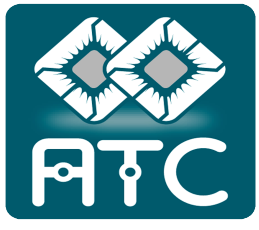
Universidad de Granada
Departamento de Arquitectura y Tecnología de Computadores,
Escuela Técnica Superior de Ingenierías

Portabilidad de aplicaciones en Astrofísica a la infraestructura de computación Grid

José Ramón Rodón Ortiz
Departamento de Física Estelar,
Instituto de Astrofísica de Andalucía – CSIC

Programa Oficial de Posgrado en Ingeniería de Computadores y Redes (P38.56.1)

Tesis doctoral



Departamento de Arquitectura y
Tecnología de Computadores,
Escuela Técnica Superior de Ingenierías
Universidad de Granada

Departamento de Física Estelar
Instituto de Astrofísica de Andalucía – CSIC

Portabilidad de aplicaciones en Astrofísica a la infraestructura de computación Grid

Memoria presentada por:

José Ramón Rodón Ortiz

para optar al grado de

Doctor por la Universidad de Granada

Programa de Doctorado:

Programa Oficial de Posgrado en Ingeniería de Computadores y Redes (P38.56.1)

Dirigida por:

Juan Carlos Suárez Yanes (DFTyC)
Julio Ortega Lopera (ETSIIT)

Granada, 15 de Enero de 2016

Como directores de la tesis titulada Portabilidad de aplicaciones en Astrofísica a la infraestructura de computación Grid, presentada por D. José Ramón Rodón Ortiz,

Dr. Julio Ortega Lopera, Doctor en Ciencias Físicas y Catedrático en el Departamento de Arquitectura y Tecnología de Computadores de la *Escuela Técnica Superior de Ingenierías de la universidad de Granada*, y Dr. Juan Carlos Suárez Yanes, Doctor en Física y perteneciente al programa de investigación Ramón y Cajal en el Departamento de Física Teórica y del Cosmos de la Facultad de Física de la Universidad de Granada

Declaran:

El doctorando José Ramón Rodón Ortiz y los directores de la tesis, el Dr. Juan Carlos Suárez Yanes y el Dr. Julio Ortega Lopera. Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

15 de Enero de 2016

Fdo:	Fdo:
Julio Ortega Lopera	Juan Carlos Suárez Yanes

José Ramón Rodón Ortiz , autor de la tesis *“Portabilidad de aplicaciones en Astrofísica a la infraestructura de computación Grid”,* autoriza a que un ejemplar de la misma quede ubicada en la Biblioteca de la Escuela Superior de Ingeniería Informática de Granada.

Fdo.: José Ramón Rodón Ortiz
15 de Enero de 2016

A María, Hugo y Ciro

Agradecimientos

Esta memoria, es resultado de varios años de trabajo duro e intensa dedicación. Un trabajo de esta envergadura jamás se podría realizar sin el apoyo, la ayuda y los consejos de múltiples personas a las que me gustaría darles las gracias.

En primer lugar, a mi pareja, consejera, psicóloga y amiga María y a mis hijos Hugo y Ciro, por estar siempre a mi lado, por su constante apoyo, enorme paciencia y por comprender mi ausencia durante horas para la realización de este trabajo.

A mi familia, tanto natural como política, por las continuas muestras de ánimo y cariño que me han ayudado a finalizar esta meta.

A mis directores de tesis, Dr. Julio Ortega y Dr. Juan Carlos Suárez, por su guía, ayuda, colaboración, conocimiento, entendimiento y comprensión para la realización de cada uno de los pasos de esta tesis. Gracias por vuestro tiempo y vuestro saber hacer.

A mi compañera de despacho, Aliciades, por la cantidad de momentos de risas, desesperación, trabajo y anécdotas durante estos seis años que me han ayudado a desestresarme de alguna manera.

A mis colaboradores más cercanos, Domi, Antonio y Rubén ya que ellos, a parte de colaborar activamente en este trabajo, son buenos amigos y consejeros.

A todos mis compañeros del Departamento del Física estelar, Javi, Zaira, Pedro, Eloy, Susana, Cristina y demás y en especial a Rafa, por motivarme intelectualmente más allá del trabajo de la tesis.

A todos mis compañeros que están o han estado en el IAA, Rubén, Sol, Haritz, Caro, Susana, Willian, Rene, Maya, Olga, Alejandro, los múltiples Frans, Emilios y a los demás (que saben perfectamente quieres son) por hacer del trabajo una tarea más agradable.

A mis compañeros del Centro de Cálculo, especialmente a Pepe Ruedas, ya que gracias a él empecé a enfocar lo que sería un trabajo de tesis.

A mis compañeros del IFCA, que es donde empezó esta aventura. Xavier, Francisco, Amalia, Jacobo, Marcos, Raquel, Alina, Lucia, Beatriz, Nuria, Serena y los demás. Nunca olvidaré los años de trabajo con vosotros. En especial a Maite por dirigirme en los primeros pasos de la tesis.

A los compañeros en el mundo Grid, por compartir su conocimiento en un tema tan complejo.

A los proyectos de investigación Grid-CSIC y CoRoT, por ofrecer los medios económicos necesarios para concluir este trabajo.

Y a todos los que algún día me han dicho... ¡Animo!

Indice

Índice de figuras

Índice de tablas

Preámbulo

Es bien conocida la importancia de la computación en la resolución de problemas, como simulaciones (de modelos físicos, de entornos, estadísticas, etc), el procesamiento de imágenes, la compresión de información, el almacenamiento, el análisis de datos, etc. Esto es incluso más evidente en ciencia.

La necesidad de altas prestaciones de computación y almacenamiento de datos en la comunidad científica ha promovido la búsqueda de nuevas soluciones computacionales. Por ejemplo, la biología [CAR14] o la física de altas energías [DAG12] son dos disciplinas científicas donde la computación de alto rendimiento es cada vez más necesaria.

Este trabajo se enmarca en el campo de la Astrofísica, donde el crecimiento exponencial de datos observacionales hacen de la computación distribuida prácticamente un requerimiento para la óptima interpretación de dichos datos en tiempos razonables. En concreto, esta Tesis cubre diversos campos de la Astronomía, desde la física estelar [GAR13] y planetaria [DAB14] hasta análisis de galaxias [PER13].

Esta Tesis se centra en el uso de infraestructuras distribuidas para coordinar recursos que no puedan manejarse mediante un control centralizado [FOS02]. Concretamente, aquí trabajamos con la infraestructura Grid, que no sólo aporta mayores recursos computacionales y de almacenamiento, sino también disponibilidad y fiabilidad en archivos críticos.

No hemos encontrado en la literatura un método global que aborde de manera sistemática la adaptación de aplicaciones a entornos distribuidos con el objeto de optimizar el proceso. En esta Tesis, además de abordar este reto, desarrollamos herramientas para poner dicho método en práctica.

Objetivos

El objetivo principal de esta Tesis doctoral es el desarrollo de un método que permita el uso eficiente de plataformas distribuidas en diversas disciplinas astrofísicas. Este uso eficiente se traduce en la optimización de los tres pasos que consideramos necesarios para ejecutar una aplicación, en nuestro caso astrofísica, en una infraestructura de computación distribuida:

- El análisis de la idoneidad de la infraestructura para satisfacer los requisitos de tiempo y almacenamiento de la aplicación.
- La ejecución simultánea de las distintas tareas que constituyen la aplicación aprovechando la disponibilidad de múltiples nodos de procesamiento.
- La gestión de la ejecución de la aplicación en la plataforma distribuida.

Para alcanzar estos objetivos se ha desarrollado una herramienta robusta de configuración y control de la computación en plataformas distribuidas que satisface requisitos de versatilidad y modularidad que son esenciales para garantizar la portabilidad de las aplicaciones astrofísicas y facilitar su manejo a usuarios que no tienen que estar familiarizados con las características de la infraestructura, y su integración en paquetes de software científico para astrofísica.

Finalmente, se ha puesto de manifiesto la utilidad del método y la herramienta que se propone en esta Tesis a través de su uso en aplicaciones astrofísicas con distinto perfil en cuanto a su carga de trabajo. En algunos de los capítulos de esta memoria se ponen de manifiesto los beneficios que han proporcionado las herramientas desarrolladas a lo largo de nuestro trabajo en esta Tesis en cuanto a mejorar la capacidad para realizar análisis de datos y evaluar de modelos en astrofísica dentro de unos límites de tiempo dados.

Estructura de la Tesis

Esta memoria representa el resultado de más de cuatro años de investigación llevada a cabo en el Instituto de Astrofísica de Andalucía y concretamente en dos de sus departamentos, primeramente en el Centro de Cálculo y a continuación en el departamento de Física Estelar. Este trabajo está dividido en cuatro capítulos:

Capítulo 1. La capacidad de cómputo Grid en Astronomía pone en contexto el trabajo de Tesis, tanto en computación distribuida como en las herramientas de adaptación a campos científicos. Además presentamos problemas planteados en la astronomía donde se necesita adaptar aplicaciones a plataformas Grid.

Capítulo 2. Adaptación de aplicaciones astronómicas a una plataforma Grid, muestra los distintos métodos usados para portar código a la plataforma Grid. Además se explica el novedoso método realizado en esta Tesis para optimizar este cometido. Presentamos varios casos donde se aplica este método, analizando exhaustivamente las soluciones tomadas, y las nuevas aportaciones que se han obtenido a partir de este trabajo.

Capítulo 3. GSG: Herramientas para Grid en la astronomía, propone un conjunto de herramientas para el uso de infraestructuras Grid en diversas aplicaciones astronómicas. Además, se presentan las soluciones aportadas para mejorar la accesibilidad a la infraestructura, su uso eficiente y las cuestiones relacionadas con la seguridad en el uso de estas plataformas. Además, se han especificando las pruebas realizadas para la evaluación de la propuesta enfrentado nuestra solución a otras alternativas.

Capítulo 4. ATILA: Plataforma para la Computación Astrosismológica, describe la adaptabilidad de las herramientas GSG al servidor de aplicaciones ATILA que permite, entre otras cosas, la conectividad con el Observatorio Virtual, considerando su diseño para el uso de multiplataformas distribuidas y su implementación modular, haciendo hincapié en las mejoras realizadas por las herramientas GSG descritas en el capítulo anterior. En este capítulo también se detalla como se ha modularizado el código para permitir ampliar el uso de esta herramienta a nuevas aplicaciones, y se enumeran las distintas estrategias tomadas para añadir nuevas funciones como la conectividad con el Observatorio Virtual, o la creación de un entorno gráfico para facilitar su uso.

Capítulo 5. Conclusiones, aportaciones y líneas futuras, enumera las aportaciones realizadas a lo largo del trabajo de investigación y propone el trabajo futuro a desarrollar en esta línea de investigación.

Capítulo 1:
La computación Grid en Astronomía.

En el mundo de la astronomía, los nuevos proyectos que aprovechan numerosas innovaciones tecnológicas en satélites, observatorios o instrumentación astronómica, multiplican los datos que se deben analizar, contribuyendo al interés en la mejora del las prestaciones computacionales.

Para conseguir este objetivo en una infraestructura distribuida es necesario que ésta disponga de una estructura institucional que le permita definir estándares y coordine los servicios y las acciones de los distintos grupos que acceden a ella.

Es esta estructura la que define los principales elementos de una plataforma Grid que se describen la Sección 1.2 del presente capítulo para facilitar la comprensión de la memoria de la Tesis. Describimos los componentes que definen la arquitectura Grid (Sección 1.2.2), el método estandarizado de funcionamiento (Sección 1.2.3), la seguridad necesaria para el uso de la plataforma (Sección 1.2.4) y los componentes de la correspondiente capa del middleware (Sección 1.2.5).

Tras presentar la arquitectura y la bases del funcionamiento de una infraestructura Grid, se enumeran los beneficios que aporta el uso de la computación distribuida en el marco de la astronomía (Sección 1.3), además de analizar el trabajo de investigación realizado hasta el momento para mejorar las prestaciones de cómputo obtenidas mediante el sistema distribuido Grid.

Después, se describen los trabajos de investigación más relevantes realizados para optimizar el uso de la computación distribuida dentro del marco astrofísico (Sección 1.3.1), centrándose en la paralelización de trabajos, y se analizan las líneas que se están explorando para la resolución de problemas computacionales en astronomía.

Finalmente, el capítulo concluye con las consideraciones relativas a la necesidad del uso de una plataforma distribuida (Sección 1.4), como puede ser Grid, en el ámbito de la astronomía para hacer posible una investigación efectiva y provechosa en este área.

1.1 Necesidad de cómputo en la Astronomía.

Las necesidades de altas prestaciones de cómputo y de una elevada capacidad de almacenamiento de datos en la comunidad científica de astronomía son debidas al incremento de los datos de alta precisión obtenidos por nuevos instrumentos desarrollados por la industria aeroespacial, complementados por diversas infraestructuras ubicadas en Tierra. Por otro lado, la automatización de métodos científicos es esencial, ya que la complejidad de los desafíos científicos aumentan y, por tanto, crece rápidamente tanto los volúmenes de datos como la necesidad de colaboración interdisciplinaria .

Estos requerimientos han promovido la búsqueda y la investigación de nuevas soluciones computacionales que difieren de las que han sido usuales en este ámbito.

Junto a esta circunstancia, los nuevos proyectos que aprovechan las numerosas innovaciones tecnológicas en satélites, observatorios e instrumentación astronómica, deben analizar volúmenes de datos cada vez mayores.

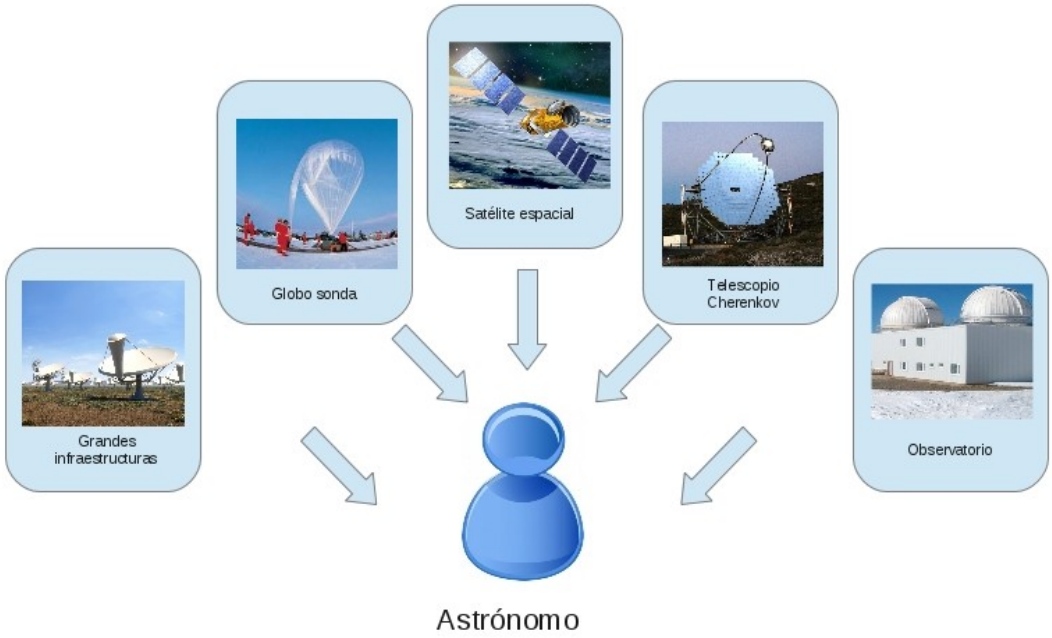


Figura 1.1: Colección de instrumentos astronómicos para producción de datos. Imágenes de izquierda a derecha: Radiotelescopios de SKA, Globo sonda de la misión Sunrise. Satélite espacial CoRot. Telescopio Cherenkov MAGIC. Observatorio de Sierra Nevada.

Como se ve en la Figura 1.1 en la colección de instrumentos se encuentran:

- Los observatorios terrestres, como el observatorio de Sierra Nevada [OSN14] (por citar de donde provienen la mayoría de los datos utilizados en está Tesis).
- Los globos sonda meteorológicos o estratosféricos (como los usados en el proyecto Sunrise [BAR11] para el estudio del Magnetismo del Sol, en el cual es partícipe el Instituto de Astrofísica de Granada (IAA)).
- Los telescopios convencionales, radiotelescopios y telescopios Cherenkov (como es el caso de los telescopios Cherenkov de proyecto MAGIC [ELS05] ubicados en el Roque de los Muchachos en la isla de La Palma y usados para la detección de Rayos gamma.)

- Las grandes estructuras distribuidas de investigación astrofísica cuyo principal exponente es el Square Kilometre Array (SKA) [JOS08] al ser el mayor radioscopio jamás desarrollado hasta el momento con miles de antenas repartidas en dos continentes, que necesitan sistemas de cómputo de gran factor de crecimiento de la carga de trabajo para procesar enormes cantidades de datos (del orden de los exabytes)
- Los satélites (orbitales, científicos y de propósitos experimentales, etc) como es el caso de satélite CoRoT [AUV09] usado para el estudio de la astrosismología y para la búsqueda de planetas extrasolares.

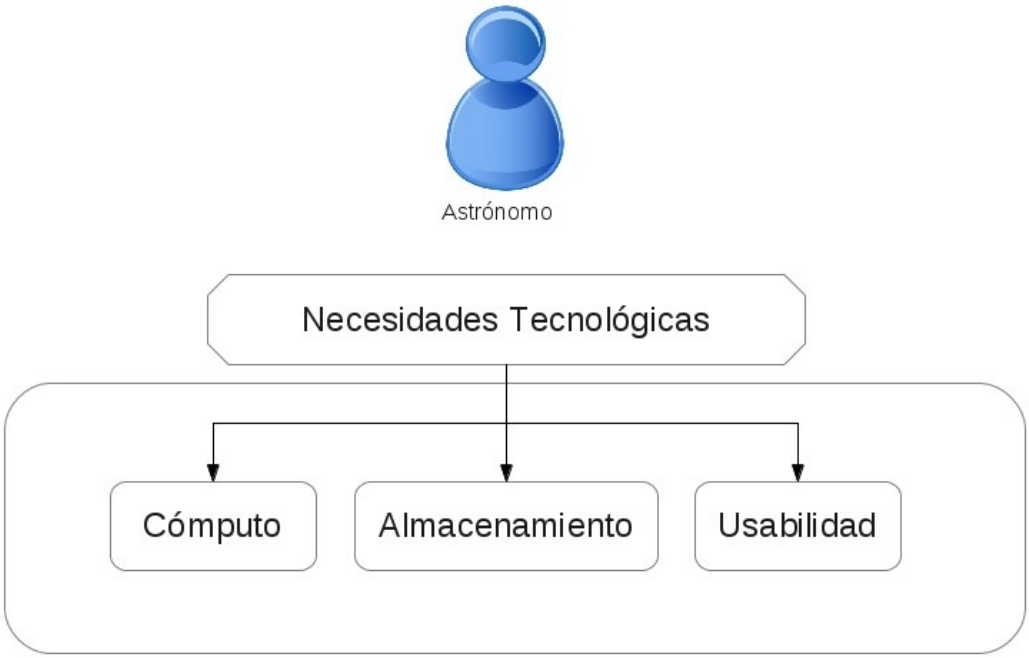


Figura 1.2: Necesidades tecnológicas en el campo de la astronomía.

Una vez obtenidos los resultados mediante los distintos tipos de instrumentación, llamados datos en crudo, las aplicaciones científicas usualmente presentan una serie de requisitos, como se observa en la Figura 1.2, que se pueden englobar en tres apartados,

- *Requisitos de cómputo*, la explotación de esta enorme cantidad de información requiere ejecutar algoritmos de análisis de datos y modelos teóricos, con un coste computacional elevado. La investigación en “Big Data” se reconoce como una importante necesidad en estos casos.

- *Requisitos de almacenamiento*, por la obligatoriedad de reunir todos los datos generados por los instrumentos astronómicos se incrementan exponencialmente cada vez que avanza la tecnología. El almacenamiento en grandes salas para alojar equipos ya no es posible en la mayoría de los proyectos. Por eso existe una tendencia a utilizar datos almacenados en infraestructuras distribuidas.
- *Requisitos de usabilidad*, ya que la comunidad astronómica es cada vez más dependiente de las TIC para comunicarse, consultar bases de datos, manipular equipos, como apoyo a la labor docente, o como complemento en las investigaciones. Las herramientas de cómputo deben ser de fácil manejo para usuarios que carecen del conocimiento técnico necesario para aprovechar convenientemente infraestructuras de cómputo avanzadas.

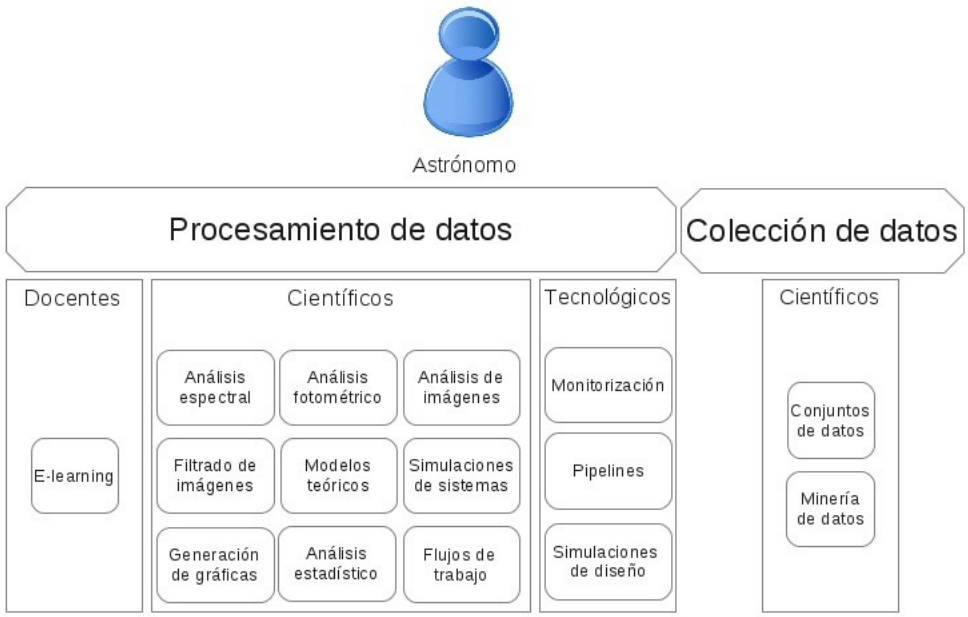


Figura 1.3: Clasificación de la naturaleza del trabajo a ejecutar en la astronomía.

Existen bastantes problemas que aparecen en las aplicaciones propias de la astronomía (Figura 1.3) que pueden beneficiarse de disponer de plataformas con capacidad de cómputo masiva. A continuación se describe cada una de las necesidades, analizándolas con más detalle.

- *El análisis espectral o espectroscopía* [BEL07]. Se basa en detectar la absorción o emisión de radiación electromagnética a ciertas longitudes de onda que se relacionan con los niveles de energía implicados en una transición cuántica. En el caso de las astronomía el objeto de estudio es el espectro de la radiación electromagnética, incluida la luz visible, que se radia desde estrellas y otros objetos celestes. La espectroscopia se puede usar para averiguar muchas propiedades de estrellas y galaxias distantes, tales como su composición química y movimiento (efecto Doppler).
- *El análisis fotométrico* [JOH53]. Es la rama de la astronomía que se dedica a medir el brillo de los diferentes astros: estrellas, planetas, satélites, asteroides, cometas, etc. Para ello se utiliza una factor de crecimiento de la carga de trabajo de brillos de las estrellas dividida en cinco magnitudes, además de incluir los astros telescópicos, invisibles al ojo humano por su extrema debilidad.
- *La reducción y el análisis de imágenes astronómicas.* La aplicación IRAF (Image Reduction and Analysis Facility) [DOU86] se basa en este tipo de herramientas.
- *El procesamiento y el filtrado de imágenes astronómicas*, consiste en un conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad, o facilitar la búsqueda de información. Mediante el filtrado se consigue obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica. Éste es el caso de herramientas usadas por el proyecto Tea-Is [TEA14] para la observación de plasmas y descargas eléctricas que se producen dentro de atmósferas planetarias.
- *La generación de modelos teórico*s relacionados con la astronomía [WOO78], basados en la investigación teórica, para ser comparados con observaciones del campo experimental. Así, el trabajo teórico es confrontado con los datos experimentalmente observados complementando así el desarrollo experimental. En la mayoría de los casos es necesario computación de alto rendimiento para generación de multitud de modelos basados en suficientes parámetros como para llevar a cabo este cometido.
- *Las simulaciones de sistemas complejos* [KUL15], que producen enormes cantidades de información de diferentes grupos, códigos, aproximaciones numéricas , física, etc, que son difíciles de manipular, analizar, extraer y publicar.

- *El análisis estadístico de resultados,* para el estudio de la recolección, análisis e interpretación de datos de una muestra representativa, ya sea para ayudar en la toma de decisiones o para explicar condiciones regulares o irregulares de algún fenómeno o estudio aplicado, de ocurrencia en forma aleatoria o condicional. En la astronomía se usa, tanto para la descripción, la visualización y el resumen de datos originados a partir de los fenómenos astronómicos, como para generar modelos, inferencias y predicciones asociadas a éstos.

- *La generación de gráficas*, para la visualización general de los datos obtenidos o de los modelos generados. Es una herramienta muy útil para la comparación de resultados en todos los ámbitos de la astronomía.

- *La utilización de flujos de trabajo en astronomía.* Para describir las actividades y los cálculos que se plantean en la resolución de problemas de astronomía. Es usado como medio para especificar y desarrollar los experimentos y será cada vez más necesario dado que en un futuro estaremos frente a una nueva generación de instalaciones y archivos relacionados con grandes cantidades de datos (ALMA [WOO03], LSST [TYS02], Pan-Starrs [KAI02], LOFAR [ROT03], SKA [JOS08]), donde los flujos de trabajo astronómicos juegan un papel importante en el método de trabajo de los astrónomos [SCH12].

Por otro lado, no solamente es necesario procesar los datos obtenidos por los distintos instrumentos de astronomía, sino que, además, la capacidad de cómputo también permite el manejo de estos. A continuación se detallan algunas tareas que, con esta finalidad, requieren de capacidad de cómputo y almacenamiento.

- *Monitorización de instrumentación astronómica.* Abarca la vigilancia de todos los servicios activos que cada instrumento ofrece. Este tarea puede consistir tanto en la vigilancia del entorno donde se encuentra el propio instrumento, como la supervisión del entorno externo, como puede ser red de conexión, almacenamiento, etc, donde está ubicado el instrumento. Tareas como la monitorización de telescopios, instrumentos a bordo de un satélite, o de un vehículo espacial se engloban dentro de esta categoría. Esta tarea se encarga del envío de alertas tan pronto como se detecta algún error. Sistemas de monitorización como Nagios [NAG14], orientado a redes, o Ganglia Monitoring System [MAS12], especializado en hardware, están muy extendidos en el campo de la astronomía. El proyecto PlanetLab [CHU03] es un ejemplo de este uso.

- L*a ejecución de pipelines para la astronomía.* Se trata del flujo de datos en un proceso con varias fases secuenciales, siendo la entrada de cada una la salida de la anterior. Uno de los casos de instrumentación astronómica tratados en esta memoria utiliza este tipo de arquitectura en *pipelines*. Se trata del proyecto "PAnoramic Near-Infrared Camera" (PANIC) [BAU08] para hacer un mapa astronómico de galaxias cercanas, entre otros fines.

- *Las simulaciones del diseño del instrumental para la astronomía.* Se trata de poner a punto la tecnología necesaria para poder disponer de los instrumentos de manera fiable y efectiva. Uno de los casos más relacionado con el trabajo desarrollado en esta Tesis es el montaje del Wind Evaluation Breadboard (WEB) [NUN08] del futuro Telescopio Gigante Europeo E-ELT (European Extremely Large Telescope) [COM11]. Este montaje necesita del diseño de un simulador para el espejo segmentado y todo el sistema de control, que requiere una gran capacidad de computo y almacenamiento.

Otras tareas que realiza la comunidad astronómica, no centradas en los temas propios de la investigación en la astronomía o en la instrumentación, pero que también están relacionados con la capacidad de cómputo y el almacenamiento son las siguientes:

- L*a comunicación docente mediante el E-learning* [MAR06], que permite la educación a distancia completamente virtualizada a través de los nuevos canales electrónicos, utilizando para ello herramientas o aplicaciones que usan plataformas digitales de formación como soporte de los procesos de enseñanza-aprendizaje.

La gran mayoría de grupos de trabajo usan el análisis de datos que requiere, no sólo mayor capacidad de cómputo para obtener el análisis en menor tiempo, sino un almacenamiento masivo para los datos analizados y sus resultados. Los casos más frecuentes son los siguientes:

- *La publicación de conjuntos de datos*, usados en un experimento astrofísico son clave para su reproducibilidad, y cada vez son más las leyes públicas y normas de revistas científicas que obligan a hacerlos públicos para evitar sesgos. Cuando estos conjuntos de datos tienen una extensión considerable, se necesita la ayuda de infraestructuras para su almacenamiento. Un caso estudiado en esta Tesis a este respecto es el uso de una infraestructura Grid para el almacenamiento de una base de datos sintéticos, “Spectral Energy Distributions” (SED) [ROB06], generados por códigos de transferencia radiativa [OSO03]. Gracias a la plataforma diseñada es posible analizar los datos recogidos por las grandes instalaciones internacionales, como puede ser el Observatorio espacial Herschel (HSO, por sus siglas en inglés) [PIL10], o el gran conjunto milimétrico de Atacama (ALMA, por sus siglas en inglés) [WOO03].
- *La minería de datos*, es una tarea muy relacionada con el punto anterior, dado que trata de descubrir patrones en grandes volúmenes de conjuntos de datos. Por tanto, busca extraer información del conjunto de datos y generar una estructura comprensible para su uso posterior. Los datos obtenidos en crudo por los distintos instrumentos deben procesarse para la obtención de información útil.

Diversos campos en la astronomía se han beneficiado de uso de infraestructuras computacionales distribuidas como puede ser la astronomía extragaláctica, el campo de la física estelar, la radioastronomía, el análisis de la estructura galáctica, o estudios relacionados con el sistema solar, entre otros muchos.

1.2 La infraestructura Grid

Para satisfacer las necesidades del campo de la astronomía existen diversas alternativas, con sus ventajas e inconvenientes. Hay alternativas como servidores de cómputo de altas prestaciones (supercomputadores) [LEB85], o sistemas constituidos a partir de servidores de computo u otros servidores dedicados que pueden configurarse como una plataforma única a través de infraestructuras Grid [FOS02] o Cloud [HAT08] o de computación voluntaria [SAR99] .

Al buscar una infraestructura que permita una acceso transparente a recursos heterogéneos, tanto computacionales como de almacenamiento, distribuida a lo largo de todo el mundo, las plataformas de tipo Grid constituyen una buena opción. Esta el la principal diferencia de las plataformas Grid frente a la arquitectura Cloud, que se presenta como un modelo centralizado con unos recursos que suelen ser de un sólo propietario.

Ademas, este tipo de plataformas permiten compartir y utilizar recursos de forma coordinada entre miles de usuarios, desde nuestro propio ordenador, y sin estar sujeto a un control centralizado debido a su carácter distribuido. De esta forma, proporcionan un mecanismo de colaboración transparente entre grupos de investigación que las hacen muy útiles para el campo de la astronomía.

Otra cualidad que respalda aun más el uso de este tipo de infraestructuras es la accesibilidad a los datos producidos desde cualquier lugar y en cualquier momento facilitando así la coordinación de grandes proyectos internacionales. Además, una infraestructura Grid, proporciona numerosos mecanismos de seguridad y autenticidad necesarios para hacerla un sistema fiable.

En las secciones siguientes (del 1.2.1 al 1.2.7), se describen las cualidades principales de las plataformas Grid.

1.2.1 Estructura institucional

En el mundo de la computación Grid existen numerosas organizaciones que administran y regulan el uso de la infraestructura. En esta sección se detalla el organigrama de las distintas entidades Grid con las que se ha interaccionado a lo largo de este trabajo.

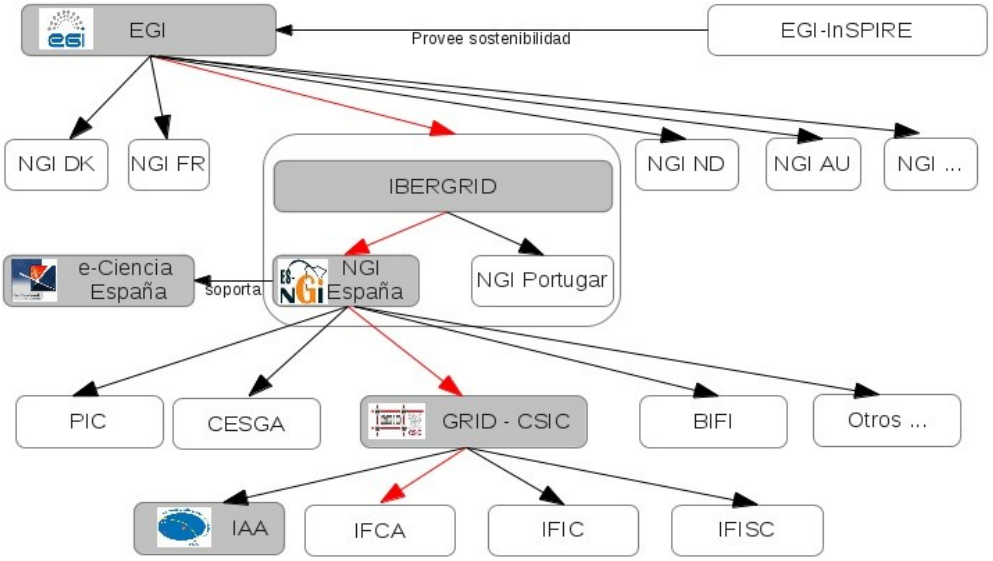


Figura 1.4: Organigrama de estructuras institucionales usadas en esta Tesis. EGI: Eurepean Grid Infrestructure. EGI-inSPIRE Integrated Sustainable Pan-European Infrastructure for Researchers in Europe. NGI: National Grid Infrastructure. PIC: Port d'Informacio Cientifica. CESGA: Centro Tecnolóxico de Supercomputación de Galicia. BIFI: Instituto de Biocomputación y Física de Sistemas Complejos, IAA: Instituto de Astrofísica de Andalucía, IFCA: Instituto de Física de Cantabria, IFIC: Instituto de Física Corpuscular. IFICS: Instituto de Física Interdisciplinar y Sistemas Complejos

Como se observa en la Figura 1.4, las distintas organizaciones que componen la infraestructura Grid utilizada tienen como raíz el proyecto **European Grid Infrastructure (EGI)** [EGI14], cuyo propósito principal es proporcionar acceso a una infraestructura de computación Grid distribuida geográficamente, disponible las 24 horas del día. Esta infraestructura está integrada por nodos computacionales de más de 50 países y proporciona el mantenimiento del middleware, asegurando el funcionamiento de una gran infraestructura de cómputo para el beneficio de una comunidad de investigadores amplia y diversa.

Como apoyo a esta infraestructura, el proyecto **EGI-InSPIRE** [EIN14] proporciona acceso unificado a las nuevas infraestructuras de computación distribuida, como el Cloud, las redes de supercomputación y las infraestructuras Grid, en beneficio de las comunidades de usuarios en el espacio europeo de investigación.

La infraestructura EGI está configurada como una federación de infraestructuras Grid nacionales. Como caso excepcional, por la cercanía y la conexión de la NGI española y la NGI portuguesa, nace la infraestructura **Ibergrid** [CAM09], cuyo objetivo principal es constituir un foro en el que los avances en el desarrollo de infraestructuras de red, tecnologías y aplicaciones puedan discutirse dentro de los principales actores de España y Portugal, pensando en futuros planes y actividades de integración en el marco de la e-ciencia europea e internacional.

En concreto, la **Iniciativa Grid Nacional Española (ES-NGI)** se compone de elementos de computación virtual y distribuida que, utilizando la tecnología Grid, permiten la interconexión de centros de recursos computacionales. Compartiendo estos recursos, y su gestión de forma conjunta y eficiente, se ofrece un servicio de soporte computacional a los investigadores, con consecuencias relevantes para la calidad de la I+D en España. Esta es la infraestructura en la que se basa la **e-Ciencia española** [ECI14] para resolver diversas tareas computacionales de forma que sea posible:

- Acceder a grandes bases de datos.
- Simular, procesar y analizar grandes colecciones de datos.
- Utilizar herramientas de colaboración (y de organización/gestión) .

De esta forma, es posible establecer un marco de colaboración entre las instituciones participantes para conseguir un desarrollo coordinado de la e-Infraestructura Grid en España. En esta linea, el Consejo Superior de Investigaciones Científicas (CSIC) desarrolla un proyecto en el que se engloba el trabajo realizado en esta Tesis. Este proyecto se denomina proyecto **Grid-CSIC**.

El proyecto Grid-CSIC está basado en la experiencia de varios centros del CSIC en la operación de infraestructuras de computación en un entorno Grid para dar soporte a líneas de investigación que requieren compartir información y recursos a través de la red de comunicaciones de la e-ciencia. Su principal objetivo es el empleo de sistemas distribuidos como un único sistema virtual, y la evolución de las redes de comunicaciones dedicadas a la investigación. Se trata de permitir el acceso a recursos geográficamente distribuidos, en diferentes dominios de administración, de forma transparente, segura, y fiable.

Un desafío importante que se plantea en este proyecto es mejorar la forma en que los investigadores utilizan los recursos, para que puedan realizar simulaciones más detalladas, procesar o transferir a velocidades elevadas grandes volúmenes de datos de modo interactivo, permitiéndoles abordar nuevos retos científicos en un contexto de colaboración internacional.

Alcanzar estos objetivos supone disponer de una potencia de cálculo elevada reduciendo costes y optimizando recursos, siendo capaces de atender picos de demanda de cálculo sin necesidad de adquirir nuevos recursos, amortizando y justificando los recursos a través de un uso más completo. Además, bajo este proyecto, se han creado varias Organizaciones Virtuales (VO) específicas dentro del Consejo Superior de Investigaciones Científicas, y se ha impulsado la red española de e-Ciencia, integrando a sus componentes dentro de la iniciativa Grid Europea (EGI), y se ha realizado proyectos de investigación que requieren capacidades que no están al alcance de un solo usuario o grupo de investigación, además de potenciar proyectos multidisciplinales en campos como,

• La simulación y análisis de datos en Física de Partículas

• La observación de modelos predictivos en Ciencias Medio Ambientales

• La Biología Computacional (Proteómica y Genómica)

• La Medicina (análisis de imágenes de diagnóstico)

• El análisis observacional en la Astronomía, que es el caso donde se ha centrado la realización de esta Tesis.

Todos los proyectos de investigación han requerido grandes recursos de cálculo, grandes bases de datos, y una colaboración distribuida a factor de crecimiento de la carga de trabajo global.

El Instituto de Astrofísica de Andalucía (IAA-CSIC) se unió al proyecto Grid-CSIC con el fin de proporcionar soporte de aplicaciones científicas en el área de astronomía. Para ello se le dotó de un nodo de computación Grid, siendo la plataforma usada en esta Tesis, permitiendo conseguir una disminución de tiempo total de ejecución de las aplicaciones gracias a la realización de numerosas herramientas que han facilitado su uso.

1.2.2 Arquitectura

A continuación se describen los principales plataformas que utilizamos en la Tesis y se propicia un acceso más fácil a los elementos de ésta y a la nomenclatura correspondiente [GLI14].

El acceso a la plataforma Grid se realiza desde una **Interfaz de Usuario (UI,** por sus siglas en inglés) instalada en el computador (PC) que utiliza el usuario. En ella se encuentra las cuentas personales y los certificados Grid requeridos de cada usuario. Desde una UI, un usuario puede recibir autenticación y autorización para usar los recursos Grid como pueden ser los sistemas de Información y de gestión computacional y de almacenamiento.

Un **elemento de computación Grid (CE,** por sus siglas en inglés) engloba recursos computacionales localizados físicamente en un lugar. Básicamente se trata de un servidor que puede estar implementado según distintos tipos de arquitecturas y posee código específico para incorporar sus recursos a la infraestructura Grid a la que pertenece.

Cada CE contiene un **sistema de administración de recursos locales (LRMS,** por sus siglas en inglés) que, para esta Tesis, soporta el sistema Maiu/Torque [STP06].

Usualmente, el CE contiene un conjunto de Nodos de Trabajo **(WNs,** por sus siglas en inglés) donde se ejecutan los trabajos enviados a la plataforma Grid. Estos elementos contienen los correspondientes a los comandos de gestión de trabajos y a las aplicaciones específicas de una Organización Virtual (VO) instaladas en un sistema de archivos compartidos accesible para cualquier usuario de la VO.

Para administrar los trabajos enviados al Grid se necesita un **sistema de gestión de carga de Trabajo (WMS,** por sus siglas en inglés) que se encarga de aceptar trabajos de usuario, asignarlos al CE más apropiado, registrando su estado y recuperando su salida. Este servicio se encuentra en servidores externos de uso centralizado para los usuarios de una determinada VO.

Los trabajos que se envían se describen usando el **Lenguaje de Descripción de Trabajo (JDL,** por sus siglas en inglés) que especifica, por ejemplo, que código ejecutar y sus parámetros, los archivos a trasladar al WN y desde el mismo, los archivos de entrada Grid que se necesitan, y todos aquellos requisitos en el CE y el WN. Un ejemplo de un código JDL se muestra a continuación:

```
Type = Job;
JobType = Normal;
Executable = start.sh;
Arguments = test.tar.gz \home;
StdError = test.err;
StdOutput = test.out;
OutputSandbox = {test.err, test.out};
Requirements = (RegExp(iaa, other.GlueCEUniqueID));
```

Código 1.1: Ejemplo de un código de Lenguaje de Descripción de Trabajo de Grid.

Otro elemento principal de una estructura Grid es el **elemento de almacenamiento (SE,** por sus siglas en inglés), que permite controlar servicios de sistema de almacenamiento masivo proporcionado el acceso uniforme a los recursos de almacenamiento. Todos los datos en un SE se consideran temporales y el usuario es responsable de transferir la información relevante a servidores específicos.

Para la gestión de los datos en un entorno Grid, los archivos pueden tener réplicas en muchos sitios diferentes. Lo ideal es que los usuarios no tenga que conocer la ubicación de un archivo, y pueden utilizar nombres lógicos para los archivos que los servicios de gestión de datos usarán para ubicarlos y acceder a ellos.

En Grid, se puede hacer referencia a los archivos mediante distintos nombres:

- **Grid Unique Identifier (GUID),**
- **Nombre Lógico de Archivo (LFN)**
- **Storage URL (SURL)**
- **Transport URL (TURL).**

Mientras los GUIDs y los LFNs identifican un archivo sin tener en cuenta su ubicación, los SURLs y TURLs contienen información sobre el lugar donde se ubica una réplica física, y la manera como se puede acceder a ella.

Las correspondencias entre LFNs, GUIDs y SURLs se guardan en un servicio denominado **Catálogo de Archivos (LCG, por sus siglas en inglés)**, mientras que los archivos en sí mismos se almacenan en los distintos SE.

Como servicio auxiliar se tiene el **servicio de información (IS, por sus siglas en inglés)** sobre los recursos Grid y su estado. Esta información es esencial para el funcionamiento de todo el Grid, puesto que a través del IS se descubren los recursos disponibles.

Cada sitio, mediante el servicio de **índice de información de Base de Datos Berkeley (BDII, por sus siglas en inglés)**, recopila información sobre todos los recursos presentes en el mismo. Además el servicio **top-BDII,** a un nivel superior, recopila toda la información proveniente de cada uno de los sitios y los almacena en una base de datos permanente. El top-BDII se puede configurar para obtener la información publicada desde los recursos en todos los sitios, o sólo desde algunos de ellos. La información publicada también se utiliza, con propósitos de monitorización y contabilidad.

La **monitorización** de los parámetros relacionados con un recurso es una actividad necesaria en cualquier plataforma distribuida. En un sistema tan heterogéneo y complejo como un Grid, esta necesidad llega a ser fundamental. El sistema de monitorización deber ser capaz de recopilar datos de los recursos del sistema, con el fin de analizar el uso, comportamiento y rendimiento del Grid, detectar y notificar fallos, violaciones de uso, y eventos definidos del usuario. En el caso del nodo usado para la Tesis se ha utilizado **Nagios** [NAG14] tanto para la monitorización local (a nivel de IberGrid) como para la global (a nivel de EGI).

El **Portal de contabilidad de EGI (APEL, por sus siglas en inglés)** [ACC14] es el elemento que analiza los parámetros más relevantes tanto de los procesos productores como consumidores de computación ejecutados en una cada uno de los nodos de trabajo (WN). Este elemento interactúa con el servicio de información del nodo (IS) y con los procesos consumidores y productores.

Una vez definidos los principales componentes se presenta una serie de conceptos básicos necesarios para comprender una infraestructura Grid.

Los usuarios de una infraestructura Grid se dividen en **Organizaciones Virtuales** (VOs) [FOS01], entidades abstractas que agrupan usuarios, instituciones y recursos en el mismo dominio administrativo [DIM09].

La afiliación a una VO le otorga al usuario privilegios específicos. Así, un usuario que pertenezca a una determinada VO astronómica puede leer los archivos de esa organización, o explotar recursos reservados para sus colaboradores. Convertirse en miembro de una VO, por lo general, requiere ser miembro de la colaboración correspondiente. El usuario debe cumplir con las reglas de la VO para obtener la afiliación, y puede expulsarse de la misma si no cumple con estas reglas.

La **seguridad Grid** es otro de los pilares en el que se sustenta la plataforma. Para explicar el proceso de seguridad Grid se define una serie de elementos básicos:

- El elemento principal de la seguridad Grid es el **certificado Grid**. Estos certificados pueden estar asociados a un recurso Grid, para autorizarle a pertenecer a una determinada infraestructura Grid, y autentifican estos recursos ante usuarios y otros servicios. También pueden estar asociados a un usuario, al que autorizan a usar los servicios de la VO y enviar trabajos al Grid. Técnicamente son credenciales de tipo X509 [LIN08] y deben estar validadas por una Autoridad de Certificación (CA) reconocida por una infraestructura Grid. En el caso de la infraestructura IberGrid ésta es IRISGrid CA.
- Los **certificados proxy** de Grid son credenciales de usuario delegadas que autentifican al usuario en cada interacción, y que tienen una duración limitada, de tal forma que se evita tener que utilizar el propio certificado personal, lo cual podría comprometer la seguridad. Así, también se evita tener que dar la clave del certificado cada vez que se use.
- Para trabajos de tiempos de ejecución considerables, el *proxy* del trabajo puede expirar antes de que el trabajo haya finalizado, causando la interrupción del mismo. Para evitar que esto suceda, existe un mecanismo de renovación vía *proxy* que mantiene el *proxy* del trabajo válido tanto tiempo como sea necesario. El servidor **MyProxy** proporciona esta función.

- El certificado *proxy* de una VO se utiliza a través del **servicio de membresía de la organización Virtual (VOMS, por sus siglas en inglés),** que permite complementar un certificado *proxy* con extensiones que contienen información acerca de la VO, los grupos de la VO a los que pertenece el usuario, y el rol que tiene el usuario. Dentro de la terminología del VOMS, un *grupo* es un subconjunto de la VO que contiene los miembros que comparten algunas responsabilidades o privilegios en el proyecto. Los grupos están organizados jerárquicamente como un árbol de directorios, comenzando desde el grupo raíz de toda la VO. Un usuario puede ser miembro de varios grupos, y un *proxy* del VOMS contiene la lista de todos los grupos a los que el usuario pertenece. Cuando se crea el *proxy* del VOMS, el usuario puede escoger uno de estos grupos como grupo “principal”.

Tabla 1.1: Servicios Grid elementales utilizados por el nodo Grid del Instituto de Astrofísica de Andalucía.

Servicio	**Definición**	**Nombre**
UI	Interfaz de Usuario.	ui.iaa.es
CE	Elemento de computación.	ce.iaa.es
LRMS	Sistema de administración de recursos locales.	torque.iaa.es
WN	Nodos de trabajo.	wnXX.iaa.es
WMS	Sistema de gestión de carga de Trabajo.	gridxb01.ifca.es wms01.egee.cesga.es
SE	Elemento de almacenamiento.	se.iaa.es
LFC	Catálogo de archivos.	gridlfc01.ifca.es lfc01.ncg.ingrid.pt
BDII	Índice de información de Base de Datos Berkeley.	bdii.iaa.es
TOP-BDII	Índice de información de Base de Datos Berkeley de nivel superior.	gridii01.ifca.es bdii.egee.cesga.es
MS	Servicio de monitorización.	Nagios
APEL	Servicio de contabilidad.	apel.iaa.es
VO	Organización Virtual.	phys.vo.ibergrid.eu
Myproxy	*proxy*s largos para usuarios.	px01.ncg.ingrid.pt
VOMS	Servicio de membresía de la organización virtual.	https://voms01.ifca.es:8443/vomses https://voms01.ncg.ingrid.pt:8443/vomses https://voms.egee.cesga.es:8443/vomses

En la tabla 1.1 se proporciona un resumen de los servicios, que constituyen la arquitectura de la plataforma Grid en la que participa el Instituto de Astrofísica de Andalucía como nodo, y que será el nodo utilizado en esta Tesis.

1.2.3 Funcionamiento

Tabla 1.2: Descripción de los estados de un trabajo Grid.

Estado	Definición
SUBMITTED	El usuario ha enviado el trabajo pero el servidor de la red no lo ha aceptado todavía .
WAITING	El Servidor de la red ha aceptado el trabajo pero el WMS no lo ha procesado todavía.
READY	Se ha asignado el trabajo a un CE pero no se ha transferido a éste todavía .
SCHEDULED	El trabajo se ha transferido al CE y está en espera en la cola del CE
RUNNING	Se está ejecutando el trabajo en el WN.
DONE	Se ha finalizado el trabajo correctamente.
ABORTED	El WMS ha cancelado el trabajo (por ejemplo porque el trabajo era muy largo, o el certificado *proxy* expiró, etc).
CANCELED	El usuario ha cancelado el trabajo.
CLEARED	Se han transferido los resultados a la Interfaz de Usuario.

Un trabajo Grid puede encontrarse en distintos estados (Tabla 1.2).

Uso de Grid

Antes de enviarse un trabajo Grid se deben realizar, en el orden adecuado, unos pasos previos para preparar el entorno de computación.

En primer lugar, el usuario debe obtener un certificado Grid a través de una Autoridad de Certificación (CA) reconocida por la infraestructura Grid que va a usar. Seguidamente debe solicitar su incorporación en una Organización Virtual adecuada para sus objetivos. Una vez admitido por el administrador de la VO correspondiente, debe crear una cuenta en la máquina donde se encuentre el UI. Estos pasos solo tienen que ejecutarse una vez para tener acceso al Grid.

Después, hay que crear un certificado *proxy* mediante el servicio VOMS o servicios de *proxy* de larga duración usando el servicio My*proxy*. Este paso debe ejecutarse la primera vez que se envía una solicitud (trabajo, consulta, archivo, etc) al Grid. Y genera un *proxy* válido durante un cierto periodo de tiempo (usualmente 12 horas) para reducir los riesgos de seguridad ante posibles plagios. Al expirar el *proxy*, se debe crear uno nuevo antes de que puedan volverse a utilizar los servicios Grid otra vez.

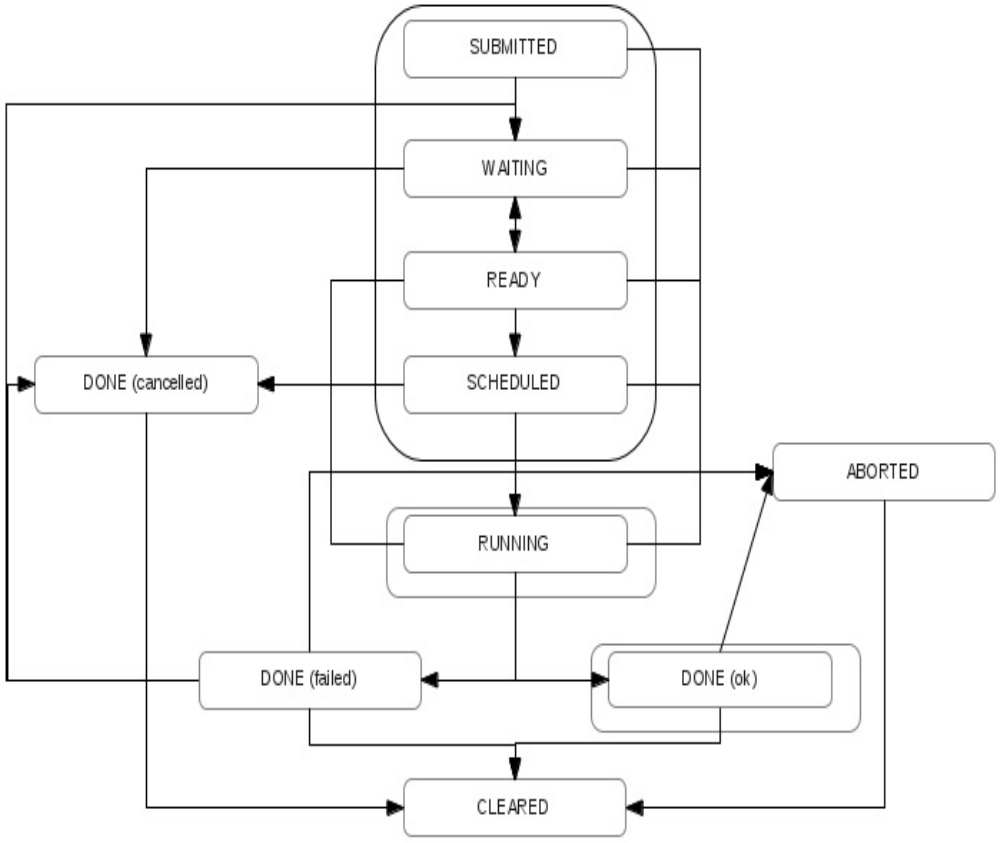


Figura 1.5: Ciclo de vida de un trabajo enviado a una plataforma Grid.

Una vez completados los pasos previos ya puede proceder al envío de un trabajo Grid. Durante su ciclo de vida, un trabajo Grid puede pasar por algunos de los estados de la Tabla 1.2. A continuación se resume el funcionamiento de una plataforma Grid tras el envío de un trabajo, tal y como se esquematiza en la Figura 1.5.

El usuario envía un trabajo Grid desde la UI a un WMS. En el archivo de descripción de trabajo (JDL) se pueden especificar los archivos que se vayan a copiar desde la UI al WN. Este conjunto de archivos se denomina Input Sandbox. Posteriormente se pasa al estado SUBMITTED.

El WMS busca el mejor CE disponible para ejecutar el trabajo. Con este fin, interroga al BDII para consultar el estado de los recursos de cómputo y de almacenamiento, y al Catálogo de Archivos Lógicos (LFC) para encontrar la ubicación de todos aquellos archivos de entrada requeridos. El trabajo pasa al estado WAITING.

El WMS prepara el trabajo para su envío, creando un *script* que se transfiere, junto con otros parámetros, al CE escogido. El trabajo pasa al estado READY.

El CE recibe la solicitud y envía el trabajo al LRMS local para su ejecución. Después, el trabajo pasa al estado SCHEDULED.

El LRMS se encarga de la ejecución del trabajo en los WNs que usa el recurso local disponible. Se copian los archivos del usuario del WMS al WN donde se ejecuta el trabajo y pasa al estado RUNNING. Mientras el trabajo se ejecuta, se puede acceder directamente a los archivos Grid desde un SE, después de copiarlos al sistema de archivos local en los WN mediante las herramientas para la gestión de datos.

El trabajo puede producir nuevos archivos de salida que pueden transferirse al Grid y ponerse a disposición de otros usuarios Grid. Transferir un archivo al Grid significa copiarlo en un SE, y registrarlo en un LFC.

Si el trabajo finaliza sin errores, los ficheros de salida, definidos en el sistema Sandbox, se transfieren nuevamente al nodo del WMS. El trabajo pasa al estado DONE.

En este momento, el usuario puede recuperar la salida de su trabajo de la UI y el trabajo pasa al estado CLEARED.

A lo largo de todo el proceso se puede consultar el estado del trabajo a través de la UI. Asimismo, es posible consultar el BDII desde al UI, para conocer el estado de los recursos.

Si el sitio al que se envía el trabajo no puede aceptarlo o ejecutarlo, el trabajo puede reenviarse automáticamente a otro CE que satisfaga los requisitos del usuario. Después de alcanzar el máximo número permitido de reenvíos, el trabajo se marcará como cancelado.

El estado de trabajo Grid denominado ABORTED implica que existe un problema en el proceso de ejecución del trabajo en el Grid. Estos errores pueden producirse debido a equivocaciones del usuario, desconfiguraciones de los componentes del Grid, fallos en el hardware o en la red, o incluso a errores en el *middleware*.

1.2.4 Seguridad

Como ya se definió en la Sección 1.2.2, dedicado a la arquitectura Grid, existen una serie de elementos que permiten que una infraestructura Grid tenga un protocolo de seguridad extenso cuyos detalles más importantes describimos a continuación.

Antes de que los recursos de una infraestructura Grid se puedan utilizar, un usuario debe leer y aceptar las reglas de uso (junto con otras reglas adicionales que la VO desee incluir) y registrar algunos datos personales en un servicio de registro.

Con el fin de identificarse para utilizar los recursos Grid, un usuario necesita tener el correspondiente certificado de usuario Grid (protegido por una clave), que se utiliza para generar y firmar un certificado temporal, denominado certificado *proxy* (o simplemente un *proxy*).

Un usuario necesita un *proxy* válido para enviar trabajos. Esos trabajos conservan sus propias copias del *proxy* para poder identificarse con los servicios Grid mientras se ejecutan.

1.2.5 Middleware

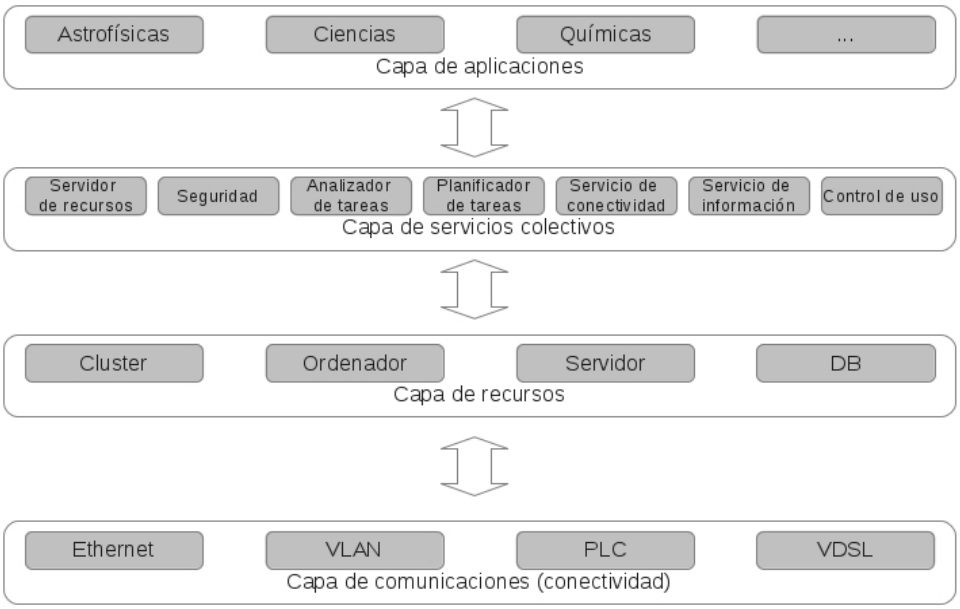


Figura 1.6: Capas de una infraestructura Grid.

El *middleware* es la capa de servicios que organiza e integra los recursos en una infraestructura Grid [BER03]. Está compuesto por múltiples módulos, y contiene cientos de miles de líneas de código. Una vez configurado, este código automatiza todas las interacciones entre los distintos elementos del Grid. Como se observa en la Figura 1.6, el *middleware* incluye una capa de protocolos de recursos y conectividad, y una capa superior de servicios colectivos.

Los protocolos de recursos y conectividad manejan todas las transacciones de red específicas para Grid, que tienen lugar entre los computadores y los recursos de la plataforma Grid. Esto se hace mediante protocolos que permiten a los elementos comunicarse unos con otros, haciendo posible el intercambio de datos, y protocolos de autenticación, que proporcionan mecanismos seguros para verificar la identidad de los usuarios y los recursos.

Los servicios colectivos incluyen:

- Directorios de actualización de recursos disponibles.
- Gestores de recursos de capacidad de cómputo y almacenamiento.
- Recursos de monitorización y diagnóstico de problemas.
- Réplicas de datos para que múltiples copias puedan estar disponibles en diferentes localizaciones para facilitar su uso.
- Servicios de membresía para saber a quién se le ha permitido hacer qué y cuándo.

Los servicios colectivos también funcionan a través de protocolos. Por un lado están los protocolos que obtienen información acerca de la estructura y el estado de los recursos en la Grid y por otro lado los protocolos de manejo, que negocian el acceso a los recursos de un modo uniforme.

Dentro de la Iniciativa Grid Europea (EGI) se presentan tres estándares de *middleware* que incluyen todos los servicios propios de un *middleware.* Estos son:

- El middleware Conector de Recursos Avanzado (**ARC,** en su siglas en inglés) [ELL07], coordinado por NorduGrid [ELL02], y con aportaciones procedentes de varios proyectos.
- El interfaz uniforme para recursos informáticos (**UNICORE,** en su siglas en inglés) [ERW02] desarrollado a través de dos proyectos financiados por el ministerio alemán de educación e investigación (BMBF) .
- El middleware **gLite** [KUN05], utilizado en los experimentos del LHC del CERN y en otras aplicaciones y proyectos científicos. Fue implementado a partir de la colaboración de más de 80 personas en 12 diferentes centros de investigación académica e industrial en Europa. Este middleware es el que se ha utilizado para el trabajo de esta Tesis debido a que es el middleware que soporta la organización virtual IberGrid.

- Finalmente, tras un esfuerzo para unificar los tres estándares, se ha creado la Iniciativa Middleware Europea (**EMI**, en sus siglas en inglés) [AIF12]. Es la base para los middleware Grid descritos y está siendo ampliamente utilizado por las comunidades de investigación científica e infraestructuras de computación distribuida en todo el mundo. El propósito de EMI es consolidar, sincronizar y apoyar las plataformas de middleware originales, evolucionándolas y extendiéndolas. Elimina los servicios redundantes o duplicados resultantes de la fusión, en favor de nuevos servicios creados para satisfacer los requerimientos del usuario o las necesidades específicas de consolidación, normalización y desarrollo de interfaces. Estas necesidades incluyen la adopción de una estructura común para la contabilidad, el intercambio de información de recursos, o la autenticación y autorización.

1.2.6 Alternativas al uso del middleware.

La capa *middleware* de Grid es a fin de cuentas, la forma estandarizada para el uso de los servicios del Grid. No obstante, debido a ciertas carencias y otros problemas, han aparecido distintos métodos de uso de una plataforma Grid alternativos o complementarios al uso del *middleware.* Hasta el momento, existen diferentes técnicas para adaptar las aplicaciones a un entorno de computación distribuida, aunque no siempre satisfacen las necesidades requeridas. Las alternativas principales se han dividido en las siguientes categorías:

- Máquinas virtuales.
- Paradigma MapReduce
- Metaplanificadores.
- Simuladores Grid.
- Paquetes de herramientas.

Una alternativa al uso del *middleware* es la **Virtualización**. Constituye una de las técnicas más usadas para la adaptación de aplicaciones a una plataforma Grid [TAF12]. Ésta aporta nuevas posibilidades para la investigación astronómica, tanto en términos de provisión de recursos, intercambio de recursos y distribución de aplicaciones científicas y de mantenimiento, ofreciendo un conjunto eficiente, estable, y confiable de servicios Grid básicos. Algunos de los ejemplos de herramientas usadas para implementar la virtualización son KVM [KIV07], Xen [BAH03], VirtualBox [VIR14] o VMWare [MUL05].

La principal ventaja frente al uso de los servicios Grid mediante un *middleware* específico es que no conlleva un alto coste de implantación, ni la necesidad de una elevada experiencia para su mantenimiento, además de proporcionar un entorno portable y acotado. Es ideal para entornos de desarrollo y de pruebas para funciones de aplicaciones en Grid, ofreciendo la posibilidad de depurar el código antes de ejecutar la aplicación en un Grid real.

Pero también existen algunas desventajas que desaconsejan su uso. Principalmente es que el sistema principal que soporta a las máquinas virtuales debe disponer de una mayor cantidad y potencia de recursos frente a una arquitectura Grid. Además, existe la posibilidad de la aparición de problemas en la compatibilidad con el hardware virtualizado.

Otra elección es el modelo de programación **MapReduce** [DEA08], que se utiliza para dar soporte a la computación paralela sobre grandes colecciones de datos en redes de computadores. MapReduce se utiliza generalmente en la computación concurrente para que problemas con grandes conjuntos de datos puedan ser procesados por un gran número de equipos (nodos) en clusters, u otras plataformas paralelas o distribuidas y se suele utilizar sistema de archivos distribuidos (HDFS). Sin embargo, MapReduce no es la solución a cualquier problema, ya que es posible que el problema no se ajuste bien a un esquema de procesado de mapas y reducción.

Proyectos en el marco docente y formativo como **VGrid** [CON12] utilizan este método ya que el uso del *middleware* de Grid no suele estar disponible para satisfacer objetivos formativos, limitando el conocimiento de esta tecnología a los estudiantes únicamente como concepto teórico, y a lo sumo a la prueba de algunos comandos de ejecución y monitorización de trabajos simples.

Otra herramienta que usa esté método es **Gridseed** [GRE08] creado para el aprendizaje y montaje de una infraestructura Grid basada en máquinas virtuales. El principal problema es que montar un Grid virtual en una única máquina implica satisfacer muchos requisitos. Estas herramientas proponen la virtualización como medio para crear un Grid funcional que puede utilizarse en un computador individual, evitando los requisitos económicos y de gestión necesarios en un Grid real.

Esta arquitectura usa características autonómicas (basadas en información de retro-alimentacion) de forma que la entidad que realiza la meta-planificacion adapte su proceso de toma de decisiones para ajustarse mejor al estado del sistema.

Por otro lado, existen metaplanificadores que mediante el uso de las características del sistema obtenidas por retro-alimentación son capaces de adaptar un proceso de toma de decisiones, ajustándose mejor al estado del sistema. Un ejemplo de esta alternativa es el metaplanificador **Gridway** [HUE05] desarrollado mediante código abierto que permite compartir recursos computacionales a gran factor de crecimiento de la carga de trabajo, de manera fiable y eficiente. Además, permite administrar diferentes sistemas de gestión de recursos locales a través de uno o varios dominios administrativos. Así pues, gestiona de la ejecución de aplicaciones distribuidas y puede usarse junto al middleware de Grid, situándose en una capa superior a los servicios ofrecidos por ésta.

Otra técnica es la basada en simuladores Grid, pensados especialmente para uso docente y formativo y no científico como es el caso de **GridSim** [SUL08], que permite modelar recursos y planificar aplicaciones para entornos de computación paralelos y distribuidos como es el caso de Grid. El gran problema del uso de este tipo de aplicaciones es que todo el potencial de la capa *middleware* de Grid no puede simularse fácilmente. Por ejemplo, carece de la gestión del entorno, mecanismos de seguridad y certificación, control de recursos, información obtenida mediante monitorización, etc.

Para usar todos los servicios de una infraestructura Grid existen paquetes que contienen colecciones de módulos y herramientas. Esta técnica se basa en utilizar una recopilación de servicios, herramientas y APIs para facilitar el uso de aplicaciones en plataformas Grid. Esta colección de herramientas resultan útiles en entornos puramente de desarrollo y de producción. Por ejemplo, existe el paquete **ISENGARD** [KUR11] que provee servicios, herramientas y APIs para simplificar la ejecución del software en la plataforma Grid. En cambio, por otro lado, oculta los servicios de gestión y de mantenimiento del Grid.

Una vez analizadas las diferentes alternativas al middleware llegamos a la conclusión que el uso de éstas mejoran ciertos campos para la ejecución de aplicaciones en Grid como son la implantación del software, el mantenimiento de servicios, la portabilidad o el uso efectivo de la paralelización. Pero creemos que estas mejoras no son suficientes como para potenciar al máximo su uso. Por eso, como trabajo incluido en esta Tesis, hemos creado el paquete de herramientas **GSG** [ROD11], descrito en el Capítulo 3 que, además de incluir las facilidades de las demás alternativas, añade una serie de módulos que homogeneizan y facilitan la usabilidad, la reutilización de código y la administración de una infraestructura Grid. Además favorecen la gestión, el control y la seguridad de este tipo de plataformas.

1.2.7 Herramientas para uso intensivo en Grid

En la actualidad se cuenta con varias herramientas para mejorar la uso y la gestión de grandes cantidades de datos en una plataforma Grid implementadas para el uso específico en proyecto de grupos de investigación. Hay un listado amplio de ellas, como por ejemplo las herramientas DIRAC [TSA10] inicialmente orientada para el instrumento LCHb [ALV08] del LHC (Large Hadron Collider) [EVA08], desarrollada en el CERN; GANGA [MOS09] cuya API se ha desarrollado para la comunidad de física de partículas; y APS [RYB12] que abastece las necesidades de las aplicaciones del instrumento ATLAS [AAD08]. Además existen otras herramientas de propósito general como SAM [KUL09], que utiliza una arquitectura basada en servicios además del paquete de herramientas Gscript [MAL08] desarrollado en Ruby [RUB14].

En el campo de la astrosismología se encuentra la herramienta Asteroseismic Modeling Portal (AMC) [MET12], y como ejemplo de interfaz para el uso de códigos astrosismológicos, ASTEC [CHR08], que usa algoritmos genéticos paralelizados.

1.3 Beneficios de Grid a la astronomía

Existen distintas plataformas que pueden satisfacer la necesidad de recursos en la investigación en astronomía, desde supercomputación, a configuraciones distribuidas de servicios como la plataforma Grid y la computación voluntaria.

Una infraestructura Grid puede dar soluciones a los problemas del campo de la astronomía, de ahí que muchos de los investigadores optan por usar esta tecnología. Además, gracias a las estructuras de Grid la comunidad astronómica, antes dispersa, tiene la posibilidad de unirse en el ámbito de las soluciones computacionales en una única Organización Virtual dedicada exclusivamente a su campo.

La tecnología Grid proporciona una computación distribuida que combina la potencia de muchas máquinas con la finalidad de obtener una capacidad ilimitada tanto de cómputo como de almacenamiento. Permite compartir recursos geográficamente distribuidos para resolver problemas a gran factor de crecimiento de la carga de trabajo de forma uniforme, segura, transparente, eficiente y fiable. Está destinada a ofrecer servicios en muchas de las líneas de trabajo de astronomía. Por ello, los científicos han adoptado la tecnología como una posible solución a sus necesidades.

Pero los científicos necesitan herramientas que les facilite abordar el proceso de adaptación al Grid de sus aplicaciones. El uso de una plataforma Grid tiene la gran ventaja de crear una red de usuarios que intercambian información y experiencias con el objetivo de adquirir la experiencia necesaria.

1.3.1 Uso del Grid en la Astronomía hoy

La plataforma Grid ha demostrado ser particularmente atractiva para el campo de la Astronomía y Astrofísica (A&A) [VIE08], ya que las aplicaciones típicas se consideran suficientemente complejas como para ser buenos candidatos a requerir del potencial de esta infraestructura.

A lo largo de la historia de una plataforma Grid los dos proyectos astronómicos más grandes e importantes son el proyecto Planck [TAF05] y el proyecto MAGIC [ELS05]. Ambos empezaron desde los inicios en Grid y fueron pioneros en el uso del Grid en la Astronomía.

La misión del satélite Planck es desarrollada por la Agencia espacial Europea (ESA) y su objetivo es detectar las anisotropías en el fondo cósmico de microondas en casi todo el cielo, con una resolución, una precisión, una estabilidad y sensibilidad sin precedentes. Uno de los cometidos principales para el centro de proceso de datos del proyecto es definir, diseñar y ejecutar una simulación completa de la misión Planck para probar el código de análisis de datos. El código de simulación debe imitar el procedimiento del satélite Planck. Las simulaciones de Planck conllevaron una alta carga computacional, absorbida por una infraestructura Grid.

El segundo proyecto se denomina MAGIC y se trata de un telescopio de rayos gamma por emisión de radiación Cherenkov en la atmósfera. Este telescopio es capaz de detectar los destellos de luz producidos en la atmósfera por rayos cósmicos. En 2008 se le unió un segundo telescopio, un clon del anterior, apodado MAGIC-II que operado junto al primero, mejora sustancialmente su resolución angular y su sensibilidad. Los datos obtenidos por estos telescopios es un territorio desconocido y presenta un reto totalmente nuevo ya que sus métodos de análisis requieren la adaptación a dominio de baja energía. El proyecto MAGIC empezó su uso en Grid en 2004 impulsado por la idea de un sistema de computación distribuida entre todos los colaboradores.

Tabla 1.3: Aplicaciones astronómicas utilizadas en una plataforma Grid.

Descripción	**Coordinación**
Simulaciones de Planck	INAF (IT).
Formación en la galaxia de alto Redshift: Cuantificación de la retroalimentación de agujeros negros en la evolución del Universo visible.	INAF (IT).
La base de datos de BaSTI.	INAF (IT).
MAGIC.	FZK (DE).
G-Eclipse.	FZK (DE).
SWIFT.	SAS (SK)
Refinamiento del escenario cosmológico de la formación del sistema solar.	SAS (SK).
Explicación de las estructuras observadas de lluvia de meteoritos a través de simulaciones de su evolución dinámica.	SAS (SK)
Resolución del problema de la migración de los cometas y de los asteroides en el sistema solar.	SAS (SK).
Evolución dinámica de partículas de polvo cósmico de forma irregular y compuesta.	SAS (SK).
Simulaciones hidrodinámicas.	UIBK (AT).
Análisis de surveys espectrales de las Misiones HERSCHEL/ALMA.	OBSP (FR).
Aplicaciones que forman parte del Proyecto Horizon.	OBSP (FR).
Mecanismos celestiales simulados y aplicados relacionados con el movimiento y el comportamiento dinamométrico de varios objetos del Sistema Solar.	OBSP (FR).
LOFAR.	RUG (NL).
Restauración de imágenes mediante la transformada de ondícula, considerando o no objetos individuales.	RUG (NL).
Detección de fuentes extragalácticas compactas y análisis de la gaussianidad de los datos de Planck.	IFCA(ES).
Aplicaciones de OrbFit.	IPB (SE)

Además de estos dos importantes proyectos, en la infraestructura Grid se han utilizado en otros proyectos de astronomía, en la tabla 1.3 se presentan varios ejemplos ya consolidados en una plataforma Grid.

A pesar de que el concepto de infraestructura Grid está ampliamente asimilado por el campo de la astronomía, aun quedan problemas abiertos como son la adaptación optimizada de aplicaciones científicas de manera automática, la usabilidad de la plataforma por personal científico sin conocimientos específicos en Grid, o la gestión eficiente de de los trabajos enviados a esta plataforma. A lo largo de esta Tesis se abordarán estos problema, exponiendo nuestras soluciones.

1.4 Conclusión

A partir de lo expuesto en el presente capítulo se puede llegar a las siguientes conclusiones:

1. En la actualidad existe una necesidad tanto de recursos computacionales como de almacenamiento en la comunidad astronómica.
2. El campo de la astronomía tiene suficiente potencial como para considerarse una de las comunidades más importantes en el uso de la computación distribuida. Como prueba de ello, múltiples campos de astronomía llevan mucho tiempo beneficiándose de este tipo de infraestructuras.
3. Las infraestructuras Grid se encuentran suficientemente desarrolladas gracias a las numerosas estructuras institucionales que consiguen dotar al Grid de una estabilidad y una seguridad difíciles de alcanzar en otros tipos de infraestructuras.
4. Las plataformas Grid están soportadas por múltiples grupos tecnológicos que han ido desarrollado herramientas, no sólo para el uso básico como puede ser middlewares o protocolos estandarizados, sino también para facilitar el uso de éstos añadiendo capas, APIs y otras herramientas que hacen más atractiva la plataforma.
5. Existen otras alternativas al Grid que ofrecen capacidad de cómputo y almacenamiento, pero o bien no se encuentran suficientemente desarrolladas o presentan múltiples carencias que son resueltas por el uso del Grid.

6. El uso cada vez más extendido de Grid por parte de aplicaciones científicas crea la necesidad de seguir distintas estrategias para la adaptación de aplicaciones astronómicas. Además, debido al carencias y escollos que aun existen para adaptar aplicaciones al entorno Grid, hemos visto la necesidad de implementar un nuevo método que optimice y automatice el proceso de adaptación de aplicaciones a distintas plataformas distribuidas como se describe en el siguiente capítulo.

Capítulo 2: Adaptación de aplicaciones astronómicas a Grid

La astronomía moderna se caracteriza hoy en día por el uso de grandes infraestructuras (observatorios, laboratorios, satélites) cuya explotación requiere el análisis masivo de datos. Para interpretar estos datos se ejecutan complejos cálculos y simulaciones. Todo ello requiere una capacidad de cálculo que sobrepasa el ordenador personal o incluso los pequeños clusters.

En este capítulo proponemos un nuevo método de adaptación de aplicaciones científicas a un entorno distribuido con la finalidad de incrementar significativamente su eficiencia en el uso de los servicios de este tipo de infraestructura, mejorando así el rendimiento de cómputo y almacenamiento.

Existen numerosos campos con aplicaciones que requieren estas necesidades, entre los que se encuentran el estudio de la física extragaláctica con los códigos para el análisis de colecciones de datos generados por satélites. Tal es el caso de los códigos para el estudio de galaxias activas (SAS) [IBA10] en el satélite XMM-Newton [AIA00], los códigos para el análisis de la evolución de las galaxias englobados en la herramienta *Starlight* [STA14] [CID09], e incluso las utilidades para simulaciones de formación de estructuras o para simulaciones de galaxias contenidas en la herramienta Gadget [SPR01].

Por otra parte, debido a la gran heterogeneidad de los códigos adaptados en una plataforma Grid, podemos encontrar códigos para problemas de física estelar, como es el caso de MESA [MES14, PAX11], que implementa cálculos de física en evolución estelar, o códigos que analizan las oscilaciones estelares teniendo en cuenta los efectos no adiabáticos y la iteración pulsaciones-convección, como es el caso de la herramienta *GraCo* [MOY08].

Otra disciplina dentro de la astronomía que se ha visto beneficiada por el uso de sus herramientas en una plataforma Grid es la astronomía solar, a través de códigos como SIR [RUI92], que implementan un algoritmo de inversión para interpretar la polarización de las líneas espectrales observadas en la fotosfera del Sol. Además de códigos para el estudio de planetas del sistema solar como LMD/MMCG [GAL09], que permite realizar el estudio de la temperatura, la dinámica y la composición de la alta atmósfera marciana, incluyendo tanto atmósfera neutra como ionosfera, e incluso fenómenos físicos tan cercanos como puede ser la atmósfera terrestre usando el código ARCoS [ARC14] para la simulación de descargas eléctricas.

Además, hemos añadido a una plataforma Grid aplicaciones pertenecientes a otras disciplinas que, aunque tienen cierta relación con la astronomía, también pueden usarse en otros campos. Este es el caso de *Ddscat* [DRA94], que permite calcular la dispersión y la absorción de la luz por partículas irregulares, y la disposición periódica de partículas irregulares que, aunque es un problema más propio de las ciencias de la Tierra, también se puede aplicar al análisis de polvo estelar. Otro ejemplo es la herramienta Superbayes [DEA06, SUP14], un paquete para obtener predicciones de cantidades observables de material, colisiones observables, y de abundancia de materia oscura muy utilizado en la física de partículas pero también de mucho interés en el campo de la cosmología.

En lo relativo a la elección del paradigma de procesamiento distribuido donde se ejecutan los códigos, hay varias alternativas. Entre ellas tenemos supercomputadores y servidores con cierto número de núcleos que pueden configurar plataformas Grid o incluso utilizarse en entornos de computación voluntaria. Para ello es necesario dividir el problema en múltiples subtareas que deben ejecutarse en este tipo de infraestructuras. Una infraestructura Grid es una de las mejores opciones porque, además de su carácter distribuido, cuenta con mecanismos de seguridad y múltiples servicios que le hacen ser una alternativa fiable y efectiva.

En la Sección 2.1 se detalla un nuevo método para la adaptación de aplicaciones a distintas plataformas distribuidas, seguidamente en las Secciones 2.3.1, 2.3.2 y 2.3.3 se describen los ejemplos de uso correspondiente a distintos perfiles de requisitos para la plataforma Grid. Finalmente en la Sección 2.4 se resumen las principales conclusiones obtenidas de este capítulo.

2.1 Metodología de implementación en plataformas distribuidas.

En esta Tesis proponemos un nuevo método de adaptación de aplicaciones científicas a un entorno distribuido con la finalidad de dar una alternativa más eficaz para utilizar los servicios a los que se tiene acceso para alcanzar los requisitos de cómputo y almacenamiento de la aplicación. Así pues, este método está pensado para casos con altas necesidades de cómputo y almacenamiento. El método que proponemos en este trabajo pretende ser válido para adaptar aplicaciones a cualquier tipo de entorno distribuido: cluster, Grid, Cloud, servidores DCE, etc.

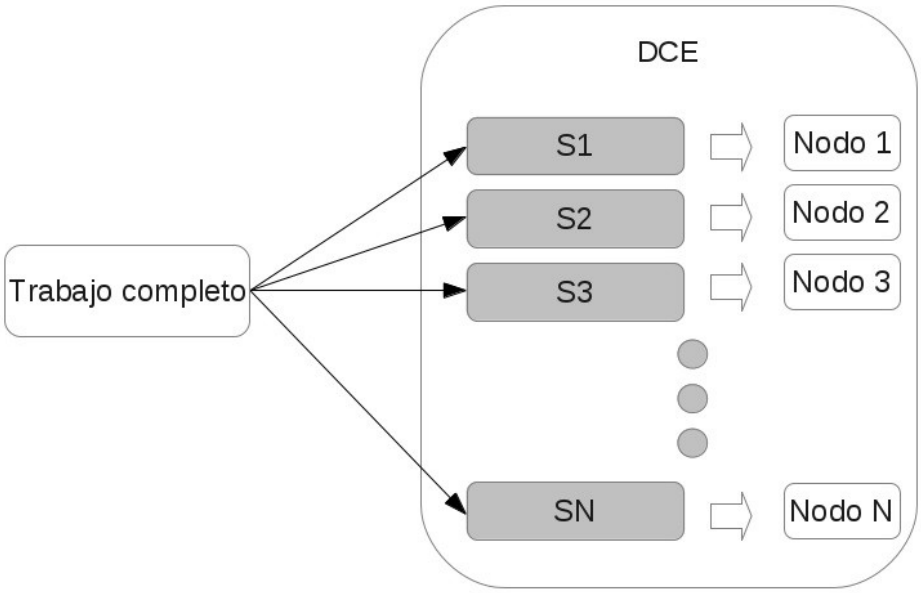


Figura 2.1: Distribución de carga del trabajo en un entorno distribuido. S1...SN son los subtrabajos enviados a cada uno de los los nodos de computación. DCE: Entornos distribuidos (Grid, Cluster, Cloud)

A través del método que proponemos se pretende conseguir una distribución equilibrada del trabajo (Figura 2.1) para su ejecución eficiente en los recursos distribuidos que constituyen la plataforma Grid de forma que se reduzca el tiempo de cómputo. Asignamos un número de trabajos a cada uno de los nodos disponibles. Hay normalmente un número mayor de trabajos que de nodos disponibles.

Para obtener este objetivo nos basamos en la realización de las siguientes acciones:

- Ofrecer una explicación, una análisis y una revisión de la solución propuesta para el uso de código científico en una infraestructura distribuida.

- Obtener la mejor forma de adaptar las aplicaciones a una infraestructura distribuida.
- Reducir el tiempo medio de ejecución de las aplicaciones usando para ello la paralelización en la ejecución de los subproblemas.
- Realizar una comparación de todas las aplicaciones implementadas en una plataforma distribuida, evaluando los tiempos medios de ejecución en un servidor dedicado y en una plataforma distribuida.

En contraste con las técnicas existentes de adaptación a DCE, el método que aquí presentamos se adapta a las aplicaciones de la manera más automatizada posible e independiente de la naturaleza del problema científico a resolver.

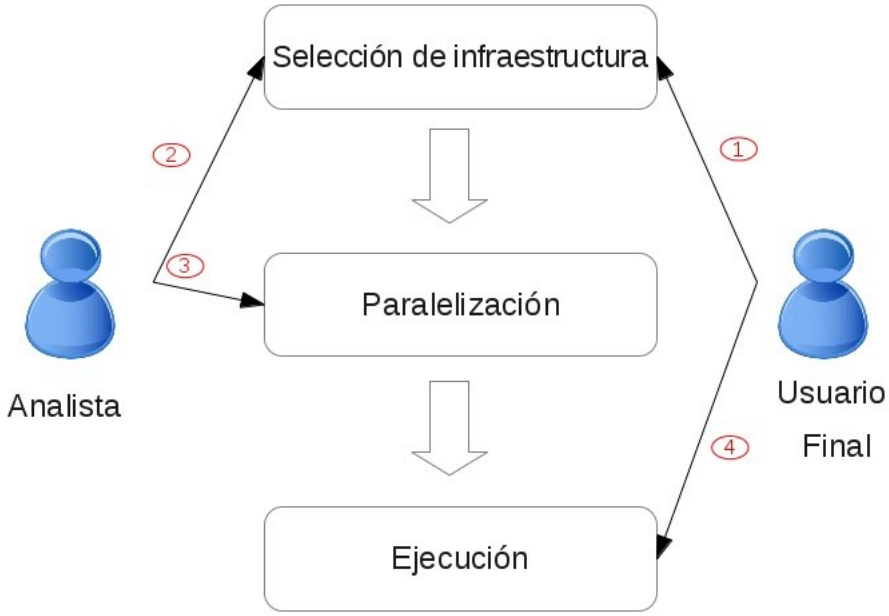


Figura 2.2: Diagrama general de casos de uso. Descripción de los pasos: 1. Imposición de necesidades. 2. Presentación de alternativas. 3. Optimización de la paralelización 4. Envío de trabajos.

Los actores principales que intervienen en el método, fases de configuración y evaluación de la aplicación son el analista de sistemas y el usuario final (Figura 2.2). En el método se preponen tres tipos de optimizaciones (de plataforma, de paralelización y de ejecución):

Optimización de plataforma

Se trata de elegir la plataforma computacional para la ejecución del trabajo en cuestión. En esta fase intervienen:

- El usuario final, que conoce los resultados esperables de la ejecución de los trabajos e impone los requisitos de cómputo y de almacenamiento. (paso 1)
- El analista del sistema, que conoce las características de cada infraestructura y presenta las mejores alternativas para el usuario. Además, selecciona la infraestructura que se utilizará para cada caso. (paso 2)

Optimización de paralelización

La segunda optimización se centra en la paralelización de la ejecución problema. El trabajo se divide en múltiples subtareas, independientes entre sí, en función de las necesidades informáticas y la infraestructura elegida. Esta optimización se hace de la forma más automática posible tal y como se recoge en el punto cuatro de la metodología.

El analista del sistema lleva a cabo esta optimización de modo transparente al usuario final. (paso 3)

Optimización de ejecución

Por último, la tercera etapa permite al usuario optimizar el envío de los trabajos por lotes, obtener informes sobre el estado de éstos, reenviar los trabajos en caso que haya errores, generar documentos con estadísticas de ejecución, etc. Para ellos utilizamos las herramientas apropiadas que hemos desarrollado, y que definen el paquete GSG [ROD11]) (paso 4).

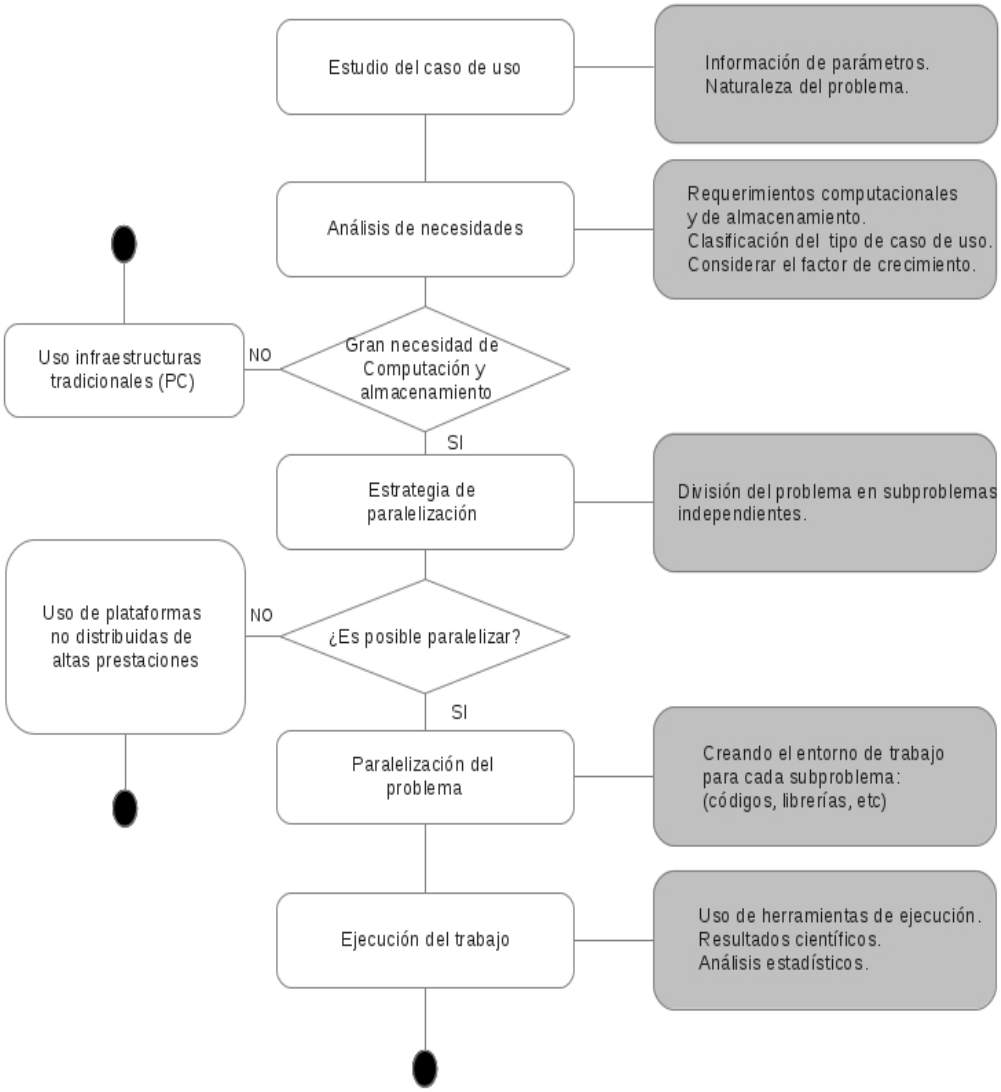


Figura 2.3: Diagrama de flujo del método propuesto para la adaptación de aplicaciones a entornos distribuidos.

Los pasos seguidos por la nueva estrategia (Figura 2.3) se describe a continuación:

1. Analizamos las necesidades de cada caso en detalle para obtener todas las características, tanto técnicas como científicas. El objetivo de esta fase es comprender la naturaleza del problema, obteniendo los parámetros que lo definen.

Tabla 2.1: Clasificación propia de la naturaleza del trabajo a ejecutar. Esta clasificación se deja abierta a otros posibles valores que aun no hemos tenido en cuenta.

Tipo	Grupo	Valores
Procesamiento de datos	Docentes	E-learning
	Científicos	Análisis espectral Filtrado de imágenes Generación de gráficas Análisis fotométrico Modelos teóricos Análisis estadístico Análisis de imágenes Simulaciones de sistemas Flujos de trabajo
	Técnicos	Monitorización Pipelines Simulaciones de diseño
Colección de datos	Científicos	Conjuntos de datos. Minería de datos.

2. Clasificamos la naturaleza (ver Tabla 2.1) cuyos valores son descritos en la Sección 1.1, englobados en los campos de la docencia, la ciencia o la tecnología.

   Las características técnicas no conllevan una clasificación, sino una información necesaria para catalogar el trabajo en las siguientes fases. Valores como el número de parámetros usados, el número de ficheros de entrada y de salida, el número de repeticiones necesarias para completar el trabajo, y otras características definen esta sección.

3. A continuación, establecemos los objetivos técnicos, en función de la infraestructura disponible (la capacidad de computación y almacenamiento requeridos), clasificándola de acuerdo con estos parámetros. Esta clasificación es una primera aproximación basada en las aplicaciones tratadas en esta Tesis, por tanto pueden ser revisadas por un factor de crecimiento de la carga de trabajo.

   El procedimiento de clasificación se basa en la información obtenida de cinco ejecuciones de un trabajo de evaluación descrito para cada caso de uso en una plataforma no distribuida (ver Sección 2.2.2). El trabajo de evaluación consiste en la ejecución del caso base de cada aplicación, con un rango de parámetros lo más pequeño posible.

Tabla 2.2: Clasificación del trabajo atendiendo a sus necesidades.

Tipo de necesidad	Computación	Almacenamiento
Básica	$T_{medio} < T_{min}$	$M_{medio} < M_{min}$
Medias	$T_{min} < T_{medio} < T_{max}$	$M_{min} < M_{medio} < M_{max}$
Altas	$T_{max} < T_{medio}$	$M_{max} < M_{medio}$

Atendiendo a los requisitos de tiempo de computación y memoria, hemos elaborado criterio que clasifica los requisitos técnicos de las aplicaciones como básicos, medios y grandes (Tabla 2.2):

• Problemas que requieren necesidades básicas: El tiempo de cálculo promedio, T_{medio}, no debe exceder más de un tiempo mínimo, T_{min}, o la necesidad de almacenamiento no debe exceder de tamaño mínimo, M_{min}.

• Problemas con necesidades medias: El tiempo de cómputo medio, M_{medio}, es más de T_{min}, pero menos de T_{max}, o el requerimiento de almacenamiento es menos de M_{min} pero no más de M_{max}.

• Tareas con grandes necesidades: Requieren más de T_{max} de ejecución o más de M_{max} de almacenamiento.

Para la realización de este trabajo de Tesis hemos especificado tiempo y tamaños de almacenamiento como sigue:

Necesidades básicas	T_{min}	1 hora	M_{min}	1 Gigabyte
Grandes necesidades	T_{max}	1 día	M_{max}	10 Gigabyte

Los tiempos de ejecución y el tamaño de almacenamiento puede variar a lo largo del tiempo en el caso que las características tecnológicas varíen.

4. La factor de crecimiento de la carga de trabajo también es un parámetro del análisis de necesidades. Definimos el factor de crecimiento de la carga de trabajo como la probabilidad que tiene el problema de hacerse más grande debido a las necesidades científicas.

   Siendo el problema definido como *Funcion*($a_1,...,a_n$), donde $a_1,...,a_n$ es el total de parámetros, definimos $N_1,...,N_n$ como el número de valores que se consideran cada uno de los parámetros que definen la función.

El factor de crecimiento de la carga de trabajo es una propiedad que se debe tener en cuenta en los casos donde la tiempo ejecución de la aplicación se multiplica si tenemos en cuenta más valores de los parámetros de entrada. Esto puede generar una gran necesidad de computo y almacenamiento, resultando crítico para la realización del trabajo. La mayoría de los casos planteados en esta Tesis se han clasificado como superlineales y, por lo tanto, el factor de crecimiento de la carga de trabajo ha sido un factor determinante a la hora de adaptar las aplicaciones a plataformas distribuidas.

Tabla 2.3: Clasificación del trabajo atendiendo a su factor de crecimiento.

Factor de crecimiento	**Descripción**
Infralineal	El aumento del número de valores, N_x ,de cualquier parámetro implica un aumento de las necesidades del problema. Por consiguiente, necesitaremos lanzar más ejecuciones para obtener los resultados esperados.
Lineal	El aumento de alguno de los valores de cualquier parámetro (no de todos) implica un aumento de las necesidades del problema.
Infralineal	El aumento del número de valores de cualquier parámetro no implica un aumento de las necesidades del problema.

El usuario final realiza esta clasificación de forma subjetiva. Aunque él debe tener en cuenta los rangos de parámetros de entrada y los requisitos científicos. (Tabla 2.3)

5. En el caso que exista una gran necesidad de recursos, se analiza si es factible o no el uso de una plataforma distribuida. En esta fase se clasifica los trabajos como:

Tabla 2.4: Clasificación del trabajo atendiendo a su divisibilidad.

Divisible	El trabajo completo se puede dividir en subtareas independientes. De esta forma, su ejecución es paralelizable.
No divisible	En el caso que el trabajo no se pueda dividir, debemos desarrollar códigos paralelos (paralelizar las tareas).

6. Cada una de esas divisiones se definen con una subtarea. Puede haber varias formas de división del trabajo. La elección de la forma óptima de división depende de cada caso de uso y del resultado del estudio. Es tarea del analista seleccionar la estrategia adecuada.

   El método de adaptación de aplicaciones tiene tres estrategias de paralelización. Cada una de ellas se diferencia por la forma como se divide el trabajo:

   - Por el número de repeticiones: Se ejecuta cada una de las repeticiones de forma paralela en distintos nodos de computación. Esta solución se elige si el trabajo se repite varias veces y el requisito de computación de cada ejecución es básica, estando definido este concepto en la Tabla 2.2.
   - Por los rangos de parámetros de entrada: Las divisiones tiene como entrada parte del rango de valores de los parámetros. Para esta división se elige los parámetros con mayor número de valores en el rango para que el número de particiones sea mayor y se obtenga mayor número de subtareas para ejecutar paralelamente. El número de subtareas (y por tanto la división de los rangos) se definirá dependiendo de los recursos disponibles en la infraestructuras. Éstas se ejecutan de forma paralela. Podemos aplicar este caso si el rango de valores de cada uno de los parámetros del problema se puede dividir en tareas independientes.
   - Por las fases de ejecución: Se puede ejecutar distintas fases del problema paralelamente en varios nodos de ejecución en el caso que éstas sean independientes. De esta manera, asociamos cada una de las fases de ejecución con una subtarea enviada a la plataforma distribuida.

7. Para adaptar las aplicaciones usadas en cada tipo de problema a un entorno distribuido, se requiere de un análisis de viabilidad que comprueba que estas aplicaciones pueden ser instaladas y ejecutadas en este entorno. Este paso es necesario, ya que una mala planificación impediría el correcto funcionamiento de la aplicación dentro del entorno distribuido.

```
Análisis_Infraestructura()

1. chech = InstalaciónSoftware()
2. check = Compatibilidad (código, infraestructura)
3. check = VerificaciónRecursos (infraestructura)
4. check = RevisiónSeguridad(certificadoUsuario)
5. check = Conexión()

6. Si check es correcto:  Procedemos al siguiente paso.
7. Sino:                  Imposibilidad de adaptación.
```

Código 2.1: Pseudocódigo para el análisis de la infraestructura.

Para este propósito, lanzamos un programa, cuyo pseudocódigo se muestra en el Código 2.2, que analiza la infraestructura en los siguientes puntos:

Línea 1: Instalación del software requerido en cada nodo de computación de la plataforma: librerías, códigos, compiladores, etc.

Línea 2: Compatibilidad de la ejecución del código con el hardware de la plataforma. El carácter heterogéneo del hardware es critico.

Línea 3: Verificación de que hay suficientes recursos de cómputo y del almacenamiento para la ejecución.

Línea 4: Revisión de la seguridad en cada uno de los nodos. Se analiza si los certificados usados y los permisos de los directorios están correctamente configurados.

Línea 5: Chequeo de los tiempos de conexión entre los diferentes servicios de la plataforma. Este análisis incluye un estudio sobre la fiabilidad de la conexión de los nodos, mediante la ejecución de códigos *benckmark* usando el sistema de monitorización Ganglia [GAN15]

8. Realizamos un análisis computacional representativo de la aplicación a adaptar descrito en la del apartado específico de este análisis en la Sección 2.1.1.

Tabla 2.5: Descripción de las distintas aplicaciones utilizadas en los casos de uso más representativos descritos en la memoria.

Aplicación	**Descripción**	**Ref.**	**Página Web**
Starlight	Spectral Synthesis code	[STA14]	http://www.*Starlight*.ufsc.br/papers/Manual_StCv04.pdf
GraCo	Granada oscillation code	[MOY08]	http://www.iaa.es/es/node/7225
Ddscat	Discrete Dipole Scattering	[DDS14]	http://code.google.com/p/*Ddscat*/

Como ejemplo para la aplicación de este método se analiza aquí los tres casos más representativos (Tabla 2.5) en cuanto a las distintas soluciones planteadas para la adaptación a dicho entorno.

Para este análisis utilizaremos la infraestructura distribuida Grid, usando para ello la infraestructura especifica Ibergrid y utilizamos nuestro paquete de herramientas "General Scripts for Grid" (GSG) [ROD11] para la ejecución de trabajos en entornos distribuidos, y más específicamente en plataformas Grid.

Tabla 2.6: Resumen de los parámetros que caracterizan la aplicación, incluyendo su definición y sus posibles valores.

Tipo	**Definición**	**Clasificación**
Naturaleza científica	Características científicas del caso	Simulaciones Análisis de datos reales Filtrado de datos Flujos de trabajo Análisis estadístico ...
Características técnicas	Obtención de las características técnicas	Nº de parámetros Nº de repeticiones Nº ficheros de E/S ...
Requerimientos computacionales	Tiempo medio de computo requerido para la ejecución de un trabajo completo	Básico Medio Alto
Requerimientos almacenamiento	Almacenamiento medio de datos obtenidos en un trabajo completo	Básico Medio Alto
Factor de crecimiento de la carga de trabajo	Relación entre el tamaño del problema y la necesidad de recursos	Superlineal. Lineal. Infralineal.
Paralelización de los trabajos	Factible o no el uso de una plataforma distribuida	Divisible No divisible
Estrategias de división del trabajo	Elección de la forma óptima de división	Número de repeticiones Rangos de parámetros de entrada Fases de ejecución

2.1.1 Análisis computacional

Con el objeto de simplificar y facilitar la comprensión del análisis computacional que aquí presento, vamos a definir unos términos de referencia:

Caso base es el elemento mínimo necesario para la ejecución en una aplicación. (e.g. un espectro, una partícula, una estrella, etc.)

Trabajo completo es el un conjunto de casos base de ejecución. Su número depende del rango de los parámetros que definen el trabajo.

Trabajo de evaluación es una parte representativa del trabajo completo de la aplicación. Debe ser suficiente extenso como para extrapolar las conclusiones del análisis al trabajo completo, y con un tiempo de ejecución no demasiado elevado como para analizarse en cada uno de los casos.

Trabajo Grid es cada una de las partes que componen un trabajo completo y que sera enviado a una plataforma Grid. Esta división depende de cada aplicación y del resultado del estudio de cada caso para una ejecución óptima. Como regla general, el tiempo de computación de trabajo Grid no debe ser significativo, pero ni reducido ni extenso. Un tiempo reducido de trabajo es ineficiente debido a que cada trabajo Grid lleva aparejado un tiempo para el envío a una plataforma distribuida. Se considera ineficiente cuando el orden del tiempo de ejecución del propio trabajo es similar al necesario para su envío. Tampoco puede ser demasiado extenso ya que aumentan las posibilidades de errores en la ejecución.

Esta división es necesaria para paralelizar de forma óptima la ejecución del código, dividiendo el problema completo en subtareas independientes entre sí que puedan enviarse a cada uno de los elementos de computación existentes en la plataforma para aprovechar su ejecución paralela para reducir el tiempo de procesamiento.

Análisis de tiempo de ejecución en una plataforma Grid.

Basándonos en los servicios de una infraestructura Grid descritos en la Tabla 1.1, la ejecución de un trabajo Grid se divide en distintas fases,

- PW: (Procesamiento en sistema de gestión de carga de Trabajos. WMS). Tiempo de preparación para la ejecución de un trabajo Grid en el Nodo de Computación Grid (WN). Es el tiempo desde que se envía el trabajo Grid hasta que llega al Elemento de Computación (CE), pasando por todos los servicios de Grid, como WMS y CE.
- WL: (Lista de espera de recursos en el CE). Tiempo que requiere el CE para asignar un WN al trabajo recibido.
- TT: (Tiempo de transferencia de datos). Tiempo necesario para transferir los datos desde el Interfaz de Usuario (UI) hasta el WN mediante el servicio SandBox o transferirlos desde el elemento de almacenamiento Grid (SE) en el caso que los fichero sean de gran tamaño.
- IS: (Tiempo de instalación de la aplicación). Tiempo que se emplea en transferir los códigos fuente, los compiladores y las librerías desde el UI al servicio WN. Se utiliza sólo si se necesita la instalación de la aplicación en el WN (gridificación dinámica).
- EX: (Ejecución efectiva en los WN). Tiempo de ejecución real de la aplicación. Una vez están todos los elementos en el WN, se procede a la fase de ejecución efectiva.
- IR: (Recogida de información y resultados). Tiempo para transferir los datos desde el WN al UI o al SE en el caso que así se requiera. Éste se emplea después del procesado y una vez obtenidos los resultados.

Tabla 2.7: Fases en la ejecución de un trabajo Grid.

PW	Procesamiento en el WMS.
CE	Lista de espera de recursos en el CE.
TT	Tiempo de transferencia de datos.
IS	Tiempo de instalación de la aplicación.
EX	Ejecución efectiva en los WN.
IR	Recogida de información y resultados.

A la hora de proceder al análisis de tiempo de cada caso, se elaboran unas tablas de tiempos cuya cabecera corresponde a las distintas fases de la ejecución de un trabajo Grid. (Tabla 2.8).

Para la comparativa de tiempos de ejecución entre las dos infraestructuras se ha utilizado la gridificación estática (Sección 2.1.2) en el caso del uso de una plataforma Grid, por esta razón no se tiene en cuenta el tiempo de instalación (IS) para el cálculo total de ejecución del trabajo de evaluación.

Por otro lado, es necesario un estudio de errores, ya que la aparición de éstos ocasiona que los tiempos de ejecución total se alarguen debido al reenvío de los trabajos fallidos.

Análisis de mejora usando la paralelización

Por otro lado, hemos usado la ley de Amdahl para analizar la mejora máxima del problema que se obtiene cuando una parte de este se ejecuta en un entorno distribuido.

$$A \leq \frac{1}{(1-F_m)+\frac{F_m}{A_m}} \quad (2.1)$$

- *A* es la ganancia en velocidad conseguida en el trabajo completo debido al envío de parte de éste a un entorno distribuido.
- F_m es el porcentaje de tiempo que la parte paralelizada del trabajo se ejecuta en la plataforma. En el caso de la ejecución en plataformas Grid, F_m tiene un valor muy cercano a 1, ya que se paraleliza la totalidad de la ejecución de los trabajos. El valor no es exactamente 1 ya que hay que tener en cuenta el tiempo necesario para la ejecución en este tipo de plataformas ($T_{overhead}$)

$$T_{overhead} = (T_{Pw} + T_{CE} + T_{TT} + T_{IR}) \quad (2.2)$$

- A_m es el factor de mejora máxima que se puede producir gracias al entorno distribuido (en este caso el número de procesadores que componen en nodo Grid).

Sabemos que la paralelización se realiza en la fase de ejecución (EX) del envío del trabajo a la plataforma Grid. El resto de fases descritas en la Tabla 2.8 se hacen de forma secuencial. Por lo tanto definimos F_m como:

$$F_m = \frac{T_{Total} - (T_{Pw} + T_{CE} + T_{TT} + T_{IR})}{T_{Total}} \quad (2.3)$$

Para que el análisis computacional sea representativo se realizan cinco ejecuciones de cada trabajo para obtener la media y la desviación típica de los trabajos. De esta forma se determina la bondad de ajuste de dos distribuciones de probabilidad entre sí, obteniendo la desviación estándar mediante el test de Kolmogórov-Smirnov [EAD71].

Para todos los casos se comparan las ejecuciones de los trabajos de evaluación en dos infraestructuras de computación. En ambos casos se han ejecutado los mismos trabajos de evaluación, con los parámetros definidos para ellos, y las mismas máquinas libres de cargas de trabajo. Las dos infraestructuras que se utilizan para el análisis computacional de los trabajos son:

- Una Infraestructura Grid con las características que se definen en las Figuras 2.6 y 2.4.
- Un servidor dedicado con las características que se definen en la Figura 2.6.

Ejecución de trabajos en un servidor dedicado frente en una infraestructura Grid.

En el caso del servidor dedicado hemos procedido a la ejecución secuencial de los trabajos para poder comparar los tiempos de ejecución de trabajos paralelizados en plataformas distribuidas frente a la misma ejecución del trabajo de forma serializada.

En una infraestructura Grid es necesario un estudio previo de paralelización para optimizar el uso de la plataforma. Además hay que tener en cuenta que los ficheros y datos necesarios para la ejecución de la aplicación deben estar previamente ubicados en una interfaz de usuario Grid (IU). El análisis de tiempos empieza desde la máquina donde se lanzan los trabajos Grid.

En todos los análisis hemos buscado una solución para la definición del trabajo Grid, dividiendo el número de casos total del trabajo completo en trabajos Grid con un tiempo de ejecución óptimo.

2.1.2 Gridificación estática vs gridificación dinámica

En cada estudio que hemos elaborado de cada caso de uso, hemos definido dos alternativas de ejecución del trabajo completo en el entorno Grid: gridificación estática y gridificación dinámica.

En la **gridificación estática** se debe instalar previamente la aplicación una sola vez en cada uno de los nodos de trabajo y posteriormente identificarlos con una etiqueta (Tag) para que, utilizando el fichero JDL, se solicite al elemento de computación que mande los trabajos sólo a los nodos que contengan esta etiqueta. Esta solución permite reducir el tiempo de ejecución efectiva en los nodos de trabajo al no tener que instalar el código cada vez que se envía a ejecutar un trabajo Grid de la aplicación.

Si la aplicación está en continua evolución, la tarea de la instalación en múltiples nodos cada vez que hay una versión nueva resulta inabarcable para los administradores de cada uno de los elementos de computación. Por tanto, sólo se ha utilizado esta solución con tiempos de ejecución pequeños y en nodos muy específicos.

Como alternativa a la solución anterior a ésta, la **gridificación dinámica,** que consiste en el envío de un paquete que contiene el código fuente de la aplicación, las librerías necesarias, un compilador compatible con la infraestructura donde se ejecuta, y un instalador (ya sea un script o un fichero binario), para hacer la instalación antes de la ejecución. Para tener todos estos elementos a disposición de un nodo de trabajo, es necesario un empaquetamiento previo.

Esta solución tiene la ventaja de poder enviar los trabajos Grid sin necesidad de filtrar los nodos que no contengan el código y las librerías necesarias para la ejecución de la aplicación. Además, permite la ejecución de la aplicación sin depender de la versión de la aplicación. Sin embargo, complica la ejecución de trabajos en Grid, ralentizando la fase de ejecución efectiva en el WN, puesto que a cada ejecución se le debe sumar el tiempo de instalación del paquete.

Como norma general, el envío de este paquete se realiza mediante el sistema SandBox [FOS02b]. Este proceso ralentiza el tiempo de procesamiento en el WMS debido al aumento del tamaño de las transferencias a la plataforma, y además incrementa la posibilidad de errores en esta fase. Para evitar esta situación se utiliza el SE. De esta forma se asegura que el paquete que contiene todos los elementos para la instalación de la aplicación pueda transferirse desde el SE, teniendo una entrada en el Catálogo de Archivos Lógicos (LFC, por sus siglas en inglés) y estando disponible para todos los elementos de computación de la Organización Virtual (VO, por sus siglas en inglés).

Hay un paquete de instalación distinto para cada tipo de arquitectura, así que, después de un análisis previo de la arquitectura, se elige el paquete de instalación requerido para cada caso. En el caso de que la arquitectura, o el sistema de archivos del nodo de computación, requiera un paquete de instalación que no se encuentre dentro del SE, el trabajo Grid se cancela y se reenvía a otro nodo de computación.

Hemos desarrollado una **solución intermedia** que unifica las dos alternativas. A la hora de mandar los trabajos se le da prioridad a los elementos de computación que tengan instalado la aplicación (método de gridificación estática), y en caso de que estos nodos estén sobrecargados de trabajos, se envían a otros nodos mediante gridificación dinámica.

2.2 Infraestructuras utilizadas

En la realización de esta Tesis hemos utilizado dos tipos de arquitecturas con diferentes recursos y procedimientos de ejecución: computación distribuida en Grid, que utiliza un tipo de arquitectura NUMA [CAR00] con nodos basados en multiprocesamiento simétrico (SMP), frente a computación centralizada en un servidor dedicado, basado en una arquitectura UMA.

Además, en cada una de las infraestructuras hemos añadido un conjunto de librerías que usan las aplicaciones instaladas. Un listado de las principales librerías de apoyo junto a una breve descripción de ellas se proporcionan en el Apéndice II.

A continuación se describen las principales características de cada una de las infraestructuras utilizadas.

2.2.1 Elemento de computación Grid

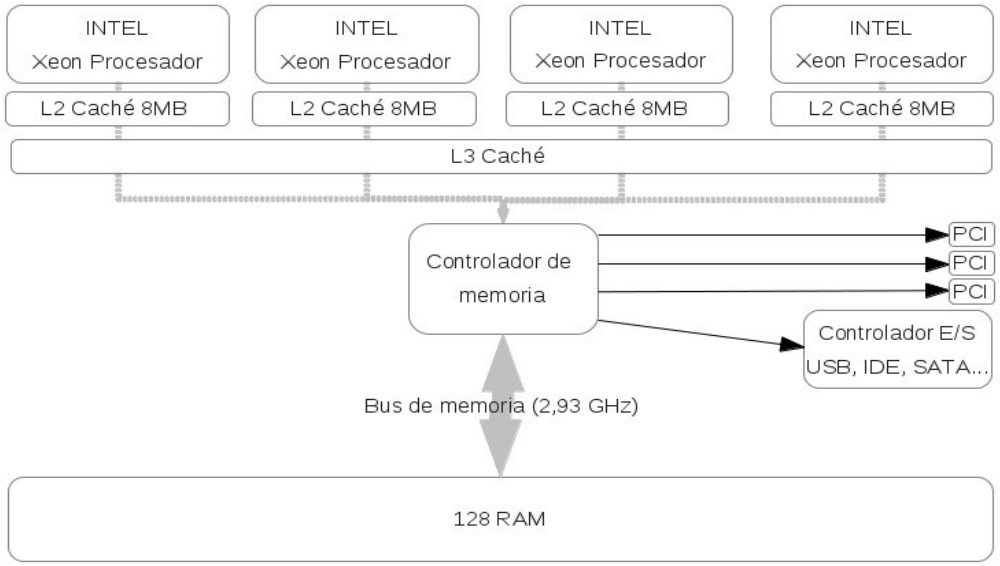


Figura 2.4: Características de un nodo de computación perteneciente a la infraestructura del Instituto de Astrofísica de Andalucía. Definido por una arqutectura UMA (SMP).

Las características de entorno para computación en Grid (Figura 2.4), se describen a continuación.

Se compone de un servidor de 32 nodos con 4 procesadores Intel Xeon Quad Core X7350 a 130W, 2.93GHz, con bus de memoria a 1066MHz, 8MB de caché L2 con soporte multiprocesador. El número total de núcleos por nodo es de 256, con un total de 128GB de RAM.

Por tanto, se dispone de 512 microprocesadores con un total de 2048 núcleos con 4TB de RAM en los 32 nodos que contiene.

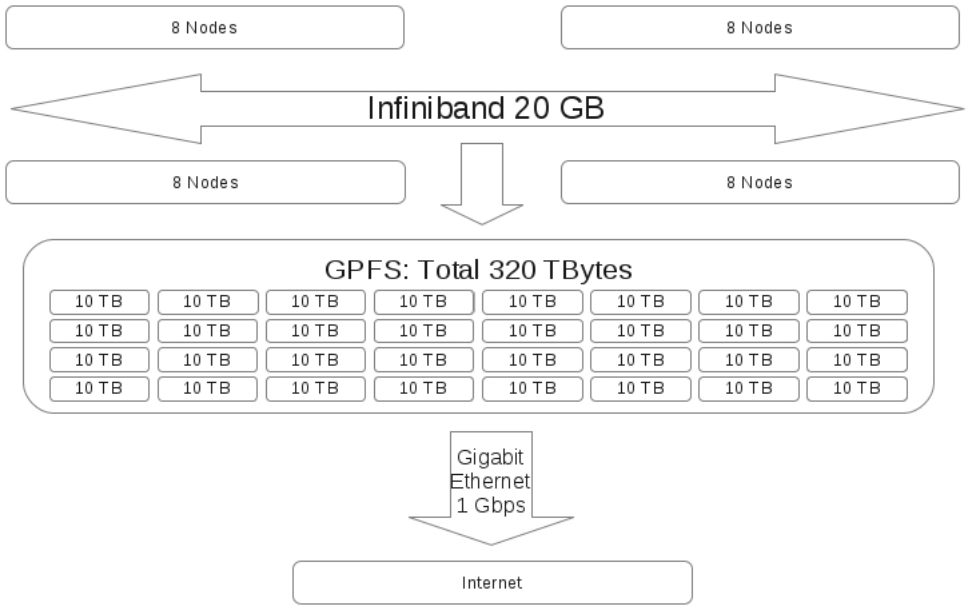


Figura 2.5: Distribución de los elementos de computación Grid del Instituto de Astrofísica de Andalucía. La unión de nodos definen una arquitectura NUMA.

Como se observa en la Figura 2.5, los recursos de almacenamiento se reparten utilizando el sistema de ficheros distribuidos GPFS, disponiéndose de un volumen total de almacenamiento de 320 TeraBytes.

La comunicación se realiza mediante el recurso de comunicación Voltaire (2 IB + 4 GE por nodo). Y para la comunicación entre nodos se utiliza una red Infiniband de 20 Gbps tanto para la comunicación de procesos MPI como para el acceso a los elementos de almacenamiento. La conexión con el exterior se realiza mediante una red Gigabit Ethernet de 1 Gbps.

2.2.2 Servidor Dedicado

Se trata de una máquina PowerEdge R910 de Dell con 80 procesadores Intel Xeon de 10 núcleos cada uno, cuyo modelo es el E7- 4850 a 2.00GHz y soporte multiprocesador (SMP). Se tiene un total de 800 núcleos con 64GB de memoria para 4 microprocesadores y posee 1TByte de disco duro de tipo SAS, conectable en caliente. La conexión con el exterior también se realiza mediante una red Gigabit Ethernet a 1 Gbps.

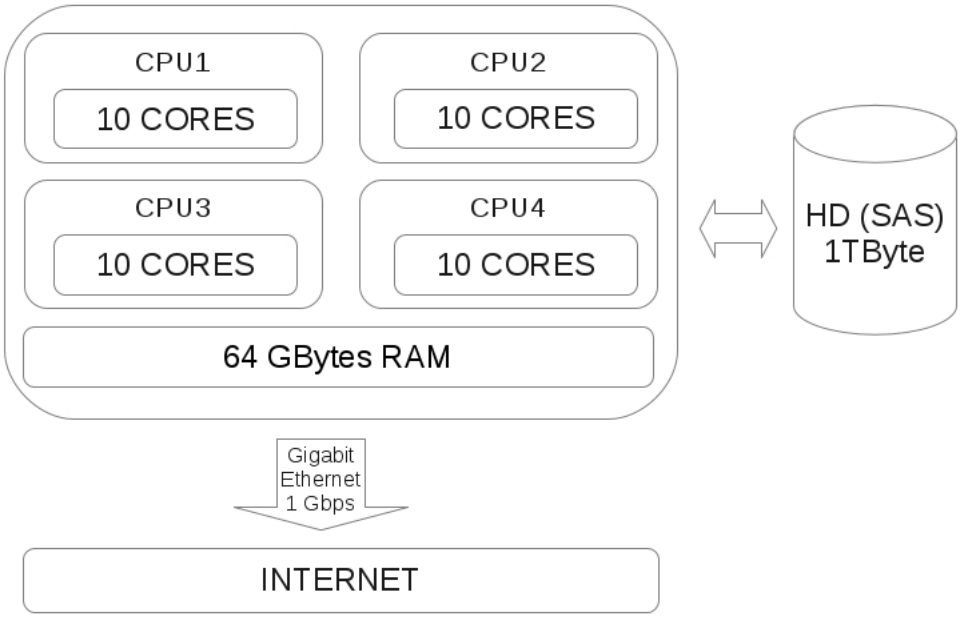


Figura 2.6: Servidor en el cual se realiza el trabajo de evaluación. Definido por una arqutectura UMA (SMP).

2.3 Ejemplos científicos del uso del método

En esta sección se describen tres ejemplos que se corresponden a las aplicaciones *Starligh* (Sección 2.3.1), *GraCo* (Sección 2.3.2) y *Ddscat* (Sección 2.3.3) que corresponden a distintos perfiles de requisitos para una plataforma Grid.

2.3.1 Estudio de poblaciones estelares con *Starlight*

El estudio de poblaciones estelares en galaxias se ve beneficiado por la disponibilidad de bases de datos de gran tamaño que contiene numerosos datos sobre espectros observados de galaxias y modelos de síntesis evolutiva [CAS90].

El análisis del espectro emitido por la galaxia es una herramienta necesaria para estimar las propiedades físicas de ésta. Se trata de la descomposición de un espectro observado en términos de una superposición de una base de poblaciones estelares simples de diversas edades y metalicidades. Este estudio permite conocer la historia de formación estelar, el enriquecimiento químico de la galaxia, su extinción, masa y propiedades cinemáticas (i.e. velocidad de dispersión).

Este caso de uso se engloba dentro del proyecto CALIFA (Calar Alto Legacy Integral Field Area) [PER13], que pretende desvelar la evolución espacio-temporal de las galaxias. Se centra en averiguar cómo crecen las galaxias al convertir el gas en generaciones sucesivas de estrellas. El objetivo de CALIFA es disponer de información física detallada acerca de una muestra numerosa de galaxias, y de bancos de datos para extraer la historia de la formación estelar en cada una de ellas. Uno de los cometidos de este proyecto es comprobar la evolución en el tiempo de las galaxias, averiguando si hay una formación de tipo “dentro - fuera”

El trabajo completo de CALIFA se compone de un sondeo de 600 galaxias. Cada una de las galaxias está contenida en un cubo de datos, siendo éste la unidad principal en el análisis, que detalla una región en el espacio definida por dos componentes espaciales y una componente espectral. Cada cubo contiene una media de 70x70 spaxels (spaxel = spectral pixel), ofreciendo un total de algo menos de 5000 espectros por cubo (~70x70). Cada espectro tiene una media de 1900 unidades espectrales, de manera que las dimensión media total del cubo es 70x70x1900.

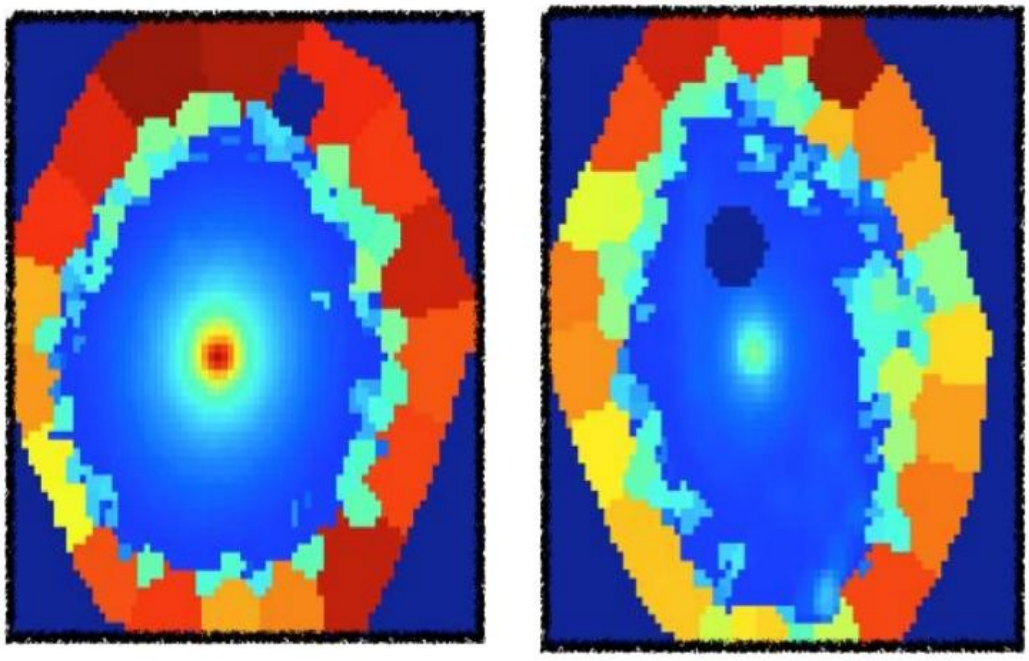


Figura 2.7: Representación en las dos dimensiones espaciales de dos cubos de galaxias (CALIFA 900 y CALIFA 277) después del aplicar el preprocesado usando teselación de Voronoi. La tercera dimensión es el espectro asociado.Figura obtenida de [PER13].

Starlight [STA14] es un código programado en Fortran designado para ajustar un espectro observado O_λ con un modelo M_λ que contiene N componentes espectrales [SAB10]. Esta librería con la que se compara puede estar hecha de otros espectros observados patrón, de modelos sintéticos evolutivos, estrellas individuales, etc. El ajuste se realiza usando varias técnicas a lo largo de diversos pasos. Mediante algoritmos de enfriamiento simulado (simulated annealing), algoritmos de Metropolis y Monte Carlo basado en cadenas de Markov (MCMC) se explora el espacio de parámetros en busca del mínimo. Los parámetros fundamentales son la extinción del espectro, la velocidad central y su dispersión, y las componentes usadas en las librerías.

El objetivo de esta aplicación es obtener el ajuste de una serie de parámetros de los espectros de un cubo de galaxias observada (como son la masa de la galaxia, el indice de extinción, la composición química, la edad de la población estelar o la luz recibida) enfrentada a una base de prototipos de espectros definidos. De esta forma, se cataloga cada muestra observada y se obtiene una librería de poblaciones estelares completa. Un ejemplo del resultado de este proceso se ve en la Figura 2.7.

Tabla 2.8: Principales parámetros de ajuste de un cubo de galaxias.

Parámetro	**Definición**
Masa (m)	Masa contenida en la galaxia
Índice de extinción (ie)	Dependiente del polvo en la galaxia.
Metalicidad (met)	Composición química.
Tiempo (t)	Edad de la población estelar
Luz	Espectro de luz recibida.

Los parámetros más relevantes ajustados para cada cubo de galaxia se muestran en la Tabla 2.8

Fases del experimento

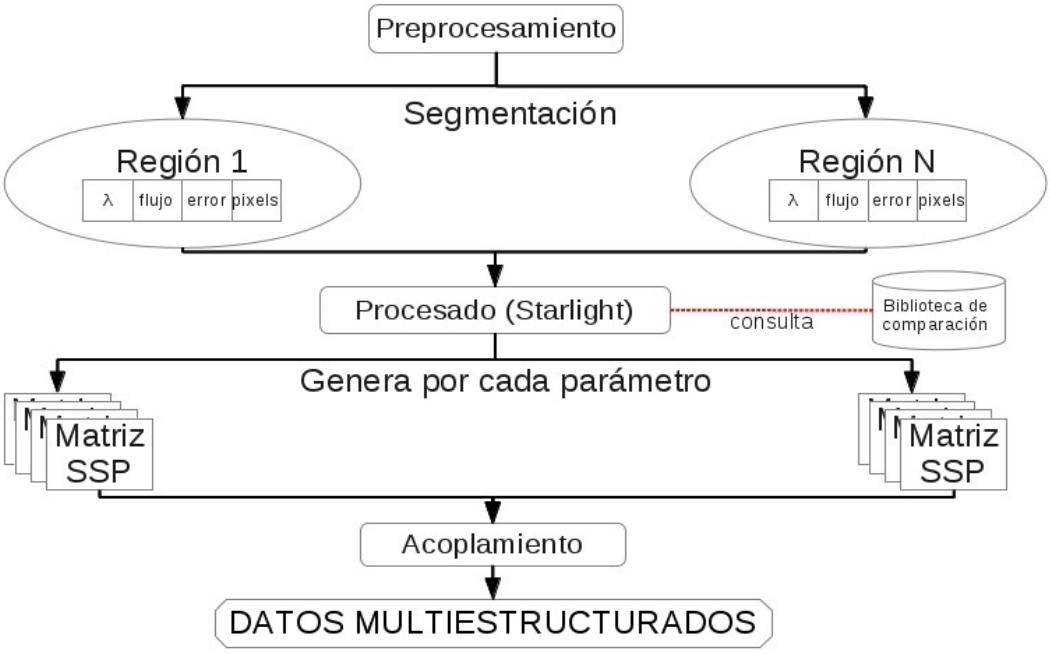


Figura 2.8: Fases de la ejecución de los cubos de galaxias pertenecientes a CALIFA.

El experimento se compone de tres fases diferenciadas (Figura 2.8). En la primera fase se produce un preprocesamiento del cubo a analizar, seguido de la propia ejecución de *Starlight*, por terminar con un agrupamiento de resultados.

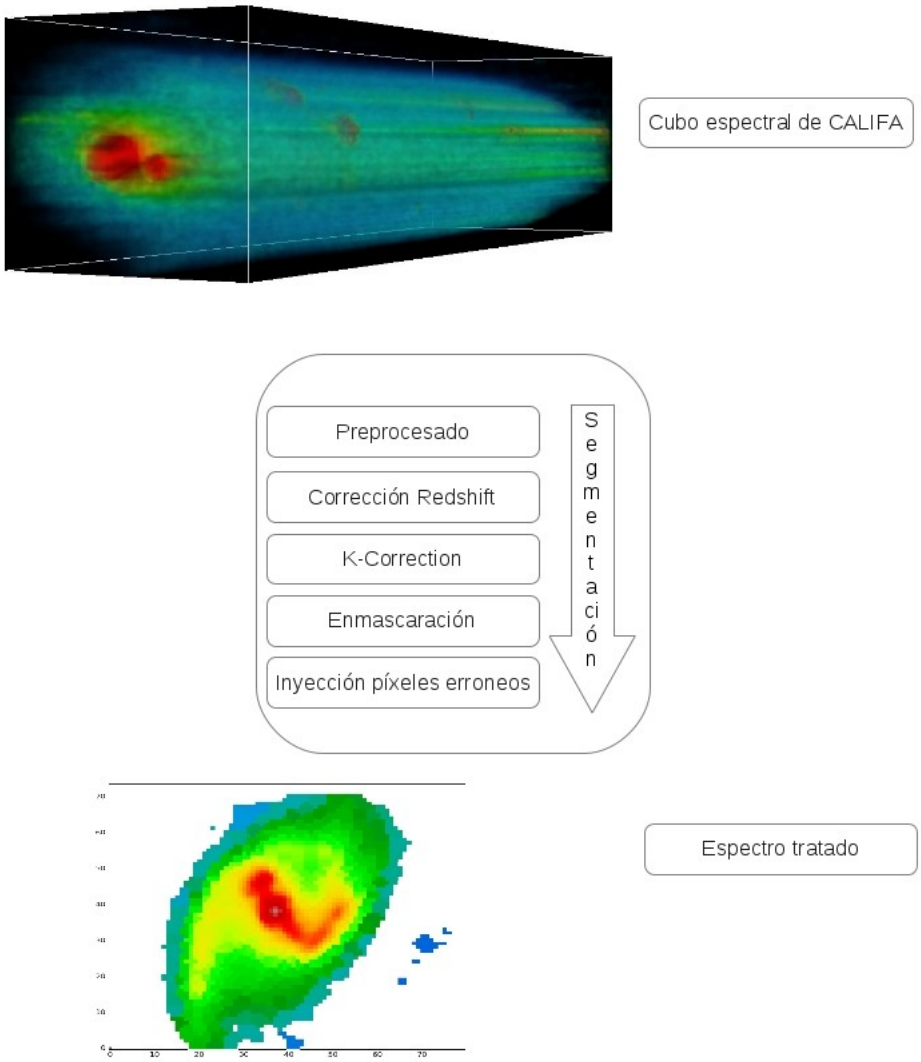


Figura 2.9: Fases del preprocesamiento del cubo espectral de CALIFA.

La fase de preprocesamiento (Figura 2.9), se encarga de proveer a la aplicación Starligh de espectros individuales previamente tratados mediante la denominada corrección *redshift* [KAN69], seguido de una corrección cosmológica de flujo K-Correction [VAN61], un enmascarado de la dimensión espacial eliminado cualquier componente ajeno a nuestra galaxia que pueda interferir en el espectro obtenido y un tratamiento cuidadoso del enmascaramiento de los elementos espectrales erróneos (asociados con la aparición de bordes negros en la configuración de la red de difracción, residuos en la corrección del espectro del cielo, líneas telúricas, etc) que pueden afectar al ajuste global del espectro.

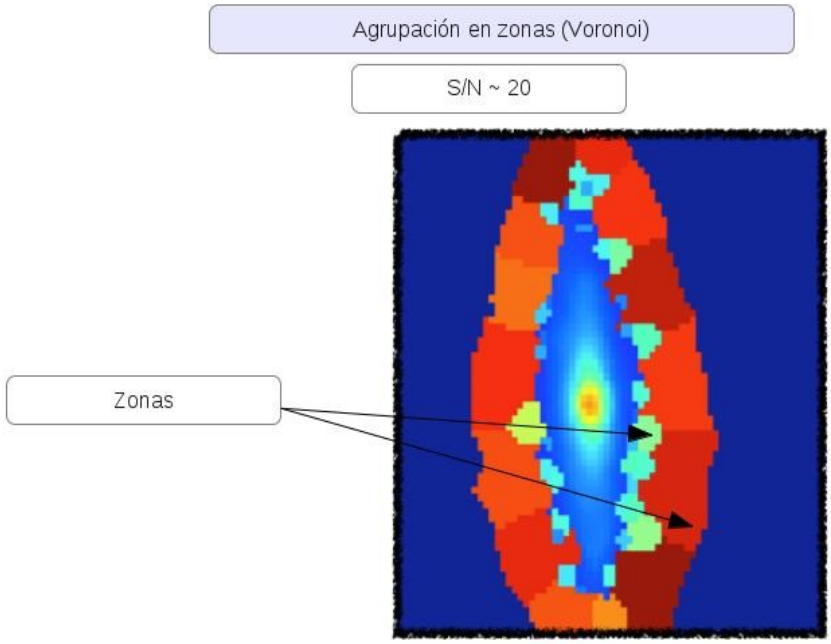


Figura 2.10: Segmentación de un cubo de galaxia en múltiples zonas mediante la selección de Voronoi. Ajustándolo a una S/R de 20. Figura basada en [PER13]

Uno de los puntos principales del preprocesamiento es segmentar el cubo en regiones con una relación señal/ruido (S/R) mínima para obtener el correspondiente mapa de segmentos. Esto se hace utilizando la técnica de teselación de Voronoi, agrupando puntos adyacentes para conseguir una S/R mínima dentro de una región conexa (Figura 2.10). De esta forma se reduce el número de espectros finales en una estructura de tres dimensiones. Como resultado, se obtiene la información del espectro de la región junto a un fichero con la información del preprocesado.

Después del preprocesado, cada cubo se almacena en ficheros binarios individuales que contienen la longitud de onda, flujo, errores de flujo y píxeles erróneos asociados a los mismos.

Starligh analiza cada uno de los espectros obtenidos individualmente en la fase de preprocesado. En cada punto se analiza el espectro para la obtención de cada uno de los parámetros relevantes (Tabla 2.8). Se trata de un análisis de hipercubos ya que hay espectros asociados a las dos dimensiones espaciales y a cada uno de los parámetros característicos de la galaxia.

Starlight realiza un ajuste de espectros sintéticos[1] a partir del espectro observado de la galaxia. Con este ajuste se calcula la matriz SSP (Single Stellar Population) para cada uno de los parámetros libres de la Tabla 2.8.

Tras obtener los espectros asociados a cada una de las zonas que contienen varios píxeles, la fase de acoplamiento solapa los resultados obtenidos para toda la región analizada. Éstos se almacenan en un gran fichero de datos multiestructurado por cada galaxia.

Análisis computacional

Trabajo completo. La aplicación toma como entrada un archivo con un espectro al que se le realiza el ajuste junto con otro archivo de parámetros de configuración. La aplicación se ejecuta una vez para cada espectro y configuración.

Estudiamos un número de galaxias, N_g, contenidas cada una en un cubo que está compuesto (una vez taselado) por N_e espectros. Para cada uno de los espectros se analiza N_p de valores de parámetros. Para la obtención del número total de ejecuciones de la aplicación necesarias para procesar un trabajo completo nos basamos en la siguiente Fórmula:

$$N_x = N_g \cdot N_e \cdot N_p \quad (2.4)$$

Definimos la media en el tiempo de procesamiento para el análisis de cada espectro como t_o, que depende del número de SSPs (Single Stellar Population) contenidas en la librería de comparación. Mientras su número sea mayor, más tiempo se necesita para la ejecución ya que se comparan más espectros sintéticos.

La necesidad de comparar con más o menos espectros viene dada por la discretización en edad y metalicidad de los espectros de la librería. Para cubrir toda la edad del universo se necesitan espectros sintéticos con distintas edades que cubran este rango para así poder comparar con el observado. Si se consideran pocas edades, se corre el riesgo de estimar de forma incorrecta la edad del espectro. De igual modo, también se necesita tener en la librería espectros de distintas metalicidades. Por ejemplo, se puede tener cubiertas un rango de 6 metalicidades y 30 edades para cada metalicidad, lo que nos daría un total de 180 espectros sintéticos de la librería sobre la que comparamos.

1 Usando una base de espectros denominada librería de comparación, compuesta por conjunto de espectros sintéticos que refleja los parámetros base para el análisis.

Para la obtención del tiempo total de la ejecución de un trabajo completo, T_c, nos basamos en la siguiente fórmula:

$$T_c = N_x \cdot t_0 \quad (2.5)$$

Análisis de los requisitos de cómputo. En el caso que nos encontramos, el preprocesado y la fase de acoplamiento (Figura 2.8) tiene un orden de complejidad O(log n). Así pues, estas dos fases no implican una carga de cómputo lo suficientemente grande como para plantearse el uso de una plataforma Grid. Si en un futuro existiese más necesidad de cómputo, estas dos fases podrían procesarse en los recursos que se añaden. Basándose en ejecuciones de similar naturaleza, la estimación que haría necesario usar la infraestructura Grid sería si el número de ejecuciones de un trabajo completo se cuadriplicase (n = 4).

Sin embargo, la fase de procesamiento, que utiliza la aplicación *Starligh,* tiene un orden de complejidad cuadrática, O(n^2). Así pues, para esta fase si es conveniente el uso de una plataforma distribuida como Grid, y para ello aplicamos el método de adaptación. De esta forma, solo centramos el análisis de ejecución y los resultados experimentales en esta fase.

Método de gridificación. *Starlight* es una aplicación en continuo desarrollo debido a que, a medida que avanza la investigación en este área de la astronomía, los requisitos del código se modifican ya sea por la inclusión de nuevos elementos, o por necesidades computacionales. Esta circunstancia implica la aparición constante de nuevas versiones. Por tanto hemos elegido el método de gridificación dinámica descrito en la Sección 2.1.2. No obstante, debido a los costes que supondría la instalación de *Starlight*, se ha optado por implementar una solución híbrida que unifica las dos alternativas.

Trabajo Grid. Estudios sobre el uso de la plataforma Grid [FER05], concluyen que el tiempo dedicado a la preparación, envío y tratamiento de los resultados supone alrededor de un 25% del tiempo total de ejecución de un trabajo en Grid. La gridifiación dinámica añade un 15% más de instalación del ejecutable y sus componentes. Teniendo esto en cuenta, los tiempos medios para el análisis de espectros individuales es t_o = 300-600 segundos, sugieren que la estrategia intuitiva de asociar cada análisis individual de espectros con un trabajo Grid no es óptima.

Para solucionar este problema, hemos optado por asociar un número prefijado de análisis de espectros a un trabajo Grid, según los recursos computacionales disponibles en ese momento. De esta forma, los trabajos Grid no tienen una duración determinada, sino que ejecutan el análisis de un espectro y, cuando terminan, proceden a analizar el siguiente.

Por ejemplo, si un trabajo completo necesita procesar 1000 espectros y se dispone de 20 nodos en la plataforma, la opción de un análisis por trabajo Grid mandará a la plataforma 1000 trabajos Grid, con sus correspondientes tiempos de preparación (25%), pero sólo se ejecutarán paralelamente 20 y el resto se mantendrá en espera. En cambio, con la solución planteada en este trabajo, se mandan 20 trabajos Grid que se ejecutan inmediatamente y cuando cada uno de ellos termine el análisis de un espectro pide el siguiente espectro disponible para su análisis. Finalmente, el trabajo Grid termina cuando no haya más espectros que procesar, analizando cada trabajo Grid un número prefijado de espectros.

Para ello es necesario que todos los espectros que se van a procesar y la base de espectros sintéticos estén en un lugar accesible para cualquier nodo de computación perteneciente a la organización virtual. El elemento idóneo para este cometido es un SE que esté cercano al CE.

Una vez se está ejecutando un trabajo Grid, éste hace una petición al SE de un espectro que no haya sido procesado. A continuación se ejecuta la aplicación *Starlight* y el resultado se almacena de nuevo en un directorio especifico del SE.

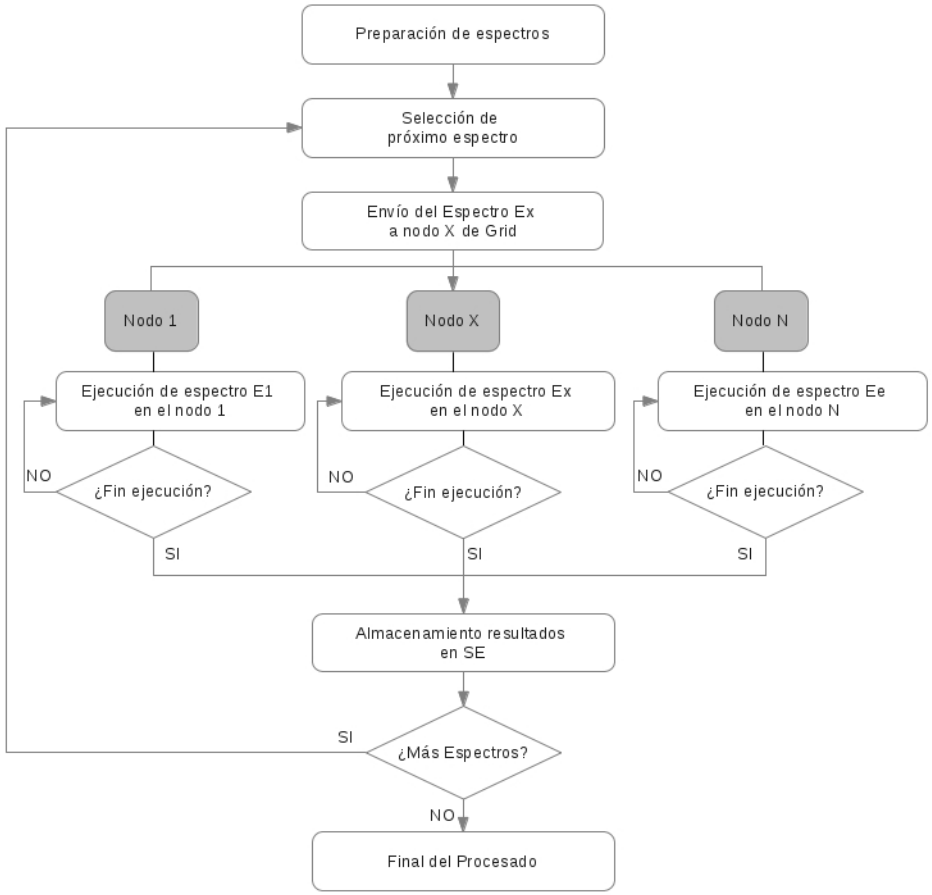


Figura 2.11: Diagrama de flujo del procedimiento de asignación de un espectro a un trabajo Grid.

```
ProgramaPrincipal () # procesado en el WN
    ...
1.  espectro = trabajoGrid.peticionEspectro();
2.  espectroProcesado = trabajoGrid.procesaEspectro(espectro);
3.  trabajoGrid.sustituyeEspectro(listaEspectrosProcesados, espectroProcesado);
    ...

espectro peticiónEspectro ();
4.   mientras existe (ListaEspectrosNOProcesados);
5.   espectro = siguienteEspectro (ListaEspectrosNOProcesados);
6.   añadir (espectroVacio, listaEspectrosProcesados, espectroVacio);
```

Código 2.2: Pseudocódigo para el mecanismo de asignación de un espectro a un trabajo Grid. Un trabajo Grid recorre los espectros contenidos en el aparado de elementos de entrada del SE (línea 4) y determina el del primer espectro que no se encuentra en el apartado de elementos procesados del SE (línea 5). Inmediatamente, genera un fichero vacío (línea 6) en el apartado de espectros procesados para que ningún otro trabajo Grid se lo asigne. Una vez ejecutado el trabajo Grid, el fichero vacío es sustituido por los resultados del análisis (línea 3) y el trabajo Grid busca el siguiente espectro procesable.

Este mecanismo (Figura 2.11), cuyo pseudocódigo se describe en el Código 2.2, necesita un procedimiento para diferenciar los espectros procesados de los que no lo están.

De esta forma, la única limitación que hay en el número de trabajos Grid enviados es el número de recursos disponibles, y su duración máxima depende del mecanismo de limitación de tiempo de ejecución de un trabajo Grid (certificados digitales, configuración de colas de ejecución, etc).

Este novedoso procedimiento es una solución óptima frente a la opción que asocia cada análisis de espectro a un trabajo Grid, ya que la fase de preparación, envío, tratamiento, e instalación solo se aplicará una vez por trabajo Grid. Además, puede aplicarse en múltiples casos que tengan en común requisitos parecidos al caso que estamos estudiando.

Trabajo de evaluación. Definimos un trabajo de evaluación como una parte del trabajo completo suficiente para evaluar los resultados de forma completa, pero sin requerir tanta necesidad de computación que haga imposible la obtención de resultados en un tiempo aceptable. Así pues, este trabajo de evaluación tiene 300 trabajos Grid que representan un 75% de la capacidad de cómputo del CE[2].

Esta forma de actuar constituye una buena práctica y permite no saturar un elemento de computo con el envío masivo de trabajos Grid.

Por otra parte, y para garantizar la optimización de recursos hemos elaborado un sistema de corrección de errores que detecta trabajos fallidos y los relanza (Código 2.3).

```
programaPrincipal () # Enviado desde el UI.
1.  trabajoGrid1.envío ()
2.  resultado = trabajoGrid1.recogidaTrabajo()
3.  Si resultado.fallido()
4.    trabajoGrid1.envío()
    Si no
5.    procesa(resultado)
```

Código 2.3: Pseudocódigo de relanzamiento de un trabajo Grid en el caso de que falle en la ejecución del trabajo. Si uno de los trabajos falla (línea 3), ya sea por un error en los servicios Grid, por un error en la comunicación, o por la expiración del tiempo de ejecución asignado, un script lanza un trabajo Grid que sustituye al actual (línea 4) y procesa el espectro que ejecutaba el trabajo cancelado con anterioridad (línea 5). De esta forma se asegura que el número de trabajos Grid en ejecución es constante hasta que la lista de espectros procesables esté terminada.

2. Este porcentaje corresponde además con los márgenes que se usan en computación masiva como buenas prácticas para evitar la saturación del sistema.

En una entorno ideal con un numero de nodos disponibles mayor a 300, y suponiendo que los 300 trabajos se ejecutan paralelamente y que su duración es idéntica mediante este mecanismo de envío en paralelo de múltiples trabajos, se consigue que el tiempo de ejecución de un trabajo completo se reduzca unas 300 veces, que coincide con el número total de trabajos Grid enviados. No obstante, hay que tener en cuenta el coste temporal de enviar los trabajos Grid a la plataforma.

Resultados experimentales

Para el análisis de los tiempos en cada una de las situaciones, hemos empleado dos tipos de arquitecturas con diversos recursos y procedimientos de ejecución: una plataforma de computación distribuida Grid, frente a un servidor de propósito específico. (Sección 2.2)

Tabla 2.9: Rango de los parámetros que definen un trabajo completo para el análisis de la evolución en galaxias utilizando el código Starligh.

Parámetro	**Valores**
Nº galaxias, N_g	600 galaxias.
Nº espectros, N_e	Alrededor de 1000 espectros por galaxia (aplicando Voronoi).
Nº parámetros, N_p	5 valores de parámetros para el análisis por espectro.

Usando el servidor descrito en la Sección 2.2.2, se ha estimado una media en el tiempo de procesamiento, $300<t_0<600$ segundos para el análisis de cada espectro que no presente dificultades en su análisis, y una media de $N_g=600$ galaxias, contenidas cada una en un cubo que está compuesto por $N_e=1000$ puntos, cada uno asociado a un espectro (ver Tabla 2.9.). Para el trabajo de evaluación analizamos $N_p=5$ valores de parámetros, con un número total de 150 SSPs. Por tanto, se estima que el número de ejecuciones son $N_x=3.000.000$.

Usando (2.5), y al estimarse $t_0=600$ s. como máximo por ejecución se obtiene un tiempo máximo estimado de computación del trabajo completo de $T_e=180.000.000$ segundos de ejecución (aproximadamente 5,7 años).

Teniendo en cuenta que el trabajo completo requiere una gran capacidad de cómputo, en el trabajo de evaluación optamos por reducir el caso base para un estudio más detallado con $N_r = 5$ repeticiones para el procesamiento de 1 galaxia con 500 espectros analizados para 1 parámetro.

Usando (2.4) y (2.5), hemos calculado que son necesarias un total de 500 ejecuciones para el trabajo de evaluación que suponen un total de 300.000 segundos (aproximadamente 83 horas), asumiendo $t_0 = 600$ s.

Evaluación en el servidor dedicado frente a la plataforma de computación distribuida Grid (IberGrid). Para el análisis nos basamos en que el CE utilizado para el procesado de trabajos tiene una capacidad media de 400 procesadores, que la organización virtual "phys.vo.ibergrid.eu" contiene una media de 1000 nodos accesibles, y que cada trabajo de evaluación abarca el análisis de una galaxia.

Hemos tenido en cuenta que el trabajo de evaluación descrito con anterioridad debe completar 500 análisis de espectros y que la media del tiempo de ejecución del análisis de un espectro es de 600 s. Procedemos a enviar un número fijo de trabajos Grid entre 100 y 1400.

Nos apoyamos en (2.6) para determinar el tiempo total de ejecución de un trabajo de evaluación T_{ev}, donde N_e es el número de análisis de espectros del trabajo de evaluación, T_g es el número de trabajos Grid lanzados y T_e es el tiempo medio de ejecución de un análisis.

$$T_{ev} = \frac{N_e}{T_g} \cdot T_e \qquad (2.6)$$

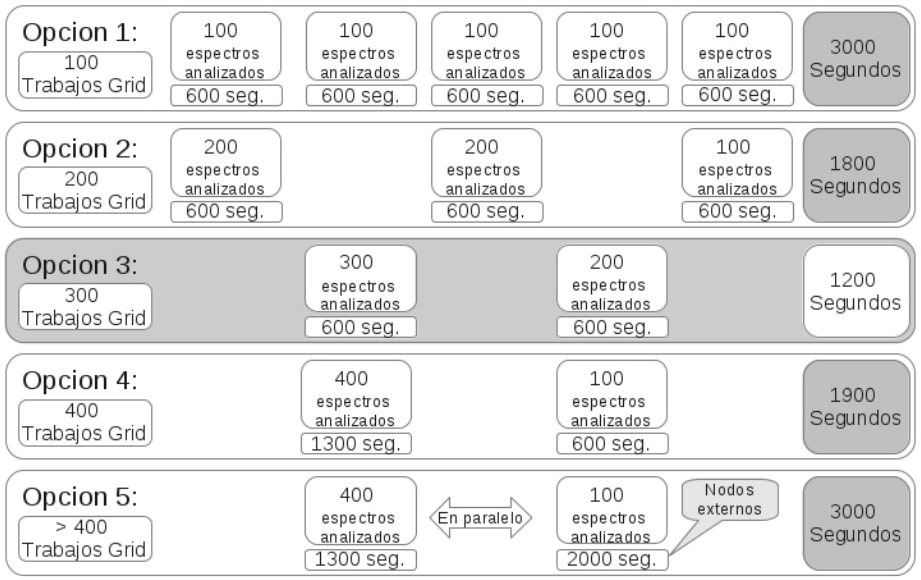


Figura 2.12: Opciones de envío de trabajos Grid para el trabajo de evaluación del análisis de 500 espectros de galáxias.

A continuación se analizan cada una de las opciones, descritas en la Figura 2.12.

Si se lanzan 100 trabajos en paralelo, la realización de los primeros 100 análisis necesitarán alrededor de 600 s. Como el trabajo de evaluación contiene 500 análisis, 400 trabajos Grid quedan a la espera de ser ejecutados. Suponiendo que todos los análisis finalizan en 600 s, la ejecución completa se realizará en 3000 s. (2.6). De esta forma concluimos que si se opta por el lanzamiento de un lote de 100 trabajos Grid, no se obtiene toda la eficiencia de la plataforma Grid.

Si se lanzan 200 trabajos en paralelo, las primeras 200 ejecuciones se realizan en paralelo y tendrán un tiempo estimado de ejecución de 600 s. Las siguientes 200 ejecuciones Grid añaden otros 600 s. Finalmente se procesan las restantes 100 ejecuciones, que requieren 600 s. de tiempo de ejecución, para completar los 500 espectros. Esto hace un tiempo aproximado de 1800 s.

Si se lanzan 300 trabajos en paralelo, se ejecuta un primer lote de 300 y otro lote de los restantes 200, cada uno de los lotes tarda en procesarse 600 s. obteniéndose un tiempo de ejecución aproximado de 1200 s.

Para lotes de más de 400 trabajos Grid, el CE usado se saturaría. Esto supone una disminución en la eficiencia de cálculo que, para los 400 trabajos, supone un aumento de 1300 s. de tiempo de ejecución más unos 600 s. en el tiempo de ejecución completa del lote de los 100 trabajos de componen el estudio.

A partir de lotes mayores de 400 trabajos, se enviarán a nodos externos pertenecientes a la VO. Para ello se usa la solución intermedia de Gridificación descrita en la Sección 2.1.2. De esta forma, los primeros 400 trabajos se envían al nodo de evaluación y consumen aproximadamente 1300 s. como se comentó anteriormente. Los 100 trabajos restantes se enviarán paralelamente a nodos externos. Su ejecución estimada es de 2000 s. ya que, al utilizarse la Gridificación dinámica, se deberá añadir al tiempo de ejecución un tiempo de instalación de la aplicación. Además, se incrementa el tiempo de espera de recursos Grid y el tiempo de transferencia de datos, debido a que los nodos externos se pueden encontrar alejados de la máquina. Así pues, la ejecución de un lote de trabajos en un nodo externo alcanza alrededor de 2000 s.

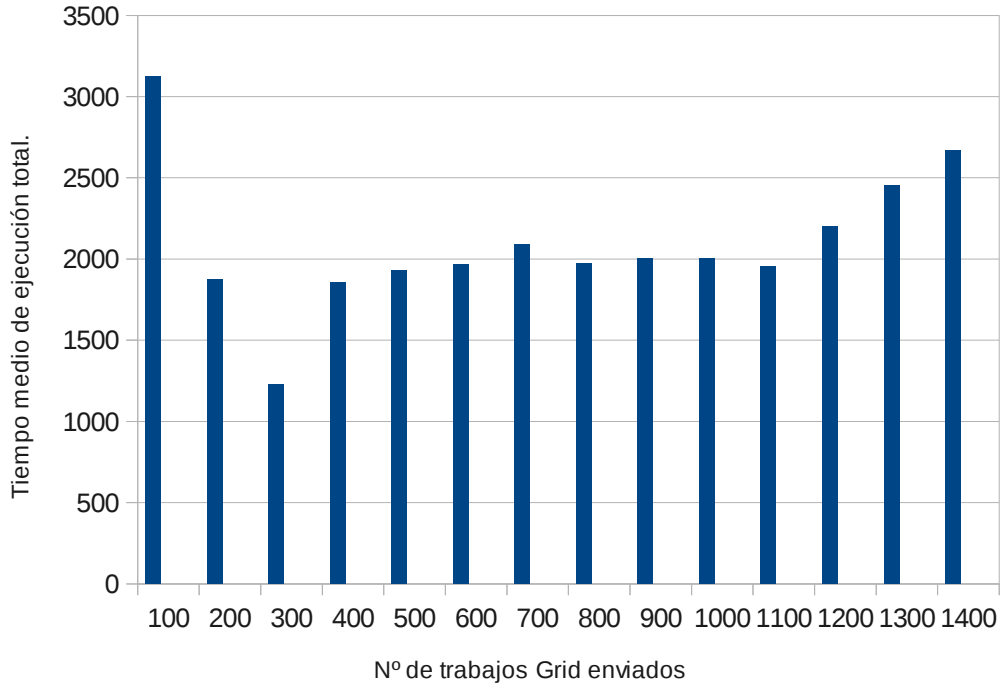


Figura 2.13: Tiempo de ejecución de un trabajo de evaluación que analiza 500 espectros frente al número de trabajos Grid enviados en un lote. Las unidades están es segundos.

Si el trabajo de evaluación necesita ejecutar un número mayor de casos, el tiempo de ejecución se mantendrá constante (unos 2000 s.) si se envía un mismo número de trabajos Grid que de casos a ejecutar, hasta llegar a los niveles de saturación de los nodos pertenecientes a la VO (400 + 1000). Como se ve en la Figura 2.13, para trabajos de evaluación que contengan más de 1100 trabajos Grid, al aumentar los trabajos enviados se incrementa el tiempo de ejecución debido a la saturación de los recursos de la VO.

El resto de los resultados de los demás casos explicados con anterioridad también se visualizan gráficamente en la Figura 2.13, llegándose a a la conclusión que lanzar paquetes de 300 trabajos Grid a la vez es la solución más óptima. Así, en este trabajo de evaluación se tienen que analizar 500 espectros, y se envían 300 trabajos. Si suponemos que la tasa de cancelaciones de los trabajos es cero, 200 trabajos Grid procesarán dos espectros y 100 solamente uno.

Tabla 2.10: Tiempos medios para el procesamiento del trabajo de evaluación ejecutado en un servidor dedicado y en una plataforma Grid lanzando 300 trabajos Grid. Las unidades están en segundos. Tiempo total: tiempo desde el inicio hasta el fin de la ejecución. Tiempo medio de un trabajo: para el servidor es un único trabajo serializado y para la plataforma Grid un trabajo Grid. Tiempo total de cálculo: tiempo total de computación, en el caso de Grid se suma todos los tiempo de cada nodo.

	Servidor dedicado	**Plataforma Grid**
Tiempo total	303.420,0	1213,0
Tiempo medio de un trabajo	299.700,0	1209,0
Tiempo total de cálculo.	299.400,0	299.580,0

El tiempo medio de un trabajo Grid es prácticamente el mismo que el tiempo total de la ejecución. Esto se debe a que el trabajo Grid siempre se mantendrá activo y e irá recibiendo nuevos espectros a analizar una vez termine de procesar cada uno de ellos.

Si observamos la comparación entre el tiempo total de ejecución del trabajo de evaluación en un servidor dedicado frente a la una infraestructura Grid Tabla 2.10, podemos concluir que el uso de la plataforma Grid reduce aproximadamente a un 13% del total del tiempo medio de ejecución de *Starlight* ejecutado en un servidor dedicado.

Tabla 2.11: Tiempos medios, mínimos y máximos de un trabajo Grid perteneciente al trabajo de evaluación definido para el caso de Starlight. Siendo PW, CE, TT, IS, EX tiempo de ejecución efectivo, IR y TOTAL La unidades están en segundos.

	PW	CE	TT	IS*	EX	IR	TOTAL
Mínimo	11,0	7,0	11,0	17,0	612,0	19,0	682,9
Máximo	31,0	19,0	12,0	29,0	1309,0	35,0	1314,0
Media	19,4	12,2	11,0	22,4	1020,2	27,6	1209,0
Desv. Estándar	7,2	4,3	0,49	4,9	285.25	5,9	253,42

Analizando con más profundidad el trabajo de evaluación en una plataforma Grid, observamos en la Tabla 2.11 que la fase de ejecución (EX) es la que consume más tiempo del trabajo de ejecución. Es justamente esta fase la que se paraleliza en un entorno Grid, obteniendo de esta forma un rendimiento mayor.

El tiempo de ejecución efectivo mínimo de un trabajo Grid es de 682,9 s. (Tabla 2.11), que corresponde al tiempo esperado de procesado de un sólo espectro más la suma del tiempo requerido para el uso de la plataforma. Si observamos el tiempo de ejecución máximo, resulta que es el tiempo esperado para la ejecución de dos espectros (1314,0 s.).

Análisis de errores. Por otro lado, los datos obtenidos del trabajo de evaluación, no se tiene en cuenta que la tasa de cancelación puede ser distinta a cero.

Tabla 2.12: Estado a la finalización de los trabajos Grid enviados a la plataforma.

Trabajos Grid enviados	313
Trabajos ejecutados con éxito	300
Trabajos terminados con error	13

Por esta razón se ha realizado un estudio de errores donde hemos enviado un trabajo de evaluación compuesto por 300 trabajos Grid, más los enviados por el mecanismo de reenvío en caso de cancelación (Tabla 2.12).

Tabla 2.13: Clasificación de errores en la ejecución de los trabajos Grid.

Trabajos con resultado erróneo	0
Trabajos sin asignación de recursos (CE)	4
Trabajos con fallo en la conexión (WMS)	5
Trabajos con ejecución errónea (WN)	1
No catalogados	3

Observando la clasificación de errores de la Tabla 2.13 hemos determinado que existe un 4,3% de errores en el envío de trabajos Grid, porcentaje asumible para la realización de este trabajo de evaluación. Además, en el contexto general de la plataforma Grid, los trabajos cuyo error de ejecución se debe a fallos en la conexión son los más abundantes, un 38,4% de los errores totales.

Después de obtener los datos de los tiempos de ejecución de la Tabla 2.10, observamos que la ganancia real obtenida es de $A_{real}=250{,}14$ (aplicando 2.1).

Sabemos que el factor de mejora es de $A_m=300$ ya que es el número máximo de cores usados en la plataforma Grid que es igual al número de trabajos enviados. Según la ley de Amdahl (2.1) debe ser mayor o igual que la real. Sabemos que $F_m\approx 1$ y por tanto cota de la ganancia es $A_{cota}\approx 300$ (aplicando 2.1). Cumpliéndose la ley de Amdahl (2.1): $(A_{real}=250{,}14)\leq(A_{cota}=300)$

En lo que se refiere a la utilización del almacenamiento en Grid, el uso de la plataforma permite almacenar todas las bases de espectros sintéticos y espectros observados para múltiples ejecuciones. Esto reduce considerablemente el tiempo de transferencia de los archivos ya que una vez sabida su ubicación, y usando el servicio de catálogo de Grid, no es necesaria ninguna transferencia. Por otro lado, el sistema de almacenamiento Grid se utiliza como repositorio de los resultados obtenidos en la ejecuciones, y el grupo de investigación que utiliza la plataforma.

Resultados de la ejecución del trabajo completo. La adaptación de la herramienta *Starligh* ha permitido que, en un tiempo reducido, el proyecto CALIFA haya estimado que las galaxias masivas, además de crecer más rápido que las menores, lo hacen de dentro afuera, es decir, desarrollando sus regiones centrales primero. [PER13]. Sin la aplicación de este método se hubiera necesitado un tiempo computación tan extenso que sería imposible su realización.

En un futuro, se pretenden realizar numerosas simulaciones sobre un mismo cubo para obtener la distribución de valores para un espectro dado. Esto hará crecer la necesitad de cómputo. Por otro lado, la nueva generación de instrumentos para la observación de galaxias permitirá aumentar el número de espectros por cubo de galaxias para analizar y se estima que de 1000 espectros se pasarán a 40.000 a lo largo de los próximos años. Debido a estos cambios nos enfrentamos a un problema dentro del denominado “Big Data”.

2.3.2 Estudio de evolución estelar con astrosismología con *GraCo*.

La astrosismología hace posible sondar el interior de las estrellas. Ello es indispensable para comprobar las teorías de estructura y evolución estelar (ver Capítulo 1). Gracias a este campo de investigación, el desarrollo de la física estelar en las últimas décadas ha sido vertiginoso. Varias misiones espaciales de fotometría (e.g. CoRoT [AUV09], Kepler [NAS09]) junto con las campañas en tierra de seguimiento, generalmente de espectroscopía de alta resolución, han sido diseñadas para el estudio del interior estelar. Esto ha supuesto un incremento de varios órdenes de magnitud la cantidad de datos disponibles, tanto observaciones como los modelos necesarios para su interpretación. La tendencia es de un crecimiento exponencial en el número de datos, como se puede deducir de las previsiones de otras misiones espaciales (TESS [RIC09], CHEOPS [BRO1] y PLATO2.0 [PLA14]) actualmente en fase de desarrollo. Es un reto para la comunidad científica el poder sacar el máximo provecho a las observaciones en un tiempo razonable. Para ello actualmente se estudian estrategias tanto para el análisis de los datos como para su interpretación y, como era de esperar, las infraestructuras distribuidas de computación juegan un papel importante.

El presente caso de uso consiste precisamente en un estudio astrosismológico que requiere del modelado masivo para la interpretación de las oscilaciones estelares de múltiples estrellas pulsantes observadas por el satélite CoRoT. Para ilustrarlo, aquí mostramos la aplicación del método a una de las estrellas de estudio, HD174966 de tipo A-F con pulsaciones delta Scuti [GAR13].

El estudio completo trata de constreñir los modelos astrosismológicos que caracterizan las estrellas observadas a partir de los datos observados, i.e. parámetros globales (luminosidad, temperatura efectiva, metalicidad, etc.) y parámetros de oscilación (frecuencias de oscilación). Se pretende caracterizar las estrellas pulsantes a partir de patrones observados en los espectros de oscilación. En particular estudiamos la denominada gran separación [GAR09, GAR13, GAR15, SUA14] para determinar la densidad media de la estrella y su estadio evolutivo. Para realizar este cometido nos basamos en la creación de dos tipos de modelos:

- *Modelos de equilibrio*, que definen las estructuras estelares mediante el estudio de cada una de las capas que las componen, teniendo cada capa distintas propiedades químicas y físicas. En el modelo evolutivo, se asocia cada modelo de equilibrio a un paso en el estadio evolutivo, creándose una traza mediante un conjunto de modelos de equilibrio (Figura 2.15). Estos modelos pueden ser implementados con aplicaciones como Cesam [MOR97], CESTAM [MAR13] o MESA [MES14]. En este caso, hemos usado la aplicación Cesam.
- *Modelos de oscilación,* que realiza perturbaciones en cada una de las capas de los modelos de equilibrio realizados con anterioridad. Estas perturbaciones pueden tener en cuenta la rotación (como los modelos generados por FILOU [SUA08]) u obviarla (como los modelos generados por *GraCo* [MOY08]). En este caso, hemos usado el código *GraCo.*

Tabla 2.14: Principales parámetros de ajuste de un modelo de estructura estelar.

Parámetro	**Definición**
Masa (m)	Masa contenida en la estrella.
Metalicidad (Fe/H)	Composición química.
Indice de convección (α)	Transferencia de calor en distintas capas. Parametrización de la eficiencia de la convección para el transporte de energía en el interior estelar.
Overshooting (d_{ov})	Parametrización del efecto en la efectividad de la convección debido a la inercia de los elementos convectivos.
Rotación (rot)	Velocidad de rotación de la estrella.

Los principales parámetros que definen este estudio astrosismológico se enumeran en la Tabla 2.14 . Además, mediante un espacio de parámetros definido para cada trabajo de ejecución se construye una maya de modelos homogéneos.

Fases del experimento

```
ProgramaPrincipal ()

(1) espectroFrq = extracciónPS (curvaLuz)
(2) listaPeriodicicidades = BúsquedaPeriodicidades(espectroFrq)
(3) info = obtenciónInfo(modelosEstructura, modelosOscilación)
```

Código 2.4: Pseudocódigo de ejecución de las distintas fases del experimento.

Este proceso se realiza en tres partes (Código 2.4): la determinación de las frecuencias propias de oscilación de las estrellas a partir de la curva de luz proporcionada por CoRoT (línea 1); el análisis de las periodicidades en dichos espectros de oscilación usando la transformada de Fourier de las posiciones de las frecuencias (línea 2), y la deducción de la densidad media, estadio evolutivo y frecuencia del modos fundamental radial de oscilación de la estrella (línea 3).

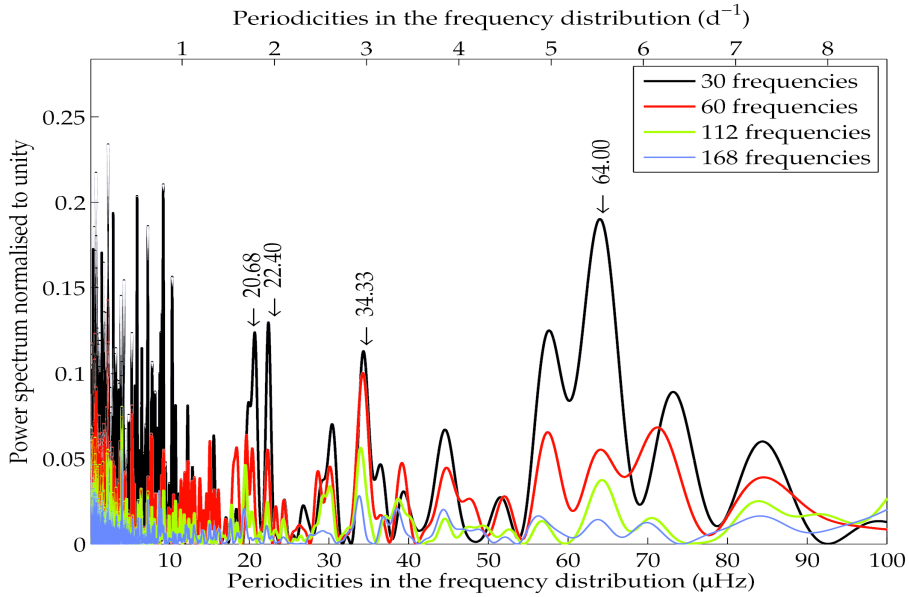


Figura 2.14: Análisis de periodicidades en la estrella HD 174966. La transformada de Fourier para un modelo de oscilación estelar sin rotación. Se identifica claramente una periodicidad en 66 μHz. Figura obtenida de [GAR13].

Cálculo de periodicidades. El cálculo de las periodicidades por el método de la transformada de Fourier, consiste en analizar el espectro de Fourier resultante de considerar sólo las posiciones de las frecuencias. Ello se consigue asumiendo que todos los modos tienen la misma amplitud igual a la unidad. Los picos resultantes son, por tanto, una medida de las periodicidades encontradas en la distribución de las frecuencias. Estas periodicidades variarán lógicamente en función del número de frecuencias usadas. Una medida de la robustez de la determinación es el análisis del comportamiento de los picos cuando se escogen los N picos de mayor amplitud del espectro de oscilación original (ver Figura 2.14). Así, línea 2 del Código 2.4 implica varios cálculos de la transformada de Fourier, en función del número de tramos elegidos.

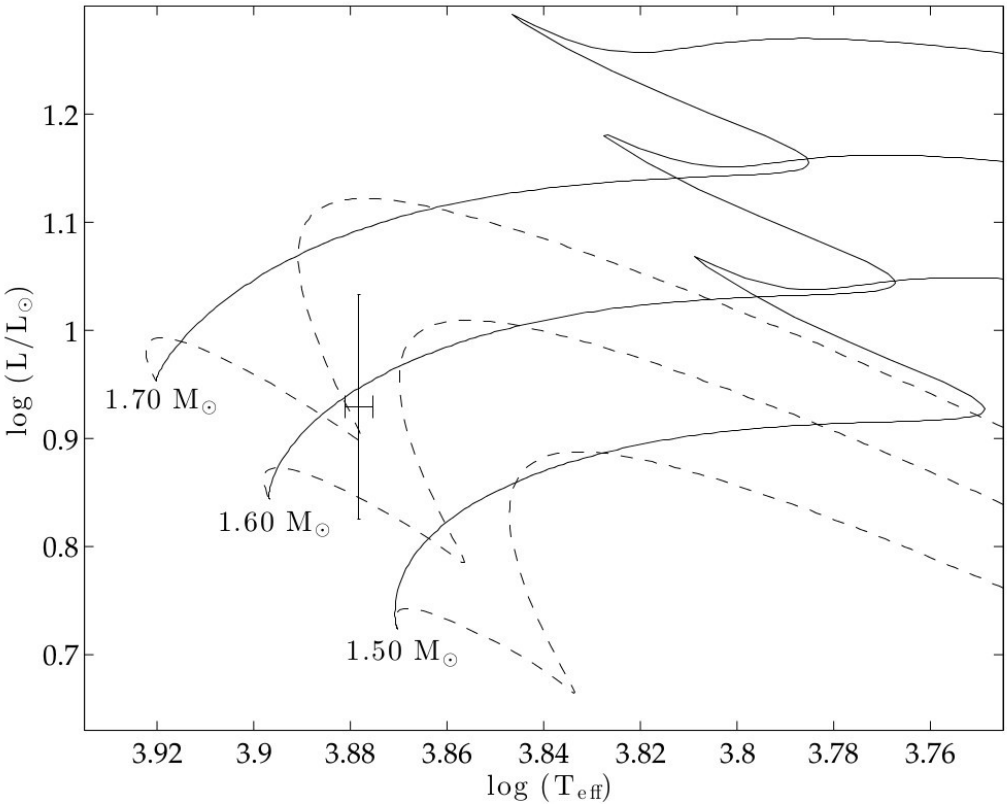


Figura 2.15: Diagrama HR que presenta trazas evolutivas para valores de masa de 1.5, 1.6 y 1.7 M . Los modelos están generados usando [FE/H]= -0.08, αML= 0.5 y dov=0.2*. Figura de la estrella HD 174966 obtenida de [GAR13]*

Deducción de la densidad media, estadio evolutivo y frecuencia del modos. Para obtener esta información generamos el modelo evolutivo mediante un modelo de equilibrio por cada paso evolutivo, empezando por un modelo base en un tiempo inicial hasta evolucionar a una determinada edad, obteniendo la traza evolutiva. Cada paso temporal de la traza parte del modelo que se obtiene a partir de la evolución del paso anterior. La Figura 2.15 representa gráficamente este proceso a través de un diagrama *Hertzsprung-Russell (*HR) que muestra el resultado de varias observaciones mediante la relación entre la magnitud absoluta de una estrella y su temperatura efectiva.

A partir de estos modelos se pueden aplicar las oscilaciones estelares que interesen en este estudio. La idea básica del proceso es perturbar los modelos de equilibrio dados por lo modelos obtenidos anteriormente para calcular los modos de pulsación que se propagan en el interior de la estrella. Cada oscilación estelar es independiente del resto, lo que facilita la ejecución paralelizada de cada uno de los casos.

Las dependencias entre tareas de las fases del experimento se muestran en la siguiente figura:

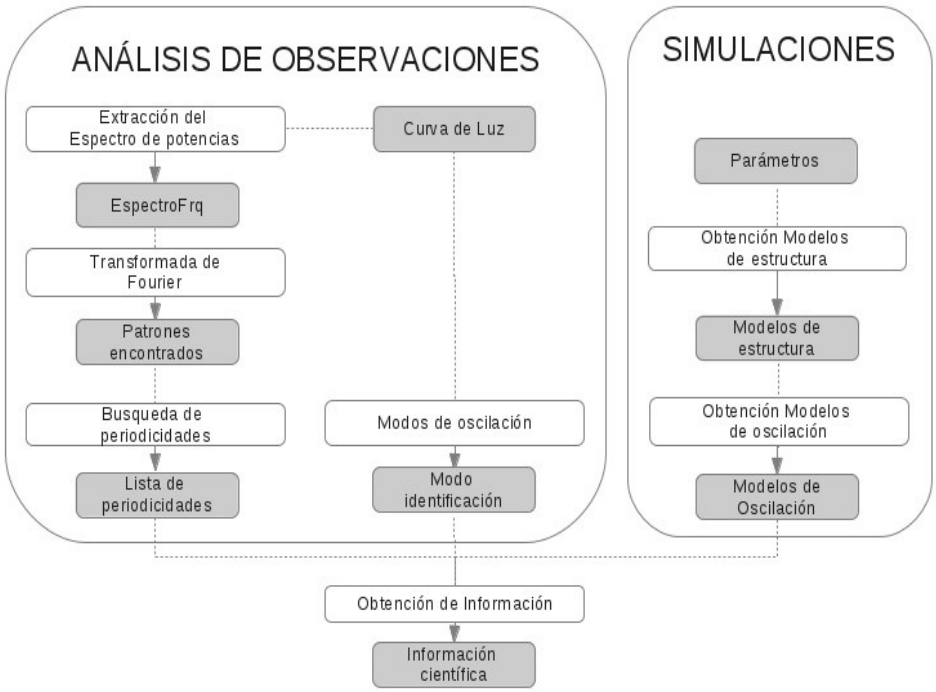


Figura 2.16: Diagrama de dependencias entre las distintas fases de experimento.

Análisis computacional.

Después de haber analizado las diferentes fases del este estudio astrosismológico mediante ejecuciones de tipo test en la infraestructura descrita en la Sección 2.2.2, hemos llegamos a la conclusión la única fase que necesita gran capacidad de cómputo es la generación de modelos de oscilación, ya que tiene un orden de complejidad cuadrática, $O(n^2)$.

Esta fase es costosa computacionalmente ya que, aunque un modelo no requiere demasiado tiempo de computación, los parámetros de entrada que definen los modelos son numerosos y su variación importante, dando lugar a un alto número de combinaciones. Por esta razón es aconsejable una infraestructura de altas prestaciones para su realización. Y, como se ha indicado, una plataforma Grid puede permitir el aprovechamiento del paralelismo que ofrece el acceso a un número apreciable de núcleos de procesamiento y recursos de almacenamiento.

Como en el momento en que abordamos este caso uso, la malla de modelos de equilibrio ya estaba calculada, nos centramos aquí en la computación sus modelos de oscilación asociados[3].

Simulación de modelos oscilación mediante *GraCo*

La simulación consiste en la elaboración de una malla equivalente a la calculada para los modelos de equilibrio pero con los modelos de oscilación correspondiente a cada uno de dichos modelos. La malla por tanto tendrá el mismo espacio de parámetros (Tabla 2.15). En promedio, GraCo invierte para un cálculo típico de oscilaciones sobre un modelo de estructura estándar, t_0, unos 300 segundos. Ello haría inviable el estudio científico propuesto. Sin embargo el uso de Grid permitiría la paralelización suficiente para cubrir el cálculo completo de la malla en un tiempo razonable.

El problema aparece cuando, una vez obtenida la red de modelos de equilibrio, aplicamos el código GraCo para obtener modelos de oscilación. GraCo no se puede paralelizar mediante técnicas como el uso de librerías MPI. Por tanto, es aconsejable usar dicho método de adaptación para el uso de esta aplicación en la plataforma Grid.

3 Debido al corto tiempo de computación de los modelos de equilibrio (alrededor de 4-5 minutos por modelo), las plataformas distribuidas no eran imprescindibles para esta parte del experimento.

Trabajo completo. Un trabajo completo en la fase de los modelos de evolución consiste en realizar modelos de oscilación sobre trazas evolutivas representativas de estrellas de tipo A-F (Figura 2.15). Para definir el cálculo del tiempo de ejecución de un trabajo completo, T_c, utilizamos el tiempo para la generación de las pulsaciones, G_p que es el tiempo necesario para para aplicar oscilaciones a una traza evolutiva. Se obtiene multiplicando el tiempo medio de procesamiento de una ejecución de *GraCo* sobre un de un modelo de equilibrio (en el caso no adiabático), t_0, y el número de modelos ejecutados (cada uno de ellos asociado a un paso evolutivo) a lo largo de una traza evolutiva, s , como se nuestra en:

$$G_p = s \cdot t_0 \quad (2.7)$$

Por otro lado, para conseguir el número de trazas evolutivas, Nt_{ev}, que se generan en un trabajo completo multiplicamos el número de posibles valores de los parámetros de la masa de la estrella, metalicidad, parámetro de convección y overshooting como se muestra (2.8) .

$$Nt_{ev} = N_M \cdot N_{[Fe/H]} \cdot N_\alpha \cdot N_{dov} \quad (2.8)$$

Por tanto, el tiempo de cálculo estimado para la ejecución del trabajo completo, T_c, al que nos hemos referido antes, se obtiene multiplicando el número de trazas evolutivas, Nt_{ev}, por el tiempo para la generación de las pulsaciones, G_p, que se estiman en un modelo evolutivo completo.

$$T_c = Nt_{ev} \cdot G_p \quad (2.9)$$

Trabajo de evaluación. El tiempo de cálculo estimado para este trabajo,T_{ev}, se obtiene multiplicando varios valores:

- El tiempo de la generación de un paso evolutivo, G_p .
- El número de valores posibles de la masa de la estrella, N_M(ev), siendo un único valor para el caso del trabajo de evaluación.
- El número de valores posibles de la metalicidad, el parámetro de convección y overshooting que no varían con respecto a los calculados para el trabajo completo.

- El número de repeticiones de la evaluación del trabajo, N_r.

$$T_{ev} = N_r \cdot N_M(ev) \cdot N_{[Fe/H]} \cdot N_\alpha \cdot N_{dov} \cdot G_p \quad (2.10)$$

Método de gridificación. Para la realización del trabajo completo se requiere una versión estable de *GraCo*. Los modelos no están definidos solamente por los parámetros utilizados sino también por el código que los utiliza. De esta forma, en la base de datos se pueden encontrar modelos de oscilación sobre una misma estrella utilizando los mismos parámetros para su realización, pero generados con códigos distintos. Los usuarios pueden beneficiarse de comparar las distintas técnicas para el modelado, y por esa razón el código debería ser el mismo para el trabajo completo.

Por este motivo se ha optado por un método de gridificación estático (Sección 2.1.2), ya que el código no cambia a lo largo del proceso de ejecución. Además, debido a la disponibilidad de un nodo de computación con la aplicación *GraCo* y las librerías necesarias pre-instaladas, se consigue un ahorro de tiempo considerable ya que no se precisa una instalación para cada uno de los trabajos lanzados a la infraestructura.

Resultados experimentales

Para el análisis de los tiempos en cada una de las situaciones, hemos empleado dos tipos de arquitecturas con diversos recursos y procedimientos de ejecución: una plataforma de computación distribuida Grid, frente a un servidor de propósito específico. (Sección 2.2)

Usando el servidor descrito en la sección 2.2.2, se ha estimado un $t_0 = 300$ segundos para una ejecución de *GraCo* en el caso no adiabático a partir de un modelo de equilibrio, y una media de $s = 200$ modelos asociados a pasos evolutivos a lo largo de una traza evolutiva. Por tanto, mediante (2.7) se estima que se necesita $G_p = 60.000$ segundos para la generación de las pulsaciones en una traza evolutiva completa.

Tabla 2.15: Rango de los parámetros que definen un trabajo completo para la realización de modelos de oscilación de una estrella utilizando GraCo.

Parámetro	**Valores**	**Número de casos**
Masa (M)	[1,25 a 2,20] Pasos de [0.2]	95
Metalicidad (Fe/H)	[-0.52 a 0.08]. Pasos de [0.2]	4
Parámetro de convección (α_{ML})	[0.5,1,1.5]	3
Overshooting (d_{ov})	[0.1, 0.2, 0.3]	3

El número de trazas evolutivas generadas en un trabajo completo se determinan mediante (2.8), obteniéndose un total de Nt_{ev} = 3420 trazas utilizando para el cálculo los valores de la Tabla 2.15

El tiempo estimado de computación según (2.9), para la realización del trabajo completo aproximando el tiempo para cada traza evolutiva a 60.000 segundos es de 57.000 horas (aproximadamente 6 años y medio). Este dato confirma que sin la adaptación de este caso científico a la plataforma Grid no hubiese sido posible la realización de éste en un tiempo reducido.

Tabla 2.16: Rango de los parámetros que definen un trabajo de evaluación para la realización de modelos de oscilación de una estrella utilizando GraCo.

Parámetro	**Valores**	**Número de casos**
Masa (M)	Valor fijo a 1.25	1
Metalicidad (Fe/H)	[-0.52 a 0.08]. Pasos de [0.2]	4
Parámetro de convección (α_{ML})	[0.5,1,1.5]	3
Overshooting (d_{ov})	[0.1, 0.2, 0.3]	3

El trabajo de evaluación se repite $N_r = 5$ veces para obtener modelos cuyos valores de parámetros de metalicidad, de convección y overshooting no varíen con respecto al trabajo completo, pero fijándose el valor de la masa. De esta forma, se tienen que ejecutar un total de $Nt_{ev} = 36$ trazas evolutivas (Tabla 2.16).

De esta manera, utilizando (2.10), el tiempo de cálculo estimado para este trabajo de evaluación es de 600 horas. En el resultado obtenido se tiene en cuenta que tiempo de generación de un paso evolutivo es de $G_p = 60000$ segundos.

Usando la plataforma descrita en la Sección 2.2.1, el computo a partir de un traza evolutiva completa mediante el el código *GraCo* requiere un tiempo de 16,6 horas aproximadamente. La duración estándar de un certificado Grid es de 12 horas y una vez alcanzado este límite, los trabajos Grid enviados se cancelan. Este escollo se soluciona mediante la utilización de un *proxy* de larga duración dado por un servidor de *proxys* de Grid, que permite alargar de esta forma el límite de tiempo para la generación de modelos en la infraestructura[4].

Por otro lado, es necesario que los modelos de equilibrio estén disponibles en el nodo de computación elegido. Como estos modelos tienen un tamaño mayor a 2 Mbytes, se ubican en un SE cercano al CE. Otros elementos necesarios para la ejecución, servicio SandBox [FOS02b], que no requieren de un servicio añadido de SE.

Una vez se está ejecutando un trabajo Grid, éste hace una petición al SE de un modelo de estructura estelar. A continuación se ejecuta la aplicación *GraCo* y el modelo generado se almacena de nuevo en un directorio especifico del SE junto a ficheros auxiliares que proporcionan datos de ejecución.

Las 36 ejecuciones Grid del trabajo de evaluación son asimilables por un CE de unos 400 núcleos (Sección 2.2.1). En el caso ideal, es decir en una ejecución sin errores de transferencias o ejecución, el tiempo de ejecución del trabajo palelizado en la plataforma Grid sería 36 veces menor, pero hay que tener en cuenta el coste temporal de enviar los trabajos Grid a la plataforma.

Tabla 2.17: Tiempos medios para el procesamiento del trabajo de evaluación ejecutado en un servidor dedicado y en una plataforma Grid lanzando el total de los trabajos Grid (36). Las unidades están en segundos. Tiempo total: tiempo desde el inicio hasta el fin de la ejecución. Tiempo medio de un trabajo: para el servidor es un único trabajo serializado y para la plataforma Grid un trabajo Grid. Tiempo total de cálculo: tiempo total de computación, en el caso de Grid se suma todos los tiempo de cada nodo.

	Servidor dedicado	**Plataforma Grid**
Tiempo total	2.199.875	67.731
Tiempo medio de un trabajo.	2.194.431	61.356
Tiempo total de cálculo.	2.194.431	2.223.937

4 Esta solución requiere el uso añadido del servidor de *proxys* y, por consiguiente, genera un aumento de complejidad en las conexiones entre los distintos módulos que actúan pero no presenta ninguna modificación en las prestaciones dadas al usuario de la plataforma

El tiempo total del trabajo de evaluación en Grid coincide con el tiempo del trabajo Grid que más ha durado. El tiempo total de cálculo en un servidor dedicado es aproximadamente el valor de multiplicar el tiempo medio de un trabajo Grid con el número de trabajos enviados a la plataforma. En este análisis vemos que se cumple este requisito.

Evaluación en el servidor dedicado frente a la plataforma de computación distribuida Grid (IberGrid). En el estudio de tiempos de computación del trabajo de evaluación ejecutado en distintas plataformas se envía la totalidad de trabajos Grid a la plataforma de una sola vez. Si observamos la comparación entre la ejecución del trabajo de evaluación en un servidor dedicado frente a la una infraestructura Grid (Tabla 2.17), podemos concluir que el uso de la plataforma Grid reduce aproximadamente a un 3% el total del tiempo medio de ejecución de *GraCo* ejecutado en un servidor dedicado.

Tabla 2.18: Tiempo medios, mínimos y máximos de un trabajo Grid perteneciente al trabajo de evaluación definido para el caso de GraCo. La unidades están en segundos.

	PW	CE	TT	IS*	EX	IR	TOTAL
Mínimo	197,0	73,0	632,0	-	51.293,0	195,0	54.074,0
Máximo	302,0	171,0	795,0	-	64.787,0	352,0	67.356,0
Media	255,6	122,8	704,6	-	60.407,8	277,8	61.356,0
Desv. Estándar	42,3	40,2	68,9	-	5.825,2	56,4	5.090,0

Analizando con más profundidad el trabajo de evaluación en una plataforma Grid, observamos en la Tabla 2.18 que la fase de ejecución (EX) es la que consume la mayoría del tiempo del trabajo Grid. Es justamente esta fase la que se paraleliza en un entorno Grid, obteniendo de esta forma un rendimiento mayor.

Tabla 2.19: Estado a la terminación de los trabajos Grid enviados a la plataforma.

Trabajos Grid enviados	36*5=180
Trabajos ejecutados con éxito	169
Trabajos terminados con error	11

Para un estudio de sobre el porcentaje de trabajos de GraCo fallidos ejecutados en una plataforma Grid, hemos enviado un el trabajo de evaluación compuesto por 180 trabajos Grid. Observamos que el porcentaje de trabajos fallidos se encuentra entorno a un 6% del total (Tabla 2.19). Este porcentaje es asimilable teniendo en cuenta los beneficios en tiempo de computación que conlleva usar este tipo de plataformas.

Tabla 2.20: Clasificación de errores en la ejecución de los trabajos Grid.

Trabajos con resultado erróneo	0
Trabajos sin asignación de recursos (CE)	3
Trabajos con fallo en la conexión (WMS)	5
Trabajos con ejecución errónea (WN)	0
No catalogados	3

Además, vemos en la clasificación de los errores (Tabla 2.20) que la naturaleza de éstos es muy diversa y no está definida su causa. Si bien, los trabajos cuyo error de ejecución se debe a fallos en la conexión son los más abundantes (alrededor de un 45% de los errores totales).

Después de obtener los datos de los tiempos de ejecución, observamos que la ganancia real obtenida es de $A_{real}=32{,}47$ (aplicando 2.1).

Sabemos que el factor de mejora es de $A_m=36$, ya que es el número máximo de cores usados en la plataforma Grid que es igual al número de trabajos enviados. Según la ley de Amdahl (2.1) debe ser mayor o igual que la real. Sabemos que $F_m\approx 1$ y por tanto cota de la ganancia es $A_{cota}\approx 36$ (aplicando 2.1). Cumpliéndose la ley de Amdahl (2.1): $(A_{real}=32{,}47)\leq(A_{cota}=36)$

En lo que se refiere a la utilización del almacenamiento en Grid, el uso de la plataforma permite tener disponible todos los modelos de estructura estelar necesarios para la obtención de los modelos de oscilación. Esto reduce considerablemente el tiempo de transferencia de los archivos ya que una vez conocida su ubicación usando el servicio de catálogo Grid, no es necesaria ninguna transferencia.

Tabla 2.21: Tabla que resume el rango de valores de los parámetros físicos de HD 174966 obtenidos utilizando los diferentes métodos discutidos en trabajo publicado en [GAR13].

Método	Fotométrico	Espectroscópico	Usando Δν
$T_{eff}(K)$	[7427, 7847]	[7505, 7605]	[7505, 7605]
$\log g$	[3.82, 4.25]	[4.16, 4.26]	[4.21, 4.23]
$[Fe/H]$	[-0.32, 0.08]	[-0.18, +0.02]	[-0.18, +0.02]
α_{ML}	[0.5, 1.5]	[0.5, 1.0]	[0.5, 1.0]
d_{ov}	[0.1, 0.3]	[0.1, 0.3]	[0.1, 0.3]
$M(M_{\odot})$	[1.35, 2.20]	[1.49, 1.58]	[1.50, 1.53]
$R(R_{\odot})$	[1.46, 3.02]	[1.50, 1.73]	[1.55, 1.61]
$L(L_{\odot})$	[5.87, 30.94]	[6.47, 8.92]	[6.8, 7.50]
$\bar{\rho}(g\,cm^{-3})$	[0.113, 0.61]	[0.43, 0.62]	[0.51, 0.57]
$Age(My)$	[434, 2244]	[826, 1306]	[926, 1206]
X_c	[0, 0.7373]	[0.4105, 0.5676]	[0.4835, 0.5352]

Resultados de la ejecución del trabajo completo. Gracias al análisis que hemos realizado se han obtenido los valores físicos más exactos posibles de los parámetros de las estrellas δ Scuti analizadas (Tabla 2.21), observadas por el satélite CoRoT. Además hemos obtenido un patrón periódico en los espectros de frecuencia con el fin de usarlo para determinar la densidad media de cada una de las estrellas.

Se confirmó la periodicidad en el espectro de frecuencias denominada gran separación como un patrón (Δν) que puede ser usado como un observable astrosismológico para estrellas tipo A-F.

La adaptación de los diversos códigos usados en este caso científicos han sido esenciales para la realización tanto de una Tesis doctoral [GAR11] como de varias publicaciones [GAR09, MAN12, PAP13, GAR13, GAR15]. Además el conjunto de las trazas evolutivas generadas han servido para otro tipo de estudios sobre estas estrellas [SUA14]. De otra forma no hubiera sido posible la realización de estos estudios en un tiempo reducido.

Además, hemos desarrollado la herramienta TOUCAN [SUA14] para la gestión de estos modelos en el Observatorio Virtual [IVOA14].

2.3.3 Estudio de la luz en partículas marcianas con *Ddscat*.

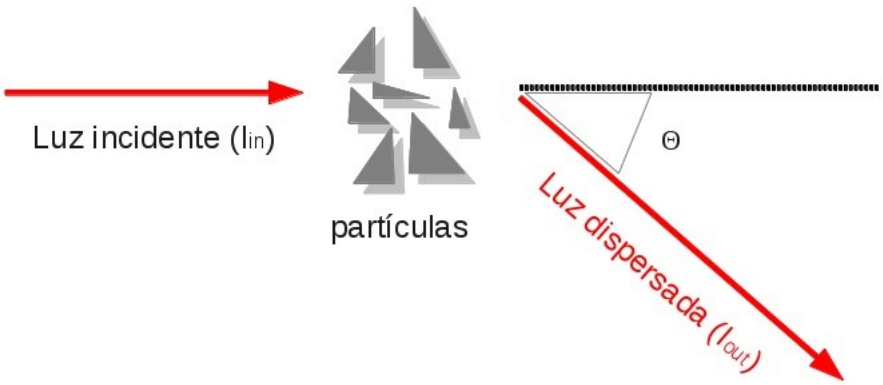


Figura 2.17: Descripción del fenómeno de dispesión.

La aplicación *Ddscat* (The Discrete Dipole Approximation for Scattering and Absorption of Light by Irregular Particles) [DRA04, DDS14] simula la luz dispersada por las partículas de minerales irregulares y la absorción por partículas aisladas (por ejemplo, partículas de polvo), como se puede observar en la Figura 2.17. Este tipo de estudios tiene aplicaciones en el campo de la fotónica y de los estudios de series de nanoestructuras. Concretamente, en el campo de la astronomía estudia el comportamiento de la luz que se dispersa en un conjunto de partículas minerales irregulares análogas al polvo marciano.

Las muestras de partículas marcianas suelen presentarse en un rango amplio de tamaños y presentan formas complicadas, heterogéneas, con diferentes formas, tamaños y cualidades físicas con un angulo de dispersión dado (Θ) [DAB14, MOR07]. Estas características implican un número elevado de grados de libertad que se traduce en numerosas operaciones, por tanto esta aplicación requiere una potencia de cálculo significativa.

Fases del experimento

Este estudio se divide principalmente en dos tareas. La primera es un análisis experimental sobre distintas muestras de calcita (partículas análogas al polvo marciano) [MUÑ10], y la segunda consiste en desarrollar simulaciones que se ajusten a los resultados experimentales para diversas estructuras de partículas; El experimento trabaja en un rango de entre 0.488 y 0.647 μm del espectro visible, aunque el método puede aplicarse a otras longitudes de onda, ampliando así el tamaño y tipo de partículas, por ejemplo partículas del espacio interestelar que dispersan la luz en el infrarrojo.

Cada partícula analizada se caracteriza por su geometría, por su parámetro de tamaño (que depende de la longitud de onda, λ, y de su radio efectivo, r[5], tal y como se muestra en (2.11)), el ángulo de incidencia de la luz, y sus propiedades de refracción debido a su composición.

$$x = \frac{2 * \pi * r}{\lambda} \quad (2.11)$$

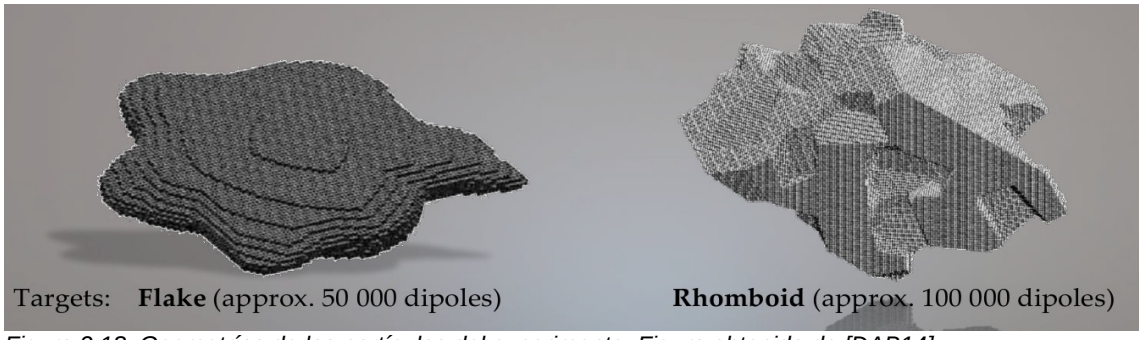


Figura 2.18: Geometrías de las partículas del experimento. Figura obtenida de [DAB14]

Estudios previos [MOR70, PET99, MER13, WOL06], muestran que los resultados de experimentos con simulación de partículas con geometrías básicas, como esferas, esferoides, elipsoides o cilindros no se ajustaban correctamente a los datos experimentales. Por esta razón, en este experimento se analizan quince geometrías distintas, cinco en forma de romboide y diez en forma de escama (Figura 2.18). Cada partícula está definida por un número de puntos y cada uno de los puntos corresponde un dipolo. Las distintas geometrías se han obtenido eliminado dipolos aleatoriamente de una figura esférica de partículas.

5 Se define como la mitad de la distancia entre dos núcleos del elemento unidos por un enlace covalente puro sencillo.

Se utilizan 50.000 dipolos de tipo romboide y 10.000 de tipo escama. Dependiendo de la propiedad de refracción del material, se distinguen tres casos que se enumeran a continuación, siendo *i* el índice de refracción de la partícula:

1. material isotrópico $m_1 = 1.644 + 0_i$ *(u.a.)*
2. material isotrópico $m_2 = 1.485 + 0_i$ *(u.a.)*
3. material birefrigente

Tabla 2.22: Comparativa de rangos de los parámetros utilizados en el análisis experimental frente a los distintos casos de simulaciones.

	Análisis experimental	**Simulaciones**
Nº geometrías	15 (5 romboides, 10 escamas)	15 (5 romboides, 10 escamas)
Radio efectivo	[0.1 – 50] µm	[0.1 – 1.2] µm
Longitud de onda	0.488 – 0.647 µm	0.647 µm
Parámetro de tamaño	0.97 – 485.31 (u.a.)	0.97 – 11,62 (u.a.)
Orientación	Ver Tabla 2.23	Ver Tabla 2.23

Tabla 2.23: Número de orientaciones dependiendo del radio efectivo de la partícula analizada. Radios truncados en 1.20 µm.

R (µm)	**Número de orientaciones**
0.10	512
0.15	512
0.20	512
0.25	512
0.30	576
0.35	576
0.40	729
0.45	729
0.60	810
0.65	900
0.75	1485
1.00	1485
1.10	1728
1.20	1728

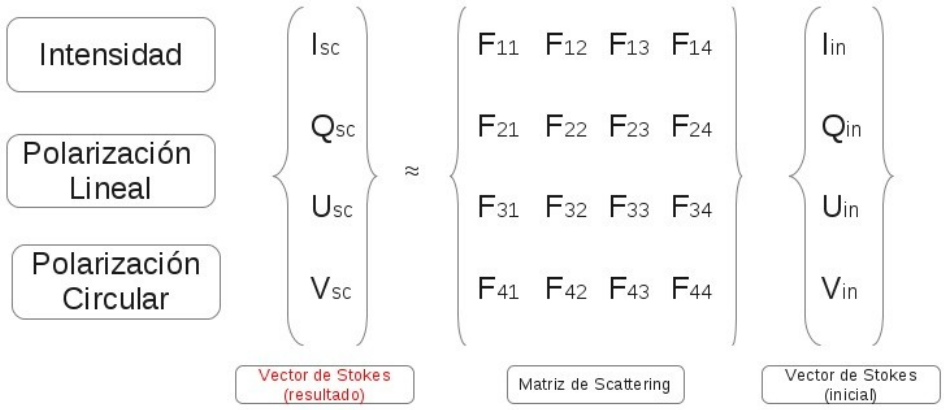


Figura 2.19: Descripción teórica del fenómeno de dispersión (Matriz de dispersión).

La finalidad del experimento es obtener la matriz de dispersión o s*cattering (*Figura 2.19). La luz dispersada y la luz incidente están relacionadas a través de dicha matriz de dimensión 4x4 que depende de la forma, el tamaño, y la composición definida a partir del índice de refracción del conjunto de las partículas, y la longitud de onda de la luz (Tabla 2.22 y Tabla 2.23).

Tabla 2.24: Componentes del haz de luz incidente o dispersada.

I	Intensidad. (u.a)
Q_{sc} y U_{sc}	Polarización lineal. (u.a)
V_{sc}	Polarización circular. (u.a)

La matriz de dispersión contiene dieciséis componentes. Estas componentes son la base de la comparación entre los resultados experimentales y las simulaciones obtenidas como se muestran en las figuras 2.22 y 2.23. Cada haz de luz, tanto el incidente como el dispersado, se describe por 4 componentes que se muestran en la Tabla 2.25.

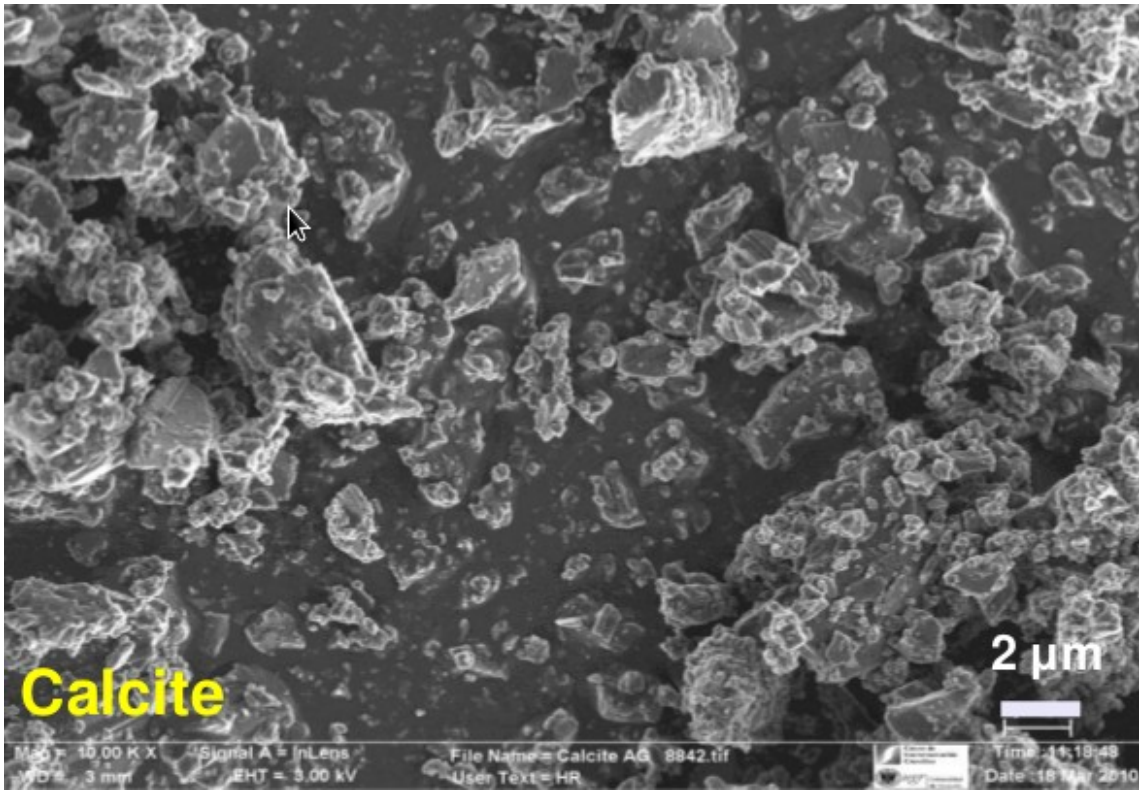


Figura 2.20: Muestra de partículas de calcita. Imagen obtenida de [DAB14]

Para el caso del análisis experimental se han utilizado muestras de calcita, como la que se observa en la Figura 2.20. Este material se ha elegido por ser similar a las muestras de polvo de Marte, tanto en su tamaño, alrededor de 1,7 µm, como en su propiedad birefringente, que permite que, una vez que la luz incide en la partícula, genere dos haces diferenciales de polarización. Esta característica se debe a que la velocidad de la luz es diferente para cada una de ellas. Una de las cualidades más importantes del código *Ddscat* es que permite simular partículas birefringentes.

En el caso de la realización de modelos mediante simulaciones, el número de

valores analizados de los parámetros pueden diferir respecto a los del análisis (Tabla 2.22) ya que el análisis para cada uno de los valores necesita una gran carga computacional para obtener la matriz de difracción a partir de estos modelos y no se dispone de infraestructura computacional suficiente.

Teniendo como referencia el servidor descrito en la Sección 2.2, los cálculos aquí descritos necesitan ser adaptados a una plataforma Grid para que éstos puedan ser ejecutados en un tiempo razonable para la investigación propuesta.

Tabla 2.25: Rango de los parámetros que definen un trabajo completo.

3 casos, N_c	material isotrópico m1 = 1.644 + 0i(indice de refracción) material isotrópico m2 = 1.485 +0i material birefrigente
15 geometrías, N_g	5 geometrías romboidales 10 geometrías en escamas
12 radios r	[0.10, 0.15, 0.20, 0.25, 0.30. 0.35, 0.40, 0.45, 0.60 ,0.65, 0.75, 1.00]
X orientaciones N_O	Depende del tamaño de la muestra: [512, 512, 512, 512, 576, 576, 576 729, 729, 810, 900, 1485, 1485]

Trabajo Completo. Los parámetros que definen un trabajo completo para este experimento son el tipo de material, su geometría, su radio efectivo y su orientación. Los rangos considerados se dan en la Tabla 2.25.

El tiempo de computación de un trabajo completo, T_{ec}, se estima teniendo el cuenta que:

- El número de casos para el estudio es N_c=3.
- El número de geometrías descritas es N_g=15.
- El tiempo de computación para cada orientación, T_o, varía dependiendo del tamaño de la muestra. Usando el servidor descrito en la Sección 2.2.2, se ha estimado un tiempo medio de procesamiento de un minuto. T_o= 60 segundos.
- El número de orientaciones para los 12 radios es N_O= 9914.
- La fórmula utilizada para su cálculo es:

$$T_{ec} = N_c \cdot N_g \cdot N_O \cdot T_o \quad (2.12)$$

Hemos llegado a la conclusión que T_{ec}=26.767.800 segundos, aproximadamente un año.

Trabajo de evaluación. El trabajo de evaluación conlleva 5 repeticiones para el procesamiento el caso de una partícula de material birefrigente para una geometría romboide (un único caso) y 12 radios a analizar correspondientes a 9914 orientaciones. El tiempo de estimado computación para el análisis una caso de una partícula de material birefrigente para una geometría romboide aproximando el tiempo de ejecución para cada orientación, T_o, a 100 segundos es de 991.400 segundos (275,38 horas).

Ddscat es una aplicación relativamente estable, es decir, el cambio de versiones no es muy frecuente. Por esta razón para la realización del trabajo completo de este esta aplicación se ha optado por el uso de la Gridificación estática.

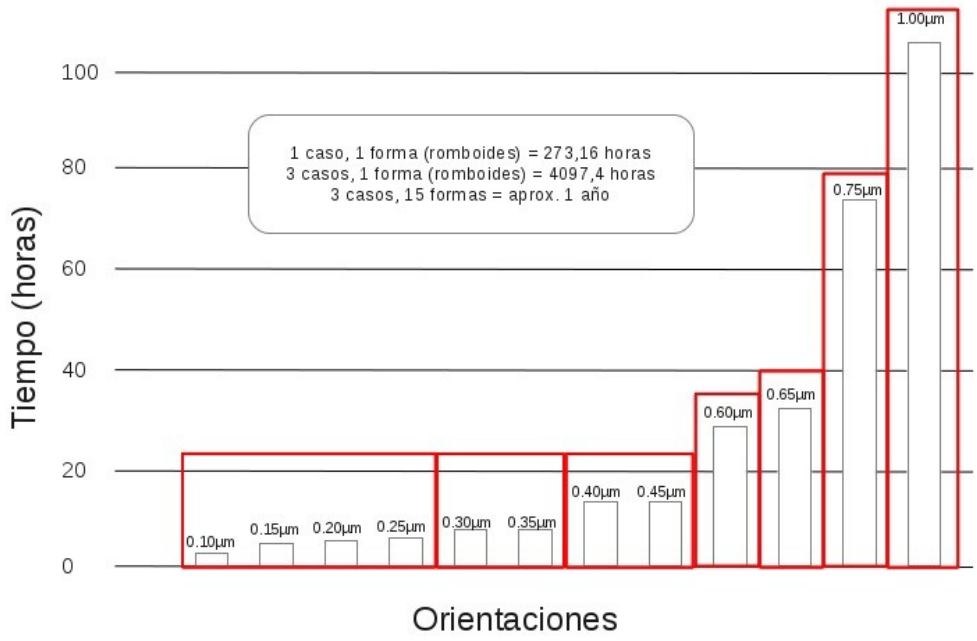


Figura 2.21: Distribución de las ejecuciones de la simulación en trabajos Grid. Gráfico obtenido de [DAB14].

Trabajo Grid. Para un caso concreto, con una geometría definida, abarcando los doce

radios efectivos previstos, y teniendo en cuenta que el número de orientaciones aumenta a la vez que aumenta el tamaño de la muestra y por tanto su tiempo de ejecución, se han dividido los casos a procesar en siete trabajos Grid (Figura 2.21).

Tabla 2.26: Tiempos medios para el procesamiento del trabajo de evaluación ejecutado en un servidor dedicado y en una plataforma Grid. Los datos para una geometría romboide en el caso de un material birefrigente. Las unidades están en segundos. Tiempo total: tiempo desde el inicio hasta el fin de la ejecución. Tiempo medio de un trabajo: para el servidor es un único trabajo serializado y para la plataforma Grid un trabajo Grid. Tiempo total de cálculo: tiempo total de computación, en el caso de Grid se suma todos los tiempo de cada nodo.

	Servidor dedicado	**Plataforma Grid**
Tiempo total	991.456,0	403.212,0
Tiempo medio de un trabajo	983,376,0	144.428,4
Tiempo total de cálculo	983.376,0	989.258,0

Si observamos la columna de los tiempo en la plataforma Grid, el tiempo del trabajo Grid que más ha durado es considerablemente mayor al tiempo total medio del trabajo Grid. Esto es debido a que el trabajo Grid correspondiente al radio de una micra requiere mucha más computación que el resto de los casos. Como este caso no se puede dividir, se le asocia un trabajo Grid a este caso y, por tanto, retrasa el tiempo total de ejecución del trabajo de evaluación. Aun así, el beneficio del uso de esta plataforma es considerable.

Análisis de tiempos. Si observamos la comparación entre la ejecución del trabajo de evaluación en un servidor dedicado frente a la una infraestructura Grid (Tabla 2.26), podemos concluir que el uso de la plataforma Grid reduce aproximadamente a un 40% del total del tiempo medio de ejecución de *Ddscat* ejecutado en un servidor dedicado.

Tabla 2.27: Tiempo medios, mínimos y máximos de un trabajo de evaluación definido para el caso de Ddscat en la plataforma Grid. La unidades están en segundos.

	PW	**CE**	**TT**	**IS***	**EX**	**IR**	**TOTAL**
Mínimo	97,0	80,0	165,0	214,0	82.923,0	150,0	83.098,2
Máximo	380,0	280,0	237,0	275,0	402.567,0	380,0	403,022,3
Media	210,8	167,2	191,2	250,2	144.428,4	238,8	144.987,3
Desv. Estándar	110,8	76,2	30,4	25,3	138.480,3	92,3	131.876,9

Hemos analizado 7 trabajos Grid. Cada trabajo de evaluación consta de 1 casos, 1 geometría (romboide) y 7 trabajos. En total se computan 7 trabajos en paralelo. La media de su tiempo total de ejecución (Tabla 2.27) es de 144.987,3 s. (40,27 horas) y el tiempo máximo de 403.022,3 s.

Analizando con más profundad el trabajo de evaluación en una plataforma Grid, en la Tabla 2.27 observamos que la fase de ejecución (EX) es la que consume la mayoría del tiempo del trabajo de ejecución. Es justamente esta fase la que se paraleliza en un entorno Grid, obteniendo de esta forma un rendimiento mayor.

Cabe resaltar que el tiempo de la fase de ejecución máximo es notablemente mayor al tiempo de la fase de ejecución medio. Esto se debe a que el mecanismo de verificación, descrito en la Sección 3.2.3, puede reenviar un mismo trabajo varias veces en el caso que se produzca un error de ejecución en la plataforma Grid.

Tabla 2.28: Estado a la terminación de los trabajos Grid enviados a la plataforma.

Trabajos Grid enviados	1575
Trabajos ejecutados con éxito	1342
Trabajos terminados con error	133

Para un estudio el estado del trabajos de evaluación de Ddscat una vez terminado hemos utilizado 5 trabajos completos con 3 casos, 15 geometrías, y 7 trabajos Grid cada uno. Por tanto se tiene un total de 1575 trabajos Grid.

Tabla 2.29: Clasificación de errores en la ejecución de los trabajos Grid.

Trabajos con resultado erróneo	25
Trabajos sin asignación de recursos (CE)	37
Trabajos con fallo en la conexión (WMS)	19
Trabajos con ejecución errónea (WN)	38
No catalogados	14

Todos los trabajos terminados con errores se reenvían completar la ejecución de forma correcta. Los trabajos con final erróneo se han clasificado tal y como se indica en la Tabla 2.29.

Con este estudio, hemos determinado que existe un 8,4% de errores en el envío de trabajos Grid. Nuevamente, se trata de un porcentaje asumible para la realización de este trabajo de evaluación. Además, la naturaleza de los errores es muy diversa y no está definida su causa. Si bien los trabajos cuyo error de ejecución se debe a una ejecución errónea en el WN de destino son los más abundantes: alrededor de un 28% de los errores totales.

Después de obtener los datos de los tiempos de ejecución, observamos que la ganancia real obtenida es de $A_{real}=2{,}45$ (aplicando 2.1).

Sabemos que el factor de mejora es de $A_m=7$ ya que es el número máximo de cores usados en la plataforma Grid que es igual al número de trabajos enviados. Según la ley de Amdahl (2.1) debe ser mayor o igual que la real. Sabemos que $F_m\approx 1$ y por tanto cota de la ganancia es $A_{cota}\approx 7$ (aplicando 2.1). Cumpliéndose la ley de Amdahl (2.1): $(A_{real}=2{,}45)\leq(A_{cota}=7)$

Resultados experimentales

El trabajo de adaptación en este caso científico ha sido clave tanto para la realización de una Tesis doctoral [DAB14B] como para la realización de varias publicaciones, entre la que destaca *"Experimental and simulated scattering matrices of small calcite particles at 647nm"* [DAB13], ya que sin esta adaptación hubiera sido imposible realizar los trabajos científicos en un tiempo moderado.

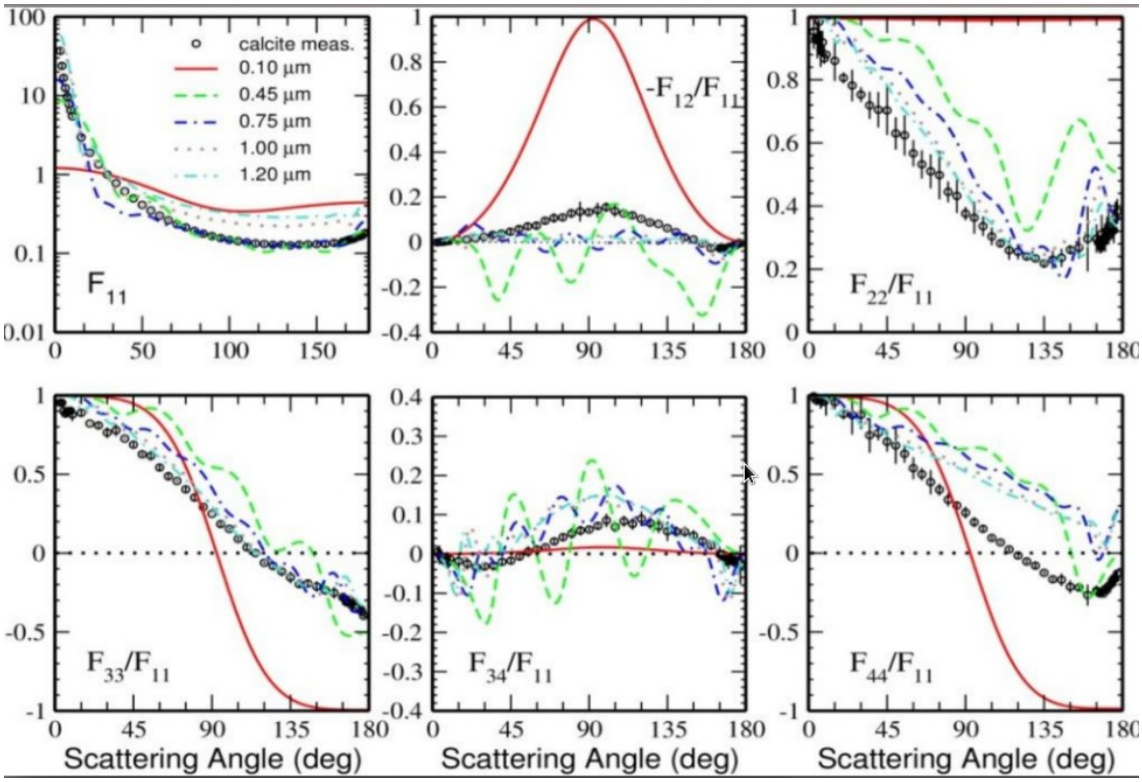


Figura 2.22: Simulaciones de dispersión frente al análisis experimental de calcita a 647 nm. Muestra de partículas de calcita para diferentes tamaños de la partícula. Figura obtenida de [DAB14]

Como resumen, en la Figura 2.22 se muestran los modelos obtenidos mediante las simulaciones. Como se puede ver, los resultados experimentales se asemejan más a los obtenidos con calcita cuanto mayor sea el tamaño de la partícula. Así, para rangos pequeños de 0.1 μm, no se ajustan bien al modelo, mientras que para tamaños de 1.2 μm, si existe una mayor concordancia.

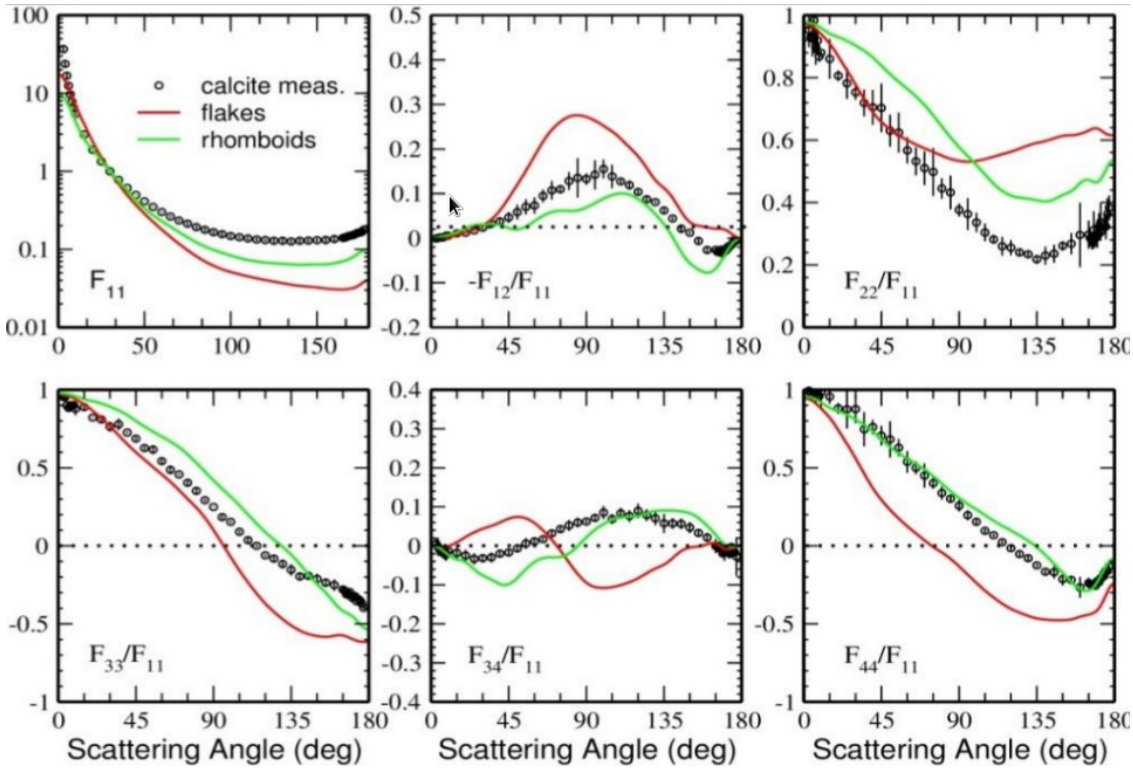


Figura 2.23: Simulaciones de dispersión frente al análisis experimental de calcita a 647 nm. Muestra de partículas de calcita para geometrías de romboides y escamas. Figura obtenida de [DAB14]

De igual forma, observando la Figura 2.23, se puede concluir que las simulaciones, tanto para geometrías de tipo romboide como para geometrías en escamas, se ajustan a los resultados experimentales en todos sus componentes, y que la unión de las dos geometrías se ajusta todavía más a los datos físicos.

2.3.4 Aplicación del método de adaptación a cada uno de los casos.

Una vez definidos los tres ejemplos de caso de uso:

- Estudio de la evolución estelar astrosismología con *GraCo* .
- Estudio de poblaciones estelares con *Starligh.*
- Estudio de la luz en partículas marcianas con *Ddscat.*

Hemos aplicando el método de implementación en la plataforma Grid (detallado en la Sección 2.1), cuyos detalles de la clasificación de los trabajos resumimos en la siguiente tabla para facilitar su comparación:

Tabla 2.30: Comparación de los parámetros para la clasificación definidos en el método de adaptación a plataformas distribuidas de los ejemplos de caso de uso estudiados en este capítulo (Starligh, GraCo y Ddscat).

Tipo	Starligh	GraCo	Ddscat
Naturaleza científica	Análisis espectral	Simulaciones de sistemas	Análisis fotométrico
Características técnicas	Nº parámetros = 5 Nº repeticiones = 5 Nº ficheros de entrada/salida = 3000 espectros individuales	Nº parámetros = 5 Nº repeticiones = 5 Nº ficheros de entrada/salida = Un fichero que contiene el modelo de equilibrio.	Nº parámetros = 5 Nº repeticiones = 5 Nº ficheros de entrada/salida = Un fichero que contiene el modelo de equilibrio.
Requerimientos computacionales	Alto	Alto	Alto
Requerimientos de almacenamiento	Alto	Alto	Alto
Factor de crecimiento	Superlineal	Lineal	Infralineal
Paralelización de los trabajos	Divisible	Divisible	Divisible
Estrategias de división del trabajo	Rangos de parámetros de entrada. N total de divisiones (T evaluación) = 300	Rangos de parámetros de entrada. N total de divisiones (T evaluación) = 150	Rangos de parámetros de entrada. N total de divisiones (T evaluación) = 7

2.4 Conclusión

En este capítulo describimos detalladamente el método propuesto en este trabajo de Tesis para el análisis y el proceso de implementación de las aplicaciones en una plataforma distribuida (en concreto en una plataforma Grid). Además, hemos analizado su uso en tres aplicaciones astronómicas diferentes.

El método que proponemos consta de unos pasos bien definidos para adaptar las aplicaciones de una manera lo más uniforme, óptima y automatizada posible. Estos pasos son los siguientes:

- Análisis de las necesidades científicas y técnicas de cada aplicación.
- Clasificación de cada trabajo dependiendo de las necesidades de recursos.
- Estudio del factor de crecimiento de la carga de trabajo de cada caso.

- Elección del entorno computacional para la ejecución de la aplicación atendiendo a las necesidades de a la aplicación.
- Elección de la estrategia óptima de la división del trabajo.
- Análisis de viabilidad de la ejecución de la aplicación en el entorno elegido.
- Envío paralelizado de trabajos.
- Uso continuado de comandos del *middleware* de Grid para la gestión de los trabajos a enviar.
- Transferencia de archivos (tanto los necesarios para la ejecución de las aplicaciones como los producidos por estas) entre los distintos elementos de las plataformas.

Como se ha dicho, este método ha contribuido a proporcionar nuevas soluciones para el desarrollo técnico y científico en diversos campos astronómicos. Contribuyendo a simplificar el acceso a la plataforma distribuida, y añadiendo de esta forma nuevas fuentes de recursos computacionales y de almacenamiento para la ejecución de diversos códigos astronómicos.

Las aplicaciones consideradas difieren entre sí en cuento a su naturaleza y necesidades. Así, hemos abordado una aplicación con una necesidad de computación continuada, como es el caso de *Starlight,* y otra aplicación, cuya necesitad no estaba sólo en la computación, sino también en los requisitos de almacenamiento debido a las grandes cantidades de datos necesarias o proporcionadas por sus ejecuciones, como es el caso de *GraCo*. O aplicaciones, como *Ddscat*, cuya necesidad depende de los parámetros de entrada, lo cual dificulta considerablemente la paralelización óptima del envío de trabajos a una plataforma Grid.

Además de la adaptación a una plataforma Grid de las herramientas citadas con anterioridad, hemos adaptado otras aplicaciones. Hemos analizado estas aplicaciones de igual manera que las descritas en la memoria, pero debido a que las soluciones de paralelización son similares a las mostradas en las secciones anteriores, hemos optado por describirlas y hacer un resumen de éstas en el Apéndice I.

Gracias a los estudios realizados, hemos llegado a la conclusión que el uso masivo de plataformas distribuidas permite disminuir considerablemente el tiempo de computación en todas las aplicaciones analizadas, además de aportar recursos de almacenamiento para grandes cantidades de datos (del orden de Tbytes).

El uso óptimo de plataformas distribuidas conlleva un envío masivo de subtrabajos creados a partir de la división de un trabajo completo. Por esta razón llegamos a la conclusión que es necesario crear una serie de herramientas que faciliten y controlen el proceso de envío, monitorización, y ejecución de todos los trabajos. Así, nació la idea de la realización el paquete de herramientas GSG, que se describe en el siguiente capítulo.

Capítulo 3: GSG, herramientas para la mejora de la capa middleware de Grid

Tras mostrar, en el capítulo anterior, el método propuesto para ayudar a la adaptación de aplicaciones astronómicas a un entorno Grid y una vez ilustrado su uso en distintas aplicaciones, se desarrolla el paso siguiente. Se trata de una utilización efectiva de este método, buscando productividad y mejora de las prestaciones computacionales y de almacenamiento de la infraestructura. Además de la posibilidad de hacer un uso más eficiente y sencillo de los entornos distribuidos por parte de los expertos en aplicaciones científicas (y más concretamente en aplicaciones en astronomía). Para ello se ha desarrollado la herramienta *General Scripts on Grid* (GSG) [ROD11]. Este paquete de herramientas utiliza la licencia GPL [GNU15], y por tanto está publicado bajo una licencia de software libre. Así pues, la información general, la documentación específica, y los códigos están disponibles en la URL:

dfe.iaa.csic.es

A través de las Sección 3.1 se resume la problemática asociada a la necesidad de un uso sencillo y rápido de los comandos utilizados para la gestión de pilas de trabajos enviadas a una infraestructura Grid y cómo GSG puede facilitar este cometido.

Posteriormente, desde la Sección 3.2.1 hasta la 3.2.7, se describen detalladamente las distintas utilidades que componen GSG especificando cada uno de los aspectos básicos de funcionamiento y la estructura de directorios usada en cada una de ellas, siendo este un punto fundamental ya que es la estructura desde donde se obtiene la información necesaria.

A continuación, en la secciones 3.2.8 y 3.2.9, se presentan una serie de utilidades añadidas a GSG. Entre ellas está el Sistema de Registros de Eventos (SRE) de gran utilidad a la hora de controlar el desarrollo de los trabajos enviados por la aplicación a una infraestructura Grid, y el Sistema de Administración Regular de Procesos (SARP) que controla la verificación del correcto funcionamiento del sistema de trabajos periódicamente.

Los destalles sobre la estructuración de los directorios, los ficheros de resultados, y los componentes necesarios para el funcionamiento de GSG se detallan en la Sección 3.3. En la Sección 3.4 se describe un ejemplo de funcionamiento de las herramientas, dejando para la Sección 3.5 un estudio de la eficiencia conseguida al usar esta plataforma.

Finalmente, en la Sección 3.6, se analizará el impacto que ha tenido el desarrollo de GSG en el ámbito de la astronomía atendiendo a las prestaciones en cuanto a tiempo de respuesta de las aplicaciones y al número de comandos utilizados para la gestión de este entorno.

A lo largo del presente capítulo se describen multitud de componentes del paquete de herramientas que utilizan acrónimos para facilitar la lectura. Se puede encontrar la definición de estos acrónimos de manera rápida en el apartado de *Glosario de Términos* ubicado al final de la memoria.

3.1 Introducción

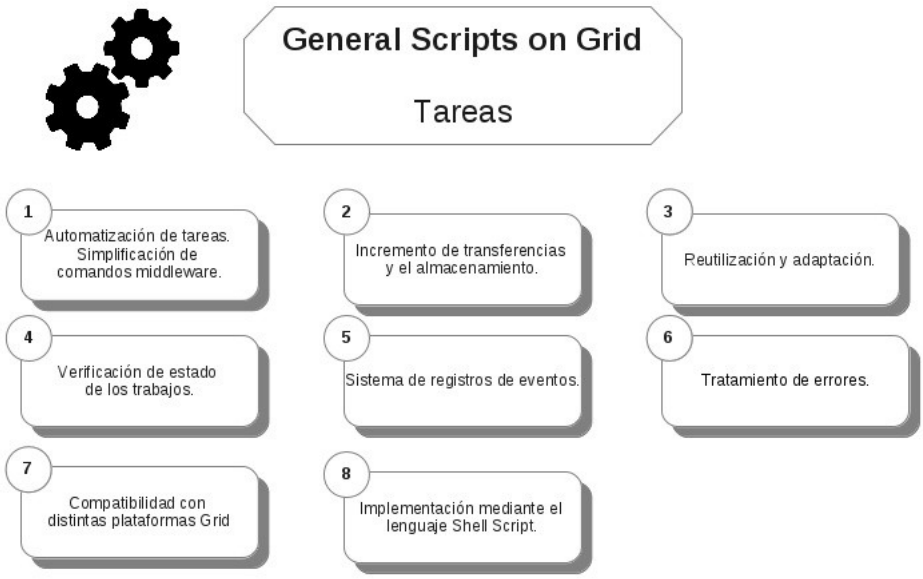


Figura 3.1: Tareas del paquete de herramienta GSG.

Entre las tareas para las que nos interesa incrementar la usabilidad y el control de aplicaciones están las que se muestran en la Figura 3.1 y se describen a continuación:

1. Automatización de tareas y simplificación de uso de comandos.

Esta tarea es necesaria ya que el uso de Grid requiere del conocimiento de muchos comandos complejos para manejar la infraestructura, ya sean de las implementaciones del *middleware* gLite [KUN05], ARC [ELL07], Unicore [ERW02] o del recientemente implementado EMI [AIF12], que la comunidad científica no suele conocer. Se requiere un tiempo de aprendizaje relativamente extenso, para asimilarlos.

Los servicios prestados por las distintas implementaciones *middleware* de una plataforma Grid se solicitan a través de una interfaz de línea de comandos (CLI, por sus siglas en inglés). Este método representa un problema para quien no posee un conocimiento técnico en este campo, ya que ha de describir, mediante estos comandos, todo el proceso de envío de trabajos a la infraestructura. Muchos de esos comandos incluyen varias opciones y secuencias predeterminadas, e incluso para el envío de trabajos se requiere un fichero de Lenguaje de Descripción de Trabajos Grid (JDL, por sus siglas en inglés). Todo esto puede tener un efecto disuasorio del uso de la infraestructura, ya que puede contemplarse como un sobrecoste añadido al trabajo del científico que, al menos en sus primeras experiencias, puede considerarse innecesario.

2. **Incremento de las transferencias y el almacenamiento de datos de diversa naturaleza dentro de un entorno distribuido Grid.**

   Para el envío masivo de datos se ha usado el procesamiento por lotes permitiendo así la ejecución de varios trabajos sin control o supervisión directa. No obstante, dado que existen algunos casos que necesitan procesarse interactivamente, también se ha incluido en GSG esta posibilidad.

   De esta manera, se consigue disminuir el tiempo de elaboración de código para el envío de trabajos y la recolección de los datos obtenidos, automatizando en gran medida estas tareas que, de otra manera, deberían realizar los usuarios. El empaquetamiento de los comandos del *middleware* gLite dentro de cada herramienta GSG permite una reducción notable de líneas necesarias para el envío masivo de trabajos.

3. **Reutilización de herramientas y adaptación inmediata y rápida a cada caso de uso.**

   GSG se ha concebido como una herramienta de carácter general, adaptable a cada caso de uso. Para ello se utiliza un módulo adaptador denominado Módulo de Acoplamiento GSG (MAG). La adaptación la realizan los administradores de cada nodo de la infraestructura Grid, facilitando así el trabajo de los científicos [ROD11].

4. **Implementación de un sistema de verificación continua y periódica del estado de los trabajos enviados.**

   Los usuarios se enfrentan a una plataforma que por su complejidad, su heterogeneidad (debido a su implementación en diferentes arquitecturas), y su dependencia de las conexiones de red (como consecuencia de su carácter distribuido), es propensa a presentar problemas de fiabilidad absoluta al servir los datos solicitados. Por tanto se requiere un sistema de verificación, continua y periódica, del estado de los trabajos enviados y de los datos obtenidos. Para ello, el SRE proporciona información sobre el estado general del sistema.

5. **Creación de un sistema de registros de eventos que facilite el intercambio de información entre la plataforma y el usuario.**

   A lo largo de todo el proceso de envío de datos, GSG tiene en cuenta lo importante que es poseer información precisa, tanto de los trabajos enviados, como del estado actual de la infraestructura. Además de obtener el recuento de tiempos y anotaciones útiles generadas al finalizar la ejecución, un usuario puede tener todo el control y la información del estado general de sus trabajos y sus archivos con una estructura perfectamente definida. Por otra parte, una vez verificados los datos, es posible conocer la ubicación de estos al anexionarse informes del proceso de ejecución en las distintas capas de una plataforma Grid.

6. **Reducción de errores producidos por circunstancias externas a la propia aplicación durante el uso de una infraestructura Grid.**

   Para reducir el número de errores debido a la plataforma, GSG contiene el SARP que consulta periódicamente el estado de los trabajos enviados y, en el caso de haber finalizado adecuadamente, analiza los datos obtenidos para comprobar si son correctos y, en ese caso, servirlos de forma transparente a la comunidad científica.

7. Compatibilidad en el uso de las herramientas en distintas plataformas Grid.

La problemática de la heterogeneidad en las plataformas Grid se resuelve en GSG gracias al uso del middleware gLite. Para resolver los problemas derivados del uso de las herramientas genéricas del *middleware* de las infraestructuras Grid, GSG incluye los principales comandos que se utilizan en el *middleware* gLite [KUN05], utilizado dentro de la plataforma Ibergrid [CAM09]. Se proporcionan así unas utilidades simples y manejables para agilizar el envío de trabajos. De hecho, la implementación de GSG para plataformas con sistemas operativos no basados en Unix no está disponible todavía al no estarlo gLite. Aunque, existen algunas alternativas para utilizar GSG en plataformas Windows, MacOS, etc, no se han considerado en el presente trabajo.

8. Uso de un lenguaje de programación para implementar las tareas anteriores.

GSG se han desarrollado utilizando programación de procesamiento por lotes (*shell script*, en inglés) [KEN84]. Se ha optado por esta opción debido a la sencillez de programación y su versatilidad, además de estar suficientemente desarrollado para la manipulación de archivos, la ejecución de programas y la impresión en pantalla. Si a esto se añade que escribir un código orientado al procesamiento por lotes es mucho más rápido que hacerlo para un código equivalente en otros lenguajes de programación, y que la selección de archivos es más cómoda, ya que la depuración puede hacerse interactivamente, se puede decir, que el uso del procesamiento por lotes ha sido la decisión más conveniente.

Las principales limitaciones de este tipo de lenguajes de programación es que no están desarrollados para soportar sistemas de manejo de datos, ni para uso de clases, multihebra, uso de matemáticas complejas, u otras características de lenguajes más desarrollados como Python[PIG04], Java [RIT93] o Perl [DOM05]. No obstante, siempre es posible abordar mediante GSG las primeras etapas de desarrollo, y convertirlo posteriormente mediante un lenguaje más potente como es Python. Este hecho se ilustra en el desarrollo del servidor de aplicaciones ATILA, descrito en el Capítulo 4.

3.2 Composición

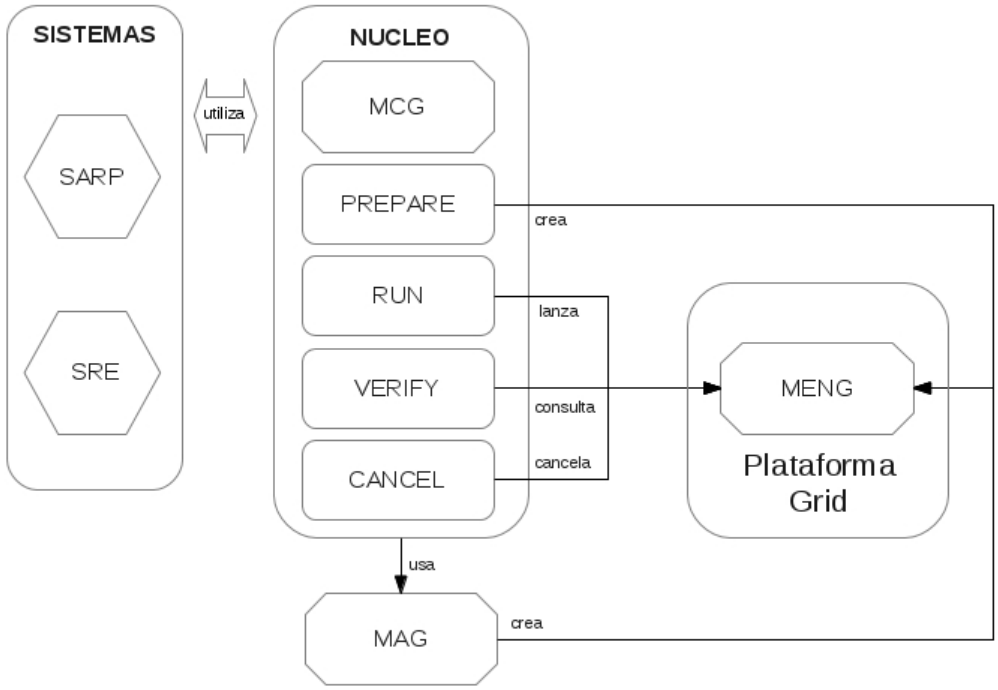


Figura 3.2: Composición del paquete de herramientas GSG

A lo largo de esta sección se describen cada uno de los elementos que componen GSG y su funcionamiento.

Tabla 3.1: Herramientas de GSG

Nombre	Breve Descripción
GSG prepare	Prepara el sistema GSG para la ejecución.
GSG run	Envía la pila de subtareas del trabajo.
GSG verify	Analiza el estado del envío en el sistema.
GSG cancel	Cancela la ejecución de las aplicaciones.

Las herramientas GSG son los componentes del paquete del mismo nombre que ejecuta el usuario desde la línea de comandos. El núcleo de GSG se basa las cuatro herramientas que se enumeran en la Tabla 3.1.

Para la ejecución de cada una de estas herramientas debe haberse ejecutado la herramienta previa de la lista de la Tabla 3.1. La ejecución de estas herramientas se detallará en la Sección 3.6 del presente capítulo.

Se denomina módulo GSG al elemento que contiene la información y los componentes necesarios para la ejecución de las herramientas GSG. Es tarea del desarrollador de los módulos (y no del usuario) definir cada uno de ellos para todos los casos. GSG utiliza tres módulos específicos.

El primero de ellos es el denominado Módulo de Acoplamiento GSG (MAG), que proporciona la utilidades necesarias para el uso de las herramientas en cada uno de los casos particulares, siendo éste el único módulo que se debe modificar en GSG según los códigos.

El segundo, cuyo nombre es Módulo de Ejecución en Nodos Grid (MENG), contiene los comandos necesarios para la ejecución mediante la aplicación original de los trabajos enviados a cada nodo Grid. Además prepara el entorno Grid para dicha ejecución.

El tercero es el Módulo de Configuración GSG (MCG), cuyo cometido es el de especificar las variables de configuración necesarias para usar las herramientas GSG.

Por otro lado, se definen los sistema GSG como los elementos que actúan sobre GSG de forma global durante todo el tiempo de ejecución de cualquier herramienta. Las herramientas GSG utilizarán dos sistemas GSG de apoyo:

El Sistema de Registro de Eventos (SRE) que es el encargado de procesar la información sobre quién ejecuta, qué se utiliza, cuándo se realiza, qué nodos de ejecución Grid se están utilizando, y las posibles anotaciones en la ejecución de cada una de las herramientas o módulos de GSG durante el tiempo de ejecución.

También se utiliza el Sistema de Administración Regular de Procesos (SARP), para ejecutar procesos de verificación de los estados de los trabajos enviados a una infraestructura Grid y actuar en consecuencia. Estos procesos se ejecutan de manera regular en un intervalo de tiempo, configurable en el MCG. Además, este módulo se basa a su vez en el SRE para la presentación de los resultados de las verificaciones.

Finalmente, el núcleo de los elementos de GSG consiste en cuatro aplicaciones generales implementadas en Shell Script [KEN84]. Éstas se ubican dentro de los directorios personales del servicio Grid denominado User Interface [FOS02b], que se describen en la Sección 3.3 del presente capítulo.

Para explicar cada uno de los elementos que definen el sistema se ha optado por utilizar un ejemplo real de utilización del paquete en una aplicación de astronomía. En concreto, se considera la aplicación *GraCo* [MOY08] para la generación de

modelos de oscilación astrosismológicos que ya se consideró en el Capítulo 2. A continuación se describen con detalle todos los elementos que se han presentado en esta sección.

3.2.1 GSG prepare

Figura 3.3: Funciones de GSG prepare.

Las principales funciones de GSG *prepare* se muestran en la Figura 3.3 y se detallan en los siguientes subapartados:

Autentificación

Para acceder a los recursos de una plataforma Grid es necesario un certificado de autentificación, personal e intransferible. Es un mecanismo de seguridad que garantiza un uso correcto de los recursos de una infraestructura Grid. Sin embargo, a su vez constituye un escollo para el envío automático de trabajos.

Figura 3.4: Certificado digital Grid Personal.

Así pues, se plantean dos estrategias para la autentificación mediante GSG. La primera es el uso de un certificado personal, de tal forma que cada usuario se autentifica con su propio certificado. Para ello es necesario solicitar este certificado a una Autoridad de Registro (RA, por sus siglas en ingles). En las plataformas ubicadas en España se trata de IRISGridCA [PKI14] perteneciente a RedIris [RED14]. En la Figura 3.4 presenta la pantalla inicial una vez que el sistema verifica que el certificado es correcto.

Una vez validados los datos por la RA, se pone el certificado a disposición de usuario. Por otro lado, se debe pertenecer a una Organización Virtual (VO, por sus siglas en inglés), debiendo ser adaptado por el administrador de la misma, tras justificar el uso la plataforma ante la organización mediante la realización de un informe de necesidades. Cada vez que se haga uso de recursos de una plataforma Grid, el usuario debe usar su certificado.

Como se ha dicho, esta situación podría interpretarse como una dificultad para acceder al recurso. Muchos usuarios creen que el proceso burocrático de la petición de certificado Grid a una VO resulta largo y tedioso. No obstante, hay que destacar que este proceso no se puede ni se debe eliminar ya que atentaría contra la seguridad del sistema. En cualquier caso, en esta Tesis también proponemos una segunda alternativa para aliviar este problema sin atentar a la seguridad.

Esta segunda opción consiste en el uso de Certificados Robots [BAR09], y es la solución adoptada para el envío masivo de trabajos usando GSG. La responsabilidad de la seguridad de los recursos Grid no recae sobre el usuarios sino sobre GSG. Es decir, el certificado se emite automáticamente por la herramienta, evitando el uso de un certificado personal. Así pues, queda a cargo de GSG la autentificación mediante contraseñas. Para ello se usa un sistema de autentificación con bases de datos centralizadas. En realidad esta estrategia es una alternativa pensada para utilizarse esporádicamente en el sistema Grid, y es más aconsejable usar la primera estrategia, ya que es la manera oficial de garantizar un uso seguro de la plataforma. En cualquier caso, GSG implementa las dos alternativas.

Debido al uso de certificados de autentificación, GSG se encuentran con otro posible escollo: el tiempo de validez del certificado. Por defecto, el certificado de autentificación Grid expira a las doce horas. La finalidad de esta restricción es que el sistema de colas de cada WN no se vea monopolizado por trabajos de un solo usuario. El problema se encuentra cuando existen trabajos de mayor duración que, al expirar los certificados, son abortados automáticamente. Para solucionarlo se aprovechan nuevamente los Certificados Robot. Si se detecta que los trabajos son de larga duración se enviarán a una cola especial en el WN ubicado en centro correspondiente. Ésta es una solución local para los problemas planteados al usar los servicios Grid en los centros configurados. En otros nodos de Grid, dependerá de la política de uso de las colas de computación.

Verificación del estado general del sistema Grid

La siguiente tarea consiste en analizar el estado general del sistema Grid, y sus recursos, en el instante previo al envío de las subtareas que forman un trabajo. Después se actúa en consecuencia, notificando el estado y las acciones realizadas mediante un correo electrónico.

Según sea el estado, se puede visualizar el número de recursos disponibles, el grado de fiabilidad de cada uno de ellos y las prestaciones que proporcionan. De esta forma se puede planificar una estrategia efectiva de envío de subtareas.

Tabla 3.2: Listado de comandos gLite para la obtención de información sobre el estado del sistema.

Comando Glite	Información obtenida
lcg-infosites -- vo “vo”	Presenta la información de los recursos disponibles para la VO elegida.
lcg-info -- list-ce -- query	Lista los elementos de computación (CE) disponibles que satisface determinadas condiciones (query)
lcg-info -- list-se -- query	Lista los SEs disponibles que satisface determinadas condiciones (query).
lcg-infosites --vo “vo” closeSE	Lista todos los elementos de computación (CE) y almacenamiento (SE) cercanos entre sí.
lcg-ManageVOTag -host “host” -vo “vo” --list	Lista todas las aplicaciones etiquetadas (tags) que se encuentra en un nodo en concreto (host) perteneciente a una organización virtual (vo).

Mediante la utilización de comandos del sistema de información del middleware, gLite, obtenemos las listas de los Servicios de Computación Grid (CE, por sus siglas en inglés) y los Elementos de Almacenamiento Grid (SE, por sus siglas en inglés) disponibles para una determinada VO. Además, también se puede acceder al listado del las aplicaciones instaladas en cada uno de ellos, los SEs cercanos físicamente a un determinado CE, y demás información útil para la planificación el envío de trabajos a una plataforma Grid. Los comandos del middleware gLite [GLI14] utilizados para este cometido se muestran en Tabla 3.2.

División del trabajo a ejecutar en subtareas

Seguidamente, para obtener el máximo rendimiento de una plataforma distribuida, como es una infraestructura Grid, es necesario paralelizar la ejecución de los códigos. La estrategia aplicada por esta herramienta es la de dividir el trabajo en subtareas independientes entre sí, y enviándolos para su ejecución simultánea en los WNs pertenecientes a una determinada VO.

Para planificar la estrategia de envío de subtareas es necesario saber como dividir los trabajos. Para ello se necesita información que debe proporcionar el sistema. De esta forma podemos plantear que, cuantos más recursos de computación libres existan, más trabajos podemos enviar. Aunque si el sistema de computación de un nodo está saturado, no sería recomendable enviar trabajos a ese nodo, y quizá sea mejor optar por la estrategia de enviar menos subtareas pero de mayor duración. Cada solución depende de las necesidades y la naturaleza de cada aplicación (en nuestro caso, las aplicaciones astronómicas).

Además, teniendo en cuenta la limitación a la que nos hemos referido antes respecto al tiempo de validez del certificado de autentificación, *GSG prepare* se encargará de dividir el trabajo de forma que las subtareas no tengan una duración estimada mayor que el tiempo de validez del certificado (por ejemplo, 10 horas para un tiempo de certificado de 12 horas).

```
./prepare.sh [m_min] [m_max] [m_delta] [z_min] [z_max] [z_delta] [a_min] [a_max]
[a_delta] [o_min] [o_max] [o_delta] [rot_min] [rot_max] [rot_delta] [working_area]
```

Código 3.1: Código de ejecución de la herramienta prepare

En el ejemplo de la aplicación *GraCo*, como parámetros de entrada en el módulo *prepare* se especifican los rangos de las variables que definen el modelo de equilibrio que usa la aplicación. La ejecución de la rutina se presenta en el Código 3.1.

Así pues, mediante esta llamada se genera ejecución para la obtención de N modelos de oscilación que difieren en la masa de la estrella (m), el índice de metalicidad (z), el índice de convección (a), el *overshooting* (o) y la rotación de la estrella (rot). De esta forma, un parámetro X (que puede ser masa, metalicidad, índice de convección, *overshooting* o rotación) se define en el rango entre los valores X_min y X_max y va variando en pasos de X_delta. Además, el conjunto de todas las ejecuciones es etiquetado por el nombre definido en el parámetro *working_area*.

La división de tareas en este caso se hace atendiendo a la creación de cada uno de los modelos. Cada modelo será ejecutado como un trabajo Grid.

Realización del fichero JDL

Para cada uno de los trabajos Grid, la herramienta GSG propone, en colaboración con el MAG, el fichero JDL que define cada una de las subtareas enviadas a la plataforma Grid de forma automática. Este fichero, cuyo ejemplo se puede ver en el Código 3.2, especifica la aplicación a ejecutar (línea 4) y los argumentos de entrada (línea 5), los ficheros de control que aportan información sobre los errores (linea 6) y sobre la ejecución (línea 7), los elementos necesarios para la ejecución (línea 8) y los resultados obtenidos (línea 9), los criterios de selección de recursos Grid disponibles en la VO seleccionada (linea 10), y el nombre que se le quiere dar al propio fichero JDL (linea 11). La realización de este fichero para cada una de la subtareas resultaría tediosa y difícil si no se conoce de antemano el lenguaje especifico para la realización de este tipo de ficheros.

Se puede observa el código generador de ficheros JDL en las siguientes líneas,

```
1. create_jdl() {
2.       echo "Type = \"Job\";" >> "$jdl_name"
3.       echo "JobType = \"Normal\";" >> "$jdl_name"
4.       echo "Executable = \"start.sh\";" >> "$jdl_name"
5.       echo "Arguments = \"$working_area".tar.gz" $wd \";" >> "$jdl_name"
6.       echo "StdError = \"GraCo.err\";" >> "$jdl_name"
7.       echo "StdOutput = \"GraCo.out\";" >> "$jdl_name"
8.       echo $set_InputSandbox >> "$jdl_name"
9.       echo "OutputSandbox = {\"GraCo.err\", \"GraCo.out\"};" >> "$jdl_name"
10.      echo "Requirements = (RegExp(\"iaa\", other.GlueCEUniqueID));" >>
11. "$jdl_name" }
```

Código 3.2: Código generador del fichero JDL.

En este fichero se incluye, además de los comandos de ejecución y los argumentos necesarios, un listado de ficheros que deben de transferirse al WN para la ejecución. Esto se hará usando el sistema SandBox [FOS02b] específico de Grid, al que ya nos referimos en la Sección 1.2.3, para el envío de ficheros entre distintos elementos.

Preparación de ficheros de entrada de la aplicación

La transferencia de ficheros de gran tamaño en la ejecución de una aplicación puede conllevar problemas debidos a que el propio sistema SandBox tiene limitado el envío de ficheros de más de un cierto tamaño (en nuestro caso, dos megabytes).

Existen dos alternativas para transferir ficheros a una plataforma Grid. La primera, a la que ya nos hemos referido, es el uso del sistema SandBox con su limitación de tamaño de fichero. Por otro parte, existe la posibilidad de ubicar los ficheros en el SE. Esta solución evita la limitación del tamaño del fichero pero incrementa el tiempo total de ejecución de los trabajos debido al tiempo necesario para ubicar el fichero en el sistema.

GSG prepare, analiza los tamaños de los ficheros necesarios para la ejecución de la aplicación y según su tamaño, los empaqueta y los ubica en el SE más cercano físicamente al CE, usando el comando *middleware* "*lcg-infosites closeSE",* donde se ejecutarán las subtareas a través de los comandos correspondientes del *middleware* gLite. En el caso de que el tamaño de los ficheros no exceda de 1 megaByte los incluye en el apartado de SandBox del fichero JDL, anexándolos al envío de la subtarea.

En el caso del ejemplo de ejecución de *GraCo*, que se detallan en la Sección 2.3.2, se contempla dos tipos de ficheros:

- Los enviados por SandBox mediante la linea de definición del fichero JDL, en la variable $set_InputSandBox definido en el módulo MAG. En el caso de *GraCo* son los ficheros:

  1. start.sh: perteneciente al módulo MENG y que contiene el código de ejecución en el nodo Grid. En caso de *GraCo* la llamada de la aplicación se hará mediante el comando "pulsaciones".
  2. frecuencia.f90: se trata de una aplicación de preprocesado para la obtención de los límites de la frecuencia del modelo.
  3. entrada_*GraCo*_Grid.dat: es el fichero de configuración de los parámetros de la aplicación *GraCo*.

  ```
  InputSandbox={\"start.sh\",\"frecuencia.f90\",\"entrada_GraCo_Grid.dat\"};
  ```

  Código 3.3: Línea del fichero JDL que refiere al sistema SandBox.

- Y los ficheros almacenados en el SE más cercano físicamente al CE. En este caso se trata del fichero que contiene el modelo de equilibrio necesario para la obtención del modelo de oscilación. El tamaño fichero puede oscilar entre 10 a 50 megabytes, y por tanto no puede enviarse mediante el sistema SandBox.

Creación de una estructura de directorios para cada una de las subtareas

Para la organización de los trabajos enviados a una plataforma Grid mediante GSG, para acceder a la información del SRE, y para ubicar los elementos que necesita la ejecución de la aplicación y los resultados que generan cada una de las subtareas de cada trabajo, es necesario crear una estructura de directorios. Así, *GSG prepare* copia los elementos necesarios tal y como se describe en la Sección 3.3, dedicada a la estructura de directorio.

Definición del MENG

A la hora de mandar trabajos a una plataforma Grid hay que tener presente que la aplicación se ejecuta en un WN que, a no ser que se especifique en los requisitos de los recursos del fichero JDL, no se conoce.

Así pues, es necesario definir perfectamente el entorno de ejecución, no sólo mediante la descripción obtenida usando el fichero JDL, sino también utilizando el MENG que define variables, procesos, configuraciones, o compilaciones previas a la ejecución de la aplicación. La descripción del módulo y su contenido se detalla en la Sección 3.2.6.

Envío de información del estado de la herramienta mediante correo electrónico

Cuando se pretende sistematizar el envío de grandes lotes de trabajos a una plataforma distribuida como es una infraestructura Grid, hay que estar informado de todos los eventos relativos al dicho envío. Cuestiones como los resúmenes de las tareas realizadas, el estado del sistema y de los trabajos, la configuración de los recursos utilizados para el envío de éstos, la ubicación de los ficheros usados, la estimación de tiempos de ejecución a lo largo del proceso, y otros aspectos a tener en cuenta, deben ser accesibles.

GSG informa, una vez terminado su cometido, de todos estos aspectos mediante la utilización del SRE, generando un informe final en formato PDF, que es enviado a la dirección de correo electrónico previamente indicada en el MCG.

3.2.2 GSG run

Esta herramienta permite enviar las subtareas que componen el trabajo, previamente tratadas por *GSG prepare*, a una infraestructura Grid. Para ello se usan los comandos de envío definidos en el *middleware* gLite. Además, *GSG run* incluye el trabajo enviado en el SARP para que éste actúe convenientemente cada vez que haya un evento.

El código de esta herramienta no depende de la aplicación usada, ajustándose así al requisito de generalidad de GSG.

El único requisito previo para la ejecución de la aplicación en una infraestructura Grid es la definición del nombre identificativo del trabajo a enviar. En el caso de la herramienta *GSG run* es "*./run.sh [working_area]*", siendo working_area el nombre identificativo definido por *GSG prepare*. Este trabajo es creado con anterioridad por *GSG prepare,* y está compuesto por las subtareas que se van a enviar. Si la herramienta no encuentra el directorio donde se ubica dicho trabajo, se notificará.

```
1. for i in $fname
   do
      cd $i/inputs/
2.    $submission_command -o job_id $jdl_name
      cd ../../logs
3.    let "subm_count += 1"
4.    echo "$subm_count" > submitted_count.log
5.    echo "$i" >> submitted_jobs.log
      echo "$i submitted"
      cd ../
   done
```

Código 3.4: Código de envío de tareas, etiquetado e información para el SRE.

Como se observa en el Código 3.4, una vez se haya enviado el trabajo a una plataforma Grid (línea 2) y se obtenga automáticamente el identificador del trabajo, se procederá al cambio de estado asociado a las subtareas, y se incrementa el número de trabajos enviados (línea 3 y 4) y la lista de trabajos enviados, definidos como *Submitted* (línea 5). Además se le asignará un identificador único al trabajo procesado, que será usado por el resto de herramientas a partir de ese momento. Esta información se incluye en el SRE.

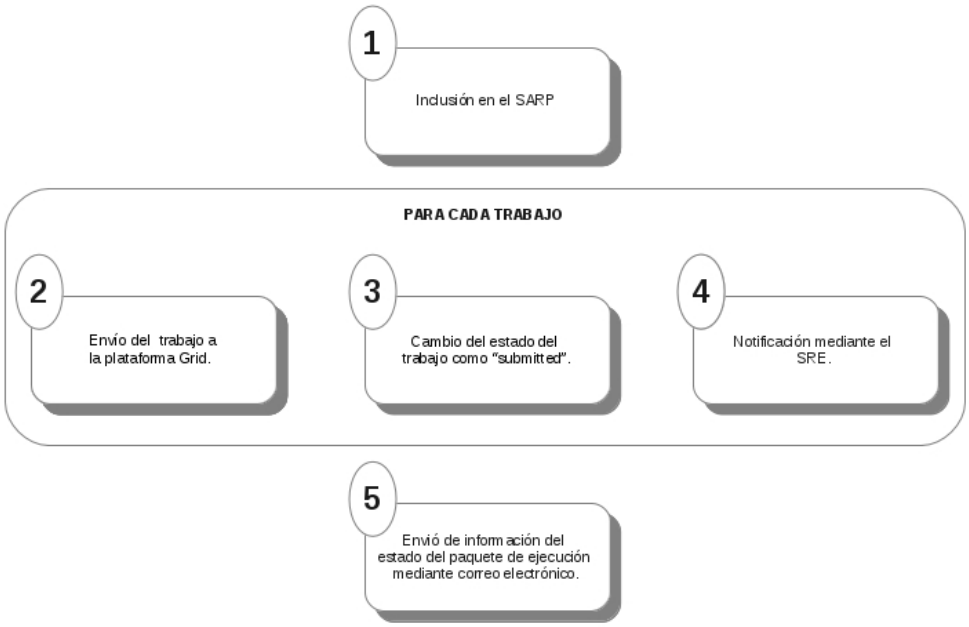


Figura 3.5: Funciones de GSG run

A continuación se detallan las funciones de GSG run que se muestran en la Figura 3.5.

Inclusión en el SARP

En el sistema al que se envían las distintas subtareas pueden ocurrir eventos muy diversos. Por ello, es importante que cualquier cambio en el estado del sistema o de los trabajos, se anote y procese adecuadamente para asegurar la estabilidad del sistema. *GSG run* incluye en SARP cada subtarea enviada, garantizando el seguimiento del estado del trabajo.

Envío del trabajo a una plataforma Grid.

El envío de un número considerable de trabajos a una plataforma Grid, puede requerir un tiempo largo en la preparación de esta tarea. Cuando los lotes de trabajos son muy extensos, es necesario un proceso de envío automático que agilice el uso de la plataforma distribuida Grid.

Precisamente el cometido de *GSG run* es enviar las subtareas del trabajo de forma automática y transparente al usuario. Para este cometido se usan los comandos de envío definidos en el *middleware* gLite, como se ve en la línea 2 del Código 3.4.

Cambio del estado del trabajo a *Submitted*.

Para un control preciso de las subtareas enviadas es necesario definir los estados de cada una de ellas. Cada subtarea debe etiquetarse de acuerdo con su estado para que cada herramienta o sistema lleve a cabo las acciones oportunas en cada caso. GSG etiqueta cada una de las subtareas enviadas como *Submitted*, usando para ello el SRE. Este estado es el inicial para todo trabajo y cuando cambie el SARP actuará en consecuencia.

Notificación mediante el SRE.

Como en el resto de herramientas, toda acción de *GSG run* debe estar incluida en el SRE. Para ello *GSG run* asignará un identificador único a cada subtarea enviada. Ese identificador se usará por el resto de herramientas a partir de ese momento.

Envío de información del estado del paquete de ejecución mediante correo electrónico.

De forma similar a *GSG prepare, GSG run* informa sobre el envío de las subtareas de un trabajo, generando un informe que incluirá cuestiones relacionadas con el tiempo de envío, el número de trabajos enviados, quién los envió, los recursos que se han asignado a cada uno de estos y su ubicación. Esta información es relevante a la hora de utilizar las herramientas que se describen a continuación.

3.2.3 GSG verify

GSG verify tiene varios cometidos y es la herramienta más compleja de todo el paquete. Entre sus objetivos se encuentra el análisis del estado de cada subtarea enviada a una plataforma Grid. También se encarga de realizar las acciones pertinentes dependiendo del estado en que se encuentre la subtarea. Ademas, el análisis de las subtareas se realiza de forma periódica utilizando para ello el SARP. Finalmente, si los trabajos han concluido correctamente, se encarga de ubicar los ficheros en el directorio indicado que se incluye en el Interfaz de Usuario (UI, por sus siglas en inglés).

Como ocurre con las demás herramientas del paquete GSG, *GSG verify* no depende de la aplicación que soporta, y el único parámetro de entrada que espera es el nombre del trabajo tratado previamente por *GSG prepare*. En el caso de la herramienta *GSG verify* es "*./verify.sh [working_area]*", siendo working_area el nombre identificativo definido por *GSG prepare*.

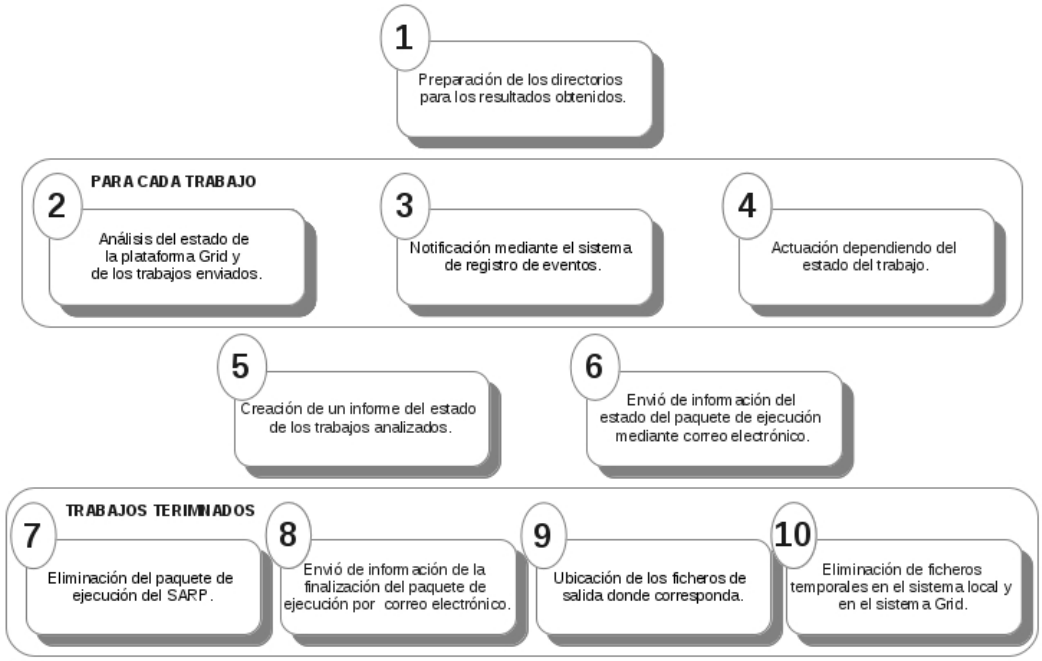


Figura 3.6: Funciones de la herramienta GSG verify.

Las tareas realizadas por esta herramienta se muestran en la Figura 3.6 y se describen a continuación.

Preparación de los directorios para los resultados obtenidos.

A lo largo de todo el proceso de ejecución de la aplicación pueden necesitarse ficheros intermedios para ubicar el estado del proceso. Tanto los ficheros generados por el SRE, como los ficheros temporales generados por la propia aplicación deben ser localizables en cualquier momento.

Para este cometido, la primera tarea de *GSG verify* es crear un sistema de directorios que contengan estos ficheros temporales y, al terminar la ejecución, los resultados de la aplicación esperados. Este sistema de ficheros se encuentra en un UI. Esta jerarquía de ficheros se detalla en la Sección 3.3 del presente capítulo dedicada a la estructura de directorios.

Análisis del estado de una plataforma Grid y de los trabajos enviados.

A lo largo de la vida de la ejecución de un trabajo, tanto el estado del sistema como el de las subtareas enviadas pueden sufrir continuas modificaciones. Se debe de estar informado de estas para obrar en consecuencia, ya sea de forma manual, o de forma automática.

Mediante el Sistema de Gestión de Carga de Trabajos (WMS, por sus siglas en inglés) incluido dentro del *middleware* gLite, *GSG verify* proporciona un informe de estado de una plataforma Grid además del de cada una de las subtareas enviadas. Esto lo hace en colaboración con el SARP que actuará de forma automática en el caso de que encuentre cambios de estado, tanto en una plataforma Grid, como en las subtareas.

Notificación mediante el SRE.

Igual que en el resto de herramientas, toda la información relevante se notifica mediante el SRE. En el caso de *GSG verify* esta información contiene el estado actual del sistema y de los trabajos enviados, los cambios efectuados en cada uno de ellos desde la última verificación, el listado de las órdenes efectuadas a consecuencia de estos cambios, un contador de tareas que se encuentran en un estado definido y, finalmente, las subtareas reenviadas o canceladas.

Actuación dependiendo del estado del trabajo.

Una vez analizadas cada una de las subtareas es necesario realizar las acciones pertinentes, de tal forma que la reubicación de ficheros, la corrección de errores, y el reenvío de trabajos sea transparente. De esta forma se consigue eliminar un proceso que, dependiendo del número de subtareas enviadas a una plataforma Grid, sería inabarcable usando métodos clásicos, como las rutinas del *middleware* gLite.

Para ello *GSG verify* proporciona, no solo mecanismos de análisis de estado de los trabajos, sino también, un protocolo de actuación para cada uno de los estados de las subtareas que se describe a continuación.

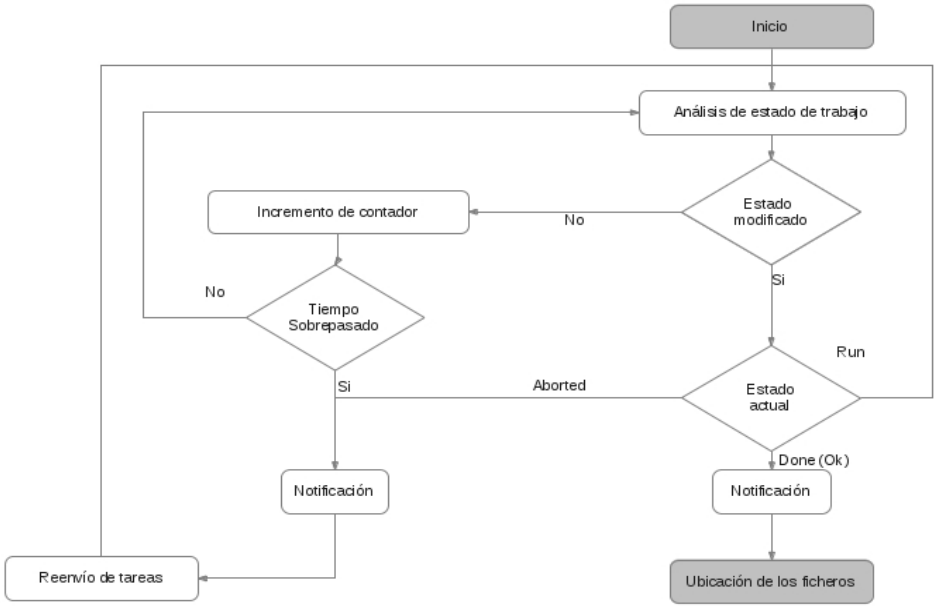


Figura 3.7: Diagrama de flujo del funcionamiento de la herramienta GSG verify.

Teniendo en cuenta el orden de ejecución de los estados que se presenta en la Figura 1.5, y al diagrama de flujo representado en la Figura 3.7, si el identificador único que reconocen cada una de las subtareas enviadas a una plataforma Grid no existe o no se encuentra disponible a la hora del análisis, eso quiere decir que el trabajo pasó al estado de *Done* o *Aborted* hace más de 48 horas, y el WMS lo ha eliminado de su base de datos. Este es un proceso relativamente frecuente cuyo objetivo es que la base de datos del WMS no mantenga indefinidamente información de trabajos enviados. El tiempo desde que un trabajo ha finalizado hasta que el WMS elimina información de su base de datos es configurable, pero no por el usuario, sino por el administrador del servicio.

Desafortunadamente, la circunstancia descrita anteriormente es muy común cuando se utilizan directamente los comandos del middleware gLite, que no tienen control sobre la terminación de sus trabajos. En la Tesis lo hemos solucionado mediante *GSG verify,* a través de la consulta periódica del estado de los trabajos. El SARP permite conocer la finalización de la subtarea en el momento en el que se produce, ya sea con resultados correctos, o con información de posibles problemas, y actúa en consecuencia antes de que el WMS elimine la información de la subtarea de la base de datos.

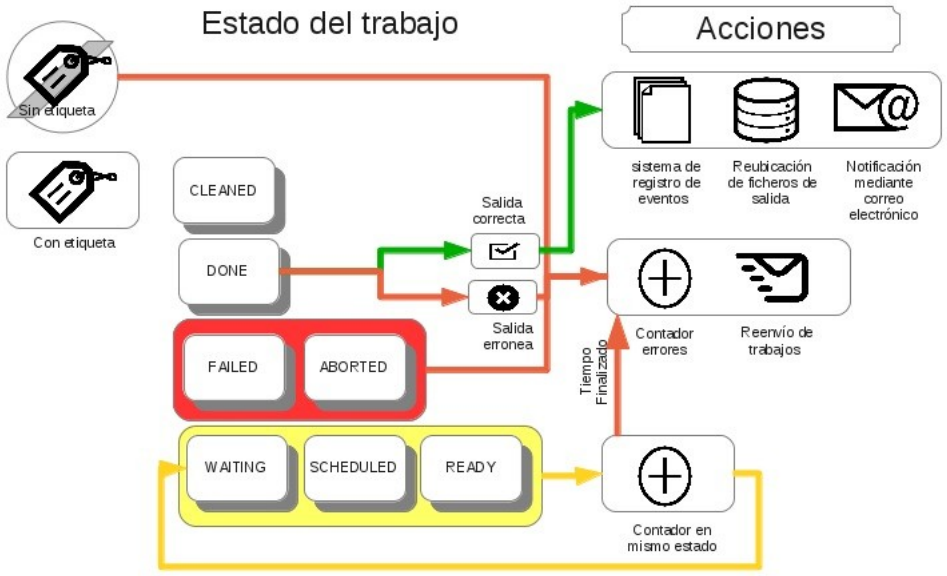


Figura 3.8: Acciones para cada uno de los estados de los trabajo GSG.

Como se puede ver en el diagrama de flujo de la Figura 3.8, si el estado del trabajo es *Cleared***,** la herramienta asume que el trabajo ha finalizado y los resultados se han recogido con anterioridad. La herramienta etiqueta el trabajo como *Skip***,** y lo elimina de la lista de trabajos a analizar por el SARP.

Si el estado del trabajo es *Done***,** las acciones de la herramienta dependerán de la correcta ejecución del trabajo. Concretamente:

Si no existen resultados, se notifica que ha habido error, se asigna el estado *Done Fail*, se reenvía el trabajo guardando el anterior identificador y se registra el trabajo con un identificador nuevo. Finalmente, el estado pasará a ser de nuevo *Submitted*.

En el caso de existir resultados, se procederá a la ubicación de estos dentro de directorio definido por *GSG prepare* usando para ello el servicio SandBox del *middleware* gLite, o en el SE, y se clasifica a través del Catálogo Grid Local (LGC, por sus siglas en inglés) posteriormente, si este fuera el caso. Seguidamente los resultados se analizarán según la aplicación, y tal y como se indica en el MAG.

En caso de que el análisis de los datos fuera correcto, pasará al estado *Cleared,* y se notificará su terminación correcta.

Si los ficheros de salida no cumpliesen los criterios de validez establecidos (como puede ser el número de ficheros de salida, su formato o su tamaño) la herramienta notifica el error y actúa de igual forma a como se describió para el caso que no existieran resultados. Esta alternativa también es la que se usa para los estados *Aborted***,** *Done Fail* o *Done* (*Cancelled*). Estos estados suelen deberse a errores en los distintos módulos o conexiones del sistema Grid.

Para los estados *Waiting***,** *Scheduled* o *Ready***,** *GSG verify* procede de la misma forma. Se sigue manteniendo el estado hasta que transcurra un tiempo especificado en el MCG, y una vez rebasado este tiempo, el sistema catalogará el trabajo como fallido, se procederá a su cancelación, se notificará esta circunstancia, y se procederá al reenvío del trabajo con un nuevo identificador y su estado pasa a ser *Submitted*. Este procedimiento se ha implementado para evitar que los trabajos se mantengan estos estados de manera indefinida. El tiempo transcurrido en dichos estados no debe ser muy elevado, ya que son estados que no están asociados a computación, y por tanto reducirían la eficiencia. De esta forma, se pasa al estado *Waiting* cuando los recursos de una plataforma Grid no están disponibles, al estado *Scheduled* cuando el WMS no ha asignado aún un identificador al trabajo, y al estado *Ready* cuando el trabajo se encuentra en CE pero no se le ha asignado un WN.

Si el número de reenvíos ocasionados por una mala ejecución, o por problemas en el sistema, sobrepasa un valor previamente configurado en el Módulo de Configuración de GSG, la herramienta notifica que el trabajo no se ha podido realizar y no lo reenvía. Este mecanismo permite evitar que un trabajo esté en el sistema indefinidamente debido a un problema que no tiene solución. Este es el caso más extremo, y su solución recaerá sobre el usuario, que puede apoyarse en la información recopilada por el sistema de SRE.

Creación de un informe del estado de los trabajos analizados.

Como en cualquier situación donde se haya realizado un proceso de análisis del sistema y de las subtareas, es conveniente tener un resumen tanto de las características del propio análisis (tiempo de inicio, número de análisis totales, etc), como del estado y de las acciones realizadas.

GSG verify, en colaboración con el SRE, realiza este informe con todos los datos que se han ido recopilando a lo largo del proceso y del uso de la herramienta. Esto se lleva a cabo creando una serie de ficheros de información por cada estado del trabajo. Estos ficheros se crean en cada uno de los WNs donde se ejecutan las subtareas. Posteriormente serán enviados a los directorios asociados dentro del UI.

Envío de información del estado del paquete de ejecución mediante correo electrónico.

Toda la información recopilada por el SRE debe remitirse al usuario por medio del correo electrónico dado que es de mucha utilidad, especialmente en esta herramienta. Para ello, igual que en todas las herramientas de GSG se envía un informe con todos los datos de la ejecución. Este informe contiene el análisis de cada uno de los trabajos, así como del sistema y de las acciones realizadas. Además, también incluye el listado de las subtareas, ordenadas por estados.

Eliminación del paquete de ejecución del SARP.

Es interesante que, una vez que terminen completamente todas las subtareas de un trabajo enviado a una plataforma Grid, se elimine cualquier ejecución periódica de procesos. Es frecuente que, si esta ejecución se lleva a cabo manualmente, acabe obviándose, y puedan quedar numerosos procesos ejecutándose periódicamente sin razón alguna.

Para evitar esta situación, y como GSG incluye el SARP, *GSG verify* procederá a la eliminación de las tareas incluidas en SARP para su análisis y seguimiento, una vez la tarea haya sido ejecutada completamente, con resultados correctos.

Envío de información de la finalización del paquete de ejecución por correo electrónico.

En el instante en el que todos las subtareas de un trabajo hayan finalizado correctamente, *GSG verify* procederá a enviar una notificación de fin de trabajo, utilizando para ello el correo electrónico.

Ubicación de los ficheros de salida donde corresponda.

Otra de las carencias que tiene el uso de una plataforma Grid es que no incluye un post-procesado después de la ejecución de los trabajos. Es función del usuario ubicar en el directorio que corresponda, cada uno los ficheros generados por cada subtarea en un WN. Mediante el fichero JDL se puede transferir desde el WN donde se ha generado, hasta el UI mediante el sistema SandBox, o transferirlo a un SE. No obstante, no es posible generar un sistema de directorios que facilite la búsqueda de los resultados.

Para resolver la carencia anteriormente descrita, *GSG verify* transfiere, cada vez que se termina una de las subtareas, los resultados generados por cada una de ellas a un sistema de directorios creados previamente por *GSG prepare*. De esta forma, se facilita la búsqueda de los resultados.

Eliminación de ficheros temporales del sistema local y del sistema Grid.

La eliminación de ficheros temporales del sistema local y del sistema Grid no es sencilla debido al envío de los trabajos de forma distribuida a diversos nodos de ejecución de una infraestructura Grid. Si no se hace de forma ordenada y meticulosa, es muy probable que muchos terminen por no borrarse.

Es cierto que el propio *middleware* gLite tiene un sistema de eliminación de ficheros temporales, en cada uno de sus WNs, que borra el directorio asociado a cada subtarea que se ejecuta en éste. Pero queda el problema de los ficheros intermedios ubicados en el SE, o en el propio UI.

Para eliminarlos de manera ordenada, GSG, una vez se ha verificado que las subtareas han terminado correctamente, procederá a la eliminación de los archivos generados por la propia ejecución de la herramienta, o por el uso de los distintos sistemas asociados a ésta. Para ello, utiliza la información de los ficheros contenidos en el SRE que se incluyó en ellos una vez que fueron creados.

3.2.4 GSG cancel

GSG cancel finaliza ordenadamente la ejecución de todas las subtareas asociadas a un trabajo gestionado por las herramientas pertenecientes a GSG. Se utiliza en casos donde, una vez empezado el proceso de computación, se detecta una configuración errónea, elementos de computación corruptos, u otra circunstancia anómala que genere resultados incorrectos. Cuando se lanza un trabajo mediante GSG, se crea un número indeterminado de subtareas que puede llegar a ser muy elevado. La eliminación mediante comandos del *middleware* de una plataforma Grid puede ser muy tediosa, y por ello, la utilización de esta herramienta es muy recomendable.

El único requisito previo para la ejecución de la aplicación en una infraestructura Grid es la definición del nombre identificativo del trabajo a enviar. En el caso de la herramienta *GSG cancel* es "*./cancel.sh [working_area]*", siendo working_area el nombre identificativo definido por *GSG prepare*.

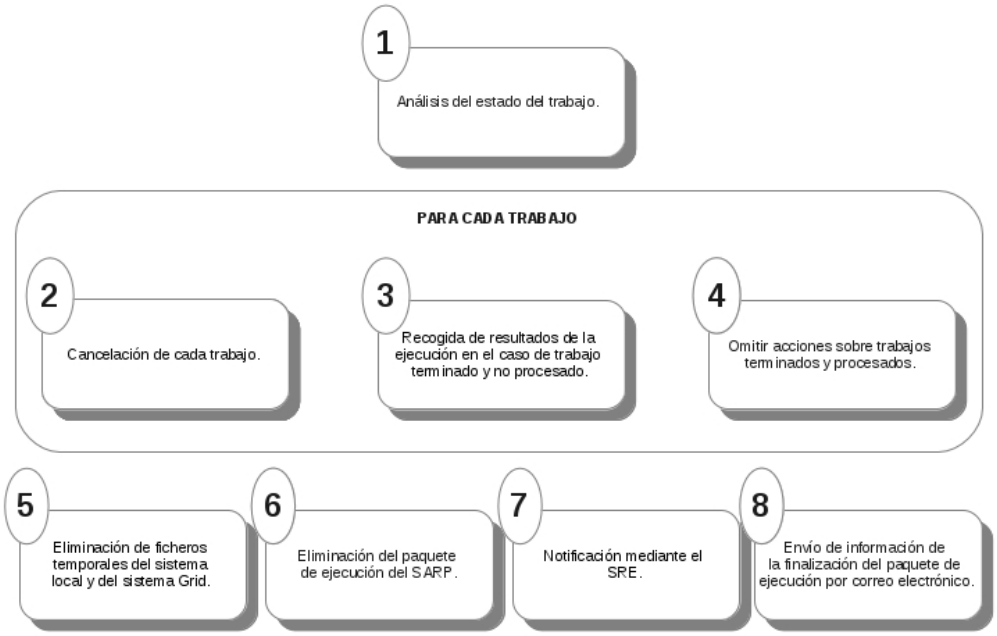


Figura 3.9: Funciones de GSG cancel

Las funciones principales de este módulo se resumen en la Figura 3.9. Aparte de la tarea principal de la cancelación de subtareas, la herramienta tiene otros cometidos como la recogida de ficheros de salida de trabajos terminados antes de la cancelación, la eliminación de ficheros temporales del sistema local y del sistema Grid, o la eliminación del trabajo del SARP.

Finalmente se notificaran las acciones realizadas mediante el SRE y mediante el envío de un correo electrónico.

Análisis del estado del trabajo.

Antes de proceder a la cancelación de un trabajo es necesario analizar el estado de ejecución de todas sus subtareas ya que, dependiendo de la información que se obtenga de este análisis, se procederá a la cancelación completa de todos los trabajos, o a una cancelación parcial de las subtareas que hayan terminado. Este proceso se debe hacer de forma automática ya que, de otra manera, el tiempo invertido puede ser demasiado elevado.

Al igual que GSG *verify*, *GSG cancel* utiliza el WMS para obtener información sobre estado de cada una de las subtareas enviadas. Una vez obtenida ésta actuará debidamente en las siguientes tareas.

Cancelación de cada trabajo.

A partir de la información del análisis de cada una de las subtareas, se cancelan todos los trabajos que se requiera. Si el proceso se hace manualmente podría haber descuidos, eliminando ficheros de salida, u obviando esta eliminación, además del tiempo requerido por esta tarea. Por esta razón, *GSG cancel* agiliza esta tarea eliminando automáticamente todos los procesos que no se encuentren en los estados *Done Ok,* o *Cleared*. Este proceso se llevará a cabo usando los comandos middleware correspondientes.

Recogida de resultados de la ejecución en el caso de trabajo terminado y no procesado.

Para los casos en que las subtareas se encuentren en el estado *Done Ok,* se puede estar interesado en recuperar los resultados obtenidos por un proceso ya terminado. Es probable que se sepa de antemano que no son correctos y se deshechen, pero también puede darse el caso de una cancelación por otros motivos, y que los resultados obtenidos sí sean relevantes.

Así pues, *GSG cancel* recupera los ficheros de salida de las subtareas finalizadas correctamente, y los ubica después donde el MCG haya estimado. Esta ubicación final puede estar en un servicio de almacenamiento de una plataforma Grid (SE) o en una ubicación en el UI, usando para ello el sistema SandBox definido en el fichero JDL.

Omitir acciones sobre trabajos terminados y procesados.

El último estado de ejecución en el que puede estar una subtarea es *Cleared*. Este estado indica que la subtarea ha terminado, ya sea manualmente, o mediante *GSG verify*. Así, la subtarea se ha procesado, y se han ubicando los ficheros de salida en el sitio pertinente, y se han actualizado los sistemas relacionados con GSG.

En este caso GSG procede a omitir acciones sobre este trabajo, ya que se ha realizado previamente una finalización ordenada, y notifica esta circunstancia al SRE.

Eliminación de ficheros temporales en el sistema local y en el sistema Grid.

Para una finalización ordenada y correcta en el caso de la cancelación masiva de trabajos, es necesario eliminar los ficheros temporales que se han podido generar a partir de la ejecución de dichos trabajos, no solo en el sistema local ubicado en el UI, sino también en cada uno de los Nodos de Ejecución de Grid o en el SE. Además, de estos ficheros, existen otros necesarios para la ejecución de trabajos y coordinación de servicios dentro de una plataforma Grid. No obstante, la eliminación de estos la lleva a cabo periódicamente el WMS.

La tarea de la eliminación de este tipo de ficheros la realiza *GSG cancel* en coordinación con el SRE que contiene el listado de los ficheros generados en cada una de las ejecuciones de las subtareas. La herramienta consulta el listado y elimina los ficheros ubicados en el UI, en el SE, o en el WN, según donde se haya sido procesada la tarea.

Eliminación del paquete de ejecución del SARP.

Después de la cancelación de las subtareas, ya no se tiene la necesidad de consultar periódicamente el estado de éstas. Así pues, el SARP no tiene que encargarse de esto. Por tanto, después de la cancelación, la herramienta eliminará el trabajo de este sistema, suprimiendo cualquier acción periódica sobre este el mismo.

Notificación mediante el SRE.

Igual que en el resto de herramientas pertenecientes al paquete GSG, toda la información relevante de la aplicación se notifica mediante el SRE. En el caso de *GSG cancel* estos registros contienen el listado de las subtareas eliminadas, de las subtareas previamente finalizadas y no procesadas (estado de *Done Ok*) y de las subtareas finalizadas y procesadas (estado de *Cleared*). Además, también están los ficheros temporales eliminados.

Envío de información de la finalización del paquete de ejecución por correo electrónico.

Después de la eliminación de todas las subtareas, *GSG cancel* procederá a mandar una notificación de trabajo finalizado utilizando el correo electrónico.

3.2.5 Módulo de Acoplamiento GSG (MAG).

La información utilizada por las herramientas GSG está dividida en dos grupos dependiendo de su carácter general o específico. La información general del paquete GSG se describe en el MCG, y la información específica de cada aplicación para la creación del MENG está contenida en el MAG.

El cometido del MAG es definir todos los parámetros y requisitos para la conexión entre GSG y las aplicaciones específicas.

Su principal tarea es facilitar el uso del paquete GSG sin que haya que modificar las herramientas que lo componen, consiguiendo así mantener su carácter general. Así, existe un módulo de este tipo por cada una de las aplicaciones que use el paquete GSG.

```
#*************************************************************
#* STRUCTURE_MODEL
#*************************************************************
1. code_used='GraCo',
2. model='#model-nad.osc'
#*************************************************************
#* PRESCRIPTIONS_TO_SEARCH_FOR_THE_MODES
#*************************************************************
3. freq_units='muhz',
4. freq_min=50,
5. freq_max=#freq,
6. l_min=0,
7. l_max=3,
8. step=0.1d0
#*************************************************************
#* PHYSICS
#*************************************************************
9. no_adiab=t,
10. mec_bc='pfinite',
11. mec_eigenfunc='pprime',
12. atm_puls=f,
13. mod_atm='edding',
14. conv_puls=f,
15. alpha_pert='gabriel',
16. rot='norot'
#*************************************************************
#*  MATH
#*************************************************************
17. rich=f,
18. int_var='lnr',
19. lawe=f
#*************************************************************
#* OUTPUTS
#*************************************************************
20. complet_ad=f,
21. complet_noad=f,
22. multicol=f,
23. rotout=f,
24. energy_out=f
#*************************************************************
#*  TEST_RUNNING
#*************************************************************
25. path=f
```

Código 3.5: Contenido de la información del módulo MAG.

El Código 3.5 presenta la definición del módulo MAG para el ejemplo de la aplicación *GraCo*. Para una mejor explicación del funcionamiento del modulo, a lo largo de esta sección se hacen las correspondientes referencias a las líneas de este código. Los parámetros definidos dentro de cada MAG son los siguientes:

Parámetros y Ficheros de entrada específicos.

Dependiendo de la naturaleza de la aplicación que se va a ejecutar, puede ser preciso conocer datos como el nombre de los ficheros, sus contenidos, sus parámetros de entrada, las variables de entorno, las librerías auxiliares y otras características necesarias para su ejecución. En el caso del ejemplo del Código 3.5, la línea 1 presenta el código y la versión usada, y la línea 2 el formato esperado del fichero de entrada. Por otro lado, desde la línea 3 hasta la 8 se presentan parámetros necesarios para la búsqueda de modos de oscilación. Específicamente, el valor de la frecuencia máxima para la búsqueda (línea 5) se determina mediante un programa llamado "pulsaciones". Otros valores relacionados con la física y la matemática utilizada por el programa se proporcionan entre las líneas 9 y 19. El número de parámetros de este tipo y su utilidad depende de la aplicación a tratar. *GSG prepare* coopera con el módulo MAG para su ejecución y para la generación del MENG.

Resultados generados por la ejecución y sus características.

El módulo define los datos esperados tras la ejecución, su número, su composición, sus parámetros y otros elementos que corresponden a un resultado correcto de una aplicación. En el caso del ejemplo, este tipo de parámetros se especifican entre las líneas 20 a la 24.

GSG verify utiliza esta información para catalogar un trabajo ejecutado correctamente en el caso de que se cumplan las características de salida definidas por este módulo. Si no se cumplen esas características, *GSG verify* indica que ha habido computación errónea y procederá en consecuencia.

Características esperadas de computación.

El módulo también define la características propias de la computación como pueden ser el tiempo medio de ejecución, las necesidades medias de memoria, computación y almacenamiento, y todas aquellas características que son útiles para GSG. En el código del ejemplo, la línea 25 especifica el directorio temporal donde se almacenan los ficheros generados. El SARP utiliza esta información para elaborar la planificación de sus ejecuciones periódicas de manera efectiva. La información especifica para el ejemplo de *GraCo* [MOY08] que está contenida en el módulo MAG, se presenta en el Código 3.5.

3.2.6 Módulo de Ejecución en Nodos Grid (MENG)

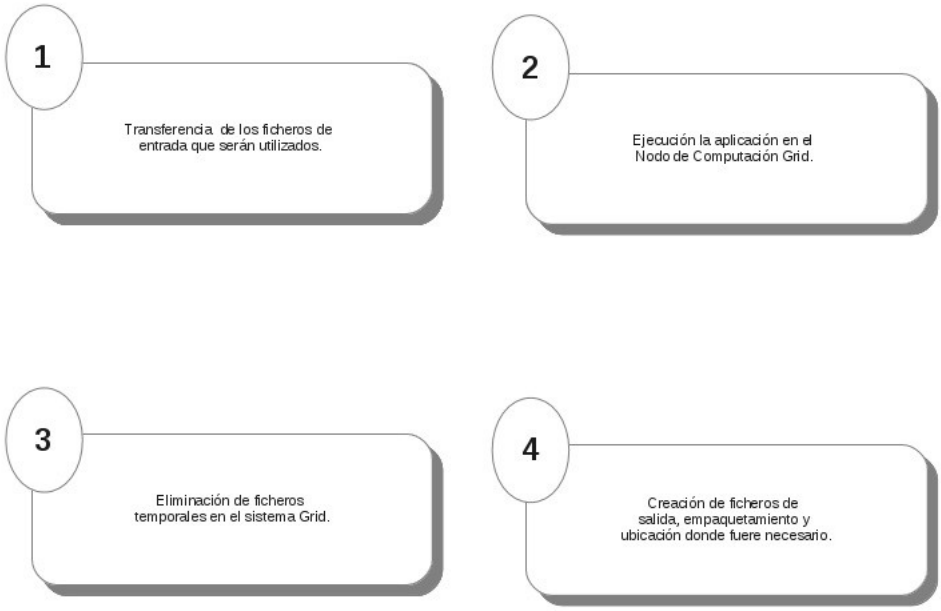


Figura 3.10: Funciones del Módulo de Ejecución en Nodos Grid.

GSG facilita la creación de este módulo atendiendo a todas las necesidades de ejecución en un nodo de computación desconocido. Sus funciones se describen en la Figura 3.10.

Para la adaptación a una aplicación específica, *GSG prepare* crea el MENG utilizado para ello el módulo MAG anteriormente descrito. El código de esta herramienta no se modifica por la aplicación usada, de esta forma se cumple el requisito de generalidad de GSG.

Cada vez que se ejecute una aplicación dentro del entorno Grid, se deben tener controlados aspectos como las variables de entorno para la ejecución, la instalación de librerías en el nodo de computación, o el propio emplazamiento de la aplicación en dicho nodo.

El mecanismo que se utiliza en las VOs de una plataforma Grid para asegurarse que dentro de sus recursos se cumplen todos los requisitos para la ejecución de un código en particular, es la definición de TAGS que identifican a ese WN como candidato para su ejecución. Se debe incluir, dentro de los requisitos del fichero fichero JDL, la condición de que el nodo contenga este TAG.

El gran inconveniente de este tipo de control es que el mantenimiento de los WNs no depende del usuario de la aplicación sino de los administradores de éstos. Cualquier modificación en las aplicaciones, como nuevas versiones, librerías o parámetros de entrada, deben notificarse a los administradores para su modificación. En un ámbito como el campo de la astronomía, donde los códigos están en continua evolución, seguir esta política no es recomendable.

Para ello GSG contiene el MENG que es el único elemento de GSG cuya ejecución se realiza en un WN. Debido a esta situación, este módulo debe ser independiente de la infraestructura donde se ejecute, es decir, los resultados no deben variar con el WN. Una infraestructura Grid se caracteriza principalmente por su heterogeneidad, que puede resultar muy ventajosa a la hora de incluir nuevos recursos, pero que debe tenerse en cuenta a la hora de controlar la ejecución de trabajos.

El MENG lo crea automáticamente *GSG prepare* que, a su vez, utiliza la información aportada por el MAG para definir las necesidades de ejecución de cada aplicación y preparar la instalación de todos los elementos necesarios dentro del nodo de computación. Estos elementos serán transferidos desde su ubicación inicial en el UI, hasta el WN correspondiente, usando para ello el sistema SandBox definido en el fichero JDL o mediante el SE en el caso de que los ficheros superen los dos megabytes. Esta limitación está definida por el propio *middleware* gLite.

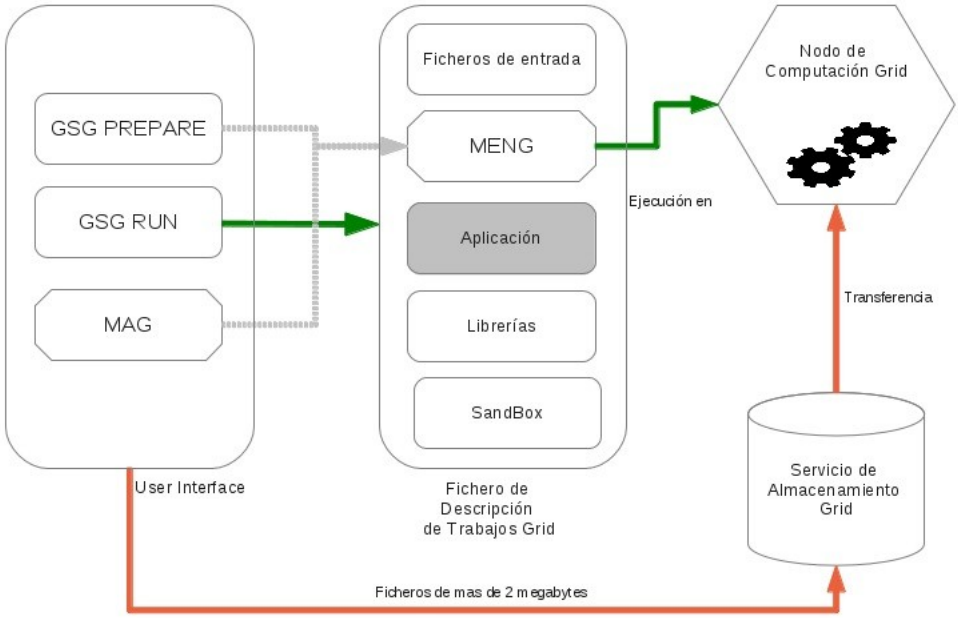


Figura 3.11: Funcionamiento del Módulo de Ejecución en Nodos Grid.

El MENG se instala en cada uno de los WNs y es transferido como fichero de entrada en el sistema SandBox definido en el fichero JDL, donde se procede a su ejecución. En su ejecución realiza las tareas descritas en la Figura 3.11.

- Transferencia de los elementos necesarios desde el SE.
- Extracción de ficheros obtenidos por SandBox o por el SE, en el caso que estén comprimidos.
- Instalación de elementos necesarios para la ejecución de la aplicación.
- Compilación, si fuera necesario, de módulos y librerías.
- Lanzamiento de la aplicación.
- Compactación de todos los ficheros generados por la aplicación, para su transferencia al SE, o al UI, mediante el sistema SandBox una vez terminada la ejecución el módulo
- Borrado de los ficheros temporales generados por la aplicación.

El único parámetro de entrada de este módulo define el nombre de la subtarea relacionada con el fichero JDL. Este nombre lo crea automáticamente *GSG prepare* y se usa como prefijo para nombrar los ficheros resultantes de cada ejecución.

3.2.7 Módulo de Configuración GSG (MCG)

El MCG define las características de ejecución de cada uno de los trabajos procesados por el paquete GSG. Se crea un módulo para cada uno de los trabajos, a partir de unos valores por defecto. El usuario puede modificarlo para adaptar las características de ejecución a sus necesidades. Este módulo tiene su ubicación dentro del directorio raíz del trabajo creado por *GSG prepare* en el UI, y todas las herramientas y los sistemas de GSG hacen uso de su información a la hora de ejecutarse, ya que ésta define el modo de actuación de cada una de las herramientas.

```
#************************************************************
#* General variables
#************************************************************
1. work_name='test_N'
2. compile =True
#************************************************************
#* General Grid variables
#************************************************************
3. export LOCAL_DIR=`pwd`
4. export LFC_HOST=lfc01.ncg.ingrid.pt
5. export my_folder=rodon
6. export my_se=se.iaa.csic.es
7. export my_vo=phys.vo.ibergrid.eu
8. submission_command="/opt/glite/bin/glite-wms-job-submit -a"
9. get_output_command="/opt/glite/bin/glite-wms-job-output"
10. status_command="/opt/glite/bin/glite-wms-job-status"
11. cancel_command="/opt/glite/bin/glite-wms-job-cancel"
12. logging_info_command="/opt/glite/bin/glite-wms-job-logging-info"
#************************************************************
#* Local variables
#************************************************************
13. source_files="GraCo.tar.gz"
14. working_home=$PWD
15. jdl_name=go.jdl
16. max_attempts=100
17. email = rodon@iaa.es
#************************************************************
#* File Paths
#************************************************************
18. input_file_path="/home/data/CESAM_EQ/"
19. output_file_path="/home/data/GraCo_OSC/"
#************************************************************
#* Working variables
#************************************************************
20. executionTime = 1
```

Código 3.6: Contenido de la información del módulo MCG.

La información configurable dentro de este módulo, cuyo ejemplo para la aplicación *GraCo* se puede ver en el Código 3.6, se clasifica de la siguiente forma:

- Parámetros de entrada generales. Entre ellos, el nombre del trabajo (línea 1) sirve para la definición del directorio base creado por *GSG prepare* y contiene subdirectorios con toda la información de cada una de las subtareas en las que se ha dividido el trabajo. Estos subdirectorios incluyen distintos tipos de ficheros, tanto de entrada como de configuración, necesarios para la ejecución de la aplicación.
- El parámetro *compile* (línea 2) indica al MENG que es necesario la compilación local de un código fuente dentro del WN. Este proceso hace que el tiempo de ejecución total en cada nodo de computación sea mayor, así que es recomendable usarlo solo cuando la arquitectura donde se ejecute sea relevante a la hora de obtener los resultados. En ese caso se debe transferir el código fuente en vez del fichero ejecutable.
- Parámetros que definen la localización de la información (líneas 3, 18 y 19), tanto local como en una plataforma Grid. En el caso de la localización de la información, en el UI se define el directorio donde se encuentran el conjunto de módulos y herramientas (working_home), además del directorio que contiene todos los trabajos enviados con GSG.
- Parámetros que definen la localización de los servicios Grid a utilizar para la ejecución del trabajo en concreto. Por ejemplo, la dirección URL donde está el Catálogo de Archivos Lógicos (LFC, por sus siglas en inglés), lfc_host (línea 4), y el directorio en concreto donde se ubicará el catálogo directorio, lfc_folder (línea 5). También se define la localización del SE por defecto, se_host (línea 6) que, en el caso de que no se indique su ubicación, correrá a cargo del WMS, que la asigna automáticamente. Por último en el módulo también se define la VO que se utiliza para el envío de trabajos, my_vo (línea 7).

- Parámetros que definen los comandos de ejecución de las distintas acciones dentro de una plataforma Grid, y según el *middleware* usado a tal efecto (líneas 8 a 12). Para las herramientas contenidas en el paquete GSG se usa por defecto los comandos del *middleware* gLite. Así pues, se utilizan los comandos de uso de éste, tanto para el envío de trabajos, la recogida de datos, el acopio de información, la cancelación de trabajos, o la visualización de los estados del sistema y de los trabajos.
- Parámetros que definen aspectos del funcionamiento de GSG, como el tiempo máximo que puede estar una subtarea en un estado intermedio (*Waiting*, *Scheduled* o *Ready)* hasta que *GSG verify* la catalogue como trabajo fallido y actúe en consecuencia (línea 23). El nombre con el que se especifica el fichero JDL (línea 16), y que es enviado por el MENG. El número máximo de intentos de reenvíos de un trabajo a una plataforma Grid se define por el nombre de *max_attemps* (línea 16). El último parámetro es el correo electrónico necesario para el envío de información por parte de las diversas herramientas o módulos (línea 17).

3.2.8 Sistema de Registro de Eventos (SRE)

GSG contiene el SRE que permite un mayor control sobre las subtareas de los trabajos que se han enviado a una plataforma Grid, y sobre el propio estado de las herramientas. Este sistema genera ficheros de información complementarios a los avisos producidos por los propios servicios de una plataforma Grid y provenientes de comandos propios del *middleware* usado. La principal ventaja de GSG es que evita las carencias informativas usuales en las herramientas estándares de los servicios de las plataformas Grid.

El SRE se estructura en tres capas :

La *capa superior*, agrupa la información generada por el SARP sobre aspectos generales de los sistemas, módulos y herramientas, incluyendo los estados, la notificaciones y los esquemas de tiempo.

La *segunda capa*, incluye la información general de cada trabajo. Esta información se genera en colaboración con *GSG verify* que, a su vez, utiliza este sistema para definir sus acciones. En concreto, la información a que nos referimos es:

- El resumen del estado actual de todas las subtareas.
- El listado de las subtareas que se encuentran en un determinado estado (status_jobs.logs). Hay un fichero por cada posible estado con la colección de trabajos que se encuentran en el estado específico, el número total de dichos trabajos, el porcentaje sobre el total de los trabajos enviados, y el tiempo que cada trabajo lleva en ese estado. Con esta información se comprueba si los trabajos están mayoritariamente en un estado concreto para obrar en consecuencia. Por ejemplo, si se observa que la mayoría de las subtareas están en estado *Waiting* puede concluir que no existen recursos suficientes en ese momento para la ejecución de sus trabajos. De esta forma podría evaluar si los requisitos solicitados, tanto de capacidad de cómputo como de almacenamiento, son correctos.
- La información sobre la ejecución de cada una de la herramientas, o módulos de GSG (scripts.log), que se usa para analizar el correcto funcionamiento de GSG.

Finalmente, la *última capa* es la más especifica para describir el estado de cada subtarea. Está compuesta por los estados por donde ha pasado la subtarea y el tiempo requerido en cada uno de ellos (steps_status), y la información específica del estado de la subtarea (job_status.log)

3.2.9 Sistema de Administración Regular de Procesos (SARP)

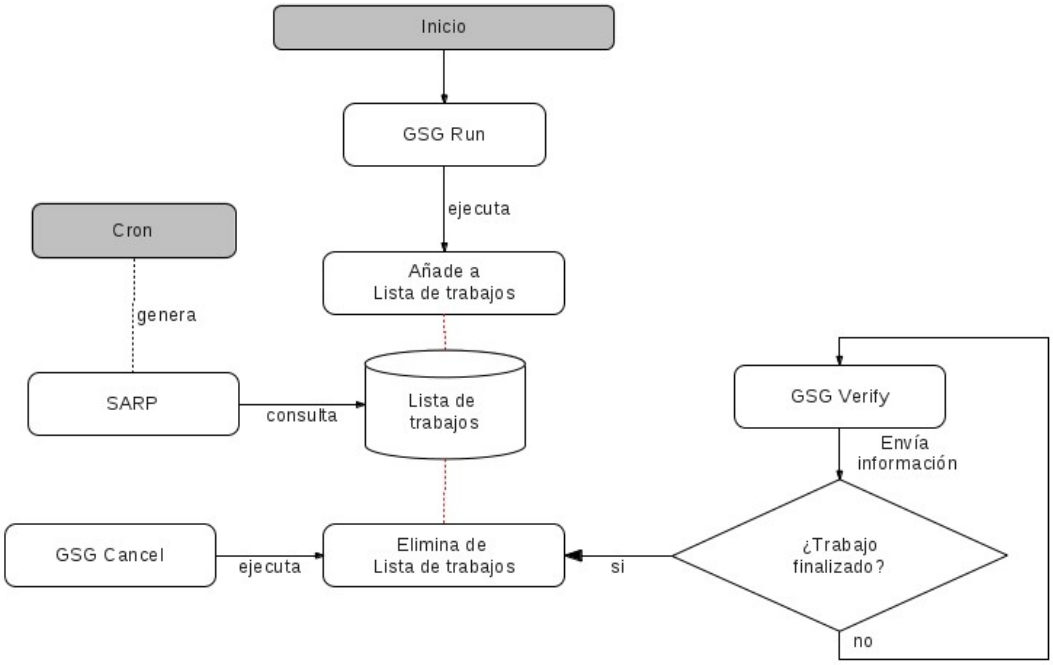


Figura 3.12: Funcionamiento del Sistema de Administración Regular de Procesos.

El SARP ejecuta periódicamente una serie de acciones para sondear el estado de las subtareas enviadas a una plataforma Grid, evitando de esta forma que haya que obtener esta información manualmente. La periodicidad con la que se ejecutan las tareas, definida en el MCG, no debe ser muy elevada ya que el sobrecosto correspondiente es mayor y puede ralentizar el sistema o incluso saturarlo. Para facilitar la visualización de como actúa este sistema se puede ver el diagrama de flujo en la Figura 3.12.

Primeramente se envían consultas a GSG mediante *GSG verify*, que aporta información sobre el estado de cada subtarea contenida en un trabajo, que a su vez se encuentra en la lista de trabajos activos del sistema. Esta lista puede modificarse tanto por *GSG run*, que agrega un trabajo, como por *GSG verify,* en el caso de una terminación exitosa, o por *GSG cancel*, que elimina el trabajo de la lista. De este modo, el SARP no volverá a ejecutar acciones relacionadas con ese trabajo.

Por otro lado, el sistema genera un fichero, *cron.log*, que contiene la información suministrada por el SARP, informando sobre los trabajos activos y finalizados, y sobre el tiempo total de ejecución.

3.3 Estructura de directorios

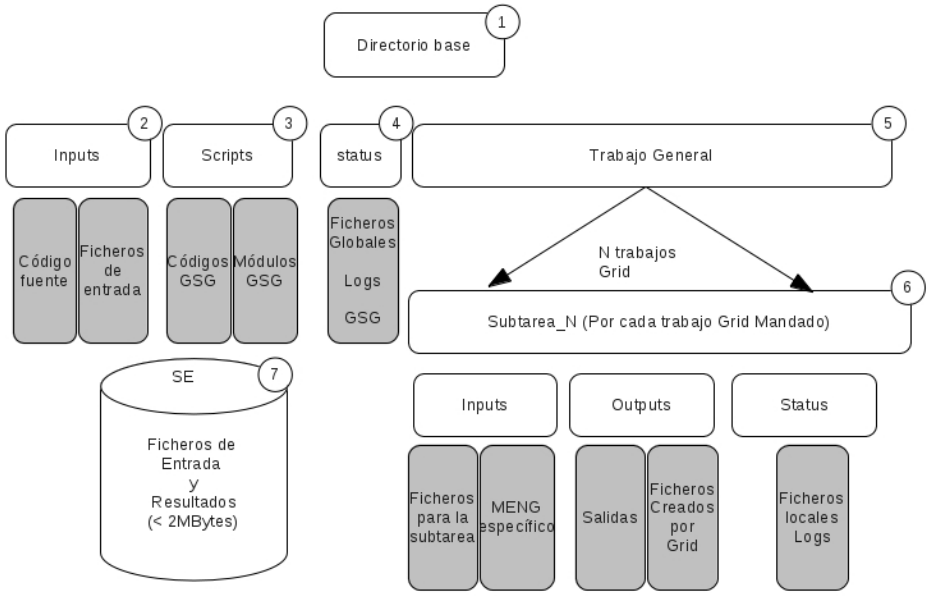


Figura 3.13: Diagrama con la estructura en árbol de directorios y subdirectorios.

Como se observa en el diagrama con la estructura en árbol de directorios y subdirectorios descrita en la Figura 3.13, para obtener una mayor claridad al visualizar estos elementos de manera gráfica se dispone de diferentes elementos. Los bloques más importantes se enumeran para facilitar la explicación de la estructura.

La estructura general de directorios parte de un directorio base (bloque 1) que contiene toda la estructura para la ejecución de GSG en el entorno Grid. Es en este directorio donde se ubicarán todos los elementos necesarios para la ejecución de la aplicación dentro de un nodo de trabajo Grid. En esta carpeta también esta incluido el MCG específico para cada trabajo, con todas las especificaciones necesarias para la ejecución de los trabajos en una infraestructura Grid (Sección 3.2.7). Los elementos contenidos en este directorio son los únicos que deben introducirse en GSG, el resto de los directorios son manipulados por el propio paquete y los sistemas asociados.

En el caso de que existiera un directorio base con el mismo nombre asignado por *GSG prepare*, éste alerta al usuario indicándole que cambie de nombre el directorio.

El directorio base contiene tres subdirectorios básicos llamados respectivamente *inputs* (bloque 2), *scripts* (bloque 3) y *status* (bloque 4), además de un subdirectorio por cada colección de trabajos generados por *GSG prepare*. El directorio base contiene, a su vez, un subdirectorio que incluye una carpeta por cada trabajo a lanzar por GSG (bloque 5). El nombre de cada una de las carpetas coincide con el proporcionado a *GSG prepare* por el MCG.

El directorio *inputs* (bloque 2) contiene todos los ficheros necesarios para la ejecución de la aplicación específica, como puede ser ficheros de configuración o anexos. Además este directorio puede contener un fichero comprimido donde se encuentra el código fuente necesario para la compilación dentro del WN. Estos ficheros serán obligatorios en el caso de que la aplicación no esté preinstalada en el WN elegido, o se requiera una compilación previa para su ejecución en infraestructuras desconocidas. La compilación previa se debe especificar en el MCG mediante el parámetro *compile*.

La carpeta *scripts* (bloque 3) contiene todos los ficheros ejecutables de las herramientas GSG y los módulos asociados a éstos. Desde esta carpeta se invoca a los módulos de GSG.

La carpeta denominada *status* (bloque 4) contiene la información general sobre el sistema Grid generada por el SRE. Dicha información se ha descrito en el apartado de SRE (Sección 3.2.8).

Cada uno de los directorios asociados a cada subtarea del trabajo (bloque 6) tiene la misma estructura. Estos directorios, son generados por *GSG prepare*, y contienen todos los elementos necesarios para el envío de trabajos, la información específica de estado de cada una de las subtareas y la captura de resultados generados por cada herramienta. Son denominados como *job_target_N,* donde el número N designa a una de las subtareas en que se ha dividido el trabajo, para su envío a los nodos de computación de una plataforma Grid donde se ejecutan en paralelo.

Cada uno de los directorios asociados a una subtarea (bloque 6) incluye varios subdirectorios. Primeramente, el directorio denominado *input* que contiene los elementos necesarios para la ejecución en una plataforma Grid que son enviados mediante el sistema SandBox. Estos elementos pueden ser datos de entrada, el fichero JDL, parámetros de entrada, configuraciones y otras informaciones relevantes. Además incluye el MENG específico para cada trabajo, y el fichero JDL, que *GSG prepare* ha creado previamente. Gracias a este método cada subtarea se puede enviar de forma independiente a una plataforma Grid.

El directorio donde se almacenan los ficheros con los resultados generados por la aplicación una vez terminado el trabajo en el entorno Grid, se llama *outputs*. Estos ficheros de resultados se gestionan mediante el sistema SandBox en el caso de ficheros que no superen los dos megabytes, o mediante un SE para ficheros de gran tamaño. En ambos casos, los ficheros se ubican en este directorio mediante las herramientas *GSG verify*, o *GSG cancel*. Además, el directorio también contiene los ficheros propios de salida de cualquier trabajo enviado a una plataforma Grid. Se trata del fichero *".out"* que contiene la información de pantalla de la aplicación, y el fichero *".err"* que contiene información en caso de error en la ejecución de la aplicación.

Por otro lado, la información específica del estado de cada subtarea se ubica en un subdirectorio llamado *status* y que genera el SRE.

Dentro del sistema de directorios cabe destacar también que los ficheros de tamaño superior a dos megabytes utilizan el SE (bloque 7), incluyéndose en el LFC que se especifica en el MCG y que sirve el sistema Grid.

3.4 Funcionamiento de GSG

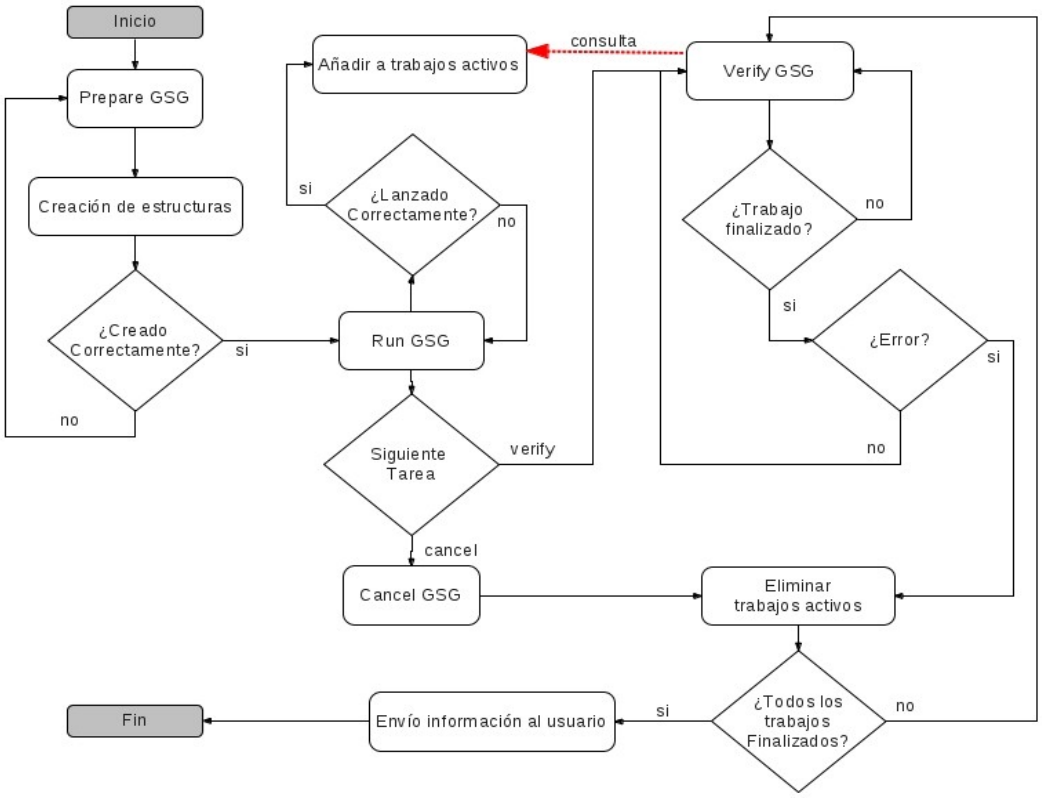


Figura 3.14: Funcionamiento General del Sistema GSG.

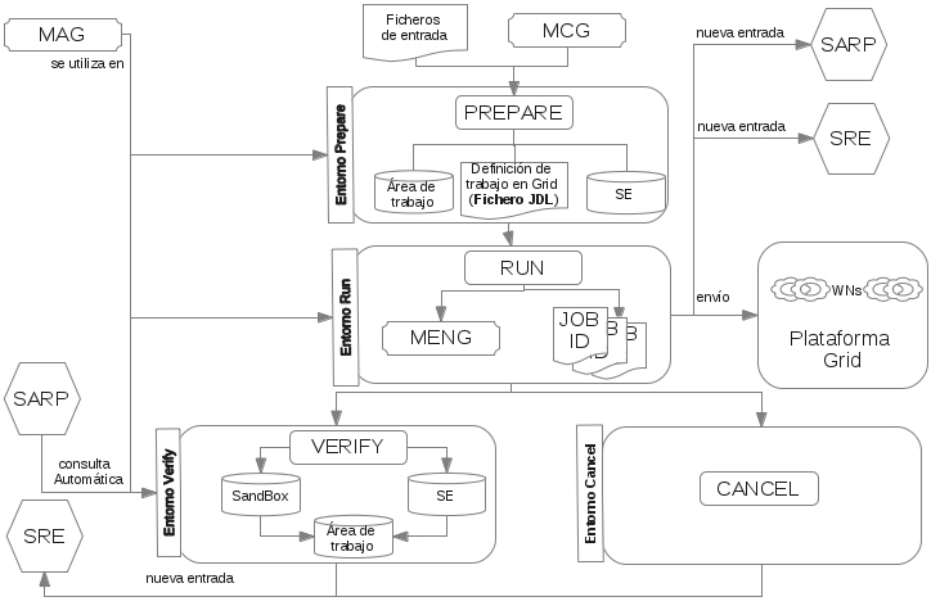


Figura 3.15: Estructura General del Sistema GSG.

En esta sección se detalla el funcionamiento de la herramienta GSG, que se resume en el diagrama de flujo de la Figura 3.14 . Por otro lado, la Figura 3.15 se usa como base para la descripción de la estructura general de GSG a lo largo del apartado.

Todas las herramientas incluidas en GSG se lanzan desde un mismo directorio denominado *directorio base* y desde el cual se ejecuta *GSG prepare* en primer lugar.

Una vez preparada la configuración física, se utiliza *GSG run* para mandar a una infraestructura Grid la colección de subtareas asociadas al trabajo. De esta forma se envía el MENG junto a los elementos necesarios para su ejecución, generando un identificador único, job_id, necesario para el posterior seguimiento del estado de la subtarea por el SARP, y las siguientes herramientas GSG. Además se añade una nueva entrada al listado de trabajos activos del SARP, para su análisis periódico.

Una vez están activas las subtareas dentro del entorno Grid, se tiene la posibilidad de usar *GSG verify* para extraer manualmente información acerca de los trabajos, ejecutar una serie de acciones según el estado de estos, y obtener, en caso de una finalización correcta, los resultados generados por la aplicación.

Este paso es opcional ya que el SARP se encarga de lanzar periódicamente *GSG verify*. De esta forma no se necesita recurrir a ningún protocolo de actuación en el caso de que un trabajo defectuoso o mal ejecutado, ya que la propia herramienta se encarga del reenvío de la subtarea en estos casos. Tampoco es obligatorio ejecutar comandos Grid para la obtención de los resultados originados ya que la herramienta se encarga de que todos los elementos de salida y la información del SRE estén colocados correctamente en su ubicación dentro de la estructura de GSG.

Por otra parte, *GSG verify* también genera información sobre el sistema. Finalmente, cuando todas las subtareas del trabajo han terminado, la herramienta envía un correo electrónico a la dirección obtenida del MCG informando que el trabajo ha finalizado. Además, en caso de que el tiempo asignado a la ejecución de los trabajos mediante el certificado Grid haya expirado, y algunas subtareas aún queden sin finalizar, también se le envía un correo electrónico informando sobre que subtareas han terminado y cuales no.

Usando *GSG cancel*, también se tiene la posibilidad de cancelar todas las subtareas asociadas a un trabajo. Así se evita que se tenga que hacerlo manualmente. Además, informa al SRE que ya no es necesario la consulta periódica del estado de las subtareas asociadas al trabajo cancelado.

3.5 Eficiencia de GSG

GSG se ha orientado hacia diversas áreas específicas de la Astronomía, en concreto las áreas beneficiadas son la Astronomía Extragaláctica, la Física Estelar, la Radioastronomía, la Estructura Galáctica y el área del Sistema Solar, donde se han ejecutado aplicaciones de distinta naturaleza en entornos distribuidos Grid gracias a GSG.

Para el estudio comparativo de tiempos, hemos usado el trabajo de evaluación utilizado para la aplicación *GraCo* descrita en la Sección 2.3.2. Para la ejecución de este trabajo de evaluación nos hemos servido de la plataforma Grid descrita en la Sección 2.2.1.

Tabla 3.3: Media y desviación estándar de los tiempos en el total de cada fase. Pr: Preparación Ln: Lanzamiento An: Análisis Rl: Relanzamiento Rc: Recogida y ubicación Cn: Cancelación. Las unidades están en segundos. Excluido el tiempo de ejecución.

Tiempo total	Media						
	Pr	Ln	An	Rl	Rc	Cn	Total
Con GSG	64,2	14,2	248,7	14,7	20,4	36,7	596,5
Sin GSG	181,0	61,3	545,6	43,5	60,5	101,3	993,0

Tiempo total	Desviación estándar						
	Pr	Ln	An	Rl	Rc	Cn	Total
Con GSG	4,6	0,6	7,7	1,6	3,3	3,7	12,3
Sin GSG	5,6	4,7	11,0	3,7	4,8	9,4	52,3

Hemos comparado el uso de GSG frente al envío manual de un conjunto de 1000 subtareas por trabajo, que se repite 10 veces en el mismo computador. Para esta comparación se ha desestimado el tiempo de ejecución de la aplicación ya que es igual en ambos casos y aquí nos centramos en el tiempo de preparación, lanzamiento, análisis de trabajos y en los casos hipotéticos de relanzamiento y cancelación si fueran necesarios. Tanto la media como la desviación estándar de los resultados de la pila de envío de trabajos se muestran en la Tabla 3.3.

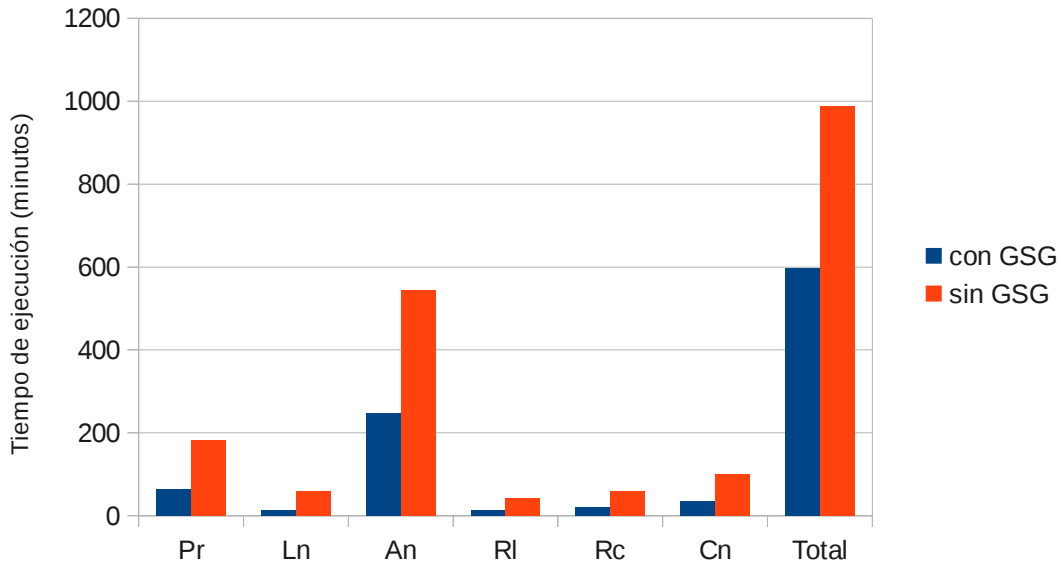


Figura 3.16: Diagrama de barras de la comparación de tiempos en las distintas fases de una pila de trabajos mandados a una plataforma Grid, excluido el tiempo de ejecución.

Observando además el diagrama de barras de la Figura 3.16 creada para comparar los tiempos de cada fase con las herramientas GSG frente a los tiempos de cada fase sin ellas, la mejora estimada en tiempo usando el paquete GSG queda demostrada ya que, en todas las fases, se observa una mejora de un rango de un 182% a un 428% dependiendo de la fase analizada. Obteniendo una mejora en la totalidad de un 165%

Además del estudio en tiempos de cada fase, hemos realizado un estudio focalizado en el análisis del número de comandos Grid con un *middleware* estándar frente al número de comandos GSG utilizados, basado en el envío de un trabajo completo dividido en un centenar de trabajos Grid. El trabajo de evaluación y la plataforma donde se ejecuta es la misma que la descrita en el estudio de tiempos de cada fase. En este caso se tienen:

- Alrededor de 1000 lineas de comando para preparar la estructura, N_p.
- 100 lineas de comando para lanzar las subtareas a una infraestructura Grid, N_l.
- 500 lineas de comando ya que 100 lineas sirven para para analizar el estado de cada una de las subtareas. Teniendo en cuenta una media de 50 veces a lo largo de una ejecución media de 150 minutos y una verificación cada 3 minutos hace el total de 500 líneas, N_a.

- 10 líneas más en el caso de la necesidad de relanzar estos trabajos a la infraestructura debido a errores en la computación. Teniendo en cuenta que un 10% de los trabajos deben realizarse por errores diversos en infraestructura o en ejecución, N_r.
- 500 lineas de comando para recoger los resultados de cada subtarea y ubicarlos en el lugar correspondiente dentro de la estructura GSG, N_o.
- 200 lineas de comando más si hay que realizar la cancelación y eliminación de la estructura de directorios, N_c.

Suponiendo el caso promedio de una secuencia de ejecuciones sin necesidad de cancelación la define la Fórmula 3.1.

$$N_{líneas\,totales} = N_p + N_l + N_a + N_r + N_o \quad (3.1)$$

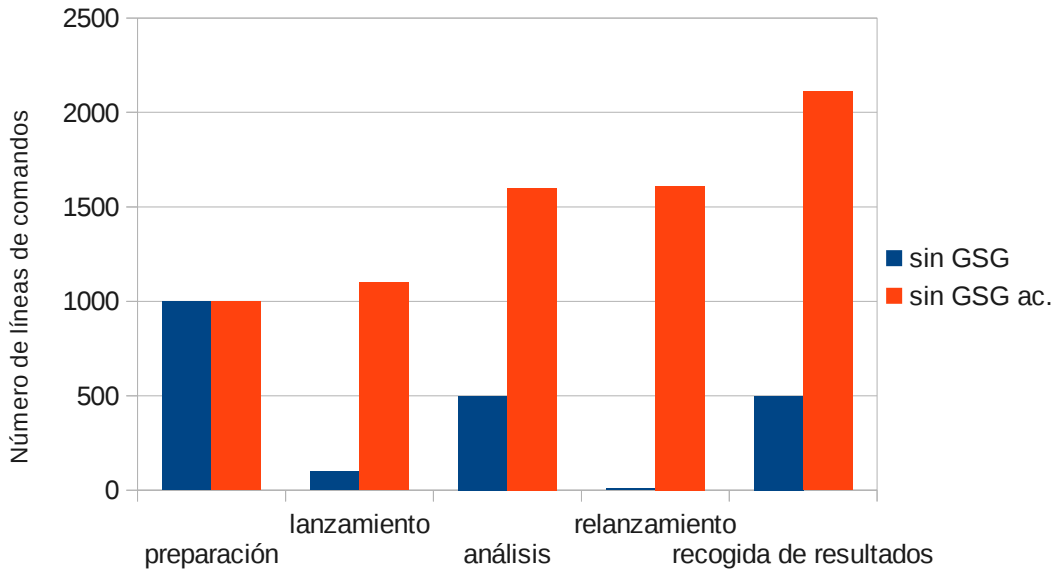


Figura 3.17: Número de comandos middleware (sin GSG) necesarios para cada tarea de GSG (un comando por tarea) representado con las barras azules. Además las barras rojas representan el número de comandos ejecutados sin GSG acumulados en cada una de las tareas.

En la Figura 3.17 podemos observar el beneficio que aporta GSG en cuanto al número de líneas de comandos ejecutadas.

Además, GSG proporciona beneficios como la creación de una estructura de datos para facilitar el acceso a los mismos, o la utilización tanto del SRE que aporta información acerca del sistema o de los trabajos, como del SARP que revisa periódicamente el estado de éstos actuando en consecuencia.

Por lo tanto, el tiempo empleado en estas tareas se ha reducido considerablemente debido a que el uso del paquete de herramientas que proporcionamos en este trabajo elimina la ejecución manual de las tareas de recogida de información y de la revisión del estado de los trabajos.

Tabla 3.4: Comandos usados en el sistema GSG.

Comandos GSG	Breve descripción
./prepare	Usa la herramienta *prepare GSG* para organizar la estructura para el lanzamiento de trabajos a una plataforma Grid.
./run	Usa la herramienta *run GSG* para la ejecución de los trabajos en el entorno Grid.
./verify	Usa la herramienta *verify GSG* para consultar el estado de los trabajos enviados a Grid así como la situación general del sistema. Además de la recogida de resultados en el caso que sea necesario.
./cancel	Usa la herramienta *cancel GSG* para la cancelación de los trabajos enviados al Grid y la recogida de resultados.

En conclusión, se tienen que ejecutar cerca de 2110 líneas de comando si no se usan herramientas como las que proporcionan GSG. En el caso de usarse GSG, una vez configurados correctamente los módulos correspondientes, hay que ejecutar cuatro comandos, descritos en la Tabla 3.4.

En este estudio queda claro el beneficio que aporta el uso de las herramientas GSG, además del valor añadido de la facilidad de uso para la gestión de grandes estructuras de trabajos dentro del entorno distribuido Grid.

Por otro lado, al usar el entorno distribuido Grid y potenciar la división de trabajos en subtareas independientes entre sí para ejecutarse en paralelo, el rango de mejora con respecto al tiempo de ejecución del mismo trabajo en un entorno serializado de las mismas características físicas que cada uno de los nodos de computación es considerable, como se puede observar en los resultados de la Tabla 2.10, la Tabla 2.17 y la Tabla 2.26 del Capítulo 2, dependiendo del grado de división y paralelismo obtenido de la aplicación. Esta mejora se debe al uso del entorno distribuido Grid.

3.6 Conclusión

Una vez descrito como adaptamos algunas herramientas astronómicas para su uso en cualquier plataforma Grid en el Capítulo 2, hemos realizado un paquete de herramientas llamado GSG. Este paquete aporta una mayor facilidad en la gestión, en la monitorización y en la ejecución de trabajos Grid, permitiendo de esta manera cubrir las necesidades computacionales y de almacenamiento de las aplicaciones que las usan.

A lo largo del capítulo hemos definido las tareas fundamentales que debe realizar un paquete de herramientas que quiera cubrir todas la necesidades en el campo de la astronomía, extendiéndose además a cualquier campo de investigación. Para completar dichas tareas, hemos diseñado las herramientas que además tienen un carácter general, esto es, mediante unas pequeñas modificaciones en algunos de los módulos descritos se consigue adaptar cualquier aplicación astronómica para su ejecución en una plataforma Grid. Se consigue, además que la mayoría de los códigos que conforma el paquete GSG sean reutilizables en cualquier aplicación.

Hemos analizado cada uno de los componentes del paquete de herramientas GSG, describiendo al detalle su funcionamiento, su estructura, y su ubicación dentro del paquete, explicando las relaciones que tiene cada componente con el resto de las herramientas.

Para mejorar la usabilidad del paquete de herramientas, hemos creado varios sistemas de apoyo al envío, a la monitorización y a la obtención de información que, aunque no son necesarios para el cometido general de las herramientas (ejecución de aplicaciones en el entorno Grid), son muy útiles para facilitar el uso del usuario de GSG.

Por otro lado, y como aportación experimental al contenido de esta memoria, hemos realizado varios estudios analizando las mejoras obtenidas con el uso de GSG frente al de los comandos Middleware. Estos estudios se han realizado tanto a partir del análisis de tiempos como del número de comandos utilizados en cada una de las fases.

Una vez descrita la herramienta GSG y justificada su utilidad en la ejecución de aplicaciones entre las que se encuentran la astronomía, se considerará su uso para implementar un servidor de aplicaciones astrosismológicas de alto nivel llamado ATILA [ROD14]. Para este cometido, GSG se ha incluido como un nuevo módulo en dicho servidor, permitiendo que las aplicaciones del servidor se ejecuten en el ámbito de la computación distribuida en Grid. La manera de conectar el paquete GSG con la aplicación ATILA se detalla en el siguiente capítulo.

Capítulo 4: Integración de herramientas GSG en aplicaciones astronómicas. El caso de ATILA.

Las herramientas de GSG fueron concebidas y desarrolladas para su uso como capa superior al middleware de las plataformas Grid. A medida que las aplicaciones adquieren cierta complejidad el uso del paquete de herramientas GSG deja de ser eficiente, en particular por la continua necesidad de supervisión y adaptación se requiere. Por ello, hemos procedido a adaptar las herramientas GSG para poner ser usadas por cualquier aplicación, posibilitando acceder a ellas desde interfaces de usuario. Como ejemplo, en esta Tesis hemos desarrollado el servidor de aplicaciones astrosismológicas ATILA [ROD14] en el que hemos adaptado e integrado las herramientas GSG.

Hemos creado el servidor de aplicaciones ATILA dentro del marco de la astrosismología [GOU85], que se define como *"la técnica astronómica que estudia las oscilaciones periódicas de las superficies de las estrellas"* [SEA14]. Los datos astrosismológicos consisten en series temporales de luminosidad de estrellas, ligados a parámetros físicos como la temperatura de la superficie de la estrella, la luminosidad, la gravedad y otras.

Se trata de facilitar la ejecución de aplicaciones astrosismológicas en un entorno multiplataforma, es decir, tanto en plataformas Grid, como otro tipo de servidores dedicados, a través del envío de trabajos paralelizados. Además, una ventaja de la herramienta es que la hemos creado siguiendo un esquema modular para que se pueda añadir mayor funcionalidad sin rehacer la herramienta desde cero. Finalmente, para facilitar el trabajo de los usuarios, se ha optado por desarrollar un entorno gráfico siguiendo el esquema Modelo Vista Controlador (MVC) [BUR97] cuya arquitectura separa los datos de la aplicación de la interfaz de usuario y del módulo encargado de gestionar los eventos y las comunicaciones.

Entre los proyectos relevantes para el desarrollo de la astrosismología se pueden citar los siguientes:

- El satélite Corot [AUV09], que fue lanzado en el año 2006 por la Agencia Espacial Europea (ESA) [ESA14] con el objetivo principal de buscar planetas extrasolares, especialmente de aquellos de un tamaño similar al terrestre, también tiene un cometido astrosismológico. Se trata de detectar los temblores que tienen lugar en la superficie de las estrellas y que alteran su luminosidad. Gracias a este fenómeno se puede calcular con bastante precisión la masa, edad y composición química de las estrellas.

- El satélite Kepler [NAS09] que fue lanzado en el año 2009 por la National Aeronautics and Space Administration (NASA) [NAS14] y cuyo objetivo principal es observar simultáneamente unas 150000 estrellas, y analizar su brillo cada 30 minutos para detectar posibles tránsitos de planetas.

- Prevemos que el servidor de aplicaciones ATILA soporte códigos de análisis de datos de futuras misiones como puede ser el instrumento CARMENES [QUI12], que tiene previsto el inicio de sus operaciones a lo largo de 2015. Se trata de un instrumento optimizado para la búsqueda de planetas alrededor de estrellas de muy baja masa. Además, se podrán detectar planetas similares a la Tierra fuera del sistema solar (Exotierras) en las zonas habitables alrededor de estrellas de tipo M (compuesto por enanas rojas y algunos tipos de estrellas gigantes y supergigantes [BOE76]) investigadas alrededor de las que transitan [KAL09]. Además, pretendemos realizar pequeñas campañas astrosismológicas para refinar los parámetros astronómicos de las estrellas con exoplanetas.

- Por otro lado se encuentra el satélite PLAnetary Transits and Oscillations of stars (PLATO) [PLA14] elegido por la ESA como futura misión, cuya fecha prevista de lanzamiento es en 2024. Su objetivo es encontrar y estudiar un gran número de sistemas planetarios extrasolares, analizando las propiedades de los planetas en la zona habitable alrededor de estrellas de tipo solar. PLATO también se ha diseñado para investigar la actividad sísmica en estrellas, lo que permite la caracterización precisa de la estrella, incluyendo su edad.

El servidor de aplicaciones ATILA contiene los códigos para el análisis de datos de instrumentos al servicio de la astrosismología. Además de códigos para el análisis de los datos de observaciones, el servidor contiene códigos para la generación de distintos tipos de modelos astronómicos, entre los que cabe destacar los códigos Cesam [MOR97] y MESA [MES14, PAX11] para la creación de modelos de estructuras estelares, o *GraCo* [MOY08] y FILOU [SUA08] para la creación de modelos de oscilación , con y sin rotación. Como detallamos en los Capítulos 2 y 3, hemos portado estas herramientas previamente a una plataforma Grid y han hecho uso de el paquete de herramientas GSG.

4.1 El servidor de aplicaciones ATILA

El servidor de aplicaciones ATILA persigue la integración de todo el conocimiento y de las técnicas propuestas por los investigadores para el análisis e interpretación de datos astrosismológicos a partir de distintos códigos.

Se trata de un servidor dinámico que pretende aunar herramientas análisis e interpretación de modelos para datos astrosismológicos. Es modular para poder ir mejorando las herramientas y/o incluyendo nuevas funcionalidades dependiendo de la actividad investigadora.

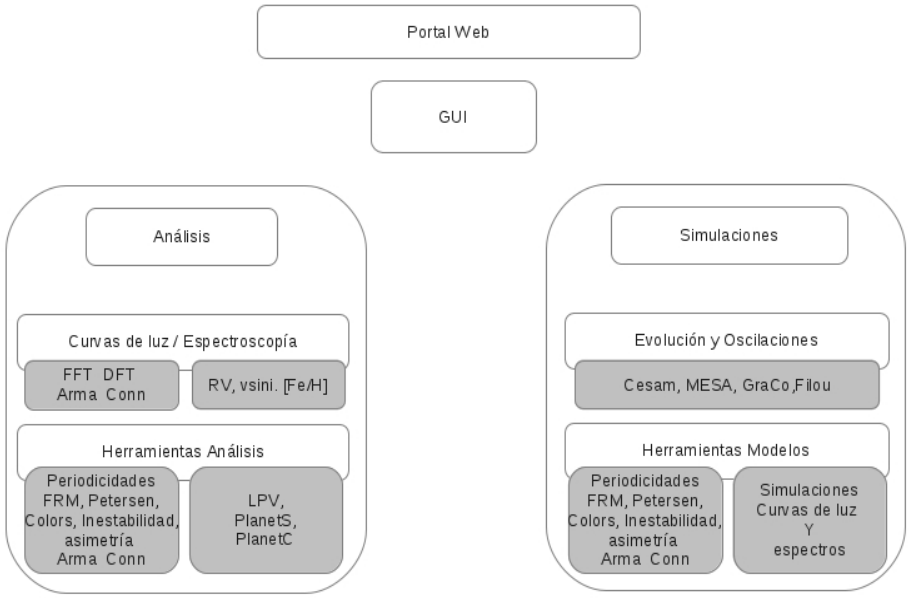


Figura 4.1: Estructura del servidor de aplicaciones ATILA

Así pues, las aplicaciones contenidas en el servidor de aplicaciones ATILA se clasifican en dos grupos (Figura 4.1):

- Aplicaciones para la gestión y el análisis de datos de observación obtenidos mediante instrumentación.
- Aplicaciones para el cálculo y la gestión de modelos de estrellas y sus múltiples parámetros físicos.

La principal característica del servidor de aplicaciones ATILA es la implementación modular de cada una de sus funciones, apoyándose en una base de datos relacional que permite la reusabilidad y la reproducibilidad de las funciones. Además, facilita la modificación de la base de datos en caso necesario permitiendo una organización estructural para su uso y comprensión, y ayuda a la implantación de nuevas funciones de forma rápida y sencilla. Para ello hemos utilizado una codificación en Python [PIG04], un lenguaje de programación orientado a objetos que se ajusta perfectamente estos requisitos.

Los módulos que añaden funcionalidad a la estructura central son necesarios, no sólo para la mejor gestión interna del programa y para la facilidad de uso del usuario final, sino que además es necesario para poder aplicar mejor el método expuesto en esta Tesis. Entre los principales módulos se encuentran:

- El módulo de conexión con infraestructuras de computación: Una de las características más importantes de este servidor es que el sistema permite ejecutar las aplicaciones dentro de distintas infraestructuras de cómputo, trazando las fases de su funcionamiento según distintas alternativas según de la infraestructura seleccionada, ya sea Grid, Cloud, o cualquier servidor específico. De esta forma, se facilita la elección de la infraestructura que se ajuste mejor a las necesidades de cómputo.
- El módulo de flujo de trabajos: Al servidor de aplicaciones ATILA también se le ha añadido la posibilidad de usar los códigos mediante flujos de trabajo. De estas manera, definimos la estructura, la sincronización, y el orden de las tareas a realizar permitiendo que el flujo de trabajo sea dinámico y modular para implementar, modificar y/o eliminar funciones de forma ágil.

1. El módulo de perfiles físicos: El servidor de aplicaciones ATILA añade también el concepto de perfiles físicos, que definen las características iniciales necesarias para la realización de un modelo. Éstos se utilizan en cada ejecución de un código en concreto. Esta característica, junto al concepto de trazabilidad de las ejecuciones implementada por el servidor, facilita la repetición de ejecuciones con distintos datos de entrada pero con una misma caracterización física que define el modelo a realizar.

- El módulo de análisis: Para gestionar y analizar los datos de observaciones recogidos por distintos instrumentos, ATILA organiza adecuadamente los resultados en colecciones de datos, consistentes en conjuntos de datos generados utilizando unas características físicas similares y afines entre sí, por ejemplo, agrupando los datos por estrellas analizadas, por instrumentos utilizados, por fechas de obtención de datos, etc. Esta función permite la creación, modificación, eliminación, análisis y/o comparación de las colecciones de datos.

- El módulo de monitorización: El servidor de aplicaciones ATILA contiene un visualizador del estado de los trabajos de cada aplicación. En el caso de la ejecución en una infraestructura Grid, este módulo está conectado con el paquete de herramientas GSG como veremos. El método de conexión se explica en la Sección 4.2)

- El módulo de base de datos: El uso de bases de datos en el servidor permite clasificar toda la información en base a las ejecuciones de los códigos, las colecciones de datos obtenidos y la física usada en cada experimento, pudiendo de esta forma generar informes útiles para el investigador.

- El módulo de conexión con el Observatorio Virtual: Se ha optado por implementar en el servidor de aplicaciones ATILA una conexión entre los resultados obtenidos por las aplicaciones y la plataforma del Observatorio Virtual [IVO14], que se creó con el objetivo de garantizar rapidez y eficiencia, tanto en el acceso a la información existente en archivos y servicios astronómicos, como en el análisis de dicha información. De esta forma, los datos se pueden mostrar de forma automática a la comunidad científica.

El servidor está conectado con una herramienta del Observatorio Virtual llamada TOUCAN (The VO gateway for asteroseismic models) [SUA14], que es un servicio abierto para gestionar conjuntos de datos de modelos astrosismológicos , y proporcionar herramientas gráficas para el análisis en línea de dichos modelos (comparación , trazado, las estadísticas, etc). Esto significa que el flujo de trabajo de ATILA debe contener un módulo para gestionar y transformar las salidas ATILA (típicamente colecciones modelo) y ser usadas por TOUCAN.

4.1.1 Integración de códigos en el servidor de aplicaciones ATILA.

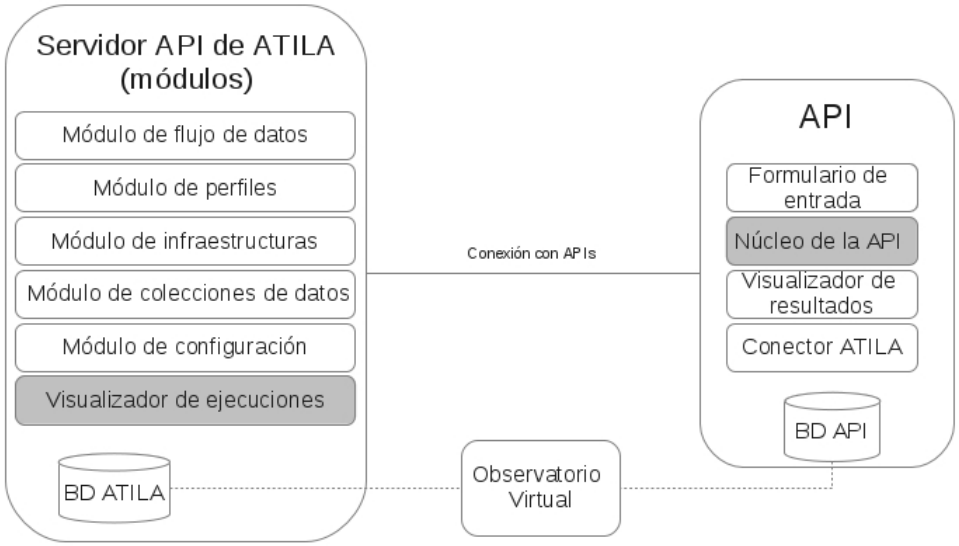


Figura 4.2: Composición modular del servidor de aplicaciones ATILA

Las características generales del servidor de aplicaciones ATILA las hemos proporcionado a través del denominado Servidor de Aplicaciones (API) de ATILA, donde cada módulo que hemos realizado implementa una de las funciones del servidor: formulario para la resolución de datos de entrada, aplicación a ejecutar, visualizador de resultados en pantalla y conector con el servidor de aplicaciones ATILA.

Cada vez que se conecta una nueva aplicación se hace mediante un paquete llamado ATILA API (AA), que proporciona una conexión rápida y transparente para el usuario. A cada uno de los paquetes ATILA API que realizamos le hemos implementado, obligatoriamente, una serie de módulos que se muestran en la Figura 4.2 dentro de la API, y que se describen a continuación:

Formulario de entrada.

Recoge de información para la ejecución de la aplicación, que consiste en una colección de formularios gráficos cuya información será incluida en una base de datos específica para cada una de las aplicaciones integradas.

Como ejemplo, en el caso del módulo de obtención de periodicidades del espectro (SPM, por sus siglas en inglés) [ROD14] incluida en ATILA, varios de los formularios se presentan como se indica en la Figura 4.3. La aplicación SPM encuentra periodicidades en el espectro generado por las curvas de luz de la estrella a analizar.

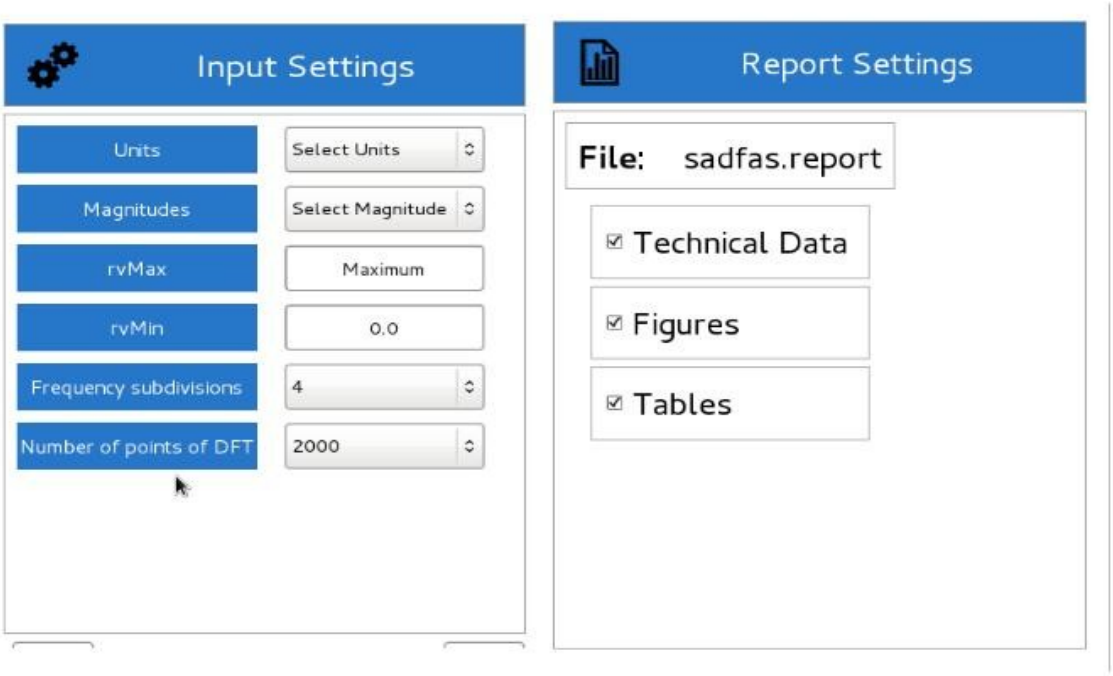


Figura 4.3: Formularios de recogida de información para la ejecución de la aplicación SPM

En el formulario de entrada se recogen datos como las unidades y las magnitudes de los datos de entrada, el rango de frecuencias a analizar y la definición de las gráficas resultantes, además de ajustes relacionados con la información recogida de la ejecución.

Núcleo de la API.

```
#Call to MIARMA
1. JFFile = name + ".jf"
2. system(miarma('JFFile'))
3. maFile = name+".agfs"
4. table = asciitable.get_reader(numpy=False)
5. maData = asciitable.read(maFile)
```

Código 4.1: Código de llamada a la aplicación externa MiArma.

La aplicación propiamente dicha también será incluida dentro del paquete ATILA API. A este módulo se le denomina núcleo de la API y puede implementarse en distintos lenguajes de programación. Esta aplicación es utilizada mediante una llamada al sistema desde ATILA. Al implementarse el módulo mediante Python [PYT14] se utiliza el paquete “system” como se presenta en el Código 4.1 que, una vez definido el nombre del fichero de entrada (línea 1), ejecuta la llamada al código MiArma (línea 2), el cual está implementado en el lenguaje de programación MATLAB [MAT14] que contiene las funciones específicas para el cálculo no analítico requerido para este caso. A continuación define el nombre del fichero de salida de MiArma (línea 3) y carga los datos en una tabla ascii (líneas 4 y 5) para su posterior gestión.

Conector ATILA

Para que el uso de paquetes ATILA API creados de forma independiente no sea un impedimento a la hora de integrar la aplicación en ATILA, se usa un el módulo *ATILA Connector* que prepara del entorno de ejecución de la aplicación dentro del servidor. Además, proporciona datos sobre el tiempo de ejecución, las necesidades computacionales y de almacenamiento de la aplicación, además de definir las variables de entorno. Como se verá posteriormente en la Sección 4.2, este módulo está muy relacionado con el Módulo de Acoplamiento GSG (MAG) .

Visualizador de resultados

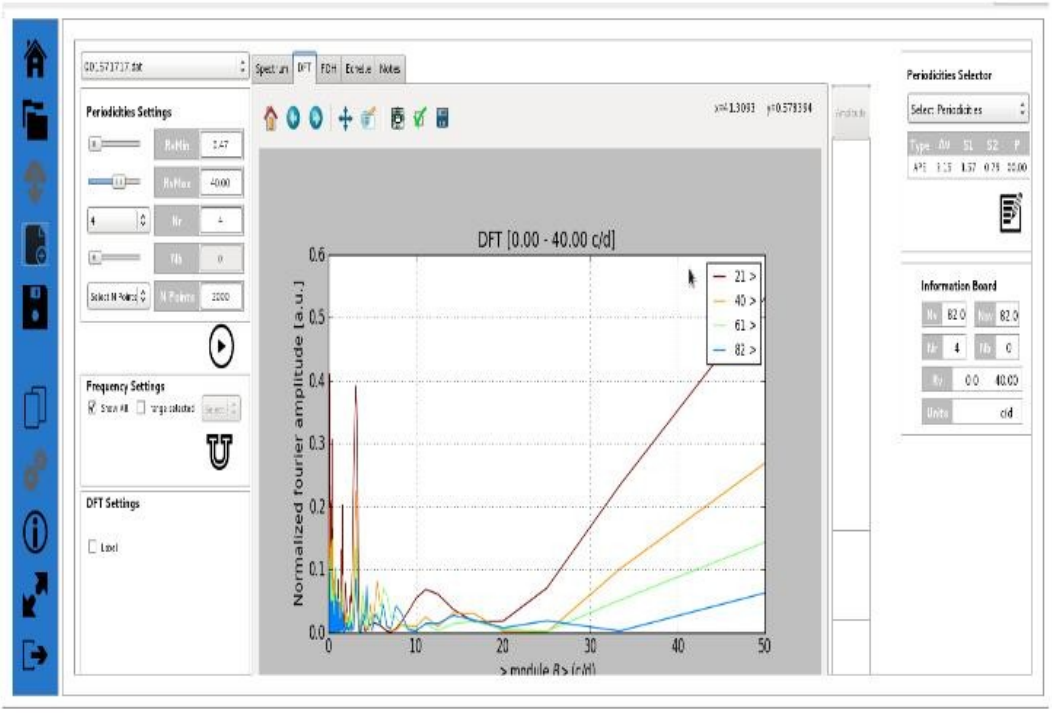


Figura 4.4: Módulo de visualización de resultados obtenidos de la aplicación SPM. La pantalla de visualización presenta una zona central donde aparecen múltiples gráficas y una serie de controles de visualización y gestión de los resultados mediante controles relacionados con la modificación de rangos de visualización, la modificación de unidades de medidas, escritura de etiquetas relevantes en las gráficas, modificaciones del rango de los espectros que generan los resultados y otros.

Cada paquete ATILA API posee un módulo de visualización de resultados que hace posible la conexión con herramientas externas para la elaboración, modificación y manejo de gráficos, informes generales, estadísticas de resultados, postprocesamiento de datos obtenidos, o cualquier utilidad que pueda añadirse a la API. La Figura 4.4 presenta un ejemplo de aplicación SPM donde se pueden observar las múltiples funciones que facilitan la obtención de información de los datos resultantes.

En el caso del ejemplo de la Figura 4.4, la pantalla de visualización presenta una zona central donde se presentan múltiples gráficas y una serie de controles de visualización y manejo de los resultados como pueden ser los controles relacionados con la modificación de rangos de visualización, la modificación de unidades de medidas, la escritura de etiquetas relevantes en las gráficas, las modificaciones del rango de los espectros que generan los resultados y otros.

La implementación de los módulos dentro de un paquete también hace posible usar cada una de las APIs de ATILA de forma independiente. Esto agiliza la depuración de errores, inconsistencias, mejora el tiempo de ejecución y facilita el uso de la herramienta ya que no incluye tantas opciones de ejecución.

4.1.2 Uso de varias plataformas de computación.

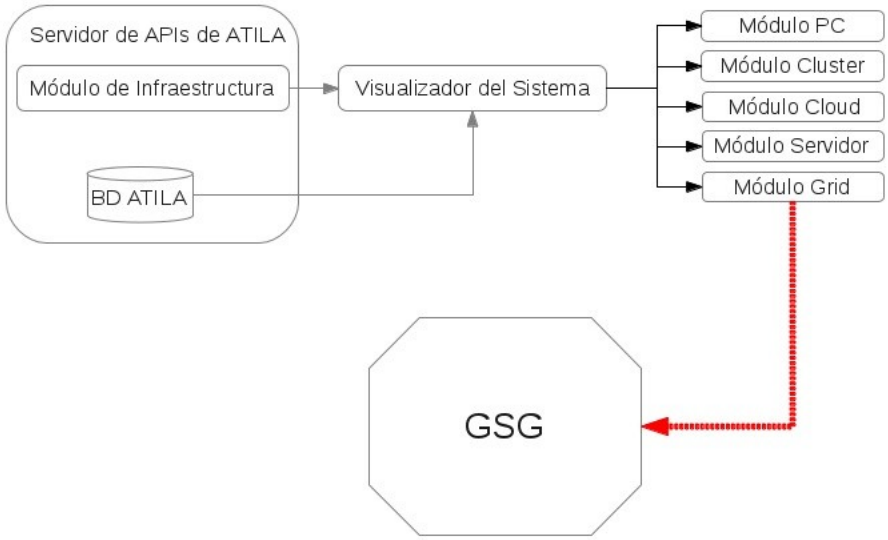


Figura 4.5: Módulo de infraestructuras ATILA

El módulo que integra esta función, llamado Módulo de Infraestructuras, contiene a su vez un submódulo por cada una de las infraestructuras que soporta ATILA, tal y como se muestra en la Figura 4.5.

Además de estos submódulos, el servidor de APIs de ATILA (SAA) contiene un sistema de visualización de estados de las distintas infraestructuras. Este sistema está implementado de forma gráfica para facilitar su comprensión. Recoge la información de cada uno de los submódulos referente al estado de la infraestructura, y la presenta en pantalla. De esta forma, el usuario, tiene toda la información para poder elegir la plataforma adecuada. El sistema presenta los recursos disponibles, los errores en tiempo real y el tiempo estimado de ejecución en cada una de las infraestructuras.

Actualmente, ATILA tiene la posibilidad de usar cinco tipos de ejecuciones asociadas a cada tipo de infraestructura. Estas son implementadas mediante los módulos para la ejecución en un ordenador local, en un cluster, en una plataforma Cloud, en un servidor dedicado y en una infraestructura Grid (Figura 4.5).

La primera de ellas es la ejecución de la aplicación en el ordenador personal huésped de la plataforma. Este es el sistema más básico y el submódulo sólo se encarga de preparar el entorno y ejecutarlo dependiendo de el sistema operativo que contenga el ordenador.

El siguiente tipo de ejecución es la que se realiza en un servidor dedicado. En este caso el submódulo se encargará del envío de los datos de entrada necesarios para la ejecución de la aplicación y su ejecución remota en dicho servidor dedicado. Esta opción también es válida para la ejecución en supercomputadores [SEV98].

Las siguientes alternativas se refiere a la ejecución en infraestructuras distribuidas. Así pues, los submódulos dedicados a cada una de ellas deben tener en cuenta sus características a la hora de enviar trabajos.

En el caso de las ejecuciones en la infraestructura de tipo cluster, los trabajos deben enviarse a una cola dedicada que se implementa a través de un gestor de recursos. Este submódulo debe ajustarse no sólo a los requisitos necesarios para la ejecución de la aplicación, sino también incluir los comandos precisos para el envío de trabajos dentro de una cola de trabajos.

En el caso de las ejecuciones en la infraestructura Cloud, se debe crear una máquina virtual con el entorno necesario para la ejecución de la aplicación, incluyendo la API de ATILA completa.

Finalmente, el modulo que gestiona el envío de una infraestructura Grid en ATILA es el sistema GSG. Para usar GSG dentro del servidor se deben conectar entre sí varios componentes de cada sistema para conseguir un funcionamiento sincronizado y correcto. Estas conexiones se explican en la siguiente sección.

4.2 Conexión del servidor de aplicaciones ATILA con el módulo GSG

Siguiendo la filosofía modular de ATILA, hemos integrado GSG como un módulo estandarizado del servidor.

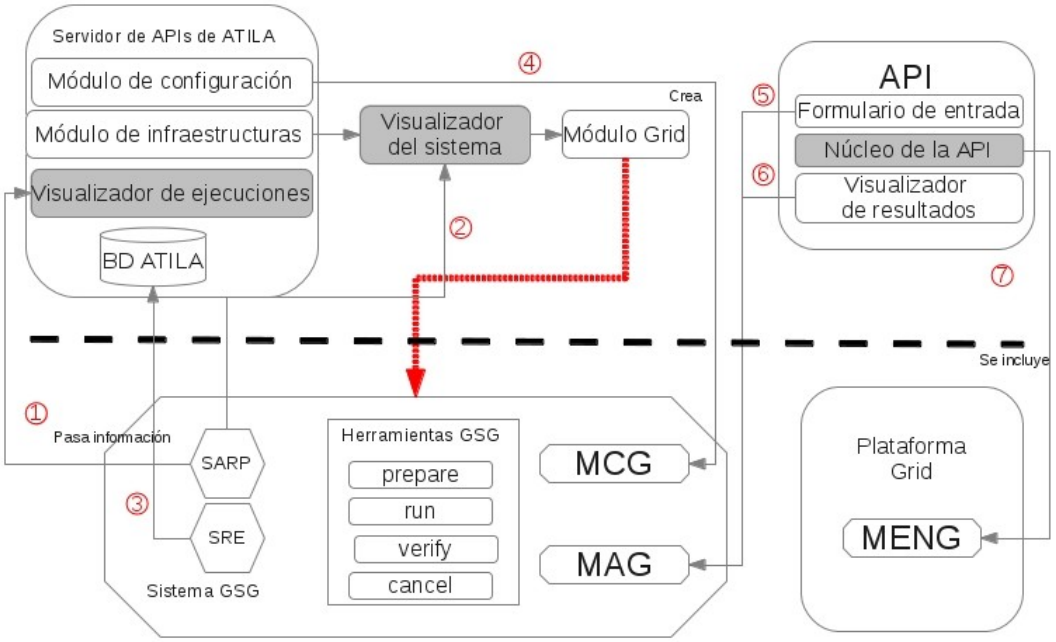


Figura 4.6: Conexión del módulo Grid de ATILA con GSG

Los distintos componentes de ATILA y del paquete GSG están conectados entre sí, garantizando mayor independencia entre los dos sistemas y a su vez facilitando todas las funciones de GSG para ATILA.

Existen diferentes conexiones entre los dos sistemas. Éstas se dividen en tres tipos:

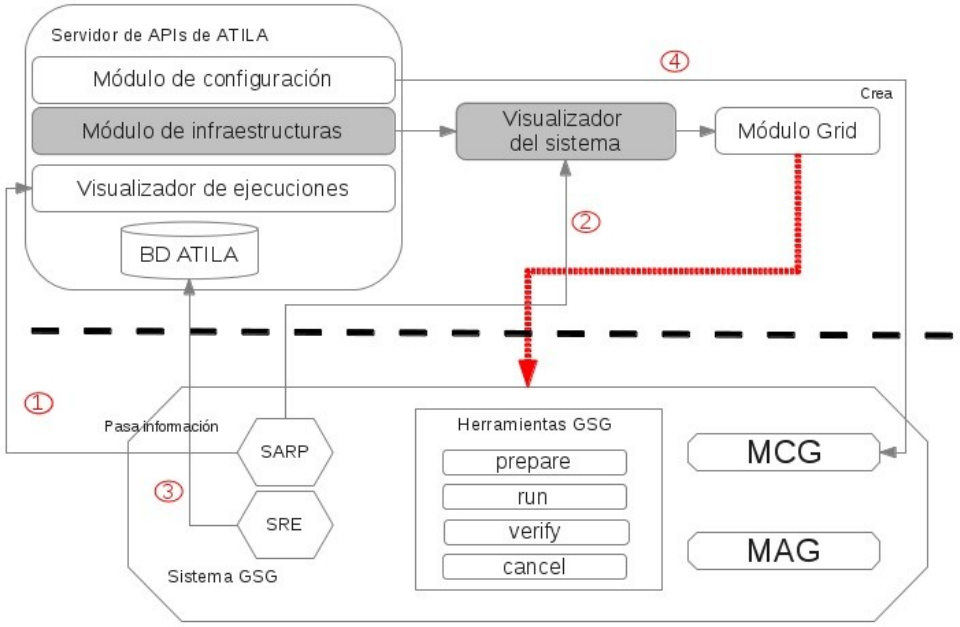


Figura 4.7: Instalación de nueva API en el servidor de aplicaciones ATILA*. Proceso de conexión entre el sistema SAA y el sistema GSG.*

Conexión SAA – GSG

Existen varias conexiones entre los componentes de los sistemas SAA y GSG. A continuación se detallan en la Figura 4.7 el proceso de conexión de cada uno de ellos.

1.- Conexión del sistema de administración regular de procesos de GSG con el módulo de visualización de trabajos. Periódicamente el SARP revisa el estado de los trabajos enviados a una infraestructura Grid y a continuación rellena el campo llamado “status” del registro, correspondiente al trabajo que se encuentra en la base de datos del SAA. Esta información es recogida por el módulo de visualización de trabajos de ATILA y se presenta en pantalla. La recogida de información de la base de datos por parte de este módulo se hace de forma periódica, ajustando el tiempo entre consultas con el fin de no saturar el sistema con continuas peticiones de información.

2.- *Conexión del SARP con el visualizador de sistemas*. El estado en tiempo real de una infraestructura Grid utilizada es enviado al módulo de visualización de sistemas usando para ello el SARP de GSG, que recopila la información necesaria y lo graba en la base de datos de SAA en un registro asociado al sistema. A continuación el módulo recoge la información de forma periódica y lo presenta en pantalla.

3.- *Conexión del sistema de registro de eventos de GSG con la base de datos general* del servidor de aplicaciones *ATILA.* Cada vez que sucede un acontecimiento en el sistema GSG, la base de datos de SAA graba la información recopilada del SRE en un registro asociado al sistema que se está usando. Posteriormente el módulo de visualización del estado de los sistemas del SAA tiene la posibilidad de presentar estos eventos, ordenados cronológicamente, mediante informes creados a partir de la BD.

4.- *Conexión del módulo de configuración de ATILA con el módulo de configuración GSG.* El módulo de configuración del SAA controla el ajuste de todos los módulos del servidor, incluido el módulo de infraestructura. En el caso de usar una infraestructura Grid, este módulo hereda todas las posibilidades de ajuste que tiene el módulo de configuración GSG. De esta forma, GSG puede configurarse internamente mediante MCG o, externamente, mediante el módulo de configuración del servidor. Este módulo creará un fichero de configuración que es procesado por el MCG de GSG.

Conexión AA – GSG

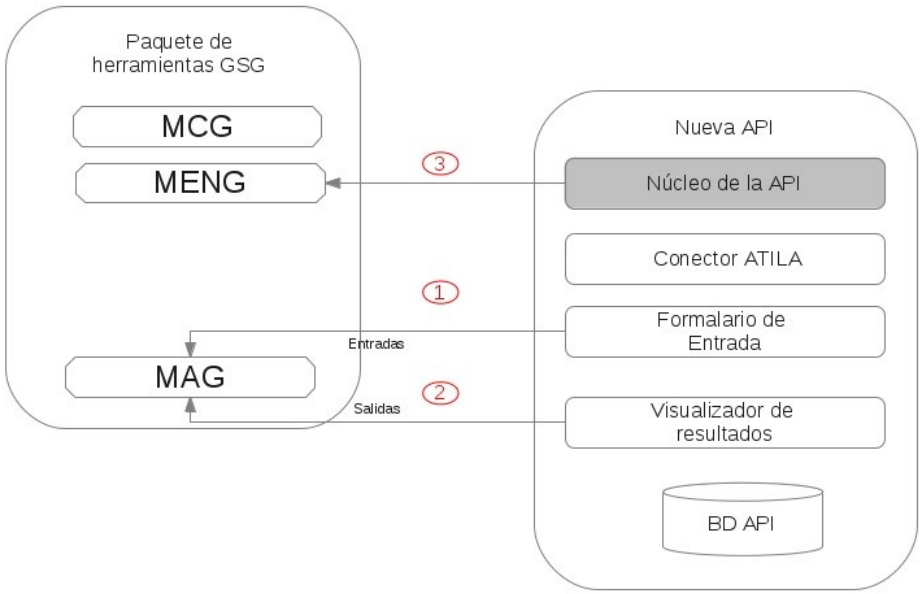


Figura 4.8: Instalación de nueva API en el servidor de aplicaciones ATILA*. Proceso de conexión entre el sistema AA y el sistema GSG.*

1.- *Conexión del módulo de recogida de información de AA con el módulo de acoplamiento GSG*. Para la utilización de una infraestructura Grid mediante GSG se debe completar la información necesaria por el módulo de acoplamiento. Para ello se usa el módulo de recogida de información del AA que, además de los datos de entrada necesarios para la ejecución de la aplicación, debe recoger la información general solicitada por el módulo de acoplamiento. La información recogida es enviada al MAG mediante un fichero intermedio.

2.- *Conexión del módulo de tratamiento de resultados de AA con el módulo de acoplamiento GSG*. Para la creación de informes generados por el visualizador de resultados sobre la ejecución de los trabajos enviados a una plataforma Grid, se utiliza la información del MAG.

3.- *Conexión del código base de cada aplicación de la AA con el módulo de ejecución en nodos Grid.* El MENG de GSG contiene, además de los componentes necesarios para la ejecución de una aplicación en el entorno Grid, el código base de la aplicación ejecutada dentro de un nodo de trabajo. Así pues, para la creación de este módulo es necesario el código completo de la aplicación, para incluirlo se utilizan comandos de traspaso de ficheros implementados en el paquete AA.

Conexión SAA – AA

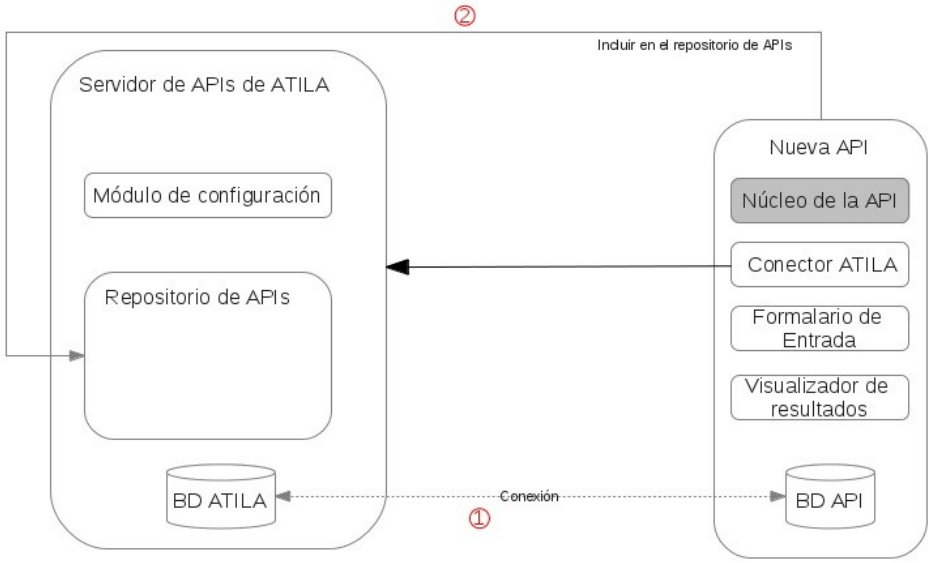


Figura 4.9: Instalación de nueva API en el servidor de aplicaciones ATILA*. Proceso de conexión entre el sistema SAA y el sistema AA.*

1.- *Conexión entre la base de datos general del SAA y la base de datos específica del AA.* Para que una API de ATILA pueda utilizar todas las funciones de SAA se usa la información contenida en su propia base de datos. Esto se hace conectando las dos bases de datos (la general de SAA y la particular de cada una de las AA). De esta forma SAA hace consultas, no sólo a su base de datos, sino también a la propia de cada AA.

2.- *Inclusión de una AA específica dentro del repositorio de aplicaciones de SAA.* SAA contiene un listado de las AA que puede manejar. Este listado está incluido dentro de la base de datos de SAA. Por otro lado, la nueva API está incluida en un repositorio que facilita la gestión de versiones compatibles con SAA. Cada vez que se incluye una nueva AA, ésta es registrada en la base de datos y se añadirá dentro del repositorio. Si se crea una nueva versión de la API, se pone en el repositorio mediante el gestor del control de versiones.

4.3 Resultados experimentales

En este apartado se analiza el uso de las aplicaciones dentro del entorno ATILA integrando el módulo GSG frente al uso de GSG gestionado individualmente. Para este cometido comparamos las distintas soluciones las aplicaciones Cesam [MOR97], *GraCo* [MOY08] y SPM [ROD14].

La aplicación Cesam se encarga de generar modelos de estructura estelar. La aplicación *GraCo* por su parte, genera modelos de pulsaciones a partir de las trazas evolutivas producidas por Cesam. Finalmente el código SPM realiza búsquedas de patrones a una frecuencia establecida mediante una transformada de Fourier.

A continuación se describe esta comparación atendiendo al coste que conlleva el uso de estas implementaciones, a través de los cuatro apartados siguientes.

La instalación de una nueva aplicación en el entorno considerado.

Si se usa el módulo GSG integrado en ATILA es necesario la creación de un módulo AA por cada una de las herramientas que componen la secuencia (en nuestro caso Cesam + *GraCo* + SPM). Para la creación de este módulo se debe implementar un formulario de entrada, una pantalla de visualización de resultados y un módulo de conexión con el servicio central de ATILA como se puede observar en la Figura 4.9. La ventaja principal del uso de esta solución es que la conexión de cada una de las API de ATILA con el módulo GSG es automática. Esto significa que los módulos MCG, MAG y MENG se crean por sí solos con la información obtenida del formulario de entrada, la pantalla de visualización y el módulo de conexión con el servicio central de ATILA.

Si la solución adoptada es la de uso del paquete GSG de forma individual, los módulos MCG, MAG y MENG deben elaborarse cada vez que se instale una nueva aplicación. En nuestro caso se hará en cada una de las tres aplicaciones.

Para concluir este apartado, observamos que la integración de GSG en ATILA en este caso no aporta grandes beneficios ya que se deben elaborar los tres módulos previamente a su funcionamiento.

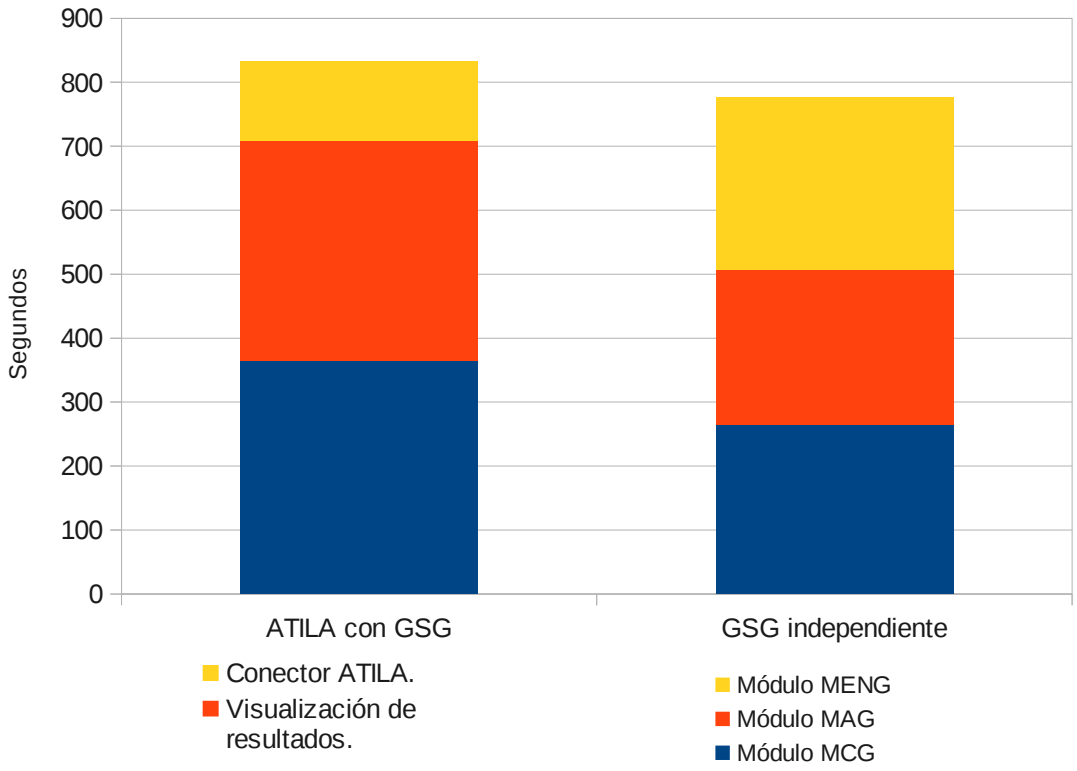


Figura 4.10: *La figura presenta la comparación del tiempo de preparación del entorno para la ejecución en el caso de utilizar ATILA con GSG integrado frente al uso de GSG independiente.. Valores en segundos.*

Preparación del entorno para la ejecución de las aplicaciones.

En el caso de la integración de GSG con ATILA, para la ejecución secuencial de las tres aplicaciones del ejemplo, sólo se debe crear un proyecto siguiendo las directrices de ATILA. Se deben elegir los ficheros necesarios de entrada para la primera aplicación, en nuestro caso Cesam. ATILA se encarga de que la salida de la primera aplicación sea la entrada de la siguiente y así sucesivamente, siendo este cometido transparente para el usuario.

Por otro lado, ATILA permite configurar una serie de características generales de uso de la aplicación y de las herramientas GSG. Unos ejemplos de estas configuraciones son los ficheros de salida de las aplicaciones, la información obtenida por el sistema, la elección de una plataforma Grid donde se va a ejecutar, etc. La configuración se realiza por medio de formularios.

En la ejecución de las distintas aplicaciones en ATILA se genera automáticamente una colección de módulos relacionados con las herramientas GSG. De esta forma el módulo MCG se crea a partir de la configuración de la aplicación mediante formularios implementados en ATILA, el módulo MAG se basa en el formulario de entrada y el visualizador de resultados de ATILA y finalmente el módulo MENG incluye internamente el núcleo de la API de cada ATILA API.

En caso de usar GSG de manera independiente, no existe conexión entre aplicaciones. Así pues, la ubicación correcta de los ficheros de entrada de cada aplicación es tarea del usuario, debiéndose situar los resultados de salida de la primera aplicación al directorio de elementos de entrada de la segunda. Esta tarea implica un mayor coste temporal.

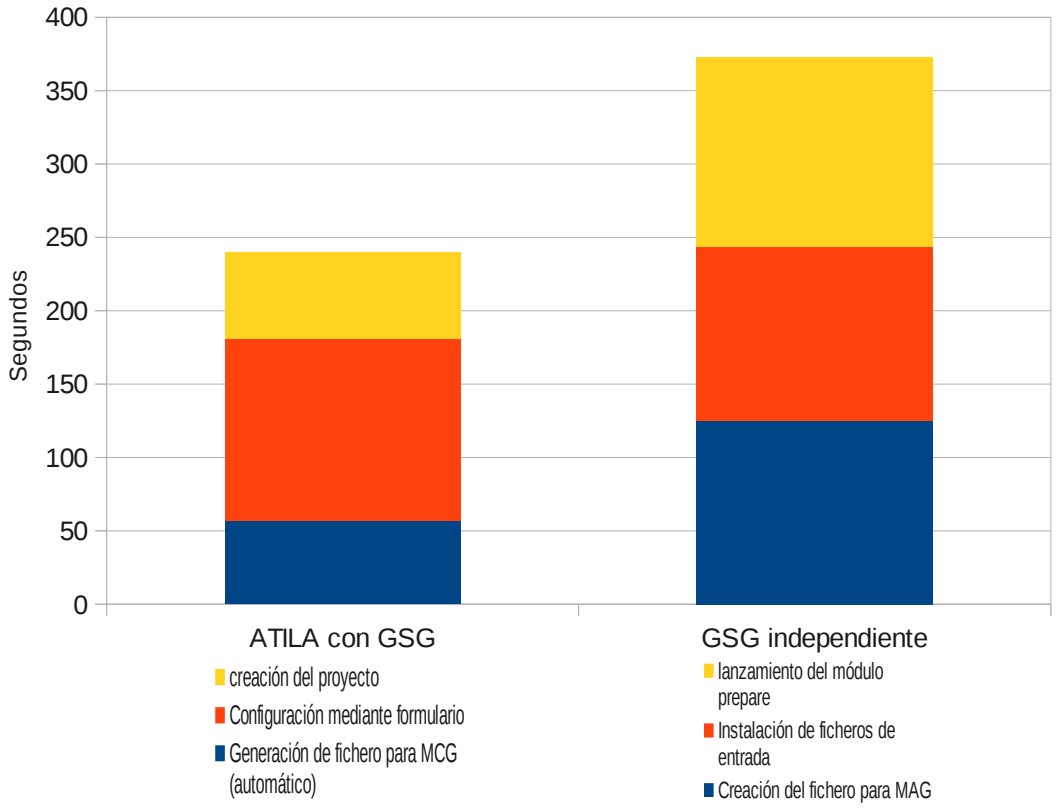


Figura 4.11: Comparación del tiempo de instalación de las aplicaciones utilizando GSG de manera independiente. Las unidades están en segundos. En la figura se compara el tiempo de preparación del entorno para la ejecución de ATILA con GSG integrado, frente a GSG usando de manera independiente.

Por otro lado, el sistema de configuración GSG no es tan avanzado como el de ATILA ya que el primero se realiza mediante la creación de los módulos MCG editando ficheros de configuración. Este método no abarca tantas opciones como ATILA, aunque sí las suficientes para configurar completamente el uso de una plataforma Grid. Por tanto, aunque sean dos opciones diferentes de configuración, el resultado es el mismo para la ejecución en plataformas Grid.

Tabla 4.1: Tiempos medios (Md) y desviaciones típicas (De) del tiempo de instalación de aplicaciones. Las unidades están en segundos. CP: creación del proyecto. CF: Configuración mediante formulario. MCG: Generación de fichero para MCG. LM: Lanzamiento del módulo prepare GSG. IF: Instalación de ficheros de entrada. MAG: Creación del fichero para MAG.

	CP		**CF**		**MCG**		**LM**		**IF**		**MAG**	
	Md	**De**	**Md**	**De**	**Md**	**De**	**Md**	**De**	**Md**	**De**	**Md**	**De**
Cesam	60	7.3	123	14,6	54	3,7	125	12,3	121	12,7	120	13,5
GraCo	67	9,6	132	12,3	56	4,1	138	15,3	135	15,7	134	14.3
SPM	57	3,2	116	11,3	59	4,2	119	9,7	143	11,2	128	11.2

Para comparar el tiempo de preparación del entono para la ejecución de ATILA con GSG integrado frente a GSG usado de manera independiente se ejecutan una batería trabajos de evaluación para 3 aplicaciones distintas: Cesam, *GraCo* y SPM. En la Tabla 4.1 se observa el resultado es la media de tiempo en cada una de las ejecuciones de cada código.

De la comparación de los tiempos de instalación de las aplicaciones se concluye que en todos los casos ejecutados de las distintas aplicaciones la mejora del tiempo de instalación usando ATILA con GSG integrada es considerable (entorno a un 45% del total).

Ejecución de aplicaciones en una infraestructura Grid.

Tabla 4.2: Parámetros de entrada para la generación de modelos.

Parámetro relacionado con la estrella	**Valor**	**Unidades**
M - Masa	[1- 10]	Masas solares
Fe/H–Índice de Metalicidad	0.02	dex
A – índice de convección	0.5	---
O – Overshooting relacionado con la convección	0.2	---
Rot – Rotación	0.65	Radianes/segundo

En el estudio para la ejecución de aplicaciones astrosismológicas en una infraestructura Grid se tiene en cuenta el tiempo medio de ejecución de las aplicaciones. Este estudio se ha realizado en una misma arquitectura, descrita en la Sección 2.2.1, con lo cual nos aseguramos de tener la mismas condiciones computacionales. Además, se han utilizado los mismos perfiles físicos para la generación de modelos variando el parámetro de la masa de la estrella del modelo de estructuras a generar y manteniendo las características físicas, descritas en la Tabla 4.2. Estos elementos son los parámetros de entrada que se definen para la ejecución de la aplicación Cesam y *GraCo* (Sección 2.3.2).

Tabla 4.3: Tabla de tiempos de las ejecuciones para las aplicaciones Cesam2K, GraCo y SMP. Las unidades están en minutos.

Aplicación	**Tiempos de ejecución**
Cesam2K	133, 150, 164, 152, 132, 163, 154, 153, 153, 161
GraCo	312, 334, 360, 312, 345, 313, 359, 372, 312, 372
SPM	73, 76, 86, 65, 59, 72, 73, 94, 80, 72

La Tabla 4.3 presenta el análisis de tiempos de ejecución de las tres aplicaciones estudiadas, que son Cesam, *GraCo* y SPM.

Tabla 4.4: Tiempos medios y desviaciones típicas de la ejecución de aplicaciones. Las unidades están en segundos.

Código	**Media de la duración en segundos**	**Desviación Típica**
Cesam	8.460,00	2.700,12
GraCo	19.800,00	8.280,67
SPM	4.020,00	780,43

Como resumen, se presenta la Tabla 4.4 de 10 ejecuciones de cada una de las aplicaciones con las medias y las desviaciones típicas asociadas.

En el caso de la utilización de GSG de manera independiente para la ejecución de aplicaciones en una plataforma Grid, cada aplicación utiliza las cuatro herramientas de que dispone dicho paquete. Estas son, *prepare, run, verify y cancel* (en el caso que sea necesario), vistas en las secciones 3.2.1, 3.2.2, 3.2.3, y 3.2.4, respectivamente. Esta circunstancia impone la necesidad de lanzar los bloques de trabajos de cada aplicación de forma independiente, ya que la conexión entre las aplicaciones que aporta ATILA es inexistente, por usar GSG de manera independiente. Las etapas a seguir son:

1	Preparación + lanzamiento GSG Cesam
2	Preparación + lanzamiento GSG *GraCo*
3	Preparación + lanzamiento GSG SPM

Este método es independiente del número de repeticiones del experimento y no hay forma de almacenar los resultados de la secuencia para el caso en el que se quiera repetir la ejecución, como ya se vio en la Sección 3.2.3.

En el caso de la utilización de ATILA, en ningún caso de los que se presentan a continuación es necesaria la transferencia de los resultados obtenidos y almacenados en el servicio Grid de almacenamiento (Storage Element – User Interface) al directorio elegido por el usuario en la máquina donde se ejecuta ATILA. Este paso lo hace automáticamente el módulo de visualización de resultados de dicho servidor, ya que tiene un código especializado para la transferencia entre el UI y la máquina desde donde se ejecuta ATILA. Por tanto no se necesita tiempo para esta tarea.

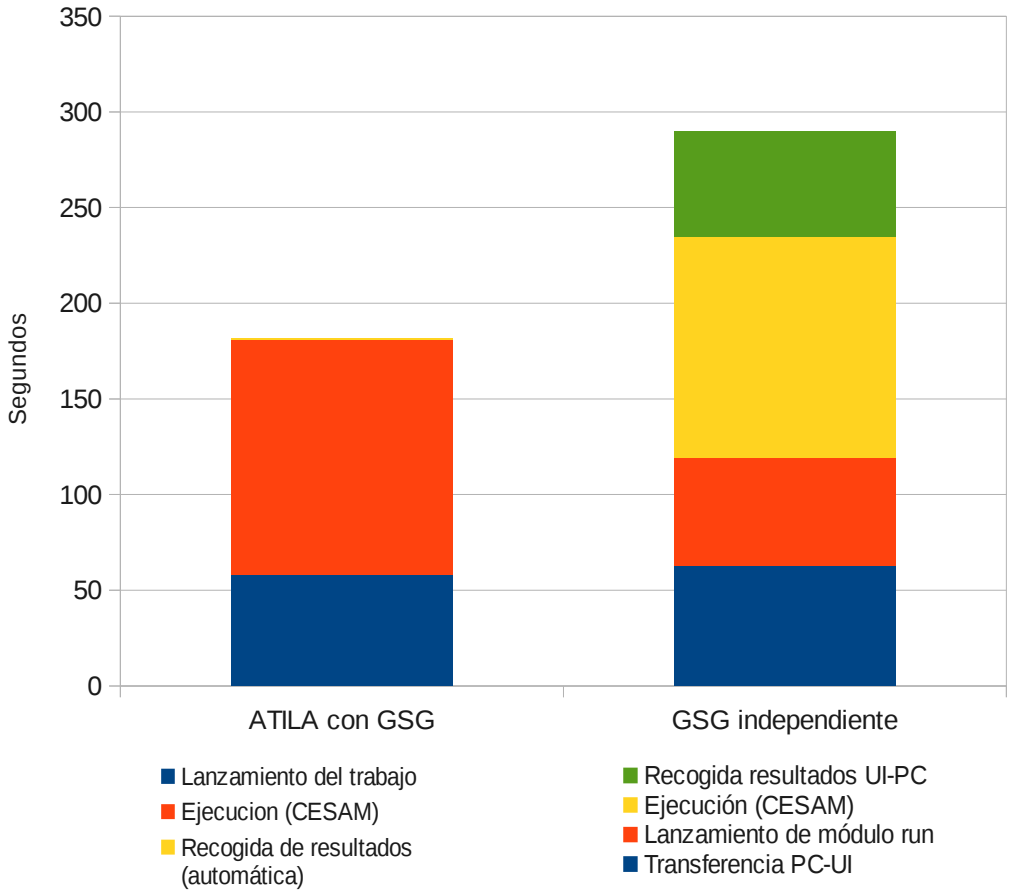


Figura 4.12: Ejecución simple, que lanza un trabajo de la aplicación Cesam al entorno Grid mediante ATILA con GSG frente a GSG independiente.

Para satisfacer la necesidad de repetir los experimentos de forma continua, el servidor de aplicaciones ATILA diferencia los siguientes casos de ejecuciones:

1.- Uso simple de una infraestructura Grid. En este caso, se ejecutan las aplicaciones de forma secuencial sólo una vez, y no se necesita repetir el experimento. Tal y como se observa en la composición de la Figura 4.12, aquí no se aprovechan las características de ATILA, ejecutándose las aplicaciones secuencialmente de la siguiente forma:

Configuración Cesam + Configuración *GraCo* + Configuración SPM
Lanzamiento secuencial de las tres aplicaciones

Como se observa en la Figura 4.12, el incremento de tiempo acumulado de todas las fases en el caso de la ejecución para la aplicación Cesam es notorio, lo cual refleja que el uso de ATILA con GSG es recomendable en el caso de ejecuciones simples de una única aplicación. En el caso de querer repetir experimentos de forma rápida y eficaz, habría que proceder según se indica en el siguiente apartado.

2.- Repetición de ejecuciones de un misma secuencia pero con distintos parámetros de entrada. En este caso, y como se observa en la Figura 4.13, si se opta por usar el método de integración de GSG con ATILA, se aprovecha el concepto de flujo de trabajo al poderse ejecutar un número indeterminado de ejecuciones para un secuencia concreta de aplicaciones. La secuencia de aplicaciones se graba y puede utilizarse tantas veces como se quiera, sin tener que configurar de nuevo sus características.

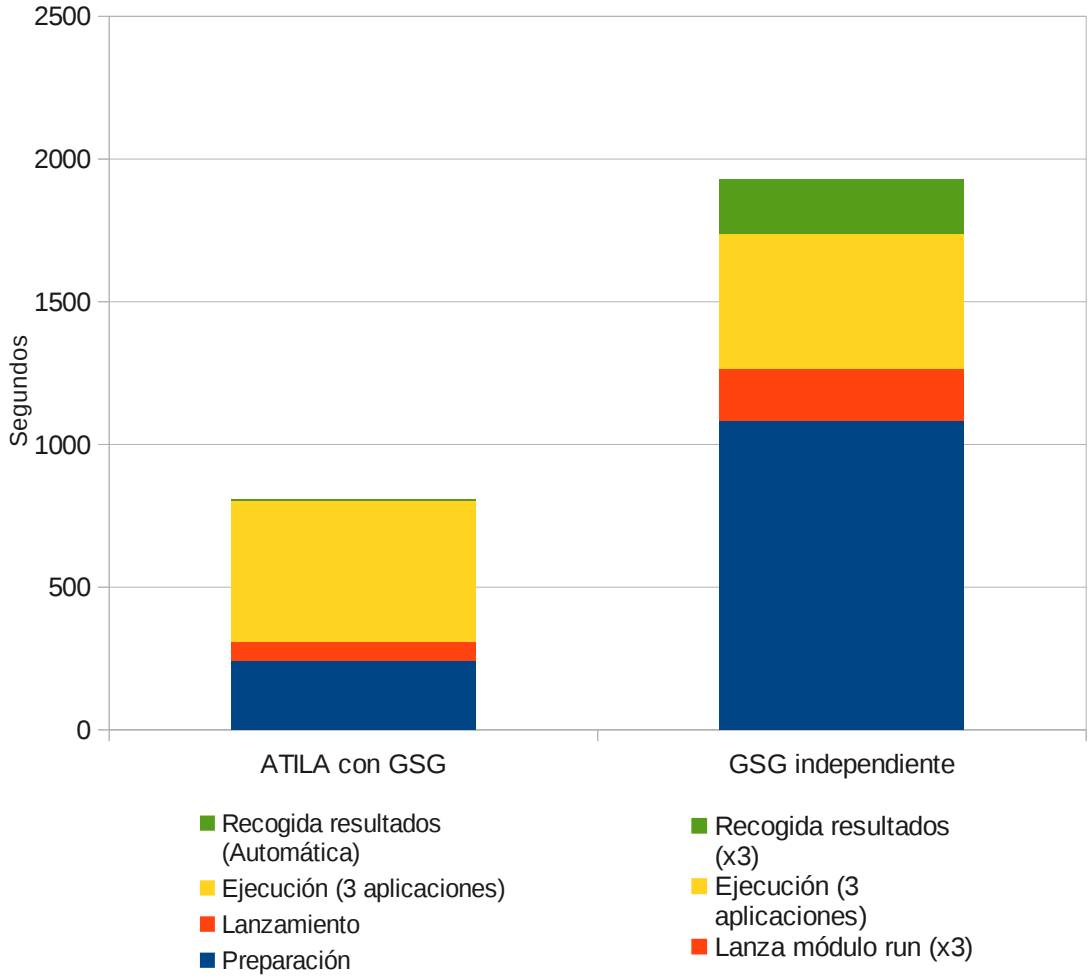


Figura 4.13: Ejecución multi-aplicación, que lanza un trabajo secuencialmente usando tres aplicaciones, Cesam + GraCo + SPM. ATILA con GSG frente a GSG independiente.

Como se observa en la Figura 4.13, el incremento de tiempo en el caso de la ejecución de varias aplicaciones ejecutadas secuencialmente para este trabajo de evaluación es de aproximadamente un 100%, lo cual refleja que el uso de ATILA con GSG es recomendable en el caso de ejecuciones multiaplicación.

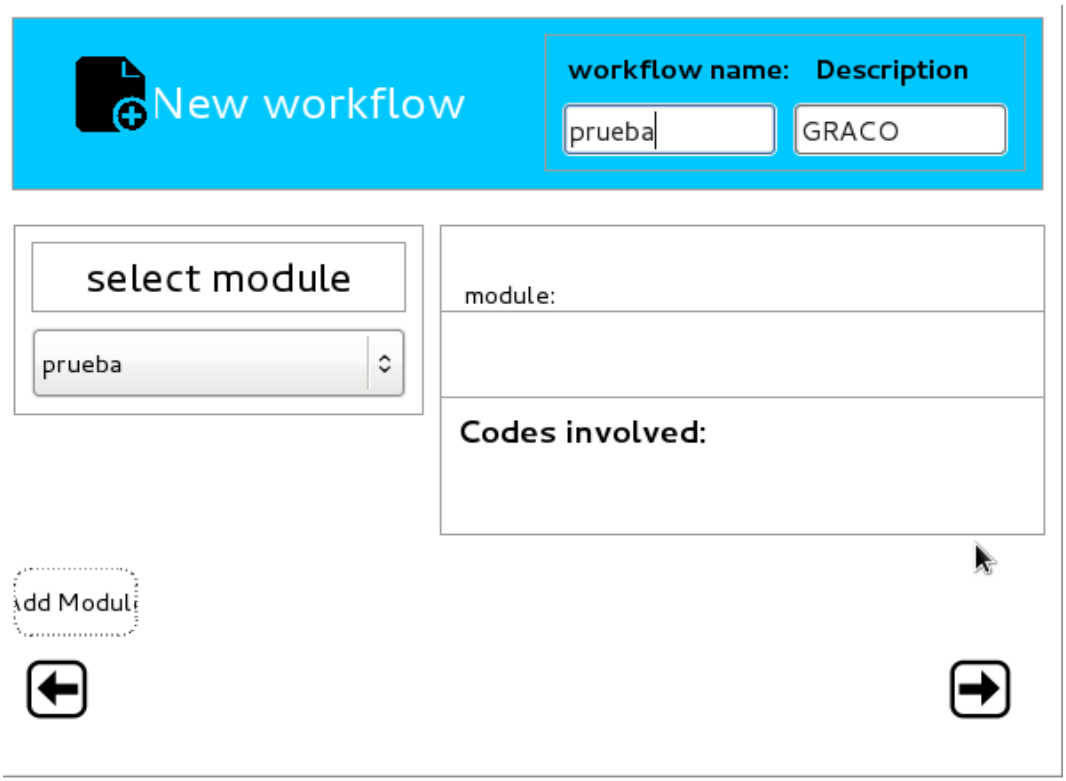


Figura 4.14: Plantalla de configuración para la creación de flujos de datos.

Como ejemplo, para el caso de usar el flujo de datos Cesam, *GraCo* y SPM descrito con anterioridad, la estructura de la ejecución se debe crear mediante el módulo de Flujo de datos perteneciente a ATILA especificando las aplicaciones a ejecutar y el orden en la secuencia de trabajo. En la Figura 4.14 se ve la pantalla de configuración de este módulo.

3.- Ejecución de aplicaciones diferentes pero con una misma colección de datos de entrada. Mediante la configuración del perfil físico se pueden seleccionar los datos de entrada que se usan para distintas aplicaciones, en el caso de utilizarse dentro de un flujo de datos, o de forma independiente. De esta forma, si una misma colección de datos generados mediante unos parámetros físicos idénticos se quiere ejecutar en distintas APIs de ATILA (AA) para comparar los resultados de cada una de ellas, simplemente se hará uso de la grabación del perfil físico de los datos mediante ATILA. La grabación del perfil físico permite automatizar este proceso.

Teniendo en cuenta el control de los trabajos ejecutados, el uso de ATILA conlleva un mayor control del estado de los trabajos y de la información de cada una de las ejecuciones ya que toda la información de estos pasos se recopila en una base de datos a la que tiene acceso ATILA en cualquier momento. Por el contrario, al usarse independientemente las herramientas GSG, la información se recoge mediante el SRE basado en la escritura de ficheros. Esta información muestra una consulta una vez finalizado el trabajo, al contrario que ocurre con la opción de usar ATILA con GSG.

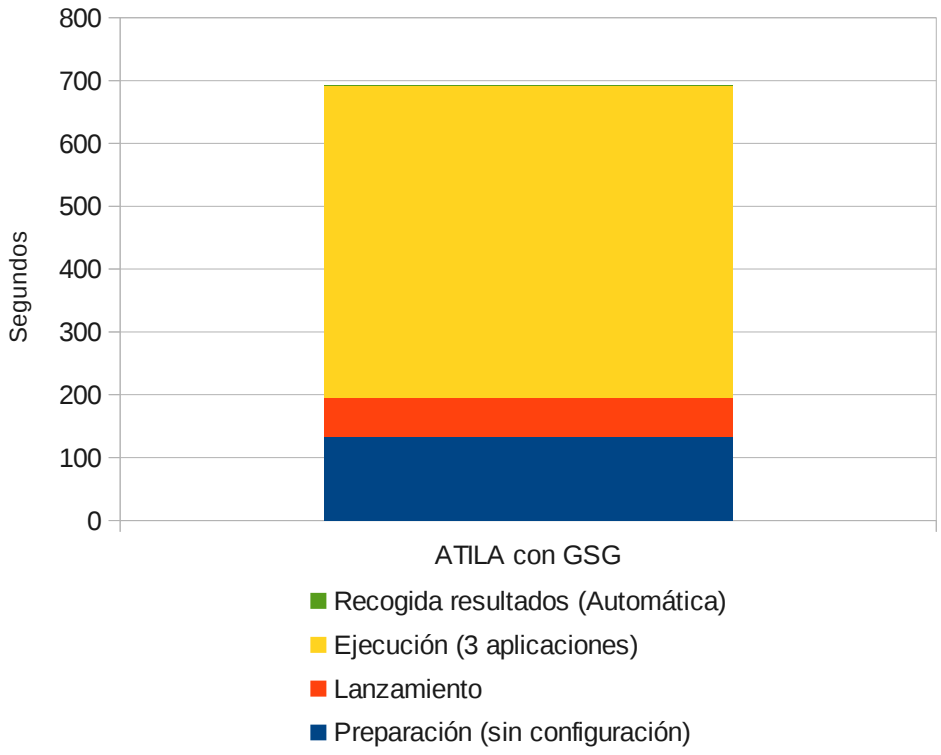


Figura 4.15: Repetición de experimento, utilizando la misma física en la configuración de las aplicaciones y se lanza de nuevo la ejecución multi-aplicación.

En cuanto a la recogida de resultados, el servidor de aplicaciones ATILA lo hace automáticamente en el ordenador donde esté instalado el programa. En el caso de la ejecución independiente de GSG, esta captura se hará dentro de la ubicación configurada dentro de servicio de interfaz de usuario (UI, por sus siglas en inglés) donde se ejecutan las herramientas. Se debe transferir de forma manual en el caso que se quiera tener dichos datos en el ordenador personal de cada usuario. Esto conlleva un tiempo añadido a la realización de los trabajos. En la Figura 4.15 se observa el tiempo completo de la realización de un experimento. El tiempo de la recogida del resultado no existe ya que se hace de forma automática por ATILA.

Ventajas añadidas.

Nótese que para este caso de ejecución el tiempo usando GSG de manera independiente es el mismo que para el caso de un lanzamiento de trabajos multiaplicación. Además el tiempo de recogida de resultados es muy reducido en comparación con el resto de tiempos.

En el ejemplo de las ejecuciones de las diversas aplicaciones existe un valor añadido que consiste en la clasificación de lo resultados obtenidos mediante una entidad llamada *Colección de Datos* de tal forma que una vez obtenidos los resultados, estos pueden consultarse más eficientemente mediante herramientas de este servidor. Esto no es posible si la ejecución se hace usando GSG independientemente, ya que al no usar una base de datos no existe la posibilidad de clasificación de los datos.

Por otro lado, si se usa GSG de forma independiente, se gana en sencillez ya que se ahorra el coste de las bases de datos, y no existen los conceptos de flujos de trabajo, perfiles físicos, proyectos o colecciones de datos. De esta forma, el paquete de herramientas GSG, usado de manera independiente, es recomendable en el caso de un envío de trabajos simple.

4.4 Conclusión

Hemos desarrollado la herramienta ATILA, que contribuye al manejo de las aplicaciones astrosismológicas añadiendo funcionalidades relacionadas con la utilización de flujos de datos, con la generación de perfiles físicos para su reutilización, con la generación de modelos, con la clasificación de los datos mediante colecciones de datos, y finalmente, con el uso de distintas infraestructuras como pueden ser la tecnología Cloud, servidores dedicados, y, por supuesto, las infraestructuras Grid.

Para el uso de este último tipo de infraestructuras, hemos conectado cada uno de los componentes del paquete de herramientas GSG con los distintos módulos de ATILA, de forma que se pueda reutilizar todo el código generado para adaptar GSG a cualquier aplicación de manera optima.

Hemos llegado a la conclusión que el uso de ATILA con el sistema GSG integrado es mucho más eficiente que el uso de las herramientas GSG utilizadas de manera independiente. Como resumen de la comparación de las dos alternativas se presenta la Tabla 2.13, que muestra cada uno de los pasos en cada una de las fases:

Tabla 4.5: Comparativa de ejecución de trabajos usando ATILA con el módulo GSG frente al uso del paquete de herramientas GSG de forma independiente.

Apartado	**Usando ATILA con integración de GSG**	**Utilizando GSG independientemente.**
Instalación de una nueva aplicación en el entorno escogido	- Creación de ATILA API. * Formulario de entrada. * Visualización de resultados. - Conector de ATILA. * ATILA conecta automáticamente AA con GSG.	- Creación de los módulos MCG, MAG, MENG.
Preparación del entorno para la ejecución de las aplicaciones	- Creación de nuevo proyecto. - Elección de datos de entrada en el directorio correspondiente. - Elección de plataforma Grid. - Automáticamente se genera módulos: * MCG a partir de la configuración. * MAG a partir del formulario de entrada y el visualizador de resultados. * MENG incluye el núcleo de la API. - Generación automática de Módulos GSG.	-Instalación de datos de entrada en el directorio requerido. * Módulo *prepare*. - Configuración del sistema mediante ficheros.
Ejecución de aplicaciones en una infraestructura Grid.	- Ejecución desde PC. - Distintos tipos de uso * Uso simple. * Misma física en datos entrada. * Repetición de ejecuciones. - ATILA tiene mayor control ya que tiene accesos a las BD. - Recogida de resultados automática e instalación en el PC correspondiente. - Lanzamiento de trabajos mediante flujos de datos. - ATILA añade flujos de datos y perfiles físicos. - Posibilidad de repetición de experimentos.	- Ejecución desde UI. * Transferencia PC – UI. - Por cada aplicación ejecución de las cuatros herramientas: * *prepare, run, verify y cancel* (en el caso que sea necesario). - Necesidad de lanzamiento de trabajos de uno en uno: * Preparación + lanzamiento GSG Cesam. * Preparación + lanzamiento GSG *GraCo*. * Preparación + lanzamiento GSG SPM. - Recogida de resultados automática. - Ubicación dentro del UI.
Ventajas añadidas en cada caso.	Clasificación de datos mediante colecciones de datos.	Simplicidad.

ATILA se ha implementado como una aplicación cliente-servidor cuyo componente cliente debe instalarse y ejecutarse en cada ordenador personal.

Finalmente, la comunidad astrosismológica se ha visto reforzada por distintos consorcios que permiten respaldar iniciativas científicas, como es el caso del propio servidor de aplicaciones ATILA. Entre los más destacados se encuentran el consorcio SpaceInn (Exploitation of Space Data for Innovative Helio and Asteroseismology) [SPA14], puesto en marcha por el HELAS (European Helio and Asteroseismology Network) [HEL14] y cuya la misión es construir, para uso de la comunidad de física estelar, una plataforma que facilite el acceso a datos, a conocimientos científicos, y la coordinación entre grupos. De esta manera se pretende asegurar el uso óptimo de los datos existentes obtenidos desde el espacio y desde observatorios terrestres, para el estudio del sol y otros estudios relacionados con la astrosismología.

Al igual que se ha utilizado el sistema GSG para proveer el uso de una infraestructura Grid dentro de la plataforma API, también pretendemos crear un nuevo paquete de herramientas que facilite el uso de la infraestructura Cloud para este servidor.

En el siguiente capítulo, se muestran las principales conclusiones, las aportaciones más relevantes en este trabajo y las líneas de trabajo futuras relacionadas con la presente Tesis.

Capítulo 5: Conclusiones, aportaciones y líneas futuras

5.1 Conclusiones

1. Hemos desarrollado un método basado en la optimización de los elementos de portabilidad de herramientas a entornos de computación distribuidos, incluido la gestión de la computación. Este método, focalizado en el ámbito de aplicaciones científicas, presenta tres aspectos fundamentales.
   - Es novedosa, ya que trata el proceso de adaptación como un problema de optimización global. Además, propone reglas objetivas para disminuir, en la medida de lo posible la intervención humana.
   - Es coherente, en cuanto la ejecución de los pasos a seguir para conseguir las optimizaciones descritas por el método.
   - Es robusta y versátil, ya que es válida para cualquier plataforma distribuida y/o aplicación y por construcción presenta pocos fallos.
2. La búsqueda de un método ya desarrollado que proporcionara las tres optimizaciones descritas como objetivo de esta Tesis fue infructuosa, haciéndose necesario realizar un método novedoso, coherente y autoconsistente y focalizado en el ámbito de aplicaciones científicas.
3. Hemos desarrollado un paquete de herramientas para el uso de Grid, llamado GSG, ampliando la usabilidad, la monitorización, la información acerca del estado, la seguridad, la optimización en la distribución de tareas, y el envío de trabajos frente al uso del middleware estándar de Grid.

4. Gracias a GSG, hemos minimizado el tiempo de gestión en entornos distribuidos, tanto de computadora como de usuario. Desarrollamos GSG con una estructura modular preparada para su integración en aplicaciones. Al implementar este desarrollo, los usuarios científicos se han beneficiado al aumentar la usabilidad del entorno Grid, viéndose incrementada la productividad científica.

5. Hemos aplicado el método a una muestra de aplicaciones astrofísicas para las que el uso de plataformas distribuidas es imprescindible. Éstas cubren una tipología de requerimientos computacionales y de almacenamiento muy diversa y complementaria, lo que nos ha permitido evaluar mejor la eficiencia del método.

6. Para cada una de las aplicaciones adaptadas a Grid, hemos generado una solución específica que ha permitido reducir significativamente el tiempo de computación y aumentar la capacidad de almacenamiento en todos los casos estudiados, llegando a multiplicar los recursos disponibles. Todos los estudios científicos descritos en esta memoria no se hubieran podido realizar sin la aplicación de este método debido a los altos requerimientos computacionales.

7. Se ha realizado diversos análisis comparativos entre un servidor dedicado no distribuido frente a una estructura distribuida Grid durante la ejecución de aplicaciones con diferentes necesidades computacionales y de almacenamiento. La aplicación de un método coherente y autoconsistente optimiza al máximo el uso de entornos distribuidos, acercándose en todos los casos analizados a la cota de máxima de mejora impuesta por la infraestructura utilizada.

8. Hemos desarrollado un servidor de aplicaciones astrosismológicas, que denominados ATILA, para el tratamiento y la ejecución de este tipo de aplicaciones. Este servidor incorpora funciones como la utilización de modelos de ejecución de flujos de datos, la generación de perfiles físicos, la generación de modelos astrosismológicos, la clasificación de los datos o el uso de múltiples infraestructuras distribuidas. Además, hemos integrado el paquete de herramientas GSG en servidor de aplicaciones ATILA, lo que permite usar automáticamente de las funcionalidades de GSG en el entorno ATILA.

9. Como consecuencia de este desarrollo, se ha mejorado la usabilidad y el tiempo de ejecución del servidor de aplicaciones ATILA al integrar el sistema GSG. Para llegar a esta conclusión se han realizado experimentos comparativos de ATILA con GSG integrado, frente al uso independiente de ATILA y GSG.

5.2 Valor añadido

La importancia de la utilización de recursos computacionales para ejecutar aplicaciones astronómicas en un tiempo razonable ha sido la principal motivación de esta Tesis. Las soluciones descritas en la memoria nos han permitido, además obtener los siguientes valores añadidos:

- Mejorar la capacidad de cómputo para abarcar nuevos retos científicos.
- Aumentar la capacidad de almacenamiento, la seguridad y la replicación de los datos.
- Facilitar de manejo de infraestructuras Grid, relativamente complicado para el usuario medio en el campo de la astronomía.
- Paralelizar la ejecución de códigos astronómicos relevantes haciendo hincapié en su organización modular.
- Conectar con la herramienta de manejo de Base de Datos astronómicos del Observatorio Virtual [IVOA14] (VO)
- Crear una herramienta generalista con el fin de usarse dentro de las infraestructuras Grid, capaz de ejecutar aplicaciones creadas en diversos campos de la astronomía. Además de facilitar el uso de esta plataforma simplificando el número de comandos, y el acceso mediante certificados robots y organizaciones virtuales generalistas contenidas en la iniciativa Ibergrid [CAM12].

5.3 Líneas de futuro.

En las actividades que contemplamos para seguir mejorando la posibilidad de utilizar eficientemente plataformas distribuidas en Astronomía destacamos:

1. La realización de un estudio sobre la adaptación de aplicaciones a plataformas distribuidas de forma automática, usando inteligencia artificial y redes neuronales con retroalimentación en cada una de las ejecuciones [CRA06]. De esta forma se dota de independencia al método, obteniendo una adaptación automática a la circunstancia.
2. La adaptación de las herramientas creadas a lo largo de esta Tesis para que se puedan usar en otras infraestructuras como pueden ser las plataformas Cloud, los servidores de supercomputación, o la computación voluntaria. Para ello habrá que adaptar las herramientas GSG según las necesidades de cada una de cada plataforma.
3. La exploración de la integración de sistemas desarrollados en la Tesis (GSG) dentro de entornos de trabajo VO [SUA14]. De esta forma conseguiríamos facilitar el descubrimiento, acceso y representación de datos usando entornos de Grid o Cloud para el procesamiento masivo y el análisis de datos.
4. La Implementación de ATILA como un servicio web que pueda utilizarse a partir de una interfaz web. De esta forma se conseguirá independencia de la arquitectura donde se ejecuta.
5. La evaluación del comportamiento y eficiencia del método propuesto y el paquete de herramientas GSG con otro tipo de sistemas. En concreto, en sistemas de dinámica planetaria cuyo cálculo pertenece a problemas numéricos de tipo Many-Body para la interacción de múltiples partículas, cuya solución no es analítica sino numérica, con la consiguiente necesidad grandes recursos de computación.

5.4 Principales publicaciones

El trabajo realizado en esta Tesis a generado diez publicaciones: cuatro en revistas internacionales, tres en congresos científicos internacionales y cuatro en congresos científicos nacionales.

5.4.1 Revistas internacionales

Título: **XMM-Newton Data Analysis with SAS Software Over Grid.** Highlights of spanish astrophysics VBook Series: Astrophysics and Space Science Proceedings pp. 479-479. JCR 2014: 2.263

Autores: M.T. Ceballos, I. Campos, P. Orviz, D. Tapiador, R. Alvarez, A. Ibarra, C. Gabriel, J.R. Rodon

Referencia: [CEB08]

Título: **The evolution of galaxies resolved in space and time: a view of inside-out growth from the CALIFA survey**. Astrophysical Journal Letters - ApJL/457268/ART/224183 http://stacks.iop.org/2041-8205/764/L1. JCR 2014: 5,339

Autores: Pérez, E.Cid Fernandes, R.Gonzalez-Delgado, R. García-Benito, J.R. Rodón, S.F. Sanchez, B. Huseman, D. Mast et al.

Referencia: [PER13]

Título: **An in-depth study of HD 174966 with CoRoT photometry and HARPS spectroscopy.** Large separation as a new observable for delta Sct stars. Astronomy & Astrophysics http://dx.doi.org/10.1051/0004-6361/201220256. JCR 2014: 4.378

Autores: García Hernández, A.; Moya, A.; Michel, E.; Rodon, J.R**.** et al.

Referencia: [GAR13]

Título: **Measuring mean densities of δ Scuti stars with asteroseismology Theoretical properties of large separations using TOUCAN.** Astronomy & Astrophysics. DOI: 10.1051/0004-6361/201322270. JCR 2014: 4.378

Autores: J. C. Suárez, JC. García Hernández, A. Moya, A. Rodrigo, C, Solano, E. Garrido,R. Rodón, JR.

Referecia: [SUA14]

5.4.2 Congresos internacionales

Título: **IAA Grid Team: porting Astrophysical applications to Grid.** 5º EGEE User Forum 2010.

Autores: Rodón JR, Benitez A, Ruedas J., Sánchez S., Ibañez JM., Cuesta AJ., Osorio. M, Sabater. J, García. A.

Referencia: [ROD10a]

Título: **User Grid made easy: Creating generic script for specific problems**. 1st EGI User Forum 2011.

Autores: Rodón, JR. Benitez, AD. Passas, M. Ruedas, J. Ortega, J.

Referencia: [ROD11a]

Título: **An example of GSG tools integration:** *Starlight* **Spectral Synthesis Code.** EGI Technical Forum 2012.

Autores: Rodón, JR. García-Benito, R.

Referencia: [ROD12]

5.4.3 Congresos nacionales

Título: **Análisis de datos de XMM con SAS sobre tecnología GRID.** VIII Reunión Científica de la SE.

Autores: M. Ceballos, I. Campos, D. Tapiador, R. Álvarez, A. de la Fuente, A. Ibarra, C. Gabriel, J.R.Rodón

Referencia: [CEB08]

Título: **IAA Grid Team: porting Astrophysical applications to Grid.** IBERGRID 2010. (Formato digital) ISBN: 978-84-9745-549-7. pag.510

Autores: Rodón, JR. Benitez, Ruedas, J. Sánchez, S. Ibañez, JM. Cuesta, AJ. Osorio, M. Sabater. J, García, A.

Referencia: [ROD10b]

Título: **GSG Tool: Creating generic scripts for problems on Astrophysics.** IBERGRID 2011. ISBN: 978-84-9745-884-9 pag.333

Autores: Rodón JR, Benitez AD, Passas M, Ruedas J, Ortega J.

Referencia: [ROD11b]

Título: **Soluciones computacionales para la simulación de sistemas planetarios.** CPESS4

Autores: Rodón, JR. Pozuelos, FJ. Suárez JC, Berdiñas ZM.

Referencia: [ROD15]

Concluye aquí este trabajo de Tesis, con la que se han realizado aportaciones significativas al campo de la implementación de aplicaciones de astronomía en plataformas distribuidas a través de la generación de nuevos métodos, herramientas y servicios que constituyen avances relevantes de cara a facilitar el uso y la explotación eficientes de las plataformas distribuidas.

Apéndice I: Aplicaciones astronómicas adaptadas a plataformas Grid

Además de las aplicaciones analizadas en los capítulos 2.3.1, 3.2 y 2.3.3, a largo de esta Tesis se ha estudiado y analizado la implementación de numerosas aplicaciones de Astronomía en plataformas Grid obteniendo mejoras gracias a la ejecución paralela de trabajos en dichas plataformas. El método de adaptación se ha aplicado de la misma forma que las aplicaciones descritas en el capítulo 2 pero al tener una solución parecida, se ha optado por no hacer un desarrollo extenso en esta memoria. A continuación se describen las demás aplicaciones astronómicas adaptadas a las plataformas Grid.

I.1 GraCo

GraCo (GRAnada oscillation Code) [MOY04, MOY08] es una aplicación que permite resolver sistemas de ecuaciones diferenciales relacionadas con la pulsación estelar.

El objetivo de este código es calcular numéricamente funciones y valores propios de modelos de oscilación adiabáticos [UNN89], es decir, que no tienen en cuenta intercambios de energía entre capas en las oscilaciones y de modelos de oscilación no adiabáticos [UNN89], que tienen en cuanta todos estos cambios como es la interacción de la atmósfera con la pulsación estelar. Entre las salidas de este código están las frecuencias, tasa de crecimiento y otros valores cuantitativos físicos de estrellas y las características propias de los modos de oscilación.

I.2 Cesam

El código **Cesam** es un conjunto de programas y rutinas que realizan cálculos, en una dimensión, de modelos de evolución estelar cuasi-hidrostática, incluyendo la difusión microscópica de especies químicas y difusión del momento angular.

La solución del equilibrio cuasi-estático se lleva a cabo por un método basado en la función de proximidad usando aproximaciones polinómicas proyectadas sobre un B-spline básico. Esto permite cálculos estables y robustos, y la restitución exacta de la solución, no sólo en puntos de la red de resultados, sino incluso para las variables discontinuas.

Otras ventajas son la monitorización usando sólo un parámetro de la precisión y la mejora por super-convergencia. Una malla automática de refinamiento ha sido diseñada para ajustar las localizaciones de puntos de la red de resultados de acuerdo con los cambios de los parámetros. Para los modelos estándar, la evolución de la composición química se resuelve mediante esquemas rígidos estables para órdenes superiores a cuatro. En las zonas de convección mixta y evolución química la solución de la ecuación de difusión emplea el esquema de Galerkin de elementos finitos. La mezcla de productos se lleva a cabo seguidamente, por una difusión turbulenta fuerte. Esto permite una restauración precisa de la atmósfera.

I.3 MMCG

MMCG (Laboratoire de Meteorologie Dynamique Mars General Circulation Model) [FOR99] calcula la evolución temporal de las diferentes variables (por ejemplo, las temperaturas, vientos, presión, densidades de distintos compuestos, etc) que controlan o describen la atmósfera marciana en diferentes puntos de una rejilla 3D que cubre toda la atmósfera desde la superficie a la exobase. A partir de un estado inicial, el modelo calcula la evolución de estas variables en distintos pasos de tiempo:

• En el instante t, se obtienen las variables X_t (por ejemplo, la temperatura) en un punto de la atmósfera.

• Se calcula la evolución (las tendencias):

$$\left(\frac{\delta X}{\delta t}\right)_1, \left(\frac{\delta X}{\delta t}\right)_2$$

Son las derivadas de cada fenómeno físico, mediante una parametrización de cada uno de estos fenómenos (por ejemplo, el calentamiento debido a la absorción de la radiación solar).

• En el siguiente paso el tiempo $(t+\delta t)$ se calcula mediante $X_{t+\delta t}$ a partir de X_t y $(\frac{\delta X}{\delta t})$. Esta es la integración de las variables en el tiempo. (Por ejemplo,

$$X_{t+\delta t} = X_t + \delta t(\frac{\delta X}{\delta t})_1 + \delta t(\frac{\delta X}{\delta t})_2 + \ldots \ )$$

Esta aplicación se usa para estudiar la temperatura, dinámica y composición de la alta atmósfera marciana, incluyendo tanto atmósfera neutra como ionosfera. Se han obtenido los resultados de la simulación que cubre un año marciano completo, centrándose en la variabilidad de la temperatura estacional, diurna y del día a día en la región exobase [GON09, GON13].

Para ejecutar el modelo, un año marciano se divide en 12 meses. Por otro lado, la ejecución de cada mes (en la versión paralela del modelos) dura entre 18 y 30 horas aproximadamente, en la plataforma descrita en la Sección 2.2.2. Teniendo en cuenta estos datos, la simulación de un año marciano lleva aproximadamente 12 días de cálculo. Teniendo en cuenta la frecuencia de envío de datos, hay que completar del orden de 10 trabajos completos por semana.

En cuanto al almacenamiento, cada mes genera datos de salidas de alrededor de 5GBytes, es decir, 60 GBytes en total por un año simulado. Por tanto, es necesario el uso intensivo del elemento de almacenamiento de la plataforma.

I.4 XMM – SAS

La herramienta **XMM-SAS** (Software Analysis System) [IBA10] es un conjunto de herramientas de análisis desarrolladas para procesar los datos obtenidos por el observatorio de rayos X XMM-Newton [CEB10].

El procesado de datos que hacen estas herramientas genera productos estándar calibrados como, por ejemplo, listas de eventos recibidos, espectros y curvas de luz del satélite, imágenes, mapas de exposición y de fondo en diferentes bandas de energía.

Los usuarios de los archivos de datos generados por XMM pueden generar nuevos resultados científicos o incluso volver a generar los productos estándar para aprovechar, por ejemplo, una nueva información de calibración.

Existen dos opciones posibles a disposición de los científicos para realizar el procesado de los datos:

- Utilizar los recursos informáticos locales (las herramientas de SAS se descargan, instalan y ejecutan localmente). Las principales desventajas de esta opción son:
  - La auto-instalación y el mantenimiento de la herramienta y sus elementos adicionales.
  - La descarga y las actualizaciones de la base de datos de calibración.
  - La sobreexplatación de recursos locales. Algunos procesamientos pueden acaparar los recursos locales si el análisis de los datos es computacionalmente exigente.
- Acceder a un sistema de procesado remoto a través de un front-end llamado RISA (Remote Interface for Science Analysis).

Una de las alternativas para ejecutar el código de análisis de los recursos locales ha sido el uso de una infraestructura Grid, que proporciona numerosos recursos computacionales distribuidos. La herramienta SAS se encontraba dentro de la organización virtual de Grid "Planck" VO [TAF05].

Para utilizar herramientas de SAS en un entorno distribuido Grid se debe aceptar mecanismos de autorización Grid y se necesitaba adaptar a la infraestructura realizando tareas como:

• La instalación de la aplicación de SAS y los elementos auxiliares para su ejecución dentro del entorno Grid.

• La instalación y automatización de la base de datos de calibración en un SE.

• La identificación de nodos de trabajo Grid que posean la aplicación SAS y estén preparados para la ejecución de la tarea, usando para ello el concepto de *TAG*

• El apoyo a los usuarios para el uso de esta infraestructura, elaborando manuales, impartiendo sesiones de formación, y desarrollando herramientas útiles para el uso de Grid, como, por ejemplo, las herramientas descritas en el Capítulo 3.

En cuanto al almacenamiento, una herramienta de SAS necesita un sistema que facilite el envío de trabajos y que conecte directamente con el archivo XMM para recuperar esos datos para procesarse.

La alternativa actual al procesado local y la utilización de Grid es un procesado remoto mediante una interfaz web (RISA), que se realiza en un cluster de ESAC, sede del Science Operations Centre de XMM-Newton. Además se ha planteado usar servicios Cloud para la ejecución de esta interfaz.

I.5 ARCoS

ARCoS (Adaptive Refinement Code for Streamers) [LUQ14] es un programa para la simulación de descargas eléctricas (tipo *streamer*) diseñado en el Centro Holandés de Matemáticas y Ciencias de la Computación (CWI).

Estas descargas son relevantes tanto en contextos de laboratorio, como para aplicaciones industriales (generación de ozono o tratamiento de superficies) [LUQ07] y en contextos geofísicos, pues son los componentes de las descargas eléctricas en la atmósfera superior (sprites).

En el caso de la geofísica, el objetivo es la simulación de descargas eléctricas en la alta atmósfera (sprites) y, en particular, el análisis de las propiedades y el modelado de procesos químicos en los *streamers* que aparecen naturalmente en las capas altas de la atmósfera terrestre. Numéricamente el mayor desafío en el modelado de *streamers* es la gran disparidad entre las escalas de longitud implicadas en ellos. Para afrontar este problema, se ha empleado un método de refinado adaptativo de mallas que ya ha dado resultados científicamente relevantes en los últimos años [MON06].

En este caso de uso se simula una descarga eléctrica en la mesosfera de tipo *sprite* [PASS14]. Los parámetros que deben establecerse corresponde a las características de las descargas, y el código proporciona ficheros de datos que describen la evolución de la carga eléctrica, la densidad electrónica etc, en el domino computacional (una sección de la mesosfera y parte de la ionosfera).

Por otro lado, se estudia el efecto de ciertos parámetros externos (por ejemplo, el campo eléctrico generado por la descarga de una tormenta) en las propiedades de un streamer (velocidad, radio, etc). Para ello, se resuelve un conjunto de ecuaciones diferenciales en derivadas parciales mediante refinado adaptativo de mallas, discretización no-lineal con un limitador de flujo, y discretización temporal de Runge-Kutta de segundo orden [MON06].

Un uso trivial de esta código es lanzar múltiples instancias de la aplicación para agilizar los estudios paramétricos. Así se pueden estudiar nuevos fenómenos físicos que resultan costosos de analizar de otra forma. Por ejemplo la bifurcación de *sprites*, que requiere un código tridimensional con gran resolución en todas las coordenadas.

El tiempo de ejecución típico para estos experimentos depende de la naturaleza del trabajo ejecutado y oscila entre 12 horas y una semana [MON06, LUQ09], utilizando la plataforma descrita en la Sección 2.2.2, y la frecuencia de envío de datos es del orden de 10 a 50 trabajos completos por mes.

En cuanto al almacenamiento, teniendo en cuenta que cada ejecución genera de 200 MBytes a 1 GBytes de datos, todas las ejecuciones pueden llegar a originar unos 500 Gbytes, obligando al uso intensivo del elemento de almacenamiento de la plataforma.

I.6 Duchamp

Duchamp [MAH12] es un programa para realizar búsquedas en los datos de radioastronomía. Si se observa una línea espectral particular aparecerán objetos discretos en tres dimensiones: dos de ellas representan la posición en el cielo, y el tercero corresponde a la frecuencia, la longitud de onda, o la velocidad.

En esta aplicación nos centramos en el estudio de diseño de GASKAP (Espectroscopia Galáctica con ASKAP) [DIC13] como parte del grupo de trabajo para la búsqueda de fuentes en imágenes espectroscópicas en radio. GASKAP se centra en el estudio de emisión de HI y OH de nuestra galaxia, utilizando el instrumento ASKAP (Australian SKA Pathfinder), uno de los precursores reconocidos para el gran proyecto internacional Square Kilometer Array (SKA).

En GASKAP, existen del orden de 10.000 fuentes individuales que ocupan un total de 100 TBytes. Evidentemente, es necesario utilizar métodos automáticos de detección de fuentes. En este momento, como parte del estudio de diseño, se intenta evaluar la idoneidad de varios algoritmos de búsqueda de fuentes en simulaciones de datos de ASKAP, con tamaños alrededor de 2-3 GBytes. Para ello se utiliza primeramente el algoritmo Duchamp, y posteriormente se realiza con 4 algoritmos diferentes (clumpfind, gaussclumps, fellwalker, reinhold) implementados en la aplicación CUPID.

Como parte del estudio de diseño de GASKAP se utilizan las imágenes espectrales obtenidas para búsqueda de fuentes en imágenes tridimensionales (plano del cielo multiplicado por velocidad).

Los resultados se utilizan en el informe de desarrollo de ASKAP. Posteriormente, se continua con los primeros datos obtenidos por las primeras antenas de ASKAP.

I.7 Gadget-2

El código de simulación **Gadget-2** (GAlaxies with Dark matter and Gas intEracT) [SPR01, SPR05] se usa para realizar simulaciones de N-cuerpos SPH (Smoothed Particle Hydrodynamics) en ordenadores masivamente paralelizados con memoria distribuida.

El código aplica la dinámica newtoniana para determinar integraciones cosmológicas en cosmologías arbitrarias, con o sin condiciones de contorno periódicas. El modelado hidrodinámico es opcional. El código es adaptativo en el espacio y en el tiempo, y su función de Lagrange hace que sea especialmente adecuado para las simulaciones de las formaciones de estructuras cósmicas.

En el caso analizado se abordan dos problemas fundamentales. En primer lugar, se estudia la formación de grupos de galaxias, desde z = 3 (*redshift*) hasta la época actual, usando simulaciones cosmológicas, de muy alta resolución y con toda la física relevante incluida (formación estelar, física de la componente gaseosa de las galaxias, realimentación a partir de la explosión de supernovas, etc) para tratar de explicar los recientes resultados observacionales de galaxias a alto *redshift*. Asimismo, también se estudian las cascadas de partículas generadas por los rayos gamma provenientes de fuentes extragalácticas para la interpretación de los datos generados por telescopios como MAGIC.

Se están realizando simulaciones dinámicas consistentes. Definiéndose, como principal objetivo, el estudio de la distribución de la luz difusa intergaláctica y la evolución de la función de luminosidad en grupos de alta densidad.

Las necesidades de cómputo que requieren las simulaciones de tipo cosmológico y la realización de simulaciones de MonteCarlo de cascadas de partículas por rayos gamma provenientes de fuentes extragalácticas son elevadas. Además, las simulaciones requieren de una cantidad de recursos nada despreciable, puesto que el adecuado reprocesamiento de los datos de salida de estas simulaciones es, en ocasiones, la parte más relevante a la hora de interpretar los resultados. El tiempo de ejecución que se necesita en cada ejecución individual comprende desde horas (análisis) hasta del orden de cien días (simulaciones), para la plataforma de computación descrita en la Sección 2.2.2.

Estas simulaciones se paralelizan a través de plataformas Grid, reduciendo así el tiempo de cálculo. El tiempo de ejecución típico para estos experimentos es de 12 horas, en la plataforma descrita en la Sección 2.2.2, con una frecuencia de envío de datos del orden de 64 a 128 trabajos completos por cada estudio, haciendo una media de 10 estudios por mes.

En cuanto al almacenamiento, teniendo en cuenta que cada trabajo genera un orden de un TBytes de datos es obligado el uso intensivo de los recursos de almacenamiento de la plataforma.

I.8 Acredition Disk

Acredition Disk [DAL05] no es una aplicación que necesite migrarse a una plataforma Grid. Se trata de un catálogo o librería de modelos de transporte radiativo de discos circunstelares en estrellas jóvenes de gran masa. Los discos son estructuras con forma de anillo o toro situada alrededor de una estrella y que están construidos por gas, polvo, y objetos rocosos o de hielo.

Esta base de datos se actualiza varias veces al año, facilitándose así la tarea de análisis de los datos recogidos por el Observatorio Espacial de Herschel o el Atacama Large Millimeter Array (ALMA).

El objetivo en este caso es la utilización de los elementos de almacenamiento de una plataforma Grid para albergar una base de datos de distribuciones de energía espectral sintéticas, e imágenes de emisión de polvo térmico que surgen del material protoestelar [DAL06].

Una vez se tiene cubierto un número de modelos que abarca el rango de condiciones físicas requerido, se hace público el catálogo almacenado generando una gran herramienta para el campo científico.

Apéndice II:Librerías utilizadas por las aplicaciones.

En este apéndice se enumeran las distintas librerías de carácter general utilizadas para la adaptación de aplicaciones astronómicas a una plataforma Grid.

Librería	Descripción	Referencia
FFTW	Es una librería que se compone de una serie de subrutinas destinadas al cálculo de transformadas de Fourier discretas en una o varias dimensiones.	http://www.fftw.org
OpenMPI	Es una implementación de código abierto de los estándares de la interfaz de paso de mensajes (Message Passing Interface, MPI) utilizada para la computación paralela o distribuida.	http://www.open-mpi.org/
GSL	La GNU Scientific Library es una librería numérica para los lenguajes C y C++ que proporciona una gran variedad de rutinas numéricas.	http://www.gnu.org/software/gsl/
OpenMP	Es una interfaz de programación de aplicaciones que permite programar aplicaciones en plataformas de memoria compartida de forma paralela.	http://openmp.org/wp/
HDF5	Engloba un modelo de datos, librerías y formato de fichero para almacenar y manejar datos. Soporta una variedad ilimitada de tipos de datos, y su diseño es flexible y eficiente para un gran volumen y complejidad de datos.	http://www.hdfgroup.org/HDF5/
IFORT	Es un compilador que optimiza la ejecución de códigos Fortran 90/95 para las CPUs Intel.	http://software.intel.com/en-us/articles/intel-compilers/
PGPLOT	Es una librería gráfica para los lenguajes Fortran o C que permite realizar gráficas.	http://www.astro.caltech.edu/~tjp/pgplot/)
WCSLIB	Es una librería de C con un conjunto de capas en Fortran, que implementa el World Coordinate System (WCS) en formato Flexible Image Transport System.	http://www.atnf.csiro.au/people/mcalabre/WCS/
CFITSIO	Es una librería que contiene subrutinas en lenguaje C y Fortran para la lectura y escritura de ficheros de datos en formato FITS.	http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html

Referencias

[AAD08] Add, G et al. “The ATLAS Experiment at the CERN Large Hadron Collider”. Journal of instrumentation. Vol. 3, Article Number: S08005, 2003

[ACC14] http://www.ngi.cesga.es/gridsite/accounting/CESGA/ngi_view.php

[AIA00] AIAA group. “The X-ray multi mirror (XMM-Newton) power system”. 35th IECEC, Vols 1 and 2. pp 221-231. 2000

[AIF12] Aiftimiei, C et al. “Towards next generations of software for distributed infrastructures : the European Middleware Initiative” Proceeding IEEE International Conference on e-Science (e-Science), 2012

[ALV08] Alves, AA et al. “The LHCb Detector at the LHC” Journal of instrumentation. Vol. 3, Article Number: S08005, 2008

[AND06] Anderson, DP. “The computational and storage potential of volunteer computing” 6th CCGrid pp.: 73-80. 2006

[ARC14] http://md-wiki.project.cwi.nl/index.php/ARCoS_code

[AST14] http://en.wikipedia.org/wiki/Asteroseismology

[AUV09] Auvergne, M et al. “The CoRoT satellite in flight: description and performance”. Astronomy & Astrophysics Vol. 506 Issue: 1 pp. 411-424. 2009

[BAH03] Barham, B. et al. “Xen and the art of virtualization”. SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles. pp. 164-177. ACM. 2003

[BAR09] Barbera, R. et al. “Grid Portal and robot certificates: a new tool for e-Science”. BMC BIOINFORMATICS. Vol.10. Article Number: S21. 2009

[BAR11] Barthol, P et al. “The Sunrise Mission” Solar Physics Vol. 268 Issue. 1 pp. 1-34. 2011

[BAU08] Baumeister, H et al. "PANIC: the new panoramic NIR camera for Calar Alto", Proc. SPIE 7014, 70142R-9. 2008

[BEL07] Beltrán V. “Espectroscopia Astronómica”. Universidad Nacional de Colombia-Sede Bogotá. 2007

[BER03] Berman, F. et al “Grid computing: Making the Global Infrastructure” a Reality. John Wiley & Sons Ltd, 2003

[BOE76] The spectral classification of M-dwarf stars. Boeshaar, P.C, Ph.D. Thesis Ohio State Univ., Columbus. 1976.

[BRA15] https://developer.ibm.com/streamsdev/docs/getting-spl-application-ready-cloud/

[BRE13] Brescia, M et al. “Astroinformatics, data mining and the future of astronomical”. Nuclear instruments & methods in physics research section a-accelerators spectrometers detectors and associated equipment. Vol. 720 pp. 92-94. 2013.

[BRO13] Broeg, C. et al. “CHEOPS: A transit photometry mission for ESA's small mission programme.” EPJ Web Conf. 47, 03005. 2013.

[BUR97] Burbeck, S “How to use Model-View-Controller (MVC)”. http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

[CAM09] Campos, I et al. “Ibergrid, Grid infrastructure for the iberian research area” ibergrid: 3rd iberian Grid infrastructure conference proceedings pp. 9-18, 2009

[CAM12] Campos, I. “Preface to special section on Ibergrid”. COMPUTING AND INFORMATICS” Volume: 31 Issue: 1 Pages: 1-2 Published: 2012

[CAR00] Carpenter, G.D. Et al. “Non-uniform memory access (NUMA) data processing system that permits multiple caches to concurrently hold data in a recent state from which data can be sourced by shared intervention”. Google Patents. https://www.google.com/patents/US6115804. 2000

[CAR14] Carapito C. et Al. “MSDA, a proteomics software suite for in-depth Mass Spectrometry Data Analysis using grid computing” Proteomics. 14(9):1014-9. doi: 10.1002/pmic.201300415. 2014.

[CAS90] Casuso, E. et al. “Síntesis evolutiva de poblaciones estelares en galaxias espirales”. Boletín Astronómico del Observatorio de Madrid. 12 (3): 35-38, 4 Ref. 1990.

[CEB08] Ceballos, MT et al. “Análisis de datos de XMM con SAS sobre tecnología GRID”. Poster. VIII Reunión Científica de la SEA. 2008.

[CEB10] Ceballos, MT. “XMM-Newton Data Analysis with SAS Software Over Grid” . Highlights of spanish astrophysics VBook Series: Astrophysics and Space Science Proceedings pp. 479-479. 2010

[CHR08] Christensen-Dalsgaard, J. “ASTEC - the Aarhus STellar Evolution Code” ASTROPHYSICS AND SPACE SCIENCE Vo. 316 Issue: 1-4 pp. 13-24, 2008

[CHU03] Chun, B, “PlanetLab: An overlay testbed for broad-coverage services” ACM Sigcomm Computer Communication Review. Vol: 33 Issue: 3 pp. 3-12. 2003

[CID09] Cid, R et al. “The star formation histories of galaxies: A tour through the *Starlight*-SDSS database”. Revista mexicana de astronomía y astrofísica, Vol 35 pp. 127-132, 2009

[CON12] Conejero, J et al. “VGrid: una infraestructura Grid virtual con fines educacionales”. Universidad de Castilla-La Mancha. Conference lecture. 2012

[COM11] Comley, P et al. “Grinding metre scale mirror segments for the E-ELT ground based telescope”. CIRP annals-manufacturing technology. Vol. 60 Iss. 1 pp. 379-382. 2011.

[CRA06] Crawa, S et al “Learning adaptation knowledge to improve case-based reasoning” Volume 170, Issues 16–17,, Pages 1175–1192. 2006.

[DAB12] Dabrowska, D.D. et al. “Effect of the n orientation of the optic axis on simulated scattering matrix elements of small birefringent particles”. Optics Letters, 37:3252. 2012.

[DAB13] Dabrowska, D.D. et all. “Experimental and simulated scattering matrices of small calcite particles at 647nm” J. Quant. Spec. Radiat. Transf., 124:62–78, 2013.

[DAB14] Dabrowska, D.D., Rodón, J.R. et al. “Scattering matrices of Martian dust analogs at 488 nm 647 nm.” A&A, 2014.

[DAB14B] Dabrowska, D.D. Thesis “Experimental and theoretical study of scattering matrix of martian dust analogous”. Granada University. 2014.

[DAG12] Dagmar Adamová, Pablo Saiz “Grid Computing in High Energy Physics Experiments” INTECH Open Access Publisher, 2012

[DAL05] P. D'Alessio et Al. " WWW Database of models of accretion disks irradiated by the central star". Revista Mexicana de Astronomía y Astrofísica, Vol. 41, pp. 61-67,2005.

[DAL06] P. D'Alessio, N. Calvet, L. Hartmann, R. Franco-Hernández, H. Servín. "Effect of dust growth and settling in T Tauri disks". The Astrophysical Journal, 638:314- 335, 2006.

[DEA06] de Austri, RR et al. "A Markov chain Monte Carlo analysis of the CMSSM". JOURNAL OF HIGH ENERGY PHYSICS, Issue: 5,Art 2, 2006

[DEE04] Deelman, E et al. "Pegasus: Mapping scientific workflows onto the Grid". Grid COMPUTING Book Series: LECTURE NOTES IN COMPUTER SCIENCE. Vol. 3165, pp. 11-20, 2004

[DIC13] Dickey, John M. et al. "GASKAP-The Galactic ASKAP Survey". Astronomical Society of Australia, Vol. 30, id.e003 20 . 2013

[DIM09] Dimou, M . "LCG User Registration and VO Management ". https://edms.cern.ch/document/428034/ . 2009.

[DOM05] Dominus, M. "Higher Order Perl". Morgan Kaufmann. Functional programming techniques in Perl. 2005

[DOU86] Doug Tody "The IRAF Data Reduction and Analysis System". National Optical Astronomy Observatories. 1986

[DRA94] Draine, B.T., & Flatau, P.J., "Discrete dipole approximation for scattering calculations", J. Opt. Soc. Am. A, 11, 1491-1499, 1994

[DRA04] Draine, B.T., & Flatau, P.J. 2004, "User Guide to the Discrete Dipole Approximation Code *Ddscat* 6.1", http://arxiv.org/abs/astro-ph/0409262v2. 2004

[DDS14] http://www.astro.princeton.edu/~draine/*Ddscat*.html

[EAD71] Eadie, W.T. et al. "Statistical Methods in Experimental Physics". Amsterdam: North-Holland. pp.269–271. 1971

[ECI14] Portal de la E-ciencia española: www.e-ciencia.es/

[EGI14] EGI home. http://www.egi.eu/

[EIN14] EGI-InSPIRE home. https://www.egi.eu/about/egi-inspire/

[ELL02] Ellert, M et al. "NorduGrid project: using Globus toolkit for building Grid infrastructure". Nuclear instruments & methods in physics research section a-accelerators spectrometers detectors and associated equipment. Vol. 502 Iss.2-3 pp.407-410. 2002.

[ELL07] Ellert, M et al. "Advanced Resource Connector middleware for lightweight computationa Grids". Future generation computer systems-the international journal of Grid computing theory methods and applications. Vol. 23 Iss.2 pp.219-240. 2007

[ELS05] Elsasser, D et al. "MAGIC and the search for signatures of supersymmetric dark matter". New Astronomy Reviews Vol. 49 Issue. 2-6 pp. 297-301. 2005

[EVA08] Evans, L et al. "LHC Machine" .Journal of instrumentation. Vol. 3, Article Number: S08001, 2008

[ERW02] Erwin, DW. "UNICORE - a Grid computing environment". Concurrency and computation-practice & experience. Vol.14 Iss.13-15 pp. 1395-1410. 2002

[ESA14] http://www.esa.int/ESA

[FER05] Lucchese, F., Yasuda, T., Lee, C. Y., Queiroz, C. A., Minetto, E., & Mungioli, A. (2005). Grid Computing in Research and Education. IBM, International Technical Support Organization. 2005.

[FOR99] Forget,F et al "Improved general circulation models of the Martian atmosphere from the surface to above 80~km". J. Geophys. Res., 1999, 104, 24,155-24,176. 1999

[FOS01] Foster, I. et al "The Anatomy of the Grid: Enabling Scalable Virtual Organizations " *International J. Supercomputer Applications*, 15(3), 2001.

[FOS02] Foster, I. "The Grid: A new infrastructure for 21st century science". PHYSICS TODAY. Vol.55 Issue: 2 pp. 42-47. 2002

[FOS02b] Foster, I. et al "Grid services for distributed system integration". Computer Vol.35, Issue: 2. pp. 27-46). 2002

[GAL09] Gonzalez-Galindo, F et al. "A ground-to-exosphere Martian general circulation model: 1. Seasonal, diurnal, and solar cycle variation of thermospheric temperatures", J.Geophys. Res., 114, E04001, doi:10.1029/2008JE003246, 2009

[GAN15] http://ganglia.sourceforge.net/

[GAR09] García Hernández, A. et al. “Asteroseismic analysis of the CoRoT δ Scuti star HD 174936”. Astronomy and Astrophysics, Volume 506, Issue 1, 2009, pp.79-83. 2009

[GAR11] García Hernández, A. Thesis: “Cuasi periodicidades en los periodogramas de estrellas Delta Scuti: un nuevo observable. Dos ejemplos de CoRoT: HD 174936 y HD 174966. Granada University. 2011.

[GAR13] García Hernández, A. Rodón, JR et al. “An in-depth study of HD 174966 with CoRoT photometry and HARPS spectroscopy. Large separation as a new observable for stars”. Astronomy & Astrophysics, Vol. 559 pp. 2013

[GAR15] García Hernández, A. et al. “Observational Δν–ρ Relation for δ Sct Stars using Eclipsing Binaries and Space Photometry”. The Astrophysical Journal Letters, Volume 811, Issue 2, article id. L29, 6 pp. 2015.

[GLI14] http://www.embnet.org/sites/default/files/quickguides/gLite.pdf

[GNU15] http://www.gnu.org/licenses

[GON09] González-Galindo, F. et al. “A ground-to-exosphere Martian general circulation model: 1. Seasonal, diurnal, and solar cycle variation of thermospheric temperatures”. Journal of Geophysical Research, 2009

[GON13] González-Galindo, F. et al. ”3D Martian Ionosphere model: I. The photochemical ionosphere below 180 km”. Journal of Geophysical Research, 2013, 118, 2105-2123. 2013

[GOU85] Gough, D “Stellar Structure – Beginnings of Asterosismology “. Nature. Vol. 314 Issue: 6006 pp. 14-15, 1985

[GRE08]Gregori,M.et al “GridSEED : A Virtual Training Grid Infrastructure”. Joint Eu-IndiaGrid/CompChem Grid Turorial on Chemical an dMaterial Science Applications. pp. 39-53.2008.

[HAY08] Hayes, B. “Cloud computing” COMMUNICATIONS OF THE ACM Vol. 51 Issue: 7 pp. 9-11. 2008

[HEL14] http://helas.astro.uni.wroc.pl

[HUE05] Huedo E. et al. “The GridWay Framework for Adaptive Scheduling and Execution on Grids”. Scalable Computing: Practice and Experience, 6(3):1-8, 2005.

[IAA14] http://www.iaa.es/es/content/grid

[IBA10] Ibarra, A et al. “XMM-Newton Science Analysis Software: Further Development and Maintenance”. Astronomical Society of the Pacific Conference Series, Vol. 434 pp.293-296, 2010

[IVO14] http://www.ivoa.net

[JOH53] Johnson, H. L et al. “Fundamental stellar photometry for standards of spectral type on the revised system of the Yerkes spectral atlas”, ApJ 117, 313. 1953

[JOS08] Johnston, S et al. “Science with ASKAP”. Experimental astronomy Vol. 22 Iss. 3 pp. 151-273. 2008.

[KAI02] Kaiser, N. et al. "pan-starrs - a large synoptic survey telescope array”. Survey and other telescope technologies and discoveries. Vol. 4836 pp. 154-164. 2002.

[KAN69] Kantowsk.R “Redshift relations of homogeneous friedmann models”.Astrophysical Journal. Vol. 155 Iss.1p1 pp.89. 1969

[KAL09] Kaltenegger, Lisa; Traub, Wesley A. "Transits of Earth-like Planets". The Astrophysical Journal. Vol 698 pp. 519–527. 2009

[KER84] Kernighan, B. et al. “Using the Shell", The UNIX Programming Environment, Prentice Hall, Inc., p. 94. 1984

[KIV07] Kivity, A. “KVM: The Linux virtual machine monitor”. OLS'07: Ottawa Linux Symposium. pp-225-230. 2007

[KOR04] Kornmayer, H et al. ”A distributed, Grid-based analysis system for the MAGIC telescope” Proceading. Computing in High-Energy Physics. p.955-958. 2004

[KUL09] Kulshrestha, A et al. "Service Oriented Architecture for Job Submission and Management on Grid Computing Resources”. (HiPC 2009) PROCEEDINGS pp. 13-19. 2009

[KUL15] Kulikov, I. et al "Complex simulation of the dynamics of astrophysical objects using hybrid supercomputers” COMPUTER PHYSICS COMMUNICATIONS Volume: 186 Pages: 71-80. 2015.

[KUN05] Kunszt, P et al. “Data storage, access and catalogs in gLite” Book Group: IEEE Computer Society. Local to Global Data Interoperability. pp. 166-170. 2005

[KUR11] Donny Kurniawan et al. “ISENGARD: an infrastructure for supporting e-science and Grid application development”.Concurrency and Computation: Practice and Experience Vol. 23, Iss. 4, pp. 390–414,2011.

[LEB85] Leblanc, TJ et al. “HPC: A model of structure and change in distributed systems”. IEEE Transactions On Computers vol. 34 Iss. 12 pp. 1114-1129. 1985.

[LIN08] Lin, G et al. “X509 digital certificates quick analyzing and validating method for safety affairs, involves establishing index table based on positions of all analyzed key fields, and memorizing location information of all key fields in index table.” Patent Number: CN101257387-A; CN101257387-B. 2008 Patent. 2008

[LUQ07] Luque, A et al. “Photoionisation in negative streamers: fast computations and two propagation modes”. APPLIED PHYSICS LETTERS Vol. 90 Iss.8 Article Number: 081501. 2007

[LUQ09] A. Luque, U.M. Ebert. “Emergence of sprite streamers from screening-ionization waves in the lower ionosphere”. Nature Geoscience, Vol. 2, 11:757-760, 2009.

[LUQ14] http://md-wiki.project.cwi.nl/index.php/ARCoS_code

[MAN12] Mantegazza, L et al. “Pulsation spectrum of δ Scuti stars: the binary HD 50870 as seen with CoRoT and HARPS”. Astronomy & Astrophysics, Volume 542, id.A24, 13 pp. 2012

[MAR06] Martel, C et al, “Modeling Collaborative Learning Activities on e-Learning Platforms”. “Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)” .2006.

[MAR13] Marques, J. “Stellar rotation and the CESTAM code”. SAC seminar. 2013

[MAS09] Massie, M et al. “Monitoring with Ganglia” Publisher: O'Reilly Media pp. 256. 2012.

[MAL08] Malawski, M et al. “High-level scripting approach for building component-based applications on the Grid”. Conference: Joint Workshop on Making Grids Works. Pp: 309-321. 2008

[MAH12] Matthew T. Whiting. “Duchamp: a 3D source finder for spectral-line data”; MNRAS 421, 3242. 2012

[MAT14] http://es.mathworks.com/products/matlab/

[MET12] Metcalfe, TS et al. “First Results from the Asteroseismic Modeling Portal” Astronomical Society of the Pacific Conference Series Vol 462, pp. 213-214, 2012

[MER13] Merikallio, S. et al. “Light scattering by the Martian dust analog, palagonite, modeled with ellipsoids” Optics Express, 21:17972–17985, 2013.

[MES14] http://mesastar.org/ http://mesa.sourceforge.net/

[MPI14] http://www.open-mpi.org/

[MON06] C. Montijn, W. Hundsdorfer, U. Ebert. “An adaptive Grid refinement strategy for the simulation of negative streamers”, Journal of Computational Physics, 219:801-835, 2006.

[MOR70] Morozhenko, AV. “Atmosphere of mars according to polarization observations”. Soviet astronomy aj USSR. Vol.13 issue: 5 pp.852 1970.

[MOR97] Morel, P. “CESAM: A code for stellar evolution calculations”. ASTRONOMY & ASTROPHYSICS SUPPLEMENT SERIES Vol: 124 Issue: 3 pp. 597-614. 1997.

[MOR08] P. Morel, Y. Lebreton. “CESAM: a free code for stellar evolution calculations”. Astrophysics and Space Science, 316:61-73, 2008.

[MOS09] Moscicki, J. et al. “GANGA: A tool for computational-task management and easy access to Grid resources”. COMPUTER PHYSICS COMMUNICATIONS Vol. 180, Issue 11, pp 2303-2316, 2009

[MOY04] A. Moya et al. “Non-adiabatic theoretical observables in a δ Scuti stars”. Astron. Astrophys, 414:1081-1090, 2004.

[MOY08] Moya, A. et al. “Granada oscillation code (*GraCo*)”. ASTROPHYSICS AND SPACE SCIENCE Vol. 316, Issue 1-4, pp. 129-133. 2008

[MRN07] Moreno, F. et al. “Comet dust as a size distribution of irregularly shaped, compact particles”, JQSRT 106, 1-3, 348, 2007.

[MUÑ10] Muñoz, O. et at. “Experimental determination of scattering matrices of dust particles at visible wavelengths: The IAA light scattering apparatus.” Journal of Quantitative Spectroscopy and Radiative Transfer, vol. 111, issue 1, pp. 187-196, 2010.

[MUL05] Al Muller, A. et al. “ Virtualization with VMware ESX Server”. ISBN: 978-1-59749-019-1. 2005

[NAG14] http://www.nagios.org/

[NAS09] Kepler: NASA's First Mission Capable of Finding Earth-Size Planets. NASA Staff. 2009

[NAS14] http://www.nasa.gov/

[NUN08] Nunez, M et al. “Wind Evaluation Breadboard electronics and software”. Advanced software and control for astronomy II. Vol. 7019 Part: 1-2 Article Number: 70190K. 2008

[OMP14] www.openmp.org/

[OSN14] http://www.osn.iaa.es/

[PAP13] Paparó, M et al. “CoRoT 102749568: mode identification in a δ Scuti star based on regular spacings”. Astronomy & Astrophysics, Volume 557, id.A27, 13 pp. 2013

[PAS14] Pascual-Granado. J. et al. “MIARMA: An information preserving method for filling gaps in time series. Application to CoRoT light curves”. submitted to A&A. 2014

[PASS14] M. Passas, J. Sánchez, A. Luque and F. J. Gordillo-Vázquez "Transient Upper Atmospheric Plasmas: Sprites and Halos" IEEE Transactions on Plasma Science, vol 42, doi: 10.1109/TPS.2014.2329320. 2014

[PAX11] Paxton, B et al. “Modules for Experiments in Stellar Astrophysics (MESA)” The Astrophysical Journal Supplement, Vol. 192, Issue 1, article id. 3, pp 35, 2011

[PER13] Perez, E. et al. “The Evolution of Galaxies Resolved in Space and Time: A View of Inside-out Growth from the CALIFA Survey”. Astrophysical Journal Letters. Vol: 764 Iss.1 Article Number: L1 2013.

[PET99] Petrova. EV. “Mars aerosol optical thickness retrieved from measurements of the polarization inversion angle and the shape of dust particles” J. Quant. Spec. Radiat. Transf., 63:667–676, December 1999.

[PIG04] Pilgrim, Mark. “Dive Into Python”. Apress. 2004

[PIL10] Pilbratt, GL et al. “Herschel Space Observatory. An ESA facility for far-infrared and submillimetre astronomy” Astronomy $ Asthophysics Vol. 518. 2010.

[PKI14] http://pki.irisgrid.es/doc/serverCertReq.es.php

[PLA14] http://sci.esa.int/plato/

[PQT14] https://wiki.python.org/moin/PyQt

[PYT14] https://www.python.org/

[QUI12] A. Quirrenbacha. “CARMENES. I: Instrument and Survey Overview”. 2012

[RED14] http://www.rediris.es/

[RIC09] Ricker, G. R. et al. “The Transiting Exoplanet Survey Satellite (TESS)”. Bull. Am. Astron. Soc.41, 193. 2009.

[RIT93] Ritchie, Dennis M. "The Development of the C Language". The second ACM SIGPLAN History of Programming Languages Conference (HOPL-II) (ACM): 201–208. 1993

[ROB06] Robitaille, TP et al. “Interpreting spectral energy distributions from young stellar objects”. Astrophysical Journal Supplement Series. Vol. 167 Iss. 2 pp. 256-285. 2006.

[ROD10a] Rodón, JR et al. “IAA Grid Team: porting Astrophysical applications to Grid”. Poster. 5º EGEE User Forum. 2010

[ROD10b] Rodon, JR et al. “IAA Grid Team: porting Astrophysical applications to Grid”. Poster. Ibergrid 2010.

[ROD11a] Rodon, JR et al. “User Grid made easy: Creating generic script for specific problem” Poster .1st EGI User Forum. 2011

[ROD11b] Rodon, JR et al. “GSG Tool: Creating generic scripts for problems on Astrophysics” IBERGrid 2011 proceedings, ISBN: 978-84-9745-884-9 pp. 333. 2011

[ROD12] Rodon, JR et al. “An example of GSG tools integration: *Starlight* Spectral Synthesis Code.” EGI Technical Forum. 2012.

[ROD15] Rodon, JR et al. “Soluciones computacionales para la simulación de sistemas planetarios” . CPESS4. 2015

[ROR05] Rodríguez,L. ”Un universo en expansión” 4ª edición. 2005.

[ROT03] Rottgering, H. “LOFAR, a new low frequency radio telescope”. New astronomy reviews. Vol. 47 Iss. 4-5 pp. 405-409. 2003.

[RUB14] https://www.ruby-lang.org/

[RUI92] B. Ruiz Cobo, J.C. Del Toro Iniesta. “Inversion of Stokes Profiles”. The Astrophysical Journal, 398:375-385, 1992.

[RUZ06] Ruiz de Austri, R *et al. “*A Markov chain Monte Carlo analysis of the CMSSM”. JHEP05. 2006.

[RYB12] Rybkin, G. “ATLAS software packaging”. CHEP2012, PTS 1-6 Book Series: Journal of Physics Conference Series Vol. 396 Article Number: 052063. 2012

[SAB10] J. Sabater. Proyecto: “SLGrid: *Starlight* Spectral Synthesis Code”. Instituciones: IAA-CSIC; CETA-CIEMAT, 2009-2010.

[SAK10] Sakellaruou, S et al “Mapping Workflows on Grid Resources: Experiments with the Montage Workflow”. Grids, P2P and Services Computing. Pp 119-132. 2010

[SAR99] Sarmenta, LFG et al. “Volunteer computingsystems using Java”. Future generation computer systems-the international journal of Grid computing-theory methods and applications. Vol.15 iss.5-6 pp.675-686.1999.

[SCH12] Schaaff, A et al. “Scientific Workflows in Astronomy”. Astronomical data analysis software and systems XXI series. Astronomical society of the pacific conference series. Vol. 461 pp. 875-878. 2012.

[SEA14] http://www.sea-astronomia.es/drupal/node/138

[SEV98] Severance, Kevin Dowd Charles, and Kevin Dowd. "High performance computing." 1998.

[SPA14] http://www.spaceinn.eu/

[SPR01] Springel V. et al. "GADGET: A code for collisionless and gasdynamical cosmological simulations", New Astronomy, 6, 51. 2001

[SPR05] Springel, V. “The cosmological simulation code GADGET-2”. Monthly notices of the royal astronomical society. Vol. 364. Iss. 4 pp.1105-1134. 2005

[STA14] www.*Starlight*.ufsc.br

[STP06] Staples, G “TORQUE resource manager”. SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing. 2006

[SUA08] Suárez, JC et al “FILOU oscillation code”. ASTROPHYSICS AND SPACE SCIENCE. Vol: 316 Issue: 1-4 Pp: 155-161. 2008.

[SUA14] Suárez, JC et al “Measuring mean densities of δ Scuti stars with asteroseismologyTheoretical properties of large separations using TOUCAN” Astronomy & Astrophysics, 2014

[SUA14b] Suárez, JC et al. “Proof of concept: A VO-Grid/Cloud architecture for massive, multi-parametric analysis of space asteroseismic data ”. Proyecto Explora Técnico. 2014

[SUL08] Sulistio, A. et al. “A toolkit for modeling and simulating data Grids: An extension to GridSim.” Concurr. Comput. Pract. Exp. , 20, 1591–1609. 2008

[SUP14] http://www.ft.uam.es/personal/rruiz/superbayes/index.php?page=main.html

[TAF05] Taffoni, G et al. ”Prototypes of a Computational Grid for the Planck Satellite ”. ASP Conference Series, Vol. 347, 2005

[TAF12] Taffoni, G et al. “Grid Infrastructure of the A&A VRC”. ASP Conference Series, Vol. 461, pp. 109-112, 2012

[TEA14] http://www.esf.org/index.php?id=9281

[TSA10] Tsaregorodtsev, A et al. “DIRAC3-the new generation of the LHCb Grid software”. 17th CHEP. Journal of Physics Conference, Vol. 219 Article: 062029. 2010.

[TYS02] Tyson, JA et al. “large synoptic survey telescope: overview” Survey and other telescope technologies and discoveries Series: Proceedings of SPIE. Vol. 4836 pp. 10-20. 2002.

[UNN89] Unno, W., et al. “Nonradial oscillation of stars”. University of Tokyo Press, 1989.

[VAN61] Vanderborght, R “The K-Correction and Stebbins-Whitford Correction In Relativistic Cosmology”. Monthly Notices Of The Royal Astronomical Society. Vol.122 Iss.2 pp. 177-180. 1961

[VIR14] ORACLE. VirtualBox. https://www.virtualbox.org/

[VIE08] Vuerli, C. et al. “Astrophysics in the EGEE Grid”. Astronomical Data Analysis Software and Systems XVII. ASP Conference Series. Vol. 394. 2008.

[WOL06] M. J. Wolff, MJ. et al. “Constraints on dust aerosols from the Mars Exploration Rovers using MGS overflights and Mini- TES”. Journal of Geophysical Research (Planets), 111:12. 2006.

[WOO78] Woodward R. Theoretical Models of Star Formation Annual Review of Astronomy and Astrophysics. Vol. 16: 555-584. DOI:10.1146/annurev.aa.16.090178.003011. 1978.

[WOO03] Wootten, A “The Atacama Large Millimeter Array (ALMA)” Large Ground-Based Telescopes, Pts 1 and 2. Series: Proceedings of the society of photo-optical instrumentation engineers. Vol. 4837 pp. 110-118. 2003.

Glosario de Términos.

AA: ATILA API.

AMC: Asteroseismic Modeling Portal. (Portal de modelado astrosismológico)

APEL: EGEE/LCG Accouning Applications. (Aplicación de gestión de cuentas EGEE/LCG)

API: Application Programming Interface. (Interfaz para la programación de aplicaciones)

ARC: Advanced Resource Connector. (Conector de recursos avanzado)

ATILA: Asteroseismology Computing Package.

BDII: Berkeley Database Information Index.

CA: Autoridad de Certificación.

CALIFA: Calar Alto Legacy Integral Field Area

CARMENES: Búsqueda de exo-Tierras alrededor de estrellas M con espectrógrafos echelle de alta resolución en el infrarrojo cercano y en el óptico desde Calar Alto.

CE: Computing Element. (Elemento de cómputo)

CESAM: Code d'Evolution Stellaire Adaptatif et Modulaire

CERN: European Organization for Nuclear Research.

CLI: Comand Line Inferfaz.

CoRoT: COnvection ROtation et Transits planétaires

CSIC: Consejo Superior de Investigaciones Científicas.

*Ddscat***:** Discrete Dipole Scattering

EGI: European Grid Infrastructure.

EGI-InSPIRE: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe

EMI: European Middleware Infrastructure.

FILOU: Oscillation Code

GraCo: Granada Oscillation Code.

Glite: Codeman of the middleware software suite developed by JRA.

GSG: General Scripts for Grid.

GUID: Grid Unique Identifier.

HCP: High Computing Performance.

IAA: Instituto de Astrofísica de Andalucía.

IS: Information System.

IBERGRID: Iberian Grid Infrastructure.

JDL: Job Description Language.

LCG: LHC Computing Grid.

LFC: Logical File Catalog.

LFN: Logical File Name.

LHC: Large Hadron Collider.

LRMS: Local Resources Management System.

MAG: Módulo de Acoplamiento GSG.

MCG: Módulo de Configuración GSG.

MENG: Módulo de Ejecución en Nodos Grid.

MESA: Stellar Evolution Code.

MS: Monitorizatioin System.

MVC: Modelo Vista Controlador.

NGI: National Grid Infrastructure.

PLATO: Planetary Transits and Oscillations of stars

RA: Register Authority.

SAA: Servidor de APIs de ATILA.

SARP: Sistema de Administración Regular de Procesos.

SE: Storage Element.

SPM: Spectra Periodicities Model.

SSP: Single Stellar Population.

SRE: Sistema de Registros de Eventos.

SURL: Storage URL.

Torque: Open source resource manager.

TURL: Transport URL.

UI: User Interface.

UGR: Universidad de Granada.

UNICORE: UNiform Interface to COmputing REsources

VO: Virtual Organization.

VOb: Virtual Observatory.

VOMS: Virtual Organization Membership Service.

WMS: Workload Management System.

WN: Worked Node.