched;
    GridTaskStudy gts;
    GridResource gr;
    TestBed tb;

    PropertyConfigurator.configure("log4j.properties");
```

```
    tb = new TestBed();

    gr = new UIPnPGridResource();
    GridResourceConfiguration grc = gr.getGridResourceConfiguration();
    grc.setVirtualOrganisation("biomed");
    gr.setGridResourceConfiguration(grc);
    grc.setWhiteListTestBed(new WhiteListLCG(true));

    tb.addGridResource(gr);

    gts = new TesisProteinCaseStudy(20);

    sched = new OrchestratorScheduler(tb, gts);
    sched.start();

    sched.waitUntilFinished();

    System.exit(0);
  }
}
```

Bibliografía

[1] J. Venter, M. Adams, *et al.*, "The sequence of the human genome," *Science*, vol. 291, pp. 1304–1351, Feb. 2001.

[2] "European heart network." http://www.ehnheart.org.

[3] C. Henriquez and A. Papazoglou, "Using computer models to understand the roles of tissue structure and membrane dynamics in arrhythmogenesis," Mar. 1996.

[4] M. R. Boyett, A. Clough, J. Dekanski, and A. V. Holden, *Modeling Cardiac Excitation and Excitability*. John Wiley & Sons, 1997.

[5] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve (reprinted from journal of physiology, vol 117, pg 500-544, 1952)," *Bulletin of Mathematical Biology*, vol. 52, no. 1-2, pp. 25–71, 1990.

[6] E. Neher and B. Sakmann, "The patch clamp technique," *Scientific American*, vol. 266, pp. 44–51, Mar. 1992.

[7] G. Beeler and H. Reuter, "Reconstruction of action potential of ventricular myocardial fibers," *Journal of Physiology-London*, vol. 268, no. 1, pp. 177–210, 1977.

[8] J. Drouhard and F. Roberge, "A simulation study of the ventricular myocardial action-potential," *IEEE Transactions on Biomedical Engineering*, vol. 29, no. 7, pp. 494–502, 1982.

[9] C. Luo and Y. Rudy, "A model of the ventricular cardiac action-potential - depolarization, repolarization, and their interaction," *Circulation Research*, vol. 68, pp. 1501–1526, June 1991.

[10] C. Luo and Y. Rudy, "A dynamic-model of the cardiac ventricular action-potential .1. simulations of ionic currents and concentration changes," *Circulation Research*, vol. 74, pp. 1071–1096, June 1994.

[11] G. Faber and Y. Rudy, "Action potential and contractility changes in [na+](i) overloaded cardiac myocytes: A simulation study," *Biophysical Journal*, vol. 78, pp. 2392–2404, May 2000.

[12] D. Geselowitz and W. Miller, "A bidomain model for anisotropic cardiac-muscle," *Annals of Biomedical Engineering*, vol. 11, no. 3-4, pp. 191–206, 1983.

[13] E. Vigmond, F. Aguel, and N. Trayanova, "Computational techniques for solving the bidomain equations in three dimensions," *Ieee Transactions on Biomedical Engineering*, vol. 49, pp. 1260–1269, Nov. 2002.

[14] M. S. Murillo, *Parallel Algorithms and Software for Time-Dependent Nonlinear Systems of Partial Differential Equations with an Application in Computational Biology.* PhD thesis, University of Colorado, 2002.

[15] S. Veronese and H. Othmer, "A computational study of wave propagation in a model for anisotropic cardiac ventricular tissue," *High-Performance Computing and Networking*, vol. 919, pp. 248–253, 1995.

[16] N. Hooke, C. Henriquez, P. Lanzkron, and D. Rose, "Linear algebraic transformations of the bidomain equations - implications for numerical-methods," *Mathematical Biosciences*, vol. 120, pp. 127–145, Apr. 1994.

[17] A. Pollard, N. Hooke, and C. Henriquez, "Cardiac propagation simulation," *Critical Reviews in Biomedical Engineering*, vol. 20, no. 3-4, pp. 171–210, 1992.

[18] J. Pormann, *A modular simulation system for the bidomain equations.* PhD thesis, Duke University, 1999.

[19] S. Veronese and H. Othmar, "Scalable implicit methods for reaction-diffusion equations in two and three space dimensions," in *Cooper Mountain Conference on Iterative Methods*, vol. 6, 1996.

[20] M. Sermesant, Y. Coudière, H. Delingette, N. Ayache, and J. Désidéri, "An electro-mechanical model of the heart for cardiac image analysis," in *Medical Image Computing and Computer-Assisted intervention (MICCAI'01)*, 2001.

[21] R. Fitzhugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysical Journal*, vol. 1, no. 6, pp. 445–&, 1961.

[22] G. Lines, M. L. Buist, A. J. Pullan, J. Sundnes, and A. Tveito, "Mathematical models and numerical methods for the forward problem in cardiac electrophysiology," *Computations and Visualization in Science*, vol. 5, pp. 215–239, 2003.

[23] G. T. Lines, *Simulating the Electrical Activity of the Heart - A Bidomain Model of the Ventricles Embedded in a Torso.* PhD thesis, University of Oslo, 1999.

[24] H. Saleheen, P. Claessen, and K. Ng, "Three-dimensional finite-difference bidomain modeling of homogeneous cardiac tissue on a data-parallel computer," *Ieee Transactions on Biomedical Engineering*, vol. 44, pp. 200–204, Feb. 1997.

[25] J. Keener and K. Bogar, "A numerical method for the solution of the bidomain equations in cardiac tissue," *Chaos*, vol. 8, pp. 234–241, Mar. 1998.

[26] J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type," *Advances in Computational Mathematics*, vol. 6, pp. 207–206, December 1996.

[27] R. Joyner, F. Ramon, and J. Moore, "Simulation of action potential propagation in an inhomogeneous sheet of coupled excitable cells," *Circulation Research*, vol. 36, no. 5, pp. 654–661, 1975.

[28] R. Coronel, "Heterogeneity in extracellular potassium concentration during early myocardial-ischemia and reperfusion - implications for arrhythmogenesis," *Cardiovascular Research*, vol. 28, pp. 770–777, June 1994.

[29] D. Porras, J. Rogers, W. Smith, and A. Pollard, "Distributed computing for membrane-based modeling of action potential propagation," *IEEE Transactions on Biomedical Engineering*, vol. 47, pp. 1051–1057, Aug. 2000.

[30] P. Franzone and L. Pavarino, "A parallel solver for reaction-diffusion systems in computational electrocardiology," *Mathematical Models & Methods in Applied Sciences*, vol. 14, pp. 883–911, June 2004.

[31] S. Lloyd, A. Simpson, L. Sastry, D. Gavaghan, and D. Boyd, "Integrative biology. Exploiting e-science to combat fatal diseases," *ERCIM News*, vol. 60, 2005.

[32] J. Pitt-Francis, A. Garny, and D. Gavaghan, "Enabling computer models of the heart for high-performance computers and the grid," *Philosophical Transactions of the Royal Society A-Mathematical Physical and Engineering Sciences*, vol. 364, pp. 1501–1516, June 2006.

[33] R. Jaenicke, "Stability and folding of domain proteins," *Progress in Biophysics & Molecular Biology*, vol. 71, no. 2, pp. 155–241, 1999.

[34] A. Jaramillo and S. Wodak, "Computational protein design is a challenge for implicit solvation models," *Biophysical Journal*, vol. 88, pp. 156–171, Jan. 2005.

[35] L. Regan and W. Degrado, "Characterization of a helical protein designed from 1st principles," *Science*, vol. 241, pp. 976–978, Aug. 1988.

[36] A. Jaramillo, L. Wernisch, S. Hery, and S. Wodak, "Automatic procedures for protein design," *Combinatorial Chemistry & High Throughput Screening*, vol. 4, pp. 643–659, Dec. 2001.

[37] C. Pabo, "Molecular technology - designing proteins and peptides," *Nature*, vol. 301, no. 5897, pp. 200–200, 1983.

[38] S. Kamtekar, J. Schiffer, H. Xiong, J. Babik, and M. Hecht, "Protein design by binary patterning of polar and nonpolar amino-acids," *Science*, vol. 262, pp. 1680–1685, Dec. 1993.

[39] D. Gordon and S. Mayo, "Radical performance enhancements for combinatorial optimization algorithms based on the dead-end elimination theorem," *Journal of Computational Chemistry*, vol. 19, pp. 1505–1514, Oct. 1998.

[40] B. Dahiyat and S. Mayo, "De novo protein design: Fully automated sequence selection," *Science*, vol. 278, pp. 82–87, Oct. 1997.

[41] B. Kuhlman, G. Dantas, G. Ireton, G. Varani, B. Stoddard, and D. Baker, "Design of a novel globular protein fold with atomic-level accuracy," *Science*, vol. 302, pp. 1364–1368, Nov. 2003.

[42] M. Dwyer, L. Looger, and H. Hellinga, "Computational design of a biologically active enzyme," *Science*, vol. 304, pp. 1967–1971, June 2004.

[43] E. Quemeneur, M. Moutiez, J. Charbonnier, and A. Menez, "Engineering cyclophilin into a proline-specific endopeptidase," *Nature*, vol. 391, pp. 301–304, Jan. 1998.

[44] U. Shukla, H. Marino, P. Huang, S. Mayo, and J. Love, "A designed protein interface that blocks fibril formation," *Journal of the American Chemical Society*, vol. 126, pp. 13914–13915, Nov. 2004.

[45] L. Looger, M. Dwyer, J. Smith, and H. Hellinga, "Computational design of receptor and sensor proteins with novel functions," *Nature*, vol. 423, pp. 185–190, May 2003.

[46] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer-Verlag, 2004.

[47] A. Jaramillo, L. Wernisch, S. Hery, and S. Wodak, "Folding free energy function selects native-like protein sequences in the core but not on the surface," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, pp. 13554–13559, Oct. 2002.

[48] B. Brooks, R. Bruccoleri, B. Olafson, D. States, S. Swaminathan, and M. Karplus, "Charmm - a program for macromolecular energy, minimization, and dynamics calculations," *Journal of Computational Chemistry*, vol. 4, no. 2, pp. 187–217, 1983.

[49] D. Benson, M. Wisz, and H. Hellinga, "Rational design of nascent metalloenzymes," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97, pp. 6292–6297, June 2000.

[50] H. Hellinga and F. Richards, "Construction of new ligand-binding sites in proteins of known structure .1. computer-aided modeling of sites with predefined geometry," *Journal of Molecular Biology*, vol. 222, pp. 763–785, Dec. 1991.

[51] K. Ogata, A. Jaramillo, W. Cohen, J. Briand, F. Connan, J. Choppin, S. Muller, and S. Wodak, "Automatic sequence design of major histocompatibility complex class i binding peptides

impairing cd8(+) t cell recognition," *Journal of Biological Chemistry*, vol. 278, pp. 1281–1290, Jan. 2003.

[52] M. Taufer, M. Crowley, D. Price, A. Chien, and C. Brooks, "Study of a highly accurate and fast protein-ligand docking method based on molecular dynamics," *Concurrency and Computation-Practice & Experience*, vol. 17, pp. 1627–1641, Dec. 2005.

[53] A. Leach, *Molecular Modelling: Principles and Applications*. Pearson Education, 1996.

[54] G. Morris, D. Goodsell, R. Halliday, R. Huey, W. Hart, R. Belew, and A. Olson, "Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function," *Journal of Computational Chemistry*, vol. 19, pp. 1639–1662, Nov. 1998.

[55] F. Agostini, D. Soares-Pinto, M. Moret, C. Osthoff, and P. Pascutti, "Generalized simulated annealing applied to protein folding studies," *Journal of Computational Chemistry*, vol. 27, pp. 1142–1155, Aug. 2006.

[56] C. Tsallis and D. Stariolo, "Generalized simulated annealing," *Physica A*, vol. 233, pp. 395–406, Nov. 1996.

[57] L. B. Costa, L. Feitosa, E. Araújo, G. Mendes, R. Coelho, W. Cirne, and D. Fireman, "Mygrid: A complete solution for running bag-of-tasks applications," in *Proceedings of the SBRC 2004 - Salão de Ferramentas (22nd Brazilian Symposium on Computer Networks - III Special Tools Session)*, 2004.

[58] C. Woods, M. Ng, S. Johnston, S. Murdock, B. Wu, K. Tai, H. Fangohr, P. Jeffreys, S. Cox, J. Frey, M. Sansom, and J. Essex, "Grid computing and biomolecular simulation," *Philosophical Transactions of the Royal Society A-Mathematical Physical and Engineering Sciences*, vol. 363, pp. 2017–2035, Aug. 2005.

[59] Y. Sugita, A. Kitao, and Y. Okamoto, "Multidimensional replica-exchange method for free-energy calculations," *Journal of Chemical Physics*, vol. 113, pp. 6042–6051, Oct. 2000.

[60] W. Sudholt, K. Baldridge, D. Abramson, C. Enticott, S. Garic, C. Kondric, and D. Nguyen, "Application of grid computing to parameter sweeps and optimizations in molecular modeling," *Future Generation Computer Systems*, vol. 21, pp. 27–35, Jan. 2005.

[61] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the Nimrod-G resource broker," *Future Generation Computer Systems*, vol. 18, pp. 1061–1074, Oct. 2002.

[62] "Human proteome folding project." http://www.grid.org/projects/hpf.

[63] "Swiss bio grid." http://www.swissbiogrid.org.

[64] "LCG: LHC Computing Grid Project." http://cern.ch/lcg.

[65] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*. Addison Wesley, 2003.

[66] S. G. Akl, *The design and analysis of parallel algorithms*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[67] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra, *MPI: The Complete Reference*. The MIT Press, Cambridge, Massachusetts, 1996.

[68] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Computing*, vol. 22, pp. 789–828, Sept. 1996.

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

[70] "Scali." http://www.scali.com.

[71] L. Dagum and R. Menon, "Openmp: An industry-standard api for shared-memory programming," *IEEE Computational Science & Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[72] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for FORTRAN usage," *ACM Trans. Math. Soft.*, vol. 5, pp. 308–323, 1979.

[73] J. J. Dongarra, J. D. Croz, I. S. Duff, and S. Hammarling, "Algorithm 679. a set of level 3 basic linear algebra subprograms," *ACM Transactions on Mathematical Software*, vol. 16, pp. 1–17, 1990.

[74] J. J. Dongarra, J. D. Croz, I. S. Duff, and S. Hammarling, "Algorithm 679. a set of level 3 basic linear algebra subprograms: model implementation and test programs.," *ACM Transactions on Mathematical Software*, vol. 16, pp. 18–28, 1990.

[75] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, third ed., 1999.

[76] Intel, "Math kernel library." http://developer.intel.com/software/products/mkl/.

[77] R. C. Whaley, A. Petitet, and J. J. Dongarra, "Automated empirical optimization of software and the ATLAS project," *Parallel Computing*, vol. 27, no. 1–2, pp. 3–35, 2001. Also available as University of Tennessee LAPACK Working Note #147, UT-CS-00-448, 2000 (`www.netlib.org/lapack/lawns/lawn147.ps`).

[78] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, "PETSc users manual," Tech. Rep. ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[79] S. Balay, V. Eijkhout, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing* (E. Arge, A. M. Bruaset, and H. P. Langtangen, eds.), pp. 163–202, Birkhäuser Press, 1997.

[80] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[81] G. Moltó, "Integración en PETSc de librerías externas para la resolución de sistemas de ecuaciones lineales. Instalación y evaluación de prestaciones," Tech. Rep. DSIC-II/06/04, Departamento de Sistemas Informáticos y Computación, 2003.

[82] P. Amestoy, I. Duff, J. L'Excellent, and J. Koster, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *Siam Journal on Matrix Analysis and Applications*, vol. 23, pp. 15–41, Apr. 2001.

[83] J. Schulze, "Towards a tighter coupling of bottom-up and top-down sparse matrix ordering methods," *BIT*, vol. 41, no. 4, pp. 800–841, 2001.

[84] G. Karypis and V. Kumar, "Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," tech. rep., University of Minnesota, Septiembre 1998.

[85] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

[86] S. Cohen and A. Hindmarsh, "Cvode, a stiff/nonstiff ode solver in c," *Computers in Physics*, vol. 10, no. 2, pp. 138–143, 1996.

[87] A. Hindmarsh, P. Brown, K. Grant, S. Lee, R. Serban, D. Shumaker, and C. Woodward, "Sundials: Suite of nonlinear and differential/algebraic equation solvers," *Acm Transactions on Mathematical Software*, vol. 31, pp. 363–396, Sept. 2005.

[88] K. Jackson and R. Sacksdavis, "An alternative implementation of variable step-size multistep formulas for stiff odes," *Acm Transactions on Mathematical Software*, vol. 6, no. 3, pp. 295–318, 1980.

[89] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Research of the National Bureau of Standards*, vol. 49, pp. 409–436, 1952.

[90] V. Sarkar, *Partitioning and Scheduling Parallel Programs for Multiprocessors.* MIT Press Cambridge, 1989.

[91] M. J. Atallah, ed., *Algorithms and Theory of Computation Handbook.* CRC Press, 1999.

[92] D. Evans, "The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices," *IMA Journal of Applied Mathematics*, vol. 4, no. 3, pp. 295–314, 1968.

[93] G. H. Golub and C. F. Loan, *Matrix Computations.* Johns Hopkins U. Press, 1996.

[94] I. Duff, A. Erisman, and J. Reid, *Direct Methods for Sparse Matrices.* Oxford University Press, 1989.

[95] M. E. Flórez, *Construcción de inversas aproximadas tipo Sparse basada en la proyección ortogonal de Frobenius para el precondicionamiento de sistemas de ecuaciones no simétricos.* PhD thesis, Universidad de Las Palmas de Gran Canaria, 2003.

[96] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, pp. 856–869, 1986.

[97] R. Fletcher, "Conjugate gradient methods for indefinite systems," *Lecture Notes in Math.*, vol. 506, pp. 73–89, 1976.

[98] M. D. García, *Estrategia para la resolución de grandes sistemas de ecuaciones lineales. Métodos de cuasi-mínimo residuo modificado.* PhD thesis, Universidad de Las Palmas de Gran Canaria, 2003.

[99] R. W. Freund, "A transpose-free quasi-minimal residual algorithm for non-hermitian linear systems," *SIAM Journal on Scientific Computing*, vol. 14, no. 2, pp. 470–482, 1993.

[100] J. W. H. Liu, "The multifrontal method for sparse matrix solution: Theory and practice," *SIAM Rev*, vol. 34, pp. 82–109, Mar. 1992.

[101] M. T. Heath, E. Ng, and B. W. Peyton, "Parallel algorithms for sparse linear systems," *SIAM Rev*, vol. 33, pp. 420–460, Sept. 1991.

[102] T. Bui and C. Jones, "A heuristic for reducing fill in sparse matrix factorization," in *6th SIAM Conf. Parallel Processing for Scientific Computing*, pp. 445–452, 1993.

[103] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," in *Supercomputing '95*, 1995.

[104] "CellML." http://www.cellml.org.

[105] D. Noble and S. J. Noble, "A model of sino-atrial node electrical activity based on a modification of the difrancesco–noble (1984) equations," *Proceedings of the Royal Society of London. Series B, Biological Sciences.*, vol. 222, no. 1228, pp. 295–304, 1984.

[106] A. Nygren, C. Fiset, L. Firek, J. Clark, D. Lindblad, R. Clark, and W. Giles, "Mathematical model of an adult human atrial cell - the role of k+ currents in repolarization," *Circulation Research*, vol. 82, pp. 63–81, Jan. 1998.

[107] C. J. Coats, "Results of the afwa mm5 optimization project," in *5th WRF / 14th MM5 User's Workshop NCAR*, 2005.

[108] N. N. Schraudolph, "A fast, compact approximation of the exponential function," *Neural Computation*, vol. 11, no. 4, pp. 853–862, 1999.

[109] N. Hooke, *Efficient Simulation of Action Potential Propagation in a Bidomain.* PhD thesis, Duke University, 1992.

[110] F. Yergeau, T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, "Extensible markup language (XML) 1.0." http://www.w3.org/TR/2004/REC-xml-20040204/, February 2004. (Third Edition).

[111] G. Moltó and J. A. Carsí, "Generación automática de wrappers para el acceso y manipulación de datos XML," in *Libro de actas del 6º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software*, pp. 280–285, 2003.

[112] W3C Consortium, "W3C XML schema." http://www.w3.org/XML/Schema, Mayo 2001.

[113] D. Goujon, M. Michel, J. Peeters, and J. Devaney, "Automap and autolink - tools for communicating complex and dynamic data-structures using MPI," *Network-Based Parallel Computing*, vol. 1362, pp. 98–109, 1998.

[114] M. P. I. Forum, *MPI: A Message-Passing Interface Standard*, 1995.

[115] M. P. I. Forum, *MPI-2: Extensions to the Message-Passing Interface*, 1997.

[116] R. Thakur, W. Gropp, and E. Lusk, "A case for using MPI's derived datatypes to improve I/O performance," *Proc. of SC98: High Performance Networking and Computing*, 1998.

[117] R. Thakur, E. Lusk, and W. Gropp, *Users Guide for ROMIO: A High-Performance, Portable MPI-IO Implementation*. Technical Memorandum ANL/MCS-TM-234. Mathematics and Computer Science Division. Argonne National Laboratory, July 1998.

[118] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: Design and analysis of algorithms*. Benjaming-Cummings, 1994.

[119] D. P. Helmbold and C. E. McDowell, "Modeling speedup (n) greater than n," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 2, pp. 250–256, 1990.

[120] L. Romero, J. M. Ferrero (Jr.), F. J. Saiz, B. Trénor, M. Monserrat, J. M. Alonso, G. Moltó, and F. Montilla, "Effects of acute ischemia on the restitution curves of myocardial tissue: A simulation study," in *Computers in Cardiology 2004*, pp. 525–528, 2004.

[121] L. Romero, J. M. Ferrero (Jr.), F. J. Saiz, B. Trénor, M. Monserrat, J. M. Alonso, G. Moltó, and F. Montilla, "Efecto de la isquemia aguda sobre las curvas de restitución del tejido ventricular," in *Libro de Actas del XXII Congreso Anual de la Sociedad Española de Ingeniería Biomédica*, pp. 261–264, 2004.

[122] L. Romero, B. Trenor, J. Ferrero (Jr.), J. Saiz, G. Moltó, and J. M. Alonso, "Estudio de la propagación y bloqueo en un tejido cardíaco virtual mediante el factor de seguridad," in *Proceedings of XXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2006)*, pp. 205–208, 2006.

[123] L. Romero, B. Trenor, J. M. Ferrero Jr., J. Saiz, G. Moltó, and J. M. Alonso, "Safety factor in simulated 2D cardiac tissue: Influence of altered membrane excitability," in *Computers in Cardiology*, vol. 33, pp. 217–220, 2006.

[124] K. Cardona, J. Saiz, J. M. Ferrero, M. Martínez, G. Moltó, and V. Hernández, "The effect of lidocaine on reentrant ventricular circuits in acute ischemic situations: A computer modelling study," in *Computers in Cardiology*, vol. 33, pp. 209 – 212, 2006.

[125] K. Cardona, J. Saiz, J. M. Ferrero (Jr.), M. A. Martínez, G. Moltó, and V. Hernández, "Efecto de la lidocaína sobre la refractariedad y la vulnerabilidad a reentradas," in *Proceedings of the XXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2006)*, pp. 201–204, 2006.

[126] K. Cardona, J. Saiz, M. Monserrat, J. M. Ferrero (Jr.), and G. Moltó, "Effects of the antiarrythmic drug lidocaine on ventricular electrical activity: A computer modelling study," in *32nd Annual International Conference on Computers in Cardiology 2005*, pp. 893–896, 2005.

[127] O. A. Henao, J. M. Ferrero (Jr.), J. Saiz, L. Romero, G. Moltó, and J. M. Alonso, "Effect of regional ischemia in arrhythmia vulnerability for heterogeneous transmural cardiac wall: A simulation study," in *Computers in Cardiology*, vol. 33, pp. 777–780, 2006.

[128] A. L. Wit and M. J. Janse, *The Ventricular Arrhythmias of Ischemia and Infarction: Electrophysiological Mechanisms*. Blackwell/Futura, 1993.

[129] M. Janse, F. Vancapelle, H. Morsink, A. Kleber, F. Wilmsschopman, R. Cardinal, C. Dalnoncourt, and D. Durrer, "Flow of injury current and patterns of excitation during early ventricular arrhythmias in acute regional myocardial ischemia in isolated porcine and canine hearts - evidence for 2 different arrhythmogenic mechanisms," *Circulation Research*, vol. 47, no. 2, pp. 151–165, 1980.

[130] M. Spach, P. Dolber, and J. Heidlage, "Influence of the passive anisotropic properties on directional differences in propagation following modification of the sodium conductance in human atrial muscle - a model of reentry based on anisotropic discontinuous propagation," *Circulation Research*, vol. 62, pp. 811–832, Apr. 1988.

[131] C. Tobón, F. J. Saiz, J. M. Ferrero (Jr.), G. Moltó, J. Gomis-Tena, and F. Hornero, "Simulación de la propagación del potencial de acción en tejido auricular humano," in *XXIII Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB2005)*, pp. 393–396, 2005.

[132] C. Tobón, J. Saiz, F. Hornero, C. Ruiz, V. Hernández, and G. Moltó, "Contribution of electrophysiological remodeling to generation of re-entries around the right pulmonary veins in the human atrium: A simulation study," in *Computers in Cardiology*, vol. 33, pp. 773–776, 2006.

[133] D. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, vol. 3. Addison-Wesley, third ed., 1997.

[134] W. Hörmann, J. Leydold, and G. Derflinger, *Automatic Nonuniform Random Variate Generation*. Springer, Berlin., 2004.

[135] J. Leydold, G. Derflinger, G. Tirler, , and W. Hörmann, "An automatic code generator for nonuniform random variate generation," *Mathematics and Computers in Simulation*, vol. 62, no. 3-6, pp. 405–412, 2003.

[136] "Sun Grid Engine." http://gridengine.sunsource.net.

[137] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor – a distributed job scheduler," in *Beowulf Cluster Computing with Linux* (T. Sterling, ed.), MIT Press, October 2001.

[138] "Innergrid." http://www.gridsystems.com.

[139] I. Foster and C. Kesselman, *The GRID: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 1999.

[140] I. Foster and C. Kesselman, *The GRID 2: Blueprint for a new computing infrastructure*. Morgan Kaufmann, 2004.

[141] I. Foster and C. Kesselman, "The grid in a nutshell," *Grid Resource Management: State of the Art and Future Trends*, pp. 3–13, 2004.

[142] C. Catlett and L. Smarr, "Metacomputing," *Communications of the ACM*, vol. 35, no. 6, pp. 44–52, 1992.

[143] R. Stevens, P. Woodward, T. DeFanti, and C. Catlett, "From the i-way to the national technology grid," *Communications of the ACM*, vol. 40, no. 11, pp. 50–61, 1997.

[144] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[145] M. Chetty and R. Buyya, "Weaving computational grids: How analogous are they with electrical grids?," *Computing in Science & Engineering*, vol. 4, pp. 61–71, July 2002.

[146] R. Buyya, D. Abramson, and S. Venugopal, "The Grid economy," *Proceedings of the Ieee*, vol. 93, pp. 698–714, Mar. 2005.

[147] "IRISGrid." http://irisgrid.rediris.es.

[148] K. Iskra, R. Belleman, G. van Albada, J. Santoso, P. Sloot, H. Bal, H. Spoelder, and M. Bubak, "The polder computing environment: A system for interactive distributed simulation," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 14, pp. 1313–1335, 2002.

[149] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, 1997.

[150] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," *Network and Parallel Computing, Proceedings*, vol. 3779, pp. 2–13, 2005.

[151] "The Globus Alliance." http://www.globus.org.

[152] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the Grid: An open grid services architecture for distributed systems integration," tech. rep., Open Grid Service Infrastructure WG. Global Grid Forum, June 2002.

[153] T. Sandholm and J. Gawor, "Globus toolkit 3 core - a grid service container framework." http://www-unix.globus.org/toolkit/3.0/ogsa/docs/gt3_core.pdf.

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

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

[156] W3C Consortium, "XML path language (XPath)." http://www.w3.org/TR/xpath, November 1999.

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

[158] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational grids," in *5th ACM Conference on Computer and Communications Security Conference*, pp. 83–92, 1998.

[159] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 2001.

[160] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet X.509 public key infrastructure: Certificate and CRL profile." RFC 2459.

[161] B. Sotomayor and L. Childer, *Globus Toolkit 4: Programming Java Services*. Morgan Kaufmann, 2005.

[162] "GLUE schema." http://glueschema.forge.cnaf.infn.it.

[163] "Berkeley Database Information Index." https://twiki.cern.ch/twiki//bin/view/EGEE/BDII.

[164] J. Vazquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente, "Una visión global de la tecnología grid." http://asds.dacya.ucm.es/jlvazquez/files/171204PaperEnlaces1.pdf.

[165] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A computation management agent for multi-institutional grids," *Cluster Computing*, vol. 5, pp. 237–246, 2002.

[166] E. Huedo, R. S. Montero, and I. M. Llorente, "A framework for adaptive execution on grids," *Journal of Software - Practice and Experience*, vol. 34, pp. 631–651, 2004.

[167] J. M. Alonso, V. Hernández, and G. Moltó, "GMarte: Grid middleware to abstract remote task execution," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 15, pp. 2021–2036, 2006.

[168] J. M. Alonso, V. Hernández, and G. Moltó, "An object-oriented view of grid computing technologies to abstract remote task execution," in *Proceedings of the Euromicro 2005 International Conference*, pp. 235–242, 2005.

[169] J. M. Schopf, "Ten Actions When Superscheduling," SchedWD 8.5, Scheduling Working Group, 2001.

[170] M. Livny, J. Basney, R. Raman, and T. Tannenbaum, "Mechanisms for high throughput computing," *SPEEDUP Journal*, vol. 11, June 1997.

[171] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java commodity Grid kit," *Concurrency and Computation-Practice & Experience*, vol. 13, pp. 645–662, July 2001.

[172] "UIPnP - user interface plug & play." http://grid-it.cnaf.infn.it/index.php?userinterface.

[173] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1," Technical Report 08, Word Wide Web Consortium (W3C), Mayo 2000. Available from http://www.w3.org/TR/soap/.

[174] S. Krishnan and L. V. Kale, "Automating parallel runtime optimizations using post-mortem analysis," in *International Conference on Supercomputing*, pp. 221–228, 1996.

[175] J. M. Alonso, V. Hernández, R. López, and G. Moltó, "A service oriented system for on demand dynamic structural analysis over computational grids," *Lecture Notes in Computer Science*, vol. 4395, pp. 13–26, 2007. Included in a Special Edition for VecPar 2006 Conference.

[176] J. Fernando, J. Sobral, and A. Proença, "JaSkel: A java skeleton-based framework for structured cluster and grid computing," in *IEEE CCGrid'06*, 2006.

[177] V. S. Jhoney A, Kuchhal M, *Grid application framework for Java*, 2005. Available Online: http://www.alphaworks.ibm.com/tech/GAF4J.

[178] M. Neary, S. Brydon, P. Kmiec, S. Rollins, and P. Cappello, "Javelin++: scalability issues in global computing," *Concurrency-Practice and Experience*, vol. 12, pp. 727–753, July 2000.

[179] F. Baude, L. Baduel, D. Caromel, A. Contes, F. Huet, M. Morel, and R. Quilici, *GRID COMPUTING: Software Environments and Tools*, ch. Programming, Composing, Deploying for the Grid. 2006.

[180] D. Caromel, W. Klauser, and J. Vayssiere, "Towards seamless computing and metacomputing in Java," *Concurrency-Practice and Experience*, vol. 10, pp. 1043–1061, Sept. 1998.

[181] A. Grimshaw and W. Wulf, "The legion vision of a worldwide virtual computer," *Communications of the ACM*, vol. 40, pp. 39–45, Jan. 1997.

[182] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience," *Concurrency and Computation-Practice & Experience*, vol. 17, pp. 323–356, Feb. 2005.

[183] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, "Reference model for service oriented architecture 1.0," Comitte Specification 1, OASIS, August 2006. http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf.

[184] "OASIS." http://www.oasis-open.org.

[185] E. Cerami, *Web Services Essentials*. O'Reilly, 2002.

[186] I. Foster, C. Kesselman, J. M. Nick, , and S. Tuecke, "Grid services for distributed systems integration.," *IEEE Computer*, vol. 35, no. 6, pp. 37–46, 2002.

[187] "Open Grid Forum." http://www.ogf.org.

[188] D. Snelling, I. Robinson, and T. Banks, "Oasis web services resource framework (WSRF) TC." Online, April 2006. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.

[189] S. Graham, A. Karmarkar, J. Mischkinsky, I. Robinson, and I. Sedukhin, "Web services resource 1.2 (WS-Resource)," oasis standard, OASIS, April 2006. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf.

[190] E. Christensen, F. Curbera, G.Meredith, and S.Weerawarana, "Web services description language (WSDL) 1.1," technical report, World Wide Web Consortium (W3C), 2001. http://www.w3.org/TR/wsdl.html.

[191] K. Lawrence and C. Kaler, "Web services security: Soap message security 1.1 (WS-Security 2004)," oasis standard, OASIS, February 2004. Available Online: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

[192] S. Graham and J. Treadwell, "Web services resource properties 1.2 (WS-ResourceProperties)," oasis standard, OASIS, April 2006. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf.

[193] L. Srinivasan and T. Banks, "Web services resource lifetime 1.2 (WS-ResourceLifetime)," oasis standard, OASIS, April 2006. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf.

[194] W. Vambenepe, S. Graham, and P. Niblett, "Web services topics 1.3 (WS-Topics)," oasis standard, OASIS, October 2006. http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.htm.

[195] S. Graham, D. Hull, and B. Murray, "Web services base notification 1.3 (WS-BaseNotification)," oasis standard, OASIS, October 2006. http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.htm.

[196] D. Chappell and L. Liu, "Web services brokered notification 1.3 (WS-BrokeredNotification)," oasis standard, OASIS, October 2006. http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.htm.

[197] N. Ferguson and B. Schneier, *Practical Cryptography*. John Wiley & Sons, 2003.

[198] "Java web start." http://java.sun.com/products/javawebstart.

[199] D. Gannon, G. Fox, M. Pierce, B. Plale, G. von Laszewski, C. Severance, J. Hardin, J. Alameda, M. Thomas, and J. Boisseau, "Grid portals: A scientist's access point for grid services (draft 1)," tech. rep., Global Grid Forum, September 2003.

[200] I. Rodero, F. Guim, J. Corbalan, and J. Labarta, "eNANOS: Coordinated scheduling in grid environments," in *Parallel Computing: Current & Future Issues of High-End Computing, Proceedings of the International Conference ParCo 2005* (G. Joubert, W. Nagel, F. Peters, O. Plata, P. Tirado, and E. Zapata, eds.), vol. 33, pp. 81–88, John von Neumann Institute for Computing, 2006.

[201] E. Elmroth and J. Tordsson, "An interoperable standards-based grid resource broker and job submission service," in *e-Science 2005. First IEEE Conference on e-Science and Grid Computing*, pp. 212–220, 2005.

[202] W. Xiaohui, D. Zhaohui, Y. Shutao, H. Chang, and L. Huizhen, "CSF4: A WSRF compliant meta-scheduler," in *The 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing (GCA'06)*, pp. 61–67, 2006.

[203] S. Venugopal, R. Buyya, and L. Winton, "A grid service broker for scheduling distributed data-oriented applications on global grids," in *Proceedings of the 2nd Workshop on Middleware in Grid Computing (MGC 04) : 5th ACM International Middleware Conference (Middleware 2004)*, October 2004.

[204] K. Nadiminti, H. Gibbins, X. Chu, S. Venugopal, and R. Buyya, "The gridbus grid service broker and scheduler (v.3.0) user guide," tech. rep., Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, 2006. Available Online http://www.gridbus.org/broker/3.0/manual.v3.0.pdf.

[205] W. Gropp and E. Lusk, "User's Guide for MPE: Extensions for MPI Programs," Tech. Rep. ANL/MCS-TM-ANL-98/xx, Mathematics and Computer Science Division. Argonne National Laboratory, 1998.

[206] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang, "Virtual clusters for grid communities," in *CCGRID 2006*, 2006.

[207] W. Cascio, "Myocardial ischemia: what factors determine arrhythmogenesis?," *J.Cardiovasc.Electrophysiol.*, vol. 12, pp. 726–729, June 2001.

[208] D. Zipes and H. Wellens, "Sudden cardiac death," *Circulation*, vol. 98, pp. 2334–2351, Nov. 1998.

[209] M. Allessie, F. Bonke, and F. Schopman, "Circus movement in rabbit atrial muscle as a mechanism of tachycardia. iii. the "leading circleçoncept: a new model of circus movement in cardiac tissue without the involvement of an anatomical obstacle," *Circ.Res.*, vol. 41, pp. 9–18, July 1977.

[210] M. Janse and A. Kleber, "Electrophysiological changes and ventricular arrhythmias in the early phase of regional myocardial ischemia," *Circ.Res.*, vol. 49, pp. 1069–1081, Nov. 1981.

[211] I. Kodama, A. Wilde, M. Janse, D. Durrer, and K. Yamada, "Combined effects of hypoxia, hyperkalemia and acidosis on membrane action potential and excitability of guinea-pig ventricular muscle," *J.Mol.Cell Cardiol.*, vol. 16, pp. 247–259, Mar. 1984.

[212] H. Morena, M. Janse, J. Fiolet, W. Krieger, H. Crijns, and D. Durrer, "Comparison of the effects of regional ischemia, hypoxia, hyperkalemia, and acidosis on intracellular and extracellular potentials and metabolism in the isolated porcine heart," *Circ.Res.*, vol. 46, pp. 634–646, May 1980.

[213] J. Zeng, K. R. Laurita, D. S. Rosenbaum, and Y. Rudy, "Two components of the delayed rectifier k+ current in ventricular myocytes of the guinea pig type. theoretical formulation and their role in repolarization," *Circ. Res.*, vol. 77, pp. 140–152, 1995.

[214] J. M. Ferrero, J. Saiz, J. M. Ferrero, and N. Thakor, "Simulation of action potentials from metabolically impaired cardiac myocytes - role of atp-sensitive k+ current," *Circulation Research*, vol. 79, pp. 208–221, Aug. 1996.

[215] J. M. Ferrero (Jr.), B. Trénor, F. Montilla, J. Saiz, J. M. Alonso, and G. Moltó, "Vulnerability to reentry during the acute phase of myocardial ischemia: A simulation study," in *Computers in Cardiology 2003*, pp. 425–428, 2003.

[216] B. Trenor, J. Ferrero, B. Rodriguez, and F. Montilla, "Effects of pinacidil on reentrant arrhythmias generated during acute regional ischemia: A simulation study," *Annals of Biomedical Engineering*, vol. 33, pp. 897–906, July 2005.

[217] A. Noma, "Atp-regulated k+ channels in cardiac muscle," *Nature*, vol. 305, pp. 147–148, Sept. 1983.

[218] J. Weiss, N. Venkatesh, and S. Lamp, "Atp-sensitive k+ channels and cellular k+ loss in hypoxic and ischemic mammalian ventricle," *Journal of Physiology-London*, vol. 447, pp. 649–673, Feb. 1992.

[219] Y. Kagiyama, J. Hill, and L. Gettes, "Interaction of acidosis and increased extracellular potassium on action potential characteristics and conduction in guinea pig ventricular muscle," *Circ.Res.*, vol. 51, pp. 614–623, Nov. 1982.

[220] C. Watson and M. Gold, "Effect of intracellular and extracellular acidosis on sodium current in ventricular myocytes," *Am.J.Physiol*, vol. 268, pp. H1749–H1756, Apr. 1995.

[221] M. Janse, A. Kleber, A. Capucci, R. Coronel, and F. Wilmsschopman, "Electrophysiological basis for arrhythmias caused by acute-ischemia - role of the subendocardium," *Journal of Molecular and Cellular Cardiology*, vol. 18, pp. 339–355, Apr. 1986.

[222] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, J. Saiz, and B. Trenor, "A grid computing-based approach for the acceleration of simulations in cardiology," *IEEE Transactions on Information Technology in Biomedicine*, 2007. To appear.

[223] A. Yatani, A. M. Brown, and N. Akaike, "Effects of Extracellular pH on Sodium Current in Isolated, Single Rat Ventricular Cells," *J. Membr. Biol.*, vol. 78, no. 2, pp. 163–168, 1984.

[224] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Three-dimensional cardiac electrical activity simulation on cluster and grid platforms," in *Proceedings of the Vecpar 2004 International Conference*, pp. 485–498, 2004.

[225] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz, "Three-dimensional cardiac electrical activity simulation on cluster and grid platforms," *Lecture Notes in Computer Science*, vol. 3402, pp. 219–232, 2005.

[226] J. M. Alonso, V. Hernández, and G. Moltó, "Experiences on a large scale grid deployment with a computationally intensive biomedical application," in *18th IEEE International Symposium on Computer-Based Medical Systems* (I. C. Society, ed.), pp. 567–569, 2005.

[227] "Automake." http://www.gnu.org/software/automake.

[228] "Autoconf." http://www.gnu.org/software/autoconf.

[229] W. Gropp, E. Lusk, D. Ashton, P. Balaji, D. Buntinas, R. Butler, A. Chan, J. Krishna, G. Mercier, R. Ross, R. Thakur, and B. Toonen, "MPICH2 user's guide," tech. rep., Mathematics and Computer Science Division. Argonne National Laboratory, December 2006.

Acrónimos

Ci^2B Centro de Investigación e Innovación en Bioingeniería, 2

API Application Programming Interface, 130

ASDS Grupo de Arquitectura y Seguridad de Sistemas Distribuidos, 194

ATLAS Automatically Tuned Linear Algebra Software, 40

BDF Backward Differentiation Formula, 43

BDII Berkeley Database Information Index, 124

BiCG BiConjugate Gradient, 55

BLAS Basic Linear Algebra Subprograms, 38

CA Certification Authority, 118

CAMAEC Computación Avanzada en la Modelización de la Actividad Eléctrica Cardiaca, 45

CAP Computación de Altas Prestaciones, 32

CE Computing Element, 123

CERN Centre Européen pour la Recherche Nucléaire, 111

CG Conjugate Gradient, 55

CN Common Name, 118

CSF Community Scheduler Framework, 175

CSR Compressed Sparse Row, 94

CV Conductivity Velocity, 86

DAM Disección Anidada Multinivel, 59

DN Distinguished Name, 118

DNI Documento Nacional de Identidad, 119

DSS Discrete Sequential Search, 103

DTD Document Type Definition, 75

E/S Entrada/Salida, 33

EDG European DataGrid, 111

EGEE Enabling Grids for E-sciencE, 111

EPR EndPoint Reference, 165

FA Fibrilación Auricular, 88

FTP File Transfer Protocol, 114

GIIS Grid Index Information Service, 116

GLUE Grid Laboratory for Uniform Environment, 124

GMarte Grid Middleware To Abstract Remote Task Execution, 127

GMRES Generalized Minimal Residual, 55

GRAM Globus Resource Allocation Manager, 115

GriPhyN Grid Physics Network, 110

GRIS Grid Resource Information Service, 116

GRyCAP Grupo de Redes y Computación de Altas Prestaciones, 2

GSI Grid Security Infrastructure, 117

GT Globus Toolkit, 114

HTTP Hypertext Transfer Protocol, 114

IS Information Service, 123

ISA Instruction Set Architecture, 180

IST Information Society Technologies, 111

JAR Java ARchive, 171

JDL Job Description Language, 124

JNLP Java Network Launch Protocol, 171

JWS Java Web Start, 170

LB Logging & Bookkeeping, 123

LCG LHC Computing Grid Project, 111

LDAP Lightweight Directory Access Protocol, 114

LFC LCG File Catalog, 123

LFN Logical File Name, 123

LHC Large Hadron Collider, 111

LRMS Local Resource Management System, 115

LTCSR Lower Triangular Compressed Sparse Row, 94

MDS Monitoring and Discovery Service, 116

MG Mínimo Grado, 59

MKL Math Kernel Library, 40

mM mili Molar, 66

MPE Multi-Processing Environment, 183

MPEG Motion Pictures Expert Group, 47

MPI Message Passing Interface, 35

MUMPS MUltifrontal Massively Parallel sparse direct Solver, 42

nnz Number of Non-Zeros, 94

NSF National Science Foundation, 110

NUMA Non Uniform Memory Access, 35

ODE Ordinary Differential Equation, 43

OGCE Open Grid Computing Environments, 175

OGF Open Grid Forum, 160

OGSA Open Grid Service Architecture, 160

OpenMP Open Multi Processing, 18

OU Organizational Unit, 118

PBS Portable Batch System, 35

PC Personal Computer, 2

PETSc Portable, Extensible Toolkit for Scientific Computation, 40

PPDG Particle Physics Data Grid Collaboratory Pilot, 110

PVM Parallel Virtual Machine, 35

R-GMA Relational Grid Monitoring Architecture, 124

RA Registration Authority (Autoridad de Registro), 119

RB Resource Broker, 122

RFT Reliable File Transfer, 116

RSL Resource Specification Language, 138

SCI Scalable Coherent Interface, 37

SE Storage Element, 123

SF Safety Factor, 86

SOA Service-Oriented Architecture, 159

SOAP Simple Object Access Protocol, 133

SPMD Single Program Multiple Data, 50

SSH Secure SHell, 205

SUNDIALS SUite of Nonlinear and DIfferential/ALgebraic equation Solvers, 43

SURL Storage Uniform Resource Locator, 123

TCP Transmission Control Protocol, 125

TFQMR Transpose-Free Quasi-Minimal Residual, 55

UI User Interface, 122

UIPnP User Interface Plug & Play, 130

URI Uniform Resource Identifier, 165

VO Virtual Organization (Organización Virtual), 112

W3C World Wide Web Consortium, 74, 75

WAN Wide Area Network, 116

WN Worker Node, 123

WS Web Services, 114

WSDL Web Services Definition Language, 160

WSRF Web Services Resource Framework, 160

XML eXtensible Markup Language, 74

XPath XML Path Language, 116