Tesis doctoral

Simulación con modelos climáticos en infraestructuras de computación distribuida

Valvanuz Fernández Quiruelas

UC
UNIVERSIDAD DE CANTABRIA

Tesis Doctoral

Simulación con modelos climáticos en infraestructuras de computación distribuida

Doctorado en Matemáticas y Computación

Departamento de Matemática Aplicada
y Ciencias de la Computación

Presentada por
Valvanuz Fernández Quiruelas

bajo la dirección de
Dr. Antonio S. Cofiño González y Dr. Jesús Fernández Fernández

Universidad de Cantabria.

Santander, 12 de mayo de 2017

Foto de portada: A Sky View of Earth From Suomi NPP (Fuente: NASA Image and Video Library). Contraportada: The Olympus Supercomputer (Fuente: Pacific Northwest National Laboratory).

A Lara.

Agradecimientos

El trabajo presentado en esta tesis comenzó hace algunos años en el Grupo de Meteorología de Santander (http://www.meteo.unican.es). Durante todo este tiempo han sido muchas las personas e instituciones que, de una forma u otra, han aportado recursos y conocimiento en el desarrollo de este trabajo.

Agradezco el tiempo invertido y la dedicación de mis directores de tesis, Dr. Jesús Fernández y Dr. Antonio S. Cofiño. Con ellos he aprendido otra forma de trabajar más sistemática y organizada que, además de ayudarme durante el desarrollo de la tesis, me ha sido de gran utilidad personal y profesionalmente.

Asimismo, este trabajo no hubiese salido adelante sin la ayuda de otros miembros del Grupo de Meteorología de Santander. Imprescindible agradecer a José Carlos Blanco su ayuda y disposición a lo largo de todo el tiempo que hemos trabajado juntos. A Dr. José Manuel Gutiérrez por creer desde el principio en el proyecto y haber facilitado mi participación en él. A Dra. Mª Dolores Frías, que, a pesar de no haber trabajado directamente conmigo, siempre ha estado dispuesta a ayudarme en mi vida personal y profesional. A Markel, Chus, Lluís y Maru, por todo el esfuerzo invertido depurando WRF4G, por su paciencia y comprensión.

También me gustaría agradecer a todas aquellas personas e instituciones que han permitido mi colaboración con el Grupo de Meteorología de Santander. En especial a Alfonso Iglesias, director del Servicio de Informática de la Universidad de Cantabria, a la comisión europea por financiar los proyectos europeos EELA (GA 026409) y EELA-2 (GA 223797) y al ministerio de economía, industria y competitividad por financiar el proyecto de plan nacional WRF4G (CGL2011-28864, cofinanciado MINECO/FEDER).

Mi agradecimiento a Gema Martínez y Gonzalo Valencia por su ayuda en el diseño de la cubierta de la tesis y a Raquel Torralba por sus correcciones. Al equipo de desarrolladores de GridWay por su excelente soporte y, en general, a todos los

desarrolladores de herramientas de código abierto utilizadas en esta tesis: CAM, WRF, Python, LaTeX, Linux, netCDF y todas aquellas que nos permiten seguir trabajando y producir conocimiento.

Por último, dedicar este trabajo a mi familia y amigos que siempre me han apoyado. Sin sus ánimos y cariño, esta tesis nunca hubiese llegado a su fin.

Valvanuz Fernández Quiruelas
Santander, 12 de mayo de 2017

Índice general

Lista de acrónimos

A continuación, se muestra la lista de acrónimos utilizada en esta tesis. En la mayoría de los casos el significado del acrónimo, además de aparecer en esta lista, se muestra en el texto de la tesis. Para aquellos acrónimos cuyo uso es más común en su forma abreviada (AMD, CERN, FTP...), se ha omitido el significado en el texto.

AEMET Agencia Estatal de Meteorología
http://www.aemet.es

AMD Advanced Micro Devices

AMGA ARDA Metadata Grid Application
http://amga.web.cern.ch/amga/

API Application Programming Interface

CAM Community Atmosphere Model
http://www.cesm.ucar.edu/models/atm-cam/

CAM4G CAM for Grid

CCLM COSMO Climate Limited area Modelling
http://www.clm-community.eu

CCSM Community Climate System Model
http://www.cesm.ucar.edu/models/ccsm4.0/

CE Compute Element

CERN Conseil Européen pour la Recherche Nucléaire
https://home.cern

CESM The Community Earth System Model
http://www.cesm.ucar.edu/models/

CORDEX Coordinated Regional Climate Downscaling Experiment
http://www.cordex.org

CORWES CORDEX-WRF-ESpaña
http://www.meteo.unican.es/es/projects/corwes/

CPD Centro de Proceso de Datos

DEGREE Dissemination and Exploitation of GRids for Earth sciencE
http://cordis.europa.eu/project/rcn/79474_en.html

DIRAC Distributed Infrastructure with Remote Agent Control
http://diracgrid.org

DMS Data Management Service

DRM4G Distributed Resource Manager for Grid
https://github.com/SantanderMetGroup/DRM4G

DRMAA Distributed Resource Management Application API

EC2 Amazon Elastic Compute Cloud

ECMWF European Centre for Medium-Range Weather Forecasts
http://www.ecmwf.int

EELA E-Infrastructure shared between Europe and Latin America
http://www.eu-eela.eu

EGEE Enabling Grids for E-sciencE
https://eu-egee-org.web.cern.ch

EGI European Grid Initiative
https://www.egi.eu

EMS Experiment Management Service

ESGF Earth System Grid Federation
https://esgf.llnl.gov

ESR Earth Science Research

FMS Flexible Modeling System
https://www.gfdl.noaa.gov/fms/

FOAM Fast Ocean Atmosphere Model
http://www.mcs.anl.gov/research/projects/foam/

FPGA Field Programmable Gate Array

FRE FMS Runtime Environment

FTP File Transfer Protocol

GCM Global Circulation Model

GISELA Grid Initiatives for e-Science virtual communities in Europe and Latin America
http://www.gisela-grid.eu

GLUE Grid Laboratory Uniform Environment

GPU Graphics Processing Unit

GRIB GRIdded Binary
https://rda.ucar.edu/docs/formats/grib/gribdoc/

GSI Grid Security Infrastructure

gUSE Grid and Cloud User Support Environment
http://guse.hu/about/architecture/ws-pgrade

HDCI Hybrid Distributed Computing Infrastructure

HIRLAM HIgh Resolution Limited Area Model
http://hirlam.org

HPC High Performance Computing

HTC High Throughput Computing

IC3 Institut Català de Ciències del Clima
http://www.ic3.cat

IPCC Intergovernmental Panel on Climate Change
http://www.ipcc.ch

JMS Job Management Service

JSDL Job Submission Description Language

LDAP Lightweight Directory Access Protocol

LFC Logical File Catalog
http://lcgdm.web.cern.ch/lfc

LHC Large Hadron Collider
https://home.cern/topics/large-hadron-collider

LRMS Local Resource Management System

MPI Message Passing Interface

NCAR National Center for Atmospheric Research
https://ncar.ucar.edu

NCEP National Centers for Environmental Prediction
http://www.ncep.noaa.gov

netCDF network Common Data Form
https://www.unidata.ucar.edu/netcdf/

NGI National Grid Initiative

NOAA National Oceanic and Atmospheric Administration
http://www.noaa.gov

OAuth Open Authorization
https://oauth.net

OGF Open Grid Forum
https://www.ogf.org

OpenMP Open Multi-Processing

OSG Open Science Grid
https://www.opensciencegrid.org

PBS Portable Batch System

PKI Public Key Infrastructure

PRACE Partnership for Advanced Computing in Europe
http://www.prace-ri.eu

RDMA Remote Direct Memory Access

RegCM Regional Climate Model
http://gforge.ictp.it/gf/project/regcm/

SAGA Simple API for Grid Applications

SAML Security Assertion Markup Language

SCI-BUS SCIentific gateway Based User Support
http://www.sci-bus.eu

SE Storage Element

SGE Sun Grid Engine

SLA Service Level Agreement

SLURM Simple Linux Utility for Resource Management

SMS Supervisor Monitor Scheduler
http://www.ecmwf.int/en/elibrary/9995-sms-supervisor-monitor-scheduler

SSD Solid-State Drive

SSH Secure Shell

SST Sea Surface Temperature

UI User Interface

UMD Unified Middleware Distribution
http://repository.egi.eu/category/umd_releases/distribution/umd-3/

UNICORE Uniform Interface to Computing Resources
https://www.unicore.eu

VDS Virtual Data System

VO Virtual Organization

VOMS Virtual Organization Membership Service

WAM WAve Model
https://www.bodc.ac.uk/data/documents/nodb/254628/

WMS Workload Manager System

WN Worker Node

WPS WRF Preprocessing System

WRF Weather Research and Forecasting
http://www.wrf-model.org

WRF4G WRF for Grid
https://meteo.unican.es/trac/wiki/WRF4G

WS-PGRADE Web Service - Parallel Grid Run-time and Application Development Environment
http://guse.hu/about/architecture/ws-pgrade

XSEDE Extreme Science and Engineering Discovery Environment
https://www.xsede.org

Acerca de este documento

Esta tesis doctoral se presenta como compendio de publicaciones editadas, de acuerdo con el epígrafe 4.4 de la Normativa de desarrollo de los artículos 11, 12, 13 y 14 del Real Decreto 56/2005, del 21 de Enero, por el que se regulan los estudios universitarios oficiales de postgrado de la Universidad de Cantabria (aprobado en Consejo de Gobierno con fecha 16 de febrero del 2007). Las publicaciones que se aportan como parte de la tesis doctoral son los siguientes artículos (publicados mediante revisión por pares):

- Fernández-Quiruelas, V.; Fernández, J.; Baeza, C.; Cofiño, A.S.; Gutiérrez, J.M. Execution management in the GRID for sensitivity studies of global climate simulations. *Earth Science Informatics*. 2, pp. 75 - 82. 2009. doi:10.1007/s12145-008-0018-z

- Fernández-Quiruelas, V.; Fernández, J.; Cofiño, A.S.; Fita, L.; Gutiérrez, J.M. Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model. *Environmental Modelling & Software*. 26 - 9, pp. 1057 - 1069. 2011. doi:10.1016/j.envsoft.2011.03.006

- Fernández-Quiruelas, V; Blanco, C.; Cofiño, A.S.; Fernández, J. Large-scale climate simulations harnessing clusters, grid and cloud infrastructures. *Future Generation Computer Systems*. 51, pp.36 - 44. 2015. doi:10.1016/j.future.2015.04.009

Conforme a la normativa vigente, esta tesis doctoral presenta una introducción al trabajo de investigación, así como una revisión del estado del arte del uso de la computación distribuida en la simulación de modelos climáticos objeto de estudio.

El núcleo central de la tesis está constituido por una copia de las separatas de los artículos que comprende esta tesis.

El documento finaliza con una sección dedicada a las conclusiones sobre el trabajo realizado y al trabajo futuro, y con otra sección que contiene las referencias bibliográficas que complementan las ya incluidas en los artículos.

En cumplimiento de la normativa, se incluye un resumen en castellano de cada uno de los artículos en un apéndice.

Motivación

El trabajo de esta tesis comenzó en 2006, año en el que empecé a colaborar con el Grupo de Meteorología de Santander participando en el proyecto Europeo *E-Infrastructure shared between Europe and Latin America* (EELA; Marechal et al., 2007), perteneciente al 6º programa marco de la Unión Europea. La labor del Grupo en el proyecto fue desarrollar el prototipo de un entorno de ejecución de experimentos con modelos climáticos para el modelo global *Community Atmosphere Model (CAM)* en la infraestructura *Grid* de EELA que pudiese ser usado para analizar el fenómeno *El Niño* en América Latina. Durante este periodo, comencé la línea de investigación que he seguido en esta tesis. En primer lugar, realicé un análisis de los requerimientos de las simulaciones climáticas para su ejecución en infraestructuras de computación *Grid*. Dada la alta tasa de fallos de la infraestructura en sus inicios, mi primer objetivo fue desarrollar un entorno de trabajo robusto capaz de detectar fallos y de reanudar las simulaciones desde el último punto de control. Para ello, desarrollé un prototipo que usaba los servicios del *middleware Grid* instalados en la infraestructura de EELA (gLite; Laure et al., 2006) para gestionar y monitorizar las simulaciones. En 2009 publiqué el artículo Fernández-Quiruelas et al. (2009a), en el que describí el trabajo realizado hasta el momento y realicé un estudio sobre el fenómeno *El Niño* en América Latina usando la infraestructura de EELA, distribuida por 10 paises.

Tras EELA, continué trabajando en la misma línea de investigación en los proyectos EELA 2 (2008-2010) y *Enabling Grids for E-sciencE* (EGEE; Gagliardi et al., 2005). En esta fase, me centré en mejorar la monitorización de trabajos y la gestión de datos. Para ello, dejé de utilizar los servicios proporcionados por gLite y desarrollé servicios propios que se adaptasen a las necesidades de las simulaciones de los modelos climáticos. El nuevo prototipo, al que denominamos CAM4G, permitió realizar un experimento mucho más complejo cuyos resultados fueron mostrados en el artículo Fernández-Quiruelas et al. (2011). El objetivo de este artículo fue mostrar a

la comunidad del clima los beneficios y limitaciones de utilizar infrastructuras *Grid* para la ejecución de simulaciones climáticas.

Una de las áreas de investigación del Grupo de Meteorología de Santander es la predicción numérica con el modelo regional *Weather Research and Forecasting (WRF)*. Dentro de este área, es común la ejecución de macro-experimentos compuestos por miles de simulaciones. Para ejecutar estas simulaciones, el Grupo tiene acceso a múltiples infraestructuras de cálculo como un *cluster* propio, supercomputadores regionales y nacionales o infraestructuras *Grid* y *Cloud*.

El flujo de ejecución de WRF es más complejo que el de CAM; está compuesto por varios módulos que han de ejecutarse en serie y cuyas configuraciones tienen numerosos parámetros. Para ejecutar un experimento con WRF en cualquiera de las infraestructuras mencionadas anteriormente, es necesario desarrollar una serie de *scripts* y configuraciones *ad-hoc* que difícilmente pueden ser reutilizados para otra infraestructura. De ahí, que la mayoría de los investigadores se limiten a usar una sola infraestructura de cálculo para llevar a cabo sus experimentos.

Por ello, la continuación natural de la línea de investigación de esta tesis fue la creación de un entorno de trabajo capaz de ejecutar experimentos en diferentes recursos computacionales simultáneamente de forma transparente. WRF4G, el entorno de trabajo desarrollado para ejecutar WRF, introdujo dos mejoras fundamentales respecto a CAM4G: la capacidad de gestionar diferentes recursos computacionales de forma transparente y la gestión desatendida y monitorización de experimentos complejos. Gracias a WRF4G, los investigadores pueden acceder de forma homogénea a múltiples recursos computacionales sin necesidad de preocuparse de la infraestructura subyacente.

El resultado de esta investigación lo publiqué en el artículo Fernández-Quiruelas et al. (2015), en el que se describe WRF4G y se muestra un experimento real en el que se ejecuta una simulación a gran escala utilizando recursos computacionales heterogéneos y distribuidos. Este artículo está orientado al campo de la computación. Tras publicarlo, escribí el artículo Fernández-Quiruelas et al. (2015), orientado a la comunidad climática, en el que mostré cómo definir y ejecutar experimentos climáticos con WRF usando WRF4G. En este artículo, hice especial hincapié en describir la configuración en WRF4G de los experimentos con modelos climáticos y meteorológicos más comunes (predicción por conjuntos, predicción retrospectiva y estudios de cambio climático) y en la configuración de recursos para acceder de forma homogénea y transparente a infraestructuras distribuidas. El artículo se ha quedado en estado de discusión en su revisión abierta en *Geoscientific Model Development*.

Además de las 3 publicaciones que comprenden esta tesis, desde 2007 hasta hoy, he participado en 13 congresos y reuniones internacionales, y he presentado el trabajo objeto de esta tesis en 7 ponencias orales. También he asistido a varios cursos sobre

Grid y he partipado en la organización del *CORWES/WRF4G hands-on Tutorial*. En 2014 realicé una estancia de 15 días en el *Institute for Advanced Simulation* del *Jülich Supercomputing Center* en Alemania, en la que trabajé en la optimización de la ejecución de WRF en infraestructuras de altas prestaciones.

A día de hoy, WRF4G dispone de su propia página web[1] y repositorio de software[2] y es utilizado por varios grupos de investigación y empresas para gestionar sus experimentos con WRF, ya sea en *clusters* locales o en otro tipo de infraestructuras distribuidas.

He realizado esta tesis al mismo tiempo que mi trabajo, a tiempo completo, en el Servicio de Informática de la Universidad de Cantabria. Por eso, su desarrollo ha sido más largo, comparado con la tesis de un doctorando con dedicación a tiempo completo.

[1] https://meteo.unican.es/trac/wiki/WRF4G
[2] https://github.com/SantanderMetGroup/WRF4G

CAPÍTULO 1

Introducción

Los modelos de simulación climática son aplicaciones complejas que hasta hace unos años sólo podían ejecutarse en infraestructuras de cálculo especializadas, al alcance sólo de grandes empresas o centros de investigación (Narayan, 2009). La mejora de rendimiento de los procesadores de propósito general que tuvo lugar a finales de los 90 y comienzos del 2000, aumentó enormemente la capacidad de cálculo disponible para la comunidad investigadora y dio lugar a dos nuevos paradigmas: la computación *Grid* y *Cloud*. El bajo coste de estas nuevas infraestructuras ha producido un cambio en el mundo de la computación. A pesar de que normalmente los investigadores utilizan una única infraestructura de cálculo, la mayoría de ellos podría tener acceso a varios *clusters* e infrastructuras *Grid* y contratar tiempo en infraestructuras *Cloud* para solventar los picos de carga.

Desde el punto de vista computacional, los modelos climáticos son aplicaciones formadas por varios componentes que se ejecutan secuencialmente intercambiando información a través de ficheros de configuración y datos. La ejecución del flujo de trabajo de los modelos es una tarea complicada y en la que es común cometer errores debido a la cantidad de parámetros y ficheros de entrada que hay que configurar.

Varias técnicas de estudio del clima, como la predicción por conjuntos (Palmer, 2002) o los experimentos de análisis de sensibilidad (Murphy et al., 2004; Board, 2009), implican ejecutar repetidamente la misma simulación perturbando ciertos parámetros. En muchos casos, las restricciones de los recursos computacionales (límites en el uso de procesador, memoria o disco) pueden requerir la división de las simulaciones en tareas más pequeñas que se ejecutan secuencialmente. En estos escenarios, la preparación y ejecución de cada simulación se convierte en una actividad

compleja, sobre todo, a medida que aumentan el número de parámetros a analizar y las dependencias entre las tareas.

Para poder resolver estos problemas, varias instituciones han creado sus propios entornos de trabajo que proporcionan un conjunto de comandos y servicios que mitigan la complejidad de configurar, ejecutar y supervisar todas las tareas involucradas en un experimento. Entre otras características, proporcionan mecanismos para ejecutar el experimento de forma desatendida, reiniciar parte del experimento en caso de fallo y dividir las simulaciones en tareas más pequeños para poder adaptarlas a las restricciones de los recursos. El *FMS Runtime Environment* (FRE; Redler et al., 2012) de la NOAA, el *Supervisor Monitor Scheduler* (SMS Redler et al., 2012) del ECMWF o el Autosubmit (Manubens-Gil et al., 2016) del IC3 son algunos ejemplos. Estos entornos de trabajo están diseñados para la ejecución de un modelo concreto en la infraestructura de computación de altas prestaciones (HPC) de la institución. Dentro de la comunidad científica, no existe consenso sobre cómo denominar a este tipo de software que automatiza la gestión de experimentos, y menos aún sobre cómo traducir al castellano estas denominaciones. Algunos de los nombres más utilizados en inglés son: *runtime environment* (entorno de ejecución), *experiment workflow manager* (gestor de flujo de trabajo de experimentos), *framework* (entorno de trabajo) o simplemente *tool* (herramienta). En esta tesis vamos a denominarlos *frameworks* o, por su traducción al castellano "entornos de trabajo".

Es importante resaltar que no todo tipo de simulaciones puede ejecutarse en infraestructuras *Grid* o *Cloud*. Muchas de las infraestructuras *Grid* a las que tienen acceso los investigadores de clima: EGI, OSG o NGI (ver sección 1.2.2), debido a su heterogeneidad y a sus componentes de propósito general, se caracterizan por estar más orientadas hacia la computación de alta capacidad (HTC; Livny et al., 1997) que al HPC. Las infraestructuras HPC disponen de redes de altas prestaciones óptimas para la concurrencia y transferencia de datos entre procesos en paralelo. Por el contrario, las infraestructuras HTC se centran en la ejecución eficiente de un gran número de tareas poco acopladas. A pesar de que, hasta hace poco, las infraestructuras *Cloud* podían considerase también HTC, están apareciendo nuevos modelos de *Cloud* más orientados al HPC (Mohammadi y Bazhirov, 2017). Aún así, la mayor parte de las infraestructuras *Cloud* y *Grid* están compuestas por equipamiento de propósito general que, al no contar con redes de altas prestaciones, no están optimizadas para tareas HPC.

Aunque la necesidad principal del modelador del clima es la ejecución en paralelo de un modelo climático, técnicas científicas recientes que involucran gran cantidad de simulaciones independientes, como la predicción por conjuntos o los experimentos de análisis de sensibilidad mencionados anteriormente, corresponden a tareas HTC, apropiadas para ser desplegadas y ejecutadas en infraestructuras *Grid* y *Cloud*.

Estos problemas han recibido una creciente atención en las últimas décadas debido a sus conexiones con el estudio de incertidumbres, como las relacionadas con la predicción estacional o el cambio climático y sus impactos en diferentes sectores socioeconómicos (Parry et al., 2007).

A diferencia de disciplinas como la física de altas energías o la biomedicina, que han sabido aprovechar las infraestructuras de computación *Grid* para llevar a cabo sus experimentos, el uso del estas infraestructuras por parte de la comunidad de Ciencias de la Tierra se ha centrado principalmente en el acceso y gestión de datos. Se han dedicado esfuerzos a desarrollar servicios genéricos para la exploración y acceso transparente a datos heterogéneos, como datos de satélites, simulaciones de modelos u observaciones (ver Cossu et al., 2010, y los documentos del proyecto europeo DEGREE) . Sin embargo, apenas se han dedicado esfuerzos al despliegue y ejecución de aplicaciones como los modelos climáticos globales. Una de las hipótesis en las que se basa esta tesis, es que la comunidad de modelización del clima puede beneficiarse del uso de infraestructuras *Grid* de tipo HTC para llevar a cabo experimentos de predicción por conjuntos y análisis de sensibilidad. La heterogeneidad y distribución geográfica de estas infraestructuras, puede llegar a complicar enormemente la gestión de las simulaciones, en especial aquellas con largos tiempos de ejecución o que involucran grandes transferencias de datos. La metodología de trabajo utilizada hasta ahora por los investigadores del clima, en la que cada experimento se configura *ad-hoc* para una determinada infraestructura de cálculo, no es válida cuando se usan infraestructuras distribuidas. Por eso, la segunda hipótesis de esta tesis se basa en que para una correcta ejecución de experimentos con modelos climáticos en infraestructuras distribuidas, es necesario el uso de un *framework* que cuente con funcionalidades similares a FRE, SMS y Autosubmit y que facilite la gestión y ejecución de distintos tipos de experimentos.

Por último, la tercera hipótesis, que podría considerarse el objetivo final de esta tesis, es demostrar que es posible aprovechar simultáneamente infraestructuras HPC, *Grid* y *Cloud* para llevar a cabo nuevos retos científicos, que no serían posibles de alcanzar con una sola infraestructura computacional. Es importante destacar que el objetivo de esta tesis no es mejorar el balance de carga entre procesos MPI, hebras OpenMP y/o carga de memoria de los modelos climáticos, sino facilitar al usuario la gestión y ejecución de experimentos que consisten en un gran número de simulaciones y tareas.

Este trabajo no sólo es útil para la comunidad climática, sino que también podría ser de interés para otras disciplinas que aún no aprovechan este tipo de recursos distribuidos. Por ejemplo, otras aplicaciones que requieren largos tiempos de ejecución, grandes transferencias de datos o flujos de trabajo complejos podrían aprovecharse de los avances realizados en esta tesis.

Para poder entender la problemática que implica la ejecución de modelos climáticos en infraestructuras de cálculo distribuidas, en la sección 1.1 se hace una introducción a la modelización del clima y a las características de los experimentos con modelos climáticos. En la sección 1.2, se describen las infraestructuras de cálculo distribuidas más relevantes: *cluster*, *Grid* y *Cloud* y se define el concepto de infraestructura híbrida de computación distribuida como la agregación homogénea de recursos *cluster*, *Grid* y *Cloud* simultáneamente. En la sección 1.3 se revisa el trabajo desarrollado por otros grupos de investigación para ejecutar simulaciones de modelos climáticos en infraestructuras distribuidas y se muestran las causas por las que no reúnen los requisitos para llevar a cabo los objetivos propuestos en esta tesis. Por último, en la sección 1.4 se resumen los principales objetivos de esta tesis.

1.1. Predicción numérica de la atmósfera

Los modelos de predicción numérica de la atmósfera son representaciones matemáticas del sistema climático que describen el comportamiento y la interacción entre sus componentes y con forzamientos externos (figura 1.1). El sistema de ecuaciones que gobierna el sistema climático no puede resolverse a través de métodos analíticos, por lo que se utilizan métodos numéricos de integración (Holton, 2004), aplicados a una malla discreta en el tiempo y en el espacio tridimensional. Los valores de las variables meteorológicas en cada uno de los puntos de la malla[1] son integrados hacia adelante en el tiempo, para dar lugar a los sucesivos estados del sistema (Müller y von Storch, 2004).

La discretización espacial no es suficiente para representar los efectos de ciertos fenómenos que ocurren a pequeña escala (procesos de radiación, microfísica, turbulencia, etc). Estos procesos no resueltos explícitamente por el mallado se abordan mediante el uso de parametrizaciones físicas (Stensrud, 2009), que son distintas aproximaciones empíricas del efecto de estos procesos sobre las variables del modelo.

El punto de partida para la predicción numérica es la caracterización del estado de la atmósfera en un instante dado. El carácter no lineal y caótico de la atmósfera hace que pequeñas variaciones en las condiciones iniciales produzcan grandes cambios en las predicciones al cabo de un tiempo. Además, no es posible disponer de condiciones iniciales precisas porque la red de observación no cubre de igual manera todo el globo (se dispone de mucha menos información de los océanos o de las variables en altura que de la superficie).

Para tener en cuenta estas incertidumbres, se utiliza la técnica de predicción por conjuntos, en la que se ejecuta el modelo con pequeñas variaciones en los datos de entrada o parametrizaciones. En la sección 1.1.3 se describen este tipo de

[1] En realidad, valores medios en las celdas de la malla.

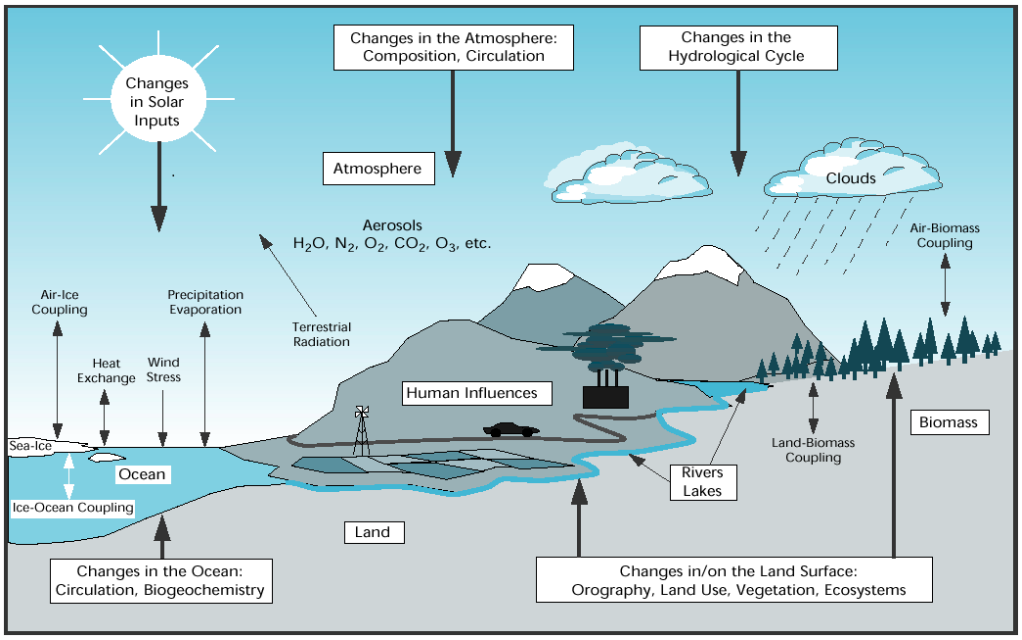


Figura 1.1: Representación esquemática de las componentes del sistema climático y sus interacciones. Fuente: *The Climate System: An Overview*. Baede et al. En: Tercer informe del IPCC (2001)

experimentos, que van a adaptarse bien a los recursos de computación distribuida.

La evolución de los modelos climáticos ha ido siempre ligada a la computación. A medida que la capacidad de procesamiento de los computadores avanzaba, se iban produciendo avances en la modelización que permitían aumentar el número de puntos de malla así como la precisión de las predicciones. En la figura 1.2 puede verse un ejemplo de la relación entre la mejora en la predicción de los modelos y los avances en la tecnología de computación a lo largo de la historia. Para evitar inestabilidades de cálculo, el paso temporal de las simulaciones tiene que estar relacionado con el tamaño de la celda. Por ello, aumentar la resolución del modelo tiene un coste computacional muy alto.

1.1.1. Modelos globales y modelos regionales

Desde el punto de vista de la región analizada, existen dos grandes tipos de modelos: globales y regionales. Los modelos globales, también conocidos como modelos de circulación general (GCM) cubren todo el globo y tienen, por lo tanto, condiciones de frontera periódicas (salvo en la superficie, donde necesitan el estado de la superficie del océano y de los continentes como condiciones de frontera). Los modelos regionales se centran sólo en una determinada parte del globo y utilizan como

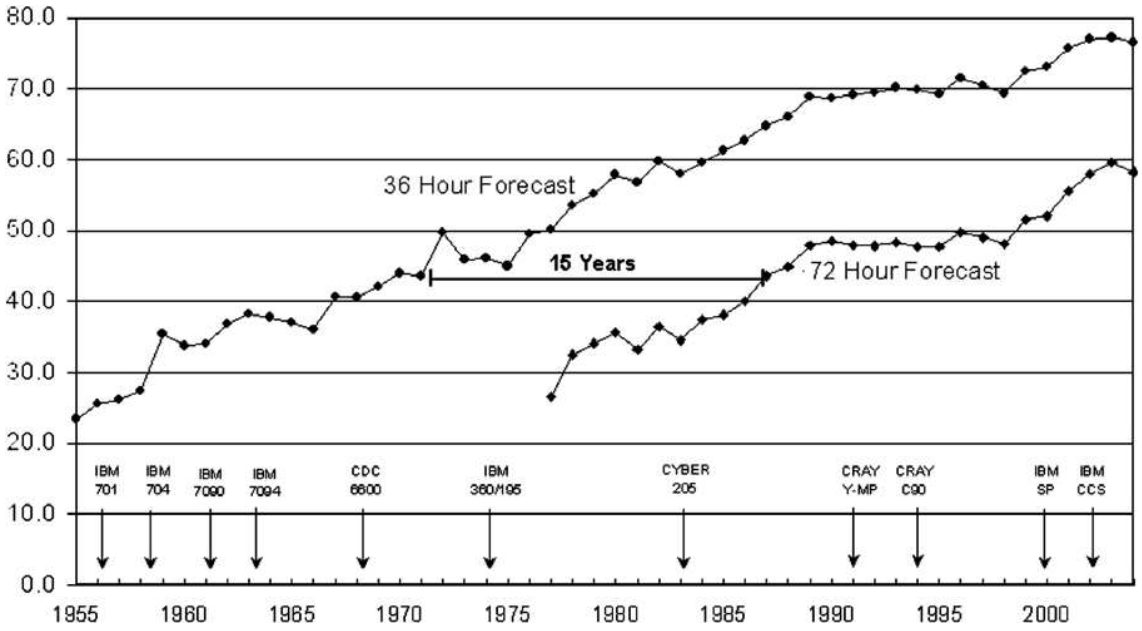


Figura 1.2: Pericia de la predicción a 36 horas (1955-2004) y 72 horas (1977-2004) producida en el NCEP. El índice de pericia de la predicción viene expresado como un porcentaje del índice (S1) de la predicción perfecta. Fuente: Lynch (2008)

condiciones de frontera la salida de los modelos globales.

En predicción meteorológica, dado el elevado coste computacional que implica ejecutar las simulaciones a escala global, éstas sólo pueden llevarse a cabo por los grandes centros de predicción numérica. Por ejemplo, el *European Centre for Medium-Range Weather Forecasts* en Europa y el *National Centers for Environmental Prediction (NCEP)* en Estados Unidos realizan predicciones globales a corto y medio plazo, que son utilizadas por los centros meteorológicos nacionales como condiciones de contorno de sus modelos de predicción, que simulan sólo su área de interés. Por ejemplo, la Agencia Estatal de Meteorología (AEMET) considera las salidas proporcionadas por el ECMWF como condiciones de contorno de su modelo regional, del consorcio HIRLAM, para obtener predicciones sobre la península Ibérica y los archipiélagos.

El modelo *Community Atmosphere Model* (CAM; Collins et al., 2004) es un modelo global desarrollado en el *National Center for Atmospheric Research* (NCAR) para las comunidades científicas de predicción del tiempo y el clima y es, además, el componente atmosférico del modelo *Community Climate System Model* (CCSM; Collins et al., 2006), que es un modelo acoplado atmósfera-océano[2].

[2] CCSM es actualmente parte de CESM (`http://www.cesm.ucar.edu`), un modelo de última generación del "Sistema Tierra".

CAM está disponible en diferentes representaciones horizontales (espectral y volumenes finitos) y resoluciones y puede ejecutar simulaciones de predicción a corto (horas o días) o largo (meses, años, décadas) plazo para analizar la variabilidad del clima presente, pasado y futuro. Al igual que la mayoría de los modelos climáticos, puede hacer uso de los paradigmas de computación paralela (Grama et al., 2003). En concreto, está preparado para utilizar paralelismo de memoria compartida con OpenMP, de memoria distribuida con *Message Passing Interface (MPI)* o ambos simultáneamente. La elección de un paradigma u otro depende de la resolución utilizada y de la infraestructura de cálculo sobre la que se ejecute el modelo. OpenMP acelera la ejecución del modelo a través de la concurrencia de procesos, pero requiere que todos los procesos accedan a los mismos bloques de memoria. Si se utiliza una resolución alta y el dominio resultante ocupa más memoria de la disponible en el nodo de cálculo, será imprescindible utilizar el paralelismo proporcionado por MPI, que partirá el dominio en bloques y los distribuirá por los nodos elegidos para realizar el cálculo.

Para los estudios de *El Niño* llevados a cabo en esta tesis, hemos utilizado la resolución T42: 128 (longitud) x 64 (latitud) y 27 niveles verticales, es decir, 221184 puntos por paso de tiempo. El modelo produce 32 variables 3D y 56 2D sobre la malla. La simulación de 1 año ejecutada en serie (sin ningún tipo de paralelismo) a una resolución T42 tarda aproximadamente 35 horas utilizando un solo núcleo de un procesador AMD Opteron(TM) Processor 6212. Si ejecutamos la misma simulación utilizando la versión MPI de CAM sobre 2 nodos de cálculo con 2 procesadores AMD Opteron(TM) Processor 6212 de 8 núcleos cada uno (32 núcleos de cálculo), la simulación tarda 96 minutos. Al aumentar la resolución a T85 (128 x 256), el tamaño del dominio resultante requiere una cantidad de memoria superior a la disponible en un solo nodo, por lo que se hace imprescindible el uso de MPI para poder utilizar la memoria de, al menos, dos nodos de cálculo. La misma simulación ejecutada a T42, que tardó 96 minutos, tardó 11 horas en ejecutarse a T85 sobre los mismos nodos.

El modelo *Weather Research and Forecasting* (WRF; Skamarock et al., 2008) es uno de los modelos atmosféricos de área limitada más populares, ampliamente utilizado por las comunidades climáticas y de predicción meteorológica. WRF es desarrollado en NCAR en colaboración con varias agencias y universidades de Estados Unidos. Además, es un modelo de dominio público y, por lo tanto, se beneficia de las contribuciones de una comunidad activa (actualmente hay más de 30.000 usuarios registrados en todo el mundo). A diferencia de otros modelos orientados a aplicaciones concretas, WRF proporciona un marco flexible y computacionalmente eficiente que permite resolver una variedad de problemas de simulación atmósférica en diferentes escalas de tiempo, desde la previsión meteorológica hasta la proyección regional del cambio climático.

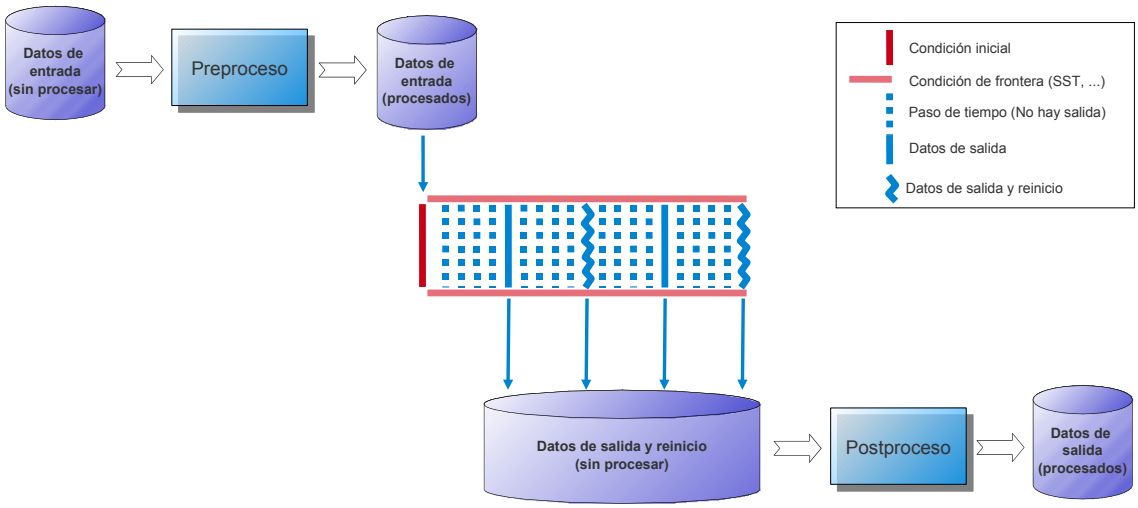


Figura 1.3: Flujo de ejecución de un modelo climático. Tras el preprocesado de los datos, el modelo comienza a simular y va generando datos de salida y de reinicio. Cuando termina se realiza un postprocesado de los datos.

Al igual que ocurre con CAM (y con el resto de los modelos climáticos), WRF puede ejecutarse usando OpenMP y/o MPI. Las características de la simulación (resolución, paso de tiempo, parametrización, horizonte de predicción...) condicionan el tipo de infraestructura computacional sobre el que pueden ejecutarse. Las simulaciones con gran demanda computacional requieren ser ejecutadas en infraestructuras HPC para optimizar el paso de mensajes entre los procesos MPI.

1.1.2. Flujo de ejecución de los modelos

Los modelos atmosféricos utilizan como punto de partida un conjunto de datos que representan el estado de la atmósfera en un momento dado (condiciones iniciales). A medida que avanza la simulación, el modelo va generando ficheros de salida cada cierto tiempo (ver figura 1.3).

Además, los modelos pueden crear ficheros de control (o de reinicio) que contienen el estado de la simulación en un momento dado. Estos ficheros permiten continuar la simulación en caso de que la ejecución se interrumpa. De esta forma, si la simulación falla, se podría continuar a partir del último fichero de reinicio disponible, en lugar de comenzar de nuevo desde el principio.

Por lo general, cada modelo utiliza un formato específico para gestionar los datos, por lo que, en la mayoría de los casos, es necesario realizar un preprocesado de las condiciones iniciales (y de frontera en el caso de los modelos regionales) para convertirlas al formato requerido por el modelo. Normalmente, el preproceso está

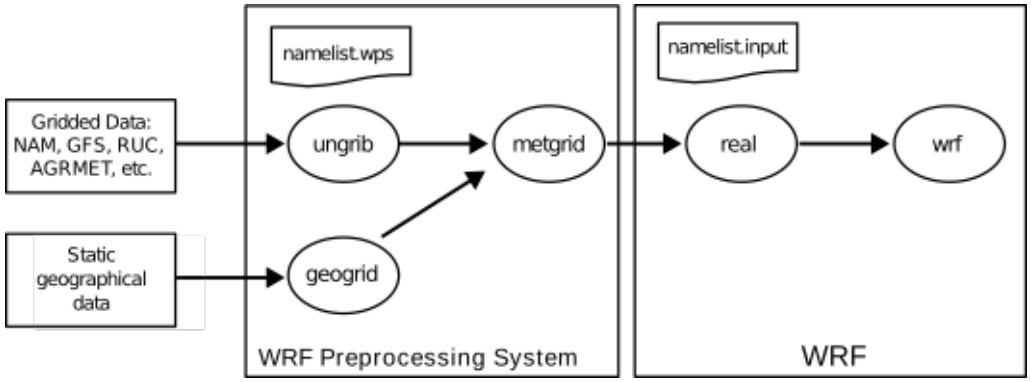


Figura 1.4: Representación esquemática del flujo de trabajo de WRF.

compuesto por la ejecución de varios programas en serie que van transformando los datos (interpolaciones, cálculo de variables derivadas,...) utilizando formatos intermedios hasta llegar al formato requerido por el modelo.

Las salidas de los modelos almacenan todas las variables simuladas. En muchos casos, estas salidas contienen datos que no son necesarios para el análisis del experimento, por lo que es común postprocesar los datos de salida para limitar las variables, o transformarlas, para obtener variables derivadas, reduciendo así el tamaño de los ficheros de salida.

El modelo CAM está formado por un único ejecutable (cam) y su flujo de trabajo corresponde con el representado en la figura 1.3. Desde el punto de vista del flujo de ejecución, WRF es un modelo mucho más complejo. La ejecución de WRF consiste en una secuencia de pasos que utiliza condiciones iniciales y de frontera obtenidas de modelos globales y de datos estáticos de alta resolución (orografía, usos del suelo,...) y produce datos meteorológicos de alta resolución consistentes con esos forzamientos.

A modo de ejemplo, podemos describir la secuencia completa del flujo de ejecución de WRF (figura 1.4): inicialmente se deben ejecutar dos programas en cualquier orden. Usando **geogrid**, el usuario define el dominio espacial del modelo eligiendo una resolución horizontal y un tamaño y localización de malla. **ungrib** convierte los datos meteorológicos globales en formato GRIB[3] a un formato intermedio binario de WRF. En el siguiente paso, **metgrid** interpola horizontalmente al dominio del modelo estos archivos intermedios, los producidos por **geogrid** y los datos meteorológicos globales.

real produce las condiciones iniciales y de contorno en el formato requerido por el núcleo numérico del modelo, que es un formato de datos netCDF (Rew y Davis, 1990) con metadatos propios de WRF. Éste núcleo numérico (**wrf**) es el último paso en el flujo de trabajo del modelo y resuelve la dinámica y la física de la atmósfera,

[3] http://origin.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc.shtml

produciendo datos meteorológicos de alta resolución en la malla elegida.

Los programas **geogrid**, **ungrib** y **metgrid** corresponden a la fase conocida como *WRF Preprocessing System (WPS)* y se configuran utilizando un archivo común (*namelist.wps*). Del mismo modo, los programas **real** y **wrf** se configuran utilizando un archivo diferente, pero común (*namelist.input*). Este modelo de flujo de trabajo no sólo se aplica a WRF, sino también a muchos otros modelos regionales, pero obviamente con distintos programas.

Tradicionalmente, el flujo de ejecución se envía en un solo trabajo que ejecuta todos los componentes. Las características del trabajo (número de procesadores para la ejecución en paralelo, memoria necesaria,...) vienen determinadas por los requerimientos del programa que más recursos requiera. Esto hace que durante la ejecución del resto de los programas los recursos estén infra-utilizados. El objetivo de esta tesis no es mejorar el balance de carga entre procesos MPI, hebras OpenMP y/o carga de memoria, sino mejorar la experiencia del usuario a la hora de gestionar y ejecutar experimentos que consisten en un gran numero de simulaciones y tareas en infraestructuras distribuidas.

1.1.3. Tipos de experimentos de simulación climática

El experimento de simulación climática más simple consiste en una sola ejecución del *workflow* del modelo. Sin embargo, las investigaciones sobre el clima o las predicciones operacionales raramente se basan en una sola ejecución del modelo. En la mayoría de los casos, para llevar a cabo este tipo de experimentos es necesaria la ejecución de muchas simulaciones con dependencias entre ellas. A continuación, describimos algunos tipos de experimentos que suelen llevarse a cabo con modelos climáticos.

Predicción por conjuntos

La alta no linealidad que caracteriza las ecuaciones que gobiernan la atmósfera, da lugar a que pequeños errores en la modelización o en los datos de entrada de los modelos, puedan hacer divergir la predicción obtenida de la realidad. Con el fin de estudiar estas incertidumbres, a finales del siglo XX, se desarrolló una nueva metodología en el estudio del clima llamada predicción por conjuntos (Hagedorn et al., 2005). En sus orígenes, este método trataba de analizar las dos principales fuentes de incertidumbre que afectan a la evolución de los modelos: las debidas a las condiciones iniciales y las debidas al modelo utilizado. Para hacerlo, se consideran perturbaciones en los datos iniciales y/o modelos, de modo que se obtiene un conjunto de predicciones que da lugar a una distribución de los posibles estados de la atmósfera. En los artículos Fernández-Quiruelas et al. (2009a); Fernández-Quiruelas

et al. (2011) de esta tesis, se realizan experimentos de predicción por conjuntos con el modelo CAM en los que el parámetro modificado es la temperatura superficial del océano. Otro ejemplo de predicción por conjuntos son los experimentos multi-física en los que se ejecuta el mismo modelo con diferentes parametrizaciones físicas. Estas parametrizaciones tratan los efectos de fenómenos de pequeña escala en las variables del modelo. Procesos como los intercambios radiativos, la condensación de agua, la turbulencia o los intercambios suelo-atmósfera son tratados por las parametrizaciones físicas. El comportamiento de las distintas parametrizaciones para un mismo proceso físico depende de muchas variables: localización geográfica, estación, variables meteorológicas de interés o incluso del estadístico que estemos considerando: e.g. media, variabilidad o extremos (Jerez et al., 2013; Fernández et al., 2007).

Es bastante habitual ejecutar experimentos multi-física antes de aplicar el modelo sobre regiones concretas. Aunque hay muchos ejemplos de experimentos multi-física de predicción por conjuntos en la literatura (Awan et al., 2011; Evans et al., 2011; García-Díez et al., 2013; Mooney et al., 2013; García-Díez et al., 2015), también existen otro tipos de experimentos de predicción por conjuntos que no son multi-física, como los que analizan el efecto de los cambios de niveles verticales del modelo, de la altura techo de la atmósfera o de otras opciones del modelo (Awan et al., 2011).

Predicción retrospectiva

Actualmente, muchas instituciones llevan a cabo predicciones de corto y medio plazo periódicamente. Sólo unas pocas de estas instituciones producen predicciones globales y otras muchas usan estas predicciones como condiciones de frontera para sus simulaciones con modelos regionales a mayor resolución. Para poder determinar los errores en la predicción, es común llevar a cabo un conjunto de simulaciones de predicciones pasadas, como si el sistema de predicción se hubiese utilizado en el pasado durante un largo periodo de tiempo. Esto se conoce como predicción retrospectiva o *hindcast*. Esta misma metodología también se utiliza para predicción estacional (Díez et al., 2011). Los *hindcast* son también un medio para producir reconstrucciones pasadas del estado de la atmósfera. La frecuente inicialización del modelo usando datos observados, evita que el modelo se desvíe de la realidad observada y mantenga una variabilidad interna reducida. Jiménez y Dudhia (2012); García-Díez et al. (2013); Menéndez et al. (2014) son ejemplos de experimentos de WRF ejecutados con este propósito. El experimento utilizado en el artículo Fernández-Quiruelas et al. (2015) para ilustrar los beneficios de usar WRF4G, es un *hindcast* de alta resolución sobre el mediterráneo basado en el realizado por Menéndez et al. (2014).

Este tipo de experimentos requiere la inicialización del modelo a intervalos regulares, lo que normalmente hace necesaria la ejecución de muchas simulaciones. Por poner un ejemplo: un *hindcast* de 20 años de predicción diaria necesita la gestión de

más de 7000 simulaciones.

Simulaciones climáticas

La simulación climática necesita analizar largos periodos de tiempo para determinar el comportamiento de la atmósfera. En las simulaciones climáticas, los modelos se usan esencialmente igual que para la predicción a corto plazo con la diferencia de que normalmente se acoplan a otros modelos que representan componentes del sistema climático más lentos (superficie terrestre, océanos,...). En los modelos regionales, los componentes más lentos normalmente se introducen a través de las condiciones de contorno.

Existen estudios recientes que utilizan el modelo WRF para producir simulaciones climáticas regionales sobre diferentes partes del globo (Nikulin et al., 2012; Mooney et al., 2013; Argüeso et al., 2012a,b; Cardoso et al., 2012; Jiménez-Guerrero et al., 2013; Vautard et al., 2013; Expósito et al., 2015; Díaz et al., 2015). La mayoría de ellas utilizan la capacidad del modelo para representar el clima observado y proyectan los cambios futuros a escala regional en respuesta al incremento de concentración de los gases de efecto invernadero.

1.2. Infraestructuras de computación distribuidas

La definición de infraestructura de computación distribuida varía notablemente en función del ámbito en el que se utilice. En sus inicios, surgió como un nuevo paradigma de computación en el que múltiples sistemas conectados a través de una red de área local trabajan para resolver un único problema (Kshemkalyani y Singhal, 2011). Este tipo de computación es lo que hoy en día conocemos como *cluster* de cálculo y el adjetivo "distribuida" hace referencia al reparto de carga de trabajo entre los nodos de cálculo.

Con la aparición de Internet, el término computación distribuida comenzó a usarse para designar a las infraestructuras que utilizaban, de forma coordinada, recursos heterogéneos y distribuidos geográficamente (Casavant y Kuhl, 1988; Thain et al., 2005). Entre ellas cabe destacar la computación *Grid* (Foster y Kesselman, 1999) y la computación voluntaria (Anderson y Fedak, 2006). Esta acepción de infraestructura distribuida, en la que el término "distribuida" hace referencia a la localización de los nodos de cálculo, es la utilizada en esta tesis.

El paradigma de computación *Cloud*, apareció unos años después, permitiendo la creación de *clusters* virtualizados y escalables con un nuevo modelo de pago por uso.

En esta sección se van a describir las infraestructuras utilizadas en este trabajo,

incluyendo el nuevo concepto de infraestructura híbrida de computación distribuida (HDCI), entendida como la infraestructura resultante de unir *clusters*, *Grid* y *Cloud*.

A pesar de que el paradigma de computación voluntaria ha sido utilizado con éxito para llevar a cabo experimentos de simulación climática muy específicos (Allen, 1999), los retos científicos a resolver para el aprovechamiento de estas infraestructuras en experimentos más generales no se han abordado en esta tesis, pero podrían ser una línea futura.

1.2.1. Cluster

Las primeras infraestructuras de cálculo distribuidas aparecieron con el desarrollo de las redes de área local. En estas infraestructuras, comúnmente conocidas como *cluster* de computación, varios ordenadores trabajan conjuntamente para resolver un problema, comunicándose a través de una red de área local. Los paradigmas de computación paralela (Grama et al., 2003), como MPI facilitan la distribución de carga entre los diferentes procesadores involucrados en el cálculo.

Para proporcionar un acceso homogéneo a los nodos de cálculo, es necesario utilizar un sistema de gestión de recursos locales (LRMS), también conocido como gestor de colas, que permita controlar la asignación de recursos y planificar y gestionar los trabajos en los nodos. En un LRMS, los usuarios preparan sus trabajos y los envían a la infraestructura para que se ejecuten tan pronto como sea posible. Una vez enviados, los trabajos esperan en la cola a ser ejecutados en los recursos a medida que éstos se van liberando, siguiendo una política de planificación basada en prioridades. Algunos de los gestores de colas más utilizados son: SGE (Gentzsch, 2001), PBS (Henderson, 1995) y SLURM (Yoo et al., 2003).

Hasta los años 80, los supercomputadores eran un solo equipo que disponía de varios procesadores y de arquitectura de memoria compartida. Con la proliferación de la informática de propósito general y sus bajos precios, poco a poco, las infraestructuras en *cluster* se fueron haciendo un hueco en el mundo de la supercomputación. A día de hoy, casi todas las infraestructuras que componen la lista de los 500 supercomputadores más potentes del mundo[4] utilizan arquitecturas en *cluster*

Los componentes más importantes de un *cluster* son:

- Nodos: los *clusters* están compuestos por los nodos de cálculo que se encargan de ejecutar los trabajos y por la interfaz de usuario, a la que se suele acceder por SSH para realizar el envío y gestión de trabajos.

[4] https://www.top500.org

- Red de interconexión: además de permitir a los procesos paralelos transmitir datos entre ellos para llevar a cabo el paralelismo de tareas, la red se puede utilizar para leer y escribir en el sistema de almacenamiento.

- Sistema de almacenamiento: los *clusters* requieren de un sistema de archivos compartido que permita a todos los nodos tener acceso a los mismos archivos. A través de este sistema de archivos compartido es como habitualmente las aplicaciones que utilizan procesos MPI para distribuir la carga entre los nodos acceden a los mismos datos.

La mayoría de los supercomputadores utilizan arquitecturas en *cluster*, lo que no quiere decir que todos los *clusters* sean supercomputadores. Generalmente, lo que distingue un *cluster* convencional de una infraestructura HPC o supercomputador (de ahora en adelante, usaremos el término infraestructura HPC para designar también a los supercomputadores) son la red de interconexión y el sistema de almacenamiento. Para que las aplicaciones paralelas escalen cuando se ejecutan en nodos diferentes, es necesario que la red de interconexión entre los nodos sea de altas prestaciones. Normalmente estas redes son de tipo *Remote Direct Memory Access (RDMA)*[5], como infiniband (Shanley y Winkles, 2003), Myrinet (Boden et al., 1995) y Omni-Path (Birrittella et al., 2015). Además, en los casos en los que las aplicaciones realizan grandes movimientos de datos, es necesario disponer de un sistema de ficheros distribuido como Lustre (Wang et al., 2009) o Ceph (Weil et al., 2006), que sea capaz de proporcionar un ancho de banda suficiente para evitar cuellos de botella en la lectura y escritura de datos.

En los últimos años, ha proliferado el uso de procesadores vectoriales como *Field Programmable Gate Array* (FPGA; Herbordt et al., 2007), Xeon Phi (Jeffers y Reinders, 2013) o unidades de procesamiento gráfico, como las tarjetas Tesla (Lindholm et al., 2008). Aunque su posible beneficio en la ejecución de simulaciones con modelos climáticos es de gran interés hoy en día, este tipo de infraestructuras no son objeto de esta tesis

1.2.2. Grid

La tecnología *Grid* (Foster y Kesselman, 1999) o computación geográficamente distribuida emergió bajo la proliferación de Internet como una alternativa integradora cuyo principal objetivo era facilitar al usuario el acceso a recursos distribuidos en distintos lugares. Cualquier tipo de recurso susceptible de ser accedido a través de Internet (*clusters*, satélites, unidades de almacenamiento, bases de datos,...), puede ser accedido a través de tecnología *Grid*. El objetivo de esta tecnología es facilitar el

[5] http://www.rdmaconsortium.org

acceso a recursos heterogéneos y distribuidos geográficamente de una manera flexible y segura a miembros de distintas instituciones.

En una infraestructura en la que se comparten los recursos a través de tecnología *Grid*, el usuario encuentra un único punto de autenticación y autorización para todos los recursos, así como servicios de exploración, monitorización y acceso seguro a éstos. En el caso de que los recursos que se compartan sean elementos de almacenamiento y computación, el usuario puede ver la infraestructura *Grid* como un macrosistema de computación formado por una gran cantidad de procesadores y almacenamiento distribuidos geográficamente (metacluster).

La tecnología *Grid* tiene dos importantes características que la distinguen de otros tipos de computación distribuida:

- Los recursos de cada sitio (unidad administrativa de *Grid* tales como instituciones o delegaciones de empresas) no están sujetos a un control centralizado global. Cada sitio tiene control sobre su equipamiento, y nadie, salvo los administradores de ese sitio pueden modificar las características de los recursos.

- Para unirse a una infraestructura *Grid*, los recursos que se van a compartir no tienen porqué ser modificados. Por ejemplo, si tenemos un *cluster* con un determinado LRMS o sistema operativo, no sería necesario cambiarlo para poder acceder a él a través de la infraestructura *Grid*, ya que la capa de software que se instala en el *cluster* se encarga de hacer todo lo necesario para integrarlo en la infraestructura *Grid*. Gracias a este software, conocido como *middleware*, los sitios pueden cumplir con los niveles de servicio (SLA) acordados por los responsables de la infraestructura.

Para que un usuario pueda acceder a recursos distribuidos geográficamente de una forma transparente, es necesario que la tecnología *Grid* proporcione unos servicios que se encarguen de realizar todas las operaciones intermedias sin su intervención. Sin la tecnología *Grid*, el usuario debería conocer sus credenciales (por ejemplo, usuario y clave) en todos los recursos y no tendría forma de saber las características de los recursos disponibles (sistema operativo, procesadores colas, etc...) si el usuario antes no se conecta a cada uno de ellos de forma independiente y obtiene esa información.

Actualmente existen varias soluciones *Grid* que, haciendo uso de diferentes servicios y *middleware*, nos facilitan el acceso a los recursos distribuidos. Cada implementación tiene distintos requerimientos y funcionalidades aunque todas ellas hacen uso de una arquitectura común basada en tres capas (figura 1.5): (1) *middleware* instalado en los recursos para homogeneizar el acceso, (2) servicios centrales que gestionan la interacción del usuario con los recursos y (3) herramientas de usuario.

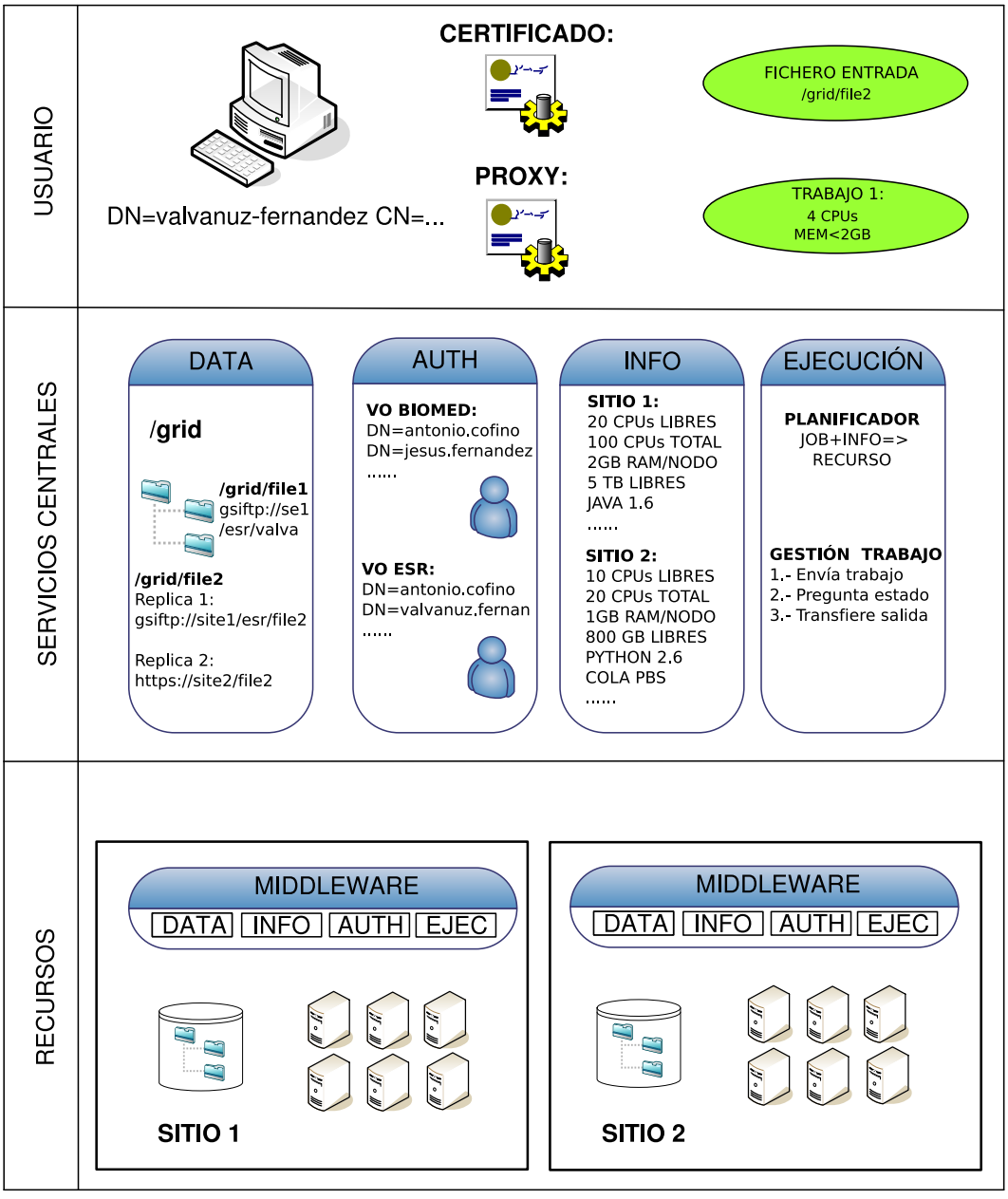


Figura 1.5: Esquema de la arquitectura de una infraestructura *Grid*. En la parte inferior podemos ver cada uno de los sitios que lo conforman, todos ellos encapsulados por el *middleware*. En la parte central encontramos los servicios centrales de la infraestructura y en la parte superior, la interfaz del usuario final, ambos descritos en la sección 1.2.2

.

Los servicios centrales se encargan de gestionar el acceso a los recursos en nombre del usuario. Para que los servicios centrales puedan interactuar con los recursos, es necesario que los recursos utilicen un *middleware* que se encargue de encapsular los servicios que ofrece a través de un protocolo *Grid* común a todos los proveedores de ese servicio (servicio de datos, de cálculo, ...). De esta forma, los servicios centrales acceden a cada tipo de recursos a través de un mismo protocolo sin necesidad de conocer las particularidades de cada uno. En Foster et al. (2001) se puede encontrar una descripción exhaustiva de todos los componentes de la tecnología *Grid*.

Las soluciones *Grid* disponen, además, de herramientas de usuario, librerías e interfaces de programación estándar (API) que facilitan al usuario la interacción con los recursos a través de los servicios centrales y del *middleware* instalado en los recursos.

La comunidad *Grid* ha hecho un gran esfuerzo en la adopción de estándares. Como resultado, en 2006 se creó el *Open Grid Forum (OGF)*. OGF se divide en grupos de trabajo que se encargan de analizar las necesidades de los usuarios y tratan de generar estándares para los distintos componentes de la tecnología *Grid*. Como resultado de estos grupos de trabajo, se han establecido algunos estándares que están siendo adoptados por las principales soluciones *Grid*.

A continuación se describen los servicios *Grid* más utilizados, así como los estándares del OGF recomendados para cada uno de estos servicios.

Autenticación y autorización

Cuando un sitio decide compartir un recurso, ya sea almacenamiento, recursos de computación o bases de datos, es necesario decidir qué se va a compartir y a quienes se les va a dar permisos para acceder o modificar los recursos compartidos. Cuando un recurso recibe una petición por parte de un usuario (por ejemplo, leer un fichero o ejecutar un trabajo en un recurso computacional), debe contactar con el servicio central de autenticación y autorización (ver figura 1.5) para identificar al usuario (autenticación) y determinar si el usuario tiene permiso para realizar la operación que solicita (autorización).

En un *cluster*, la autenticación suele hacerse a través de un usuario y contraseña y la autorización por pertenencia a un grupo. Para el acceso a sitios web, es común el uso de sistemas de autenticación y autorización digital como OpenId (Recordon y Reed, 2006), OAuth (Hammer-Lahav et al., 2011) o SAML (Lewis, 2009).

En *Grid*, para autenticar a los usuarios en los recursos, se utiliza *Grid Security Infrastructure* (GSI; Foster et al., 1998) basada en la infraestructura de clave pública PKI (Adams y Lloyd, 1999) en la que se identifica a los usuarios y a los servicios a través de certificados estándar X509 (Welch et al., 2004). Este certificado identificará al usuario de forma única en el *Grid* y será utilizado cada vez que quiera realizar

cualquier operación. Para evitar tener que escribir la clave con la que está protegido el certificado cada vez que realice una operación, se creará un *proxy* (copia autofirmada de certificado) de duración limitada (normalmente 12 horas), que será utilizado automáticamente en todas las operaciones (Tuecke et al., 2004). El uso del *proxy*, permitirá también a los servicios centrales interactuar con los recursos en nombre del usuario.

Para poder autorizar el acceso a los recursos, una vez que se ha identificado a los usuarios, es necesario saber qué permisos tienen sobre estos (autorización). Para autorizar usuarios en *Grid*, el OGF establece el estándar *Virtual Organization Membership Service* (VOMS; Alfieri et al., 2004). En VOMS los usuarios son identificados a través de sus certificados y son organizados en Organizaciones Virtuales (VO; Alfieri et al., 2004) en función de su perfil (trabajan en el mismo proyecto, organización o ámbito de investigación). Cuando un recurso recibe una petición de un usuario, comprueba a qué VO pertenece y decide si tiene permisos para acceder.

Información y exploración de recursos

Para poder hacer un uso eficiente de los recursos, los servicios centrales necesitan conocer el estado y características de éstos. Por eso, todos los recursos publican información sobre su estructura y estado (configuración, carga de trabajo, política de uso,...) a través de los servicios de información instalados por el *middleware*. OGF recomienda publicar la información de los recursos usando el esquema GLUE (Andreozzi et al., 2009), un sistema de información estándar para representar de forma uniforme el estado del *Grid*.

El servicio central de información recoge la información de los recursos y la publica para que el resto de los servicios accedan a ella. Normalmente, esta información también se publica a través de la web para que los usuarios puedan explorarla. Por ejemplo, el estado y características de la infraestructura EGI (ver sección 1.2.2) puede consultarse en `https://operations-portal.egi.eu/vapor`. También es posible hacer consultas directamente al servicio de información utilizando las herramientas de usuario o la API.

Ejecución de trabajos

Dentro de los recursos que pueden ejecutar trabajos podemos encontrar una gran variedad de sistemas, desde pequeños *clusters* o nodos de computación sueltos a grandes supercomputadores. Gracias al *middleware*, el usuario no tiene que preocuparse del tipo de LRMS utilizado por el sistema en el que se va a ejecutar su trabajo.

Cuando un usuario envía un trabajo al *Grid*, el planificador consulta el servicio

de información para ver el estado de los recursos y decide a qué recurso enviar el trabajo. Este trabajo es gestionado por el servicio de ejecución que se encarga de enviar el trabajo y monitorizar su estado hasta su finalización.

Como existen gran variedad de soluciones *Grid*, que no son compatibles entre ellas, el OGF ha definido varias interfaces estándar para el envío y gestión de trabajos. Entre ellas, cabe destacar *Simple API for Grid Applications* (SAGA; Goodale et al., 2006)) y *Distributed Resource Management Application API* (DRMAA; Tröger et al., 2007). SAGA es una API orientada a aplicaciones que asegura su compatibilidad con diferentes infraestructuras de cálculo (*cluster*, *Grid* y *Cloud*), ofreciendo una capa extra de abstracción, mientras que DRMAA es una especificación de alto nivel para el envío y control de trabajos a gestores de infraestructuras distribuidas como *Grids* o *clusters*. OGF también define un lenguaje para el envío de trabajos sencillos a estas infraestructuras llamado *Job Submission Description Language* (JSDL; Anjomshoaa et al., 2005)

Gestión de datos

De la misma forma que los usuarios pueden acceder a muchos nodos de computación, el *Grid* también proporciona acceso a gran cantidad de recursos de almacenamiento distribuidos. Los servicios de almacenamiento se encargan de localizar, transferir y gestionar los datos. El primer problema al que hay que enfrentarse es la elección del protocolo con el que realizar la transferencia.

No es posible transferir los datos usando protocolos como FTP (Postel y Reynolds, 1985) o SSH (Ylonen, 1996), ya que carecen de soporte para la infraestructura GSI y no podrían autenticar a los usuarios. Por eso, se ha creado un protocolo estándar para la transferencia de ficheros en *Grid* llamado GridFTP (Allcock et al., 2005). Este protocolo, basado en FTP y con soporte para GSI (descrita en la sección 1.2.2), se caracteriza por un alto rendimiento y un transporte de datos fiable. GridFTP es a la vez el estándar recomendado por el OGF y el estándar *de facto* utilizado por casi todas las soluciones *middleware* para la transferencia de datos.

Cuando se dispone de grandes cantidades de datos distribuidas en muchos lugares diferentes y sólo accesibles a través de Internet, se corre el riesgo de perder el acceso a algunos datos debido a fallos en la red o a falta de servicio en los sitios. Además, debido al carácter distribuido de la arquitectura *Grid*, es esperable que los datos también se encuentren distribuidos, por tanto, para un usuario, es difícil saber en qué sitio se encuentra un determinado fichero de datos.

Para que el usuario pueda ver todos los recursos de almacenamiento como una sola unidad de almacenamiento virtual, independientemente de donde estén localizados, es necesario el uso de un catálogo que mantenga una relación entre los nombres lógicos de la estructura virtual y los lugares donde están almacenados, similar a una

estructura de sistema de ficheros residentes en un dispositivo de bloques. De esta forma, podríamos tener los datos de un determinado experimento científico distribuidos por unidades de almacenamiento de todo el mundo, sin que el usuario tenga que preocuparse de dónde han ido a parar los datos. El uso de un sistema de catálogo también facilita la gestión de réplicas, haciendo posible la optimización de recursos y su disponibilidad. Así, si un sitio falla y todos los datos están replicados, se podría seguir accediendo a todos ellos.

Infrastructuras Grid

La tecnología *Grid* puede ser usada para propósitos muy diferentes, desde proporcionar acceso único a recursos de una misma empresa o institución distribuidos geográficamente, hasta la creación de infraestructuras de computación y almacenamiento de gran capacidad que aglutinan recursos de numerosos centros de investigación y universidades.

En esta sección presentamos dos tipos de infraestructuras *Grid*. Las primeras, denominadas e-infraestructuras (EGEE, OGF y ESGF), son grandes infraestructuras orientadas al HTC, en las que se comparten recursos de múltiples sitios, mientras que las segundas (por ejemplo, XSEDE y PRACE), orientadas al HPC, se crearon con el objetivo de proporcionar acceso homogéneo a grandes supercomputadores y cuentan con pocos recursos muy potentes. Además de proporcionar una gran capacidad de cálculo y almacenamiento, uno de los objetivos de las e-infraestructuras es promover el uso de nuevos métodos de cooperación entre los investigadores, facilitando el intercambio resultados y proporcionando nuevas formas de trabajo colaborativas que favorezcan la circulación de conocimiento (Hey y Trefethen, 2005). Por ello, las e-infraestructuras promueven la incorporación de nuevas instituciones y disciplinas y tratan de simplificar el acceso a los investigadores. Por el contrario, conseguir acceso a las infraestructuras *Grid* de supercomputación es difícil y requiere realizar solicitudes periódicas justificando la necesidad de su utilización.

A continuación, se describen las infraestructuras *Grid* más relevantes hoy en día:

NGI y EGI El germen de las infraestructuras *Grid* europeas fue el proyecto europeo *Enabling Grids for E-sciencE (EGEE)*. EGEE surgió en 2004 como solución al problema de almacenamiento y análisis de los datos que se producirían con el Gran Colisionador de Hadrones (LHC) del CERN. Tras el éxito de EGEE, se iniciaron otros proyectos europeos para promocionar el uso del *Grid* fuera de Europa. Entre ellos se encontraba *E-Infrastructure shared between Europe and Latin America (EELA)*, en el que participaban 21 instituciones europeas y latino americanas.

Actualmente, la mayoría de los países europeos cuenta con su propia infraestructura *Grid* nacional (NGI) que conecta recursos HTC de diferentes universidades y

centros de investigación. Las NGI europeas se han unido, a través del *middleware* UMD (ver sección 1.2.2), formando la *European Grid Initiative (EGI)*, que es la infraestructura sucesora de EGEE. Al igual que PRACE, EGI forma parte del mapa europeo de infraestructuras de computación (Lossau, 2012). Tanto las NGI como EGI organizan a los investigadores por disciplinas científicas en VO. Un investigador del clima español, podría acceder a los recursos de la VO *Earth Science Research (ESR)* de EGI y a los de la VO earth.vo.ibergrid.eu de la NGI de España y Portugal. Aunque las NGI suelen agrupar recursos de un solo país, excepcionalmente España y Portugal se han unido para formar la NGI IberGrid.

Open Science Grid (Pordes et al., 2007, OSG) es la infraestructura HTC más grande de Estados Unidos. Al igual que EGI, organiza a los usuarios por VO. OSG basa su *middleware Grid* (*Virtual Data Toolkit*[6]) en *Globus Toolkit*, aunque también incluye componentes de gLite, Condor y VOMS. Actualmente, está compuesto por 100 universidades y centros de investigación que proporcionan entre cientos de miles y millones de procesadores.

XSEDE (*Extreme Science and Engineering Discovery Environment*) está orientada a la investigación científica que demanda recursos HPC. Actualmente proporciona acceso a los supercomputadores de 9 centros de investigación de Estados Unidos.

PRACE [7] (*Partnership for Advanced Computing in Europe*) es un consorcio compuesto por los centros de supercomputación más importantes de Europa agregados en dos niveles (*Tier-0* y *Tier-1*) que proporciona acceso distribuido a los supercomputadores a través de tecnología *Grid*.

Earth System Grid Federation (ESGF; Williams et al., 2011) es una iniciativa *Grid* mundial que trata de resolver el problema de la gestión, acceso y exploración de datos de Ciencias de la Tierra en entornos distribuidos. Aunque la infraestrucutura *Grid* generada permite realizar tareas de procesado de datos, la infraestructura fue diseñada para optimizar el acceso y localización de los datos. Cuenta con una completa colección de servicios y herramientas que facilitan al usuario el acceso a los datos en los diferentes centros de investigación asociados.

ESGF es la infraestructura utilizada para distribuir los datos generados por las simulaciones llevadas a cabo para los informes de evaluación del Grupo Intergubernamental de Expertos sobre el Cambio Climático (IPCC).

[6] http://vdt.cs.wisc.edu
[7] http://www.prace-ri.eu

Ejemplos de middleware Grid

Actualmente existen numerosos ejemplos de *middleware Grid*: *Globus Toolkit* (Foster, 2006), *Unified Middleware Distribution (UMD)*[8], UNICORE (Erwin y Snelling, 2002), *Advance Resource Conector* (Ellert et al., 2007), etc. Todos ellos ofrecen software que facilita el acceso transparente a los recursos usando una arquitectura similar a la explicada anteriormente (servicios de autorización y autenticación, gestión de datos y trabajos y servicios de información). UMD es el *middleware* utilizado en la infraestructura EGI. Dado que esta infraestructura ha sido la más utilizada para desarrollar y probar el *framework* presentado en esta tesis, vamos a hacer especial hincapié en él. También vamos a describir el *middleware* de *Globus Toolkit* por ser el más extendido y un estándar *de facto*.

Globus Toolkit (Foster, 2006) fue desarrollado en los años 90 en Estados Unidos y contiene un completo conjunto de servicios *Grid*. Además de proveer servicios de alto nivel como la gestión de datos o trabajos, proporciona a los usuarios herramientas para desarrollar sus propios servicios. La arquitectura es modular, lo que permite a cada usuario instalar sólo los componentes que necesita.

Está basado en Linux y contiene API para C, Java y Python. De los *middlewares* descritos en este trabajo, Globus es el más versátil. A diferencia de UNICORE y UMD, concebidos para crear grandes e-infraestructuras, los componentes de Globus pueden ser usados en ámbitos empresariales o en pequeños grupos para dar soluciones al acceso de recursos distribuidos.

UMD y gLite La solución *Grid* gLite (Laure et al., 2006) fue creada para la iniciativa EGEE. Además de desarrollos propios, integraba componentes basados en versiones antiguas de *Globus Toolkit*. Con la evolución de EGEE a EGI, gLite se convirtió en UMD.

UMD incluye servicios de seguridad, de información y monitorización, de gestión de datos y de gestión de trabajos, cada uno de los cuales tiene su propio rol y ha de ser instalado en máquinas independientes (no es posible instalar dos roles en un mismo equipo). Para interaccionar con los recursos, los usuarios han de utilizar un equipo con el rol del Interfaz de Usuario (UI). Para unir los recursos al *Grid*, es necesario instalar los roles *Compute Element (CE)* y *Storage Element (SE)* en los *clusters* de cálculo y unidades de almacenamiento, respectivamente. Cada servicio central también requiere un rol independiente. Así, el *Workload Manager System (WMS)* se encarga de planificar y gestionar los trabajos entre los CE y el *Logical*

[8] http://repository.egi.eu/category/umd_releases/distribution/umd-3

File Catalog (LFC) de mantener un sistema de ficheros virtual cuyo almacenamiento físico son los SE.

1.2.3. Cloud

El paradigma de computación *Cloud* surgió como un modelo de abstracción y aprovisionamiento dinámico de recursos computaciones que se ofrece bajo demanda a través de Internet (Fox et al., 2009). El ejemplo más claro, y que de alguna manera propició el nacimiento de la computación *Cloud*, es la tecnología *Amazon Elastic Compute Cloud (EC2)*. EC2 es un servicio de alojamiento en Internet que ofrece servidores virtuales y almacenamiento de tamaño modificable. Su modelo de facturación por tiempo de uso lo hace ideal para escenarios en los que se necesitan servidores de forma puntual.

A los servidores virtuales utilizados en *Cloud* se les denomina instancias. Los proveedores *Cloud* ofrecen instancias con diferentes características hardware y permiten elegir entre una gran variedad de preconfiguraciones de sistema operativo y software. La facilidad de mantenimiento y despliegue del *Cloud* ha llevado a muchas empresas a migrar al *Cloud* los servicios que anteriormente tenían alojados en su Centro de Proceso de Datos (CPD). Sin embargo, el uso del *Cloud* para la ejecución de aplicaciones de investigación científica es bastante reducido. A continuación mostramos las principales ventajas y desventajas del uso del *Cloud* (Armbrust et al., 2010; Fox et al., 2009):

Ventajas:

- Pago por uso: el modelo de facturación por tiempo de uso del *Cloud* permite a las empresas/instituciones alquilar recursos de acuerdo con sus necesidades. Gracias a esto, no es necesario hacer un desembolso inicial para adquirir equipamiento, ni dimensionarlo teniendo que predecir los posibles picos de carga que se puedan producir en un futuro.

- La facilidad de despliegue en el *Cloud* esconde la complejidad de mantener y gestionar una infraestructura computacional, tanto a nivel hardware como software.

- Escalabilidad y disponibilidad instantánea: las infraestructuras *Cloud* permiten aumentar o disminuir la capacidad en minutos, según cambien las necesidades de los usuarios.

- Flexibilidad: la facilidad para elegir entre distintos tipos de instancias, sistemas operativos y paquetes de software, facilita el despliegue de las aplicaciones.

Además, como las instancias son virtuales, se pueden mover entre diferentes infraestructuras *Cloud* de forma sencilla.

Desventajas:

- Transferencias y almacenamiento de datos: en el *Cloud*, además de facturar por las instancias y por el almacenamiento compartido, se factura por las transferencias de datos hacia/desde el *Cloud*. Si las aplicaciones que se van a ejecutar tienen gran cantidad de datos de entrada, el coste de transferirlos y mantenerlos en el almacenamiento compartido puede ser muy elevado. Además, el almacenamiento en *Cloud* plantea problemas de privacidad y confidencialidad debido a que el proveedor de servicios tiene necesariamente acceso a todos los datos y podría revelarlo de forma accidental o deliberada o usarlo para fines no autorizados.

- Tradicionalmente, el hardware utilizado por las instancias *Cloud* era de propósito general. Esto, junto con la sobrecarga debida a la virtualización, hacía que muchas aplicaciones con requerimientos de ancho de banda de entrada/salida, uso intensivo de memoria RAM o bajas latencias en las comunicaciones entre procesos, no pudiesen usar estas infraestructuras.

Recientemente, varios proveedores *Cloud* han creado instancias optimizadas para el cálculo científico. Tanto las soluciones HPC de EC2[9] como las de Azure[10], ofrecen una amplia gama de instancias para HPC que disponen de redes infiniband de baja latencia, gran capacidad de memoria, discos SSD y/o unidades de procesamiento gráfico (GPU).

A pesar de que se han hecho algunos avances en el campo de las estandarizaciones para el acceso al *Cloud*, la mayoría de las infraestructuras *Cloud* ofrece sus propias API para gestionar los recursos.

Aunque hasta ahora sólo hemos mencionado *Clouds* públicos, muchas instituciones cuentan con *Clouds* privados en sus instalaciones. El objetivo de los *Clouds* privados es la flexibilización de los servicios a la hora del despliegue y mantenimiento y la escalabilidad. Para la creación de *Clouds* privados pueden utilizarse soluciones como OpenNebula (Milojičić et al., 2011) y OpenStack (Sefraoui et al., 2012). Cuando los *Clouds* privados de las instituciones y empresas no pueden satisfacer los picos de carga de trabajo, es común el uso conjunto de infraestructuras públicas y privadas. Estos entornos mixtos se conocen como *Clouds* híbridos.

[9] https://aws.amazon.com/hpc/

[10] https://docs.microsoft.com/en-us/azure/virtual-machines/windows/a8-a9-a10-a11-specs

1.2.4. Infraestructuras híbridas de computación distribuida

Actualmente, los investigadores tienen acceso a múltiples recursos computacionales de tipo HPC, *Grid* y *Cloud*. Poder acceder a todos estos recursos simultáneamente, permitiría combinar los mejores atributos de cada uno de los paradigmas y daría acceso a una enorme capacidad de cómputo. Como se ha visto en la sección 1.2.2, la forma de proporcionar un acceso transparente a estos recursos sería utilizando un *middleware Grid*.

El primer problema que surge al tratar de homogeneizar este acceso, es la variedad de *middlewares Grid* disponible actualmente, que se convierte a su vez, en una fuente de heterogeneidad cuando queremos unir diferentes infraestructuras *Grid*. Los *middleware* que encontramos hoy en día, no disponen de interfaces comunes para interactuar entre ellos, por lo que, llegados a este punto, se hace necesario definir el concepto de federación, que hace referencia a la agregación transparente de diferentes infraestructuras *Grid*. Existen varios ejemplos en la literatura en los que se estudian las federaciones *Grid*, pero quizás, el que mejor las caracteriza es Vázquez Blanco (2013). En este trabajo, además de ofrecer soluciones para la federación *Grid*, se propone una "arquitectura de aprovisionamiento dinámico basada en el metaplanificador GridWay, en el cual se combinan técnicas de ambos mundos, *Grid* y *Cloud*, para conseguir una infraestructura adaptable a la demanda, flexible en el tipo de aplicaciones que pueda ejecutar y con capacidad de crecimiento a proveedores *Cloud* públicos."

La mayor parte de las infraestructuras utilizadas por la comunidad del clima son de tipo HPC, que, por lo general, no proporcionan una interfaz de acceso *Grid*, ni permiten su instalación y, normalmente, la única forma de acceder a ellas es a través de SSH. Aunque la solución propuesta por Vázquez Blanco (2013) soluciona el acceso homogéneo a *Grid* y *Cloud*, no ofrece mecanismos para acceder directamente a infraestructuras que no utilizan *middleware Grid*. Así, se hace necesario definir el concepto de infraestructura híbrida de computación distribuida (HDCI), como la infraestructura resultante de unir *clusters* (que no proporcionan acceso a través de un *middleware*), *Grid* y *Cloud*. Mateescu et al. (2011); Hwang et al. (2016) muestran ejemplos de HDCI y hacen una caracterización de los paradigmas HPC, *Grid* y *Cloud* en la que se analizan las ventajas y desventajas de cada uno de ellos y la problemática que surge al intentar aprovechar estas infraestructuras al mismo tiempo.

Para poder dar un acceso transparente a los recursos de una HDCI, es necesario disponer de unos servicios equivalentes a los mostrados para el *Grid* en la sección 1.2.2: autenticación y autorización, información, gestión de datos y gestión de trabajos. Es importante destacar que todas las infraestructuras distribuidas (*cluster*,

Cloud, *Grid* y HDCI), necesitan estos servicios para proporcionar un acceso homogéneo a sus recursos. Lo que diferencia los servicios utilizados en unas infraestructuras de otras, es la complejidad de la infraestructura. Por ejemplo, en un *cluster* la autenticación suele hacerse a través un sistema de identidad basado en LDAP (Howes et al., 2003) o kerberos (Neuman y Ts'o, 1994), mucho más sencilla que la gestión de certificados X509 y VO que es necesaria para el *Grid*.

Al no ser siempre posible la instalación de un *middleware Grid* en los recursos HPC, los *frameworks* de HDCI se conectan a través del protocolo SSH a las infraestructuras HPC utilizando la cuenta del usuario con la que se va a ejecutar en ellas. Una vez conectados, ejecutan los procesos necesarios para monitorizar la infraestructura y gestionar los datos y los trabajos. Algunos ejemplos de *frameworks* que permiten la unión de HDCI son: DIRAC (Tsaregorodtsev et al., 2008), gUSE/WS-PGRADE (Farkas y Kacsuk, 2011), Condor-G (Frey et al., 2002) o DRM4G (Blanco Real, 2017). Es importante destacar que, a diferencia de las distintas soluciones LRMS de los *clusters*, que básicamente disponen de las mismas funcionalidades, las soluciones HDCI son muy heterogéneas en cuanto a arquitectura y funcionalidades. Por ejemplo, gUSE/WS-PGRADE es un portal web al que los usuarios se conectan para gestionar sus flujos de trabajo, mientras que Condor-G es una herramienta de línea de comandos que permite distribuir las cargas de trabajo entre diferentes infraestructuras.

Uno de los objetivos de esta tesis, es poder llevar a cabo nuevos retos científicos optimizando el uso de una HDCI. Tras realizar un análisis de las opciones disponibles, la única herramienta que se adaptaba a los requerimientos de las simulaciones climáticas resultó ser GridWay (Huedo et al., 2001). Su único inconveniente era que no disponía de mecanismos para acceder a infraestructuras de tipo *cluster*, por lo que decidimos implementar una nueva funcionalidad en GridWay que facilitase la incorporación de este tipo de recursos. Desarrollar un software de estas características que cumpla con requisitos de robustez y escalabilidad, es un trabajo que está fuera del alcance de esta tesis. Por ello, este nuevo software, llamado DRM4G, ha sido objeto de otra tesis doctoral (Blanco Real, 2017) realizada dentro del mismo grupo de investigación.

1.3. Predicción numérica de la atmósfera en Grid y HDCI

Uno de los principales objetivos de esta tesis consiste en mostrar a la comunidad científica los beneficios de las e-infraestructuras *Grid* y de las HDCI, para llevar a cabo experimentos con modelos climáticos. En esta sección, se hace un análisis de las iniciativas con objetivos similares a los de esta tesis y se muestra por qué ninguna de las soluciones presentadas cumple con los requisitos exigidos.

Las primeras investigaciones con modelos climáticos llevadas a cabo a través de un *middleware Grid* se publicaron en Nefedova et al. (2006). Este trabajo es posiblemente, el más avanzado de todos los que vamos a presentar en esta sección. Muestra cómo adaptando el entorno de trabajo *Virtual Data System* (VDS; Kricheff et al., 2001) para cumplir los requerimientos del modelo FOAM (Jacob et al., 2001), es posible la ejecución de grandes experimentos de predicción por conjuntos. Para demostrarlo, ejecutaron un experimento de predicción por conjuntos en 2 supercomputadores de la infraestructura TeraGrid. El principal inconveniente de este trabajo es que fue diseñado para entornos de ejecución controlados y de alto rendimiento, muy diferentes a las e-infraestructuras de computación *Grid*. Como se verá en el desarrollo de esta tesis, algunas de las características de las e-infraestructuras, como la alta tasa de fallos, el bajo ancho de banda entre los sitios, o los límites de utilización de los recursos y su heterogeneidad, dificultan enormemente la ejecución de simulaciones climáticas. Estos problemas no suelen encontrarse en infraestructuras *Grid* controladas como TeraGrid o XSEDE.

Bretherton et al. (2009) desarrollaron un *framework* basado en servicios web que podía ser utilizado para ejecutar diferentes modelos climáticos. Esta solución ofrecía características avanzadas como la abstracción que facilita la ejecución de diferentes modelos o el control del flujo de ejecución de los trabajos. Sin embargo, algunas cuestiones importantes, como la planificación de tareas, la gestión eficiente de datos o la recuperación de simulaciones fallidas, no se trataban.

Varios trabajos se han centrado en adaptar un modelo para realizar un experimento concreto en una infraestructura *Grid* concreta (Davidović et al., 2010; Lagouvardos et al., 2010; Sulis, 2009; Yalew et al., 2013; De Almeida et al., 2013). Aunque todos estos trabajos han sido ejecutados en una e-infraestructura, no resuelven los problemas relacionados con la heterogeneidad de recursos ni con la gestión de datos. En su lugar, proporcionan una solución *ad-hoc* que sólo funciona en un conjunto muy limitado de recursos. Este enfoque tiene dos desventajas: que la solución no es reutilizable para otros experimentos llevados a cabo con el mismo modelo y que no puede sobreponerse a los errores, con lo que muchas simulaciones no finalizan correctamente.

Durante los últimos años, varios proyectos, entre los que cabe destacar *SCIentific gateway Based User Support (SCI-BUS)*, se han centrado en promover el uso de portales web que faciliten a los investigadores la ejecución de sus flujos de trabajo en diferentes tipos de infraestructuras (entre los que se encuentran *clusters*, *Grid* y *Cloud*). El *middleware* más utilizado para ello es gUSE/WS-PGRADE (Balasko et al., 2013). Con él, se han realizado dos nuevas herramientas (Danovaro et al., 2014; Davidović et al., 2013) para la ejecución de modelos climáticos en infraestructuras distribuidas. Si bien, estas herramientas permiten la ejecución en diferentes tipos de

infraestructuras, al igual que las anteriores, sólo proporcionan soluciones *ad-hoc*, que no tienen capacidad para gestionar los problemas derivados de la heterogeneidad y distribución geográfica de los recursos.

1.4. Objetivos de la tesis

Tras esta introducción del estado del arte de los conceptos a tratar, en la presente sección se exponen de forma concreta los principales objetivos de esta tesis:

- Analizar los requerimientos de las simulaciones con modelos climáticos para su ejecución en infraestructuras heterogéneas y distribuidas (HDCI).
- Estudiar el flujo de ejecución de los experimentos que se llevan a cabo con modelos climáticos y diseñar una metodología que permita al investigador definir cualquier tipo de experimento y sus flujos de ejecución de forma sencilla.
- Diseñar la metodología necesaria para configurar y gestionar estos experimentos en HDCI de forma que podamos aprovechar todos los recursos computacionales a nuestro alcance de forma simultánea. Basándonos en dicha metodología, implementar una prueba de concepto de un *framework* que permita la ejecución de un modelo climático concreto en HDCI. Este *framework* ha de separar la configuración de los experimentos de su ejecución en los recursos y ha de ser capaz de ejecutar diferentes tipos de experimentos de forma transparente, eficiente y escalable.
- Mostrar a la comunidad de clima los beneficios y desventajas de utilizar e-infraestructuras *Grid* para la ejecución de experimentos con modelos climáticos.
- Establecer el procedimiento a seguir para adaptar otros modelos climáticos a este entorno de trabajo y cuantificar el esfuerzo que supone. Identificar los cambios que habría que implementar para que otras disciplinas científicas pudiesen aprovechar este entorno de trabajo para ejecutar sus aplicaciones.

CAPÍTULO 2

Publicaciones

2.1. Workflow management in the GRID for sensitivity studies of global climate simulations

Cita completa

Fernández-Quiruelas, V.; Fernández, J.; Baeza, C.; Cofiño, A.S.; Gutiérrez, J.M. Execution management in the GRID for sensitivity studies of global climate simulations. *Earth Science Informatics*. 2, pp. 75 - 82. 2009. doi:10.1007/s12145-008-0018-z

Resumen

En este primer artículo se muestra cómo la tecnología *Grid* actual se encuentra en un estado inmaduro y no está adaptada para la comunidad de Ciencias de la Tierra. La alta tasa de fallos de las infraestructuras *Grid* y los límites de tiempo de ejecución de los trabajos, requieren la creación de nuevo *middleware* capaz de manejar estos problemas. El artículo muestra la aplicación GRID-CAM, un gestor de flujo de trabajo para llevar a cabo experimentos de predicción por conjuntos con el modelo CAM en *Grid*. GRID-CAM cumple con los principales requerimientos que los modelos climáticos demandan para su ejecución en *Grid*: tolerancia a fallos, capacidad de reinicio de las simulaciones, monitorización y gestión de datos y metadatos. Con el fin de demostrar la capacidad de la aplicación GRID-CAM se ha realizado un experimento para analizar el fenómeno *El Niño* en la infraestructura *Grid* de EELA.

Earth Sci Inform (2009) 2:75–82
DOI 10.1007/s12145-008-0018-z

METHODOLOGY ARTICLE

Execution management in the GRID, for sensitivity studies of global climate simulations

V. Fernández-Quiruelas · J. Fernández · C. Baeza · A. S. Cofiño · J. M. Gutiérrez

Received: 17 September 2008 / Accepted: 18 December 2008 / Published online: 17 January 2009

Abstract Recent trends in climate modeling find in GRID computing a powerful way to achieve results by sharing geographically distributed computing and storage resources. In particular, ensemble prediction experiments are based on the generation of multiple model simulations to explore, statistically, the existing uncertainties in weather and climate forecast. In this paper, we present a GRID application consisting of a state-of-the-art climate model. The main goal of the application is to provide a tool that can be used by a climate researcher to run ensemble-based predictions on the GRID for sensitivity studies. One of the main duties of this tool is the management of a workflow involving long-term jobs and data management in a user-friendly way. In this paper we show that, due to weaknesses of current GRID middleware, this management is complex task. Those weaknesses made necessary the development of a robust workflow adapted to the requirements of the climate application. As an illustrative scientific challenge, the application is applied to study the El Niño phenomenon, by simulating an El Niño year with different forcing conditions and analyzing the precipitation response over south-American countries subject to flooding risk.

Communicated by: H. A. Babaie

V. Fernández-Quiruelas · J. Fernández · A. S. Cofiño (✉)
Department of Applied Mathematics and Computer Sciences,
Universidad de Cantabria,
Santander, Spain
e-mail: antonio.cofino@unican.es

C. Baeza
Center for Mathematical Modeling, Universidad de Chile,
Santiago, Chile

J. M. Gutiérrez
Instituto de Física de Cantabria, CSIC-UC,
Santander, Spain

Keywords CAM model · Climate models · El Niño phenomenon · GRID computing · Workflow management

Introduction

The EU-funded project EELA (E-Infrastructure shared between Europe and Latin America) aims at bringing the e-Infrastructures of Latin American countries to the level of those established in Europe, identifying and promoting a sustainable framework for e-Science (http://www.eu-eela.org). The present paper describes the new developments achieved as a result of porting a climate application to the GRID under the EELA framework with the goal of analyzing el Niño phenomenon, which is a key factor for Latin-American (LA) climate prediction. El Niño has a special interest due to its direct effect in the Pacific coast of South America and, in particular, in Peru and Chile (EELA LA partners).

For this reason, the climate applications in EELA were designed around this phenomenon with the main objective of developing a simulation and analysis tool especially useful for LA partners.

GRID technologies emerged in the 1990s as a way to share computer resources and other scientific equipment across geographically distributed locations in a user-transparent way (Foster and Kesselman 1999). By sharing computer resources it is meant not only to share their storage capacity, but also the computer power, which would be used to run applications. The user transparency relies on what is referred to as "middleware", a software layer between the applications and the GRID infrastructure.

A number of research and commercial projects have developed different middleware solutions and applications

(e.g. the EGEE project (www.eu-egee.org) is the reference in GRID development in Europe). New applications ported to the GRID demand new services which are not always available in the existing middleware.

In this paper, we present a new paradigmatic example on the area of climate simulation which demands solutions in terms of, e.g., job duration and workflow management.

We selected a Global Circulation Model as the first application to be ported to the GRID, since any further simulation or analysis step would require a global simulation as starting point. Particular features of the GCM posed specific problems for the GRID, such as experiments lasting beyond proxy certificates lifetime, control of jobs, etc. Using the existing middleware solutions we created a new application developing extra middleware to run the GCM in the GRID with a specific workflow, solving most of the problems encountered.

The paper is organized as follows. A brief overview of the climate models is given next, followed by a summary of the specific benefits obtained from the GRID and also the requirements posed on the GRID. Then, the GRID-CAM application is described introducing the existing middleware solutions used and the new developments performed to achieve the deployment of the climate model on the GRID. Finally, some scientific results achieved with GRID-CAM are presented.

Climate modeling

Dynamical climate models are mathematical models that numerically solve the nonlinear equations governing the atmosphere on a global lattice with horizontal resolutions ranging from 50 to 300 km, depending on the application. These models require a set of initial conditions (values of climate variables—wind, pressure, temperature, etc,—on a lattice of points at the starting time) to propagate the solution forward in time.

In order to analyze the atmospheric part of the global climate system, we selected the CAM model (Community Atmosphere Model), which is the latest in a series of global atmosphere models developed at NCAR for the weather and climate research communities (Collins et al. 2004). CAM can be run for short (hours, days) or long periods of time (decades, centuries) to investigate the present, past or future climate variability. A great percentage of the total variability (70–90%) of global climate is obtained when an atmospheric model is forced with the oceanic fluxes; say the sea surface temperature and ice cover. As a first step, we ported the CAM model, which is an atmosphere-only model and requires lower boundary conditions at the surface (sea surface temperature). This model enables a wide range of experiments.

The port of a fully coupled model (such as the CCSM model which comprises CAM as the atmospheric part) is left as a natural future work.

The model can be run either in parallel (using MPI) or as a single process. The single-process version has been deployed and run in the EELA testbed with T42 resolution: 128 (longitude)×64 (latitude) and 27 vertical levels, i.e. 221184 points per time step. The model produces 32 3D and 56 2D variables over the lattice. The simulation of a year takes approximately 48 hours of wall clock time in a 3 GHz Intel Pentium D processor, while 100 years would take around 7 months of wall clock time. The model produces 197 MB per time step, i.e. more than 720 GB per century. These figures increase when running the model at a more state-of-the-art resolution. T42 was state-of-the-art one decade ago and is only in use for very long simulations (centuries), but we used it for the development process.

The application we designed aims to perform sensitivity experiments by running an ensemble of CAM simulations with perturbed the sea surface temperatures as boundary conditions.

Benefits of GRID

Climate models are complicated computer programs which require large amounts of CPU power. Most of them are parallelized. However, the GRID cannot make the most of this kind of parallelism, since the latency across geographically distributed computers would render the program completely inefficient. Apart from computer parallelism, climate science is recently making use of a large number of simulations, referred to as an "ensemble", of the same phenomenon in order to assess the uncertainty inherent to the simulation (Hagedorn et al. 2005; Palmer 2002). Ensembles of simulations with varying parameters are also used for sensitivity experiments and many other applications. Each simulation in an ensemble is independent of the others and can be run asynchronously. This kind of parametric jobs is well suited for the GRID, since each simulation can be carried out in different nodes and the results are made available as a uniform data set in the Logical File Catalogue (LFC), ready to be analyzed.

Unlike volunteer computing projects, such as *climateprediction.net* (Allen 1999) where the GCM needs to be simplified and most of the results thrown away to avoid the overloading of the volunteer hosts, the GRID allows running a full state-of-the-art model and store the regular output information.

Requirements and workflow management

Nowadays, it is uncommon the use of GRID computing to run long-term jobs, due to the high rate of job failure and

the CPU-time limitations for the jobs on the local management system (typically only jobs lasting less than 48 h are allowed). These problems become critical for long simulations such as those performed with climate models and other Earth Science applications. Thus, unlike many other applications ported to GRID, earth science applications need to make use of advanced techniques in workflow management. In particular, the climate application described in this paper has the following requirements:

- Failure aware: Due to the nature of GRID there are several reasons which may cause job failures in the testbed, including heterogeneity of resources, CPU-time limited queues, etc.
- Checkpointing for restart: The complexity of the climate model runs may require jobs to be restarted in a different working node due, for instance, to the excessive duration of the job. Unlike other applications, it is not affordable to throw away an interrupted simulation.
- Monitoring: Since the climate simulations last for a long time, we need to know what is happening with the simulation once it has been sent to the testbed: whether the model is running or not, which time step is being calculated, which files have been uploaded to Storage Elements, which one is the last restarting point, etc.
- Data and Metadata storage: The goal of our application is the generation of output information that can be easily accessed by users, so data and metadata should be stored in an appropriate form.

The above requirements made necessary the development of a goal-oriented workflow manager in order to run the experiments with a minimum of human intervention. Therefore, we developed the GRID-CAM application which is a "GRID workflow management layer for CAM simulation".

The requirements of the climate model described in this section are only an example of the needs of many other Earth Science applications.

The GRID-CAM application

In this section we briefly introduce and define the different components involved in a typical climate simulation on the GRID. We define an *experiment* as an ensemble of simulations (parametric jobs) designed to answer some scientific question (a single execution is the simplest experiment); each of the ensemble executions is called a *realization* and requires a set of input data to run the model in the prescribed simulation period (typically several years). A particular type of experiment is that related to climate sensitivity studies. In this case, the different sets of input data are obtained from a single one including certain user-defined perturbations to form the ensemble (perturbed initial or boundary conditions, perturbed parameters, perturbed radiative forcing, etc.).

The lowest level component of our application is a *job*. This component matches with a standard GRID job and cannot be related one to one with a realization since realizations cannot be guaranteed to finish in a single job. In general, a realization requires several jobs to be completed, each one restarted from the previous one. As the job is running, the model generates information (files and metadata) that has to be available from every other component of the GRID: restart files (for failure recovering), current simulation time step, number of restarts, job id (for monitoring purposes), statistical information, output data, etc. Hereinafter, all the data and metadata generated by the models will be referred to as output information.

Figure 1 shows a scheme illustrating the relationships between the main components of the application.

Therefore, climate simulation on the GRID requires the management of a complex workflow formed by experiments composed of realizations split across jobs. This workflow is not trivially managed by the currently available GRID middleware, so a new layer is necessary for a proper execution of climate simulations.

Grid middleware used

gLite middleware

The gLite middleware is an integrated set of components designed to enable resource sharing in GRID (Burke et al. 2007).The core components of the gLite architecture are the following:

- User Interface (UI): It is the access point to the GRID.
- Computer Element (CE): A set of computing resources localized at a site (i.e. a cluster, a computing farm).
- Worker node (WN): The cluster nodes where the jobs are run.
- Storage Element (SE): Separate service dedicated to store files.

The Logical file catalog (LFC) is a GRID secure catalog containing logical to physical file mappings. The primary function of the LFC is to provide central registration of data files distributed amongst the various Storage Elements. On the other hand, AMGA (Koblitz et al. 2007) is the gLite Metadata Catalogue, and we use it just as a classical GRID-enabled database where we store all the data and metadata information required by the application and the user.

We also used GridWay (http://www.gridway.org/), which is a GRID meta-scheduler which provides a scheduling

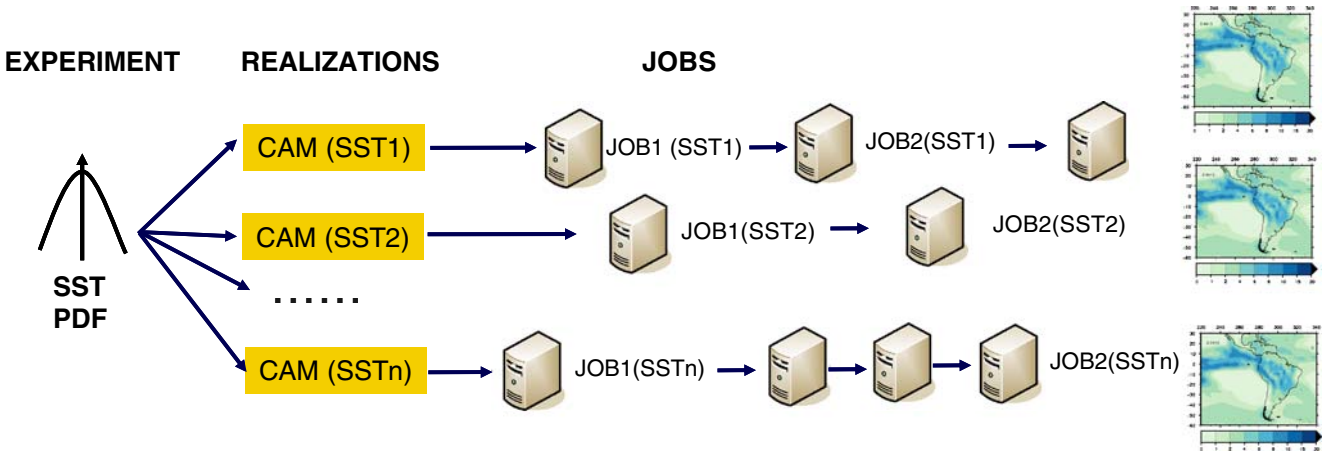


Fig. 1 GRID-CAM Application main components. In order to run an experiment (ensemble of simulations) we will create several realizations that will be simulated by several jobs in cascade

framework similar to that found on local Resource Management systems, supporting resource accounting, fault detection and recovery and the definition of state-of-the-art scheduling policies.

In addition to these existing middleware products, some GRID developments were necessary in order to deploy the climate application and to develop the appropriate workflow elements. These new components are described in the following sections.

The Grid Enabling Layer (GEL)

Climate models are mature applications with thousands of lines of code (usually Fortran). We introduced small modifications to the code to perform system calls to specific applications which are in charge of interacting with the GRID on behalf of the climate model. To this aim, we developed a new software layer, referred to as GRID Enabling Layer (GEL), which provides the model with the ability to interact with the GRID. The slightly modified source code of the model plus its GEL conform a fully featured GRID application. Since climate models are developed by external institutions, this approach is the best suited to keep up with the most recent updates with the least effort: only the small modifications to interact with the GEL need to be introduced at key points of any new release.

The GEL provides the following capabilities:

- Realization monitoring: Since our simulations last for a long time, we need to know their status once they have been sent to the testbed: If the model is preparing the WN or running, which step of time is calculating, which files has uploaded to SE-LFC, which is the last restart point, etc. This is analyzed in detail in the next section.
- Restart management: Each time CAM finishes simulating a time-step, the GEL uploads the restart files to the nearest SE and registers them in the LFC. It also publishes the restart field associated to this experiment in the AMGA database. This way, if the job fails and the realization is rescheduled to another WN, it will continue calculating from this time-step on.
- Data and Metadata management: In order to store all the output and restart information generated by the model, we need that the events and files are permanently registered in a place accessible from any component of the GRID (AMGA and LFC-SE).

The above issues were solved by introducing Fortran system calls at 4 specific points of the CAM source code. These calls execute the GEL scripts which carry out the previously mentioned tasks. The slight modifications to the Fortran source code also allow us use fast-development programming languages for the GEL scripts (we used a mixture of shell, Perl and Python).

Workflow management layer (WML)

As we pointed out in "Requirements and workflow management", there are several reasons which may break the flow of the job. In fact, during the first stages of the porting of the application, we did not account for all these possible errors. The result was that very few of our simulations finished.

To better understand the WML, we describe next the steps a GRID job goes through in our system since a user sends it to the scheduler until it is finished.

When a user submits a job from the UI, the scheduler sends it to a CE. Once there, the job takes the pending status while the CE finds a WN for the job in the local site.

If everything goes fine, the job starts its execution in the WN and finishes with the SUC final state (see Fig. 2).

Unfortunately, there are several situations where the job does not reach this status. In order to create a robust workflow, we identified the failure points that were not managed by the gLite middleware and we designed a workflow able to overcome the following situations:

- Due to misconfigurations in local sites, the CE is publishing free WNs when, in fact, there are not. Then, the job can be in pending status in the CE indefinitely. To overcome this, if a job remains for a long time (currently set to more than 5 min) in pending status in the CE it passes to an error state (SUSP) and the job will be rescheduled.
- When a job is running in the WN, it can crash as normal executions crash in a cluster (Core dumps, power failures, …). In these cases, the job is finished with the ERROR state and rescheduled to another CE.
- Even if the job is in active state in the CE (it is supposed to be running in the WN), sometimes the WN is not executing it. If after being in active state in the CE for more than 5 min, the job has not been started, it will be passed to the MID.ERR state and rescheduled.
- When the WNs are not well configured (e.g. cannot access the LFC, SE or AMGA), the job is passed to MID.ERR state and rescheduled to a different site.
- Some clusters do not distribute the load in a proper way among the WNs. When a job detects that the CPU assigned to it is less than a given threshold (currently set to 30%), it is passed to the PERF state and is rescheduled.
- The local queue kills our job when it has reached the maximum allowed time in the cluster (usually 48 hours). In this situation, the job takes the WALL (local queue walltime) status and is rescheduled.

All these final status (SUC, SUSP, ERR, MID.ERR, PERF, WALL) are stored in the AMGA database for each job for further analysis.

After analyzing several job managers, we found that GridWay meta-scheduler was the one that best fulfilled our requirements, since GridWay detects job failures for all of the problems mentioned before, and it is able to reschedule the failed jobs to another cluster. Moreover, once the re-scheduled job starts to run in the WN, a component of the WML developed queries the AMGA database to find the latest restart files for this realization in order to continue the simulation started by the previous job. We have also adopted an additional monitoring feature provided by GridWay: while the job is running in the WN, a monitor script (running also in the WN) checks the status of the job. This monitor can copy the output and error files of our job to the UI with a given frequency. In this way, from the UI we can accurately determine any failure in each of our realizations. When an ensemble of simulations is sent to the GRID, each realization of the ensemble is converted to a GridWay job that is sent to the scheduler. When GridWay receives the jobs, it searches the WN better suited to our application needs and chooses the best among them. To do so, it uses a powerful scheduling policy that takes into account the user requested requirements (memory, CPU, etc.) and a heuristic scheduling based on the jobs sent in the past. For instance, if all jobs sent to a CE failed, GridWay will not try to send jobs to that site for a long period.

Finally, in order to manage the issue of the expiration of the proxy, which affects every job lasting longer than 48 h, we used the *myproxy* credential management system as a provisional solution.

Monitoring with AMGA

The AMGA database has two different tasks in the application. On one hand, it is used to store the information generated by the experiments executed in the GRID. On the other hand, it is used for monitoring purposes, storing all the status information about each of the jobs as metadata information. The tables and relationships used by GRID-CAM are shown in the Fig. 3.

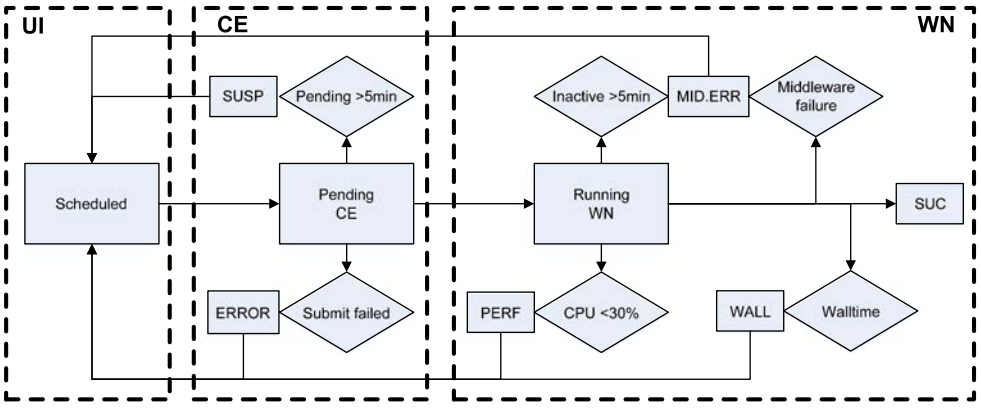


Fig. 2 Flow chart with conditions (*diamonds*) for possible states (*squares*) and state-transitions (*arrows*) of a job. The elements are grouped by GRID component (*dashed squares*)

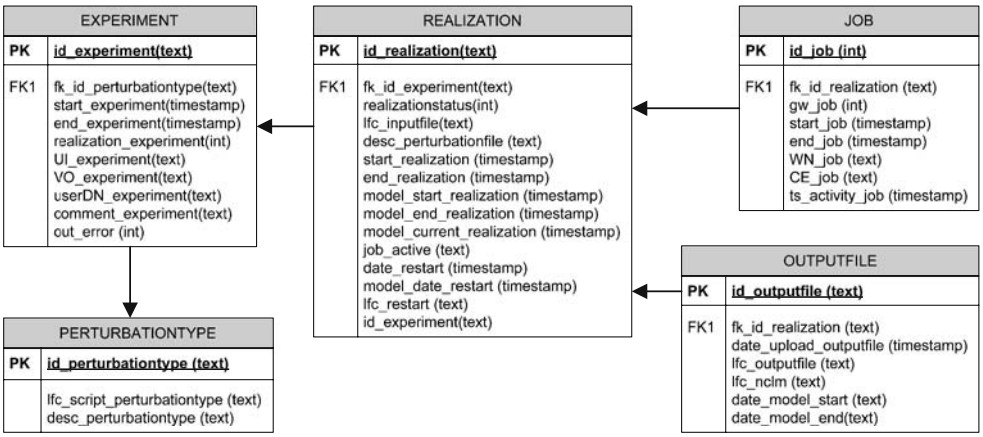


Fig. 3 Structure of the AMGA database used to store metadata and status information for the GRID-CAM application

Some of them are also relevant to the workflow, as described below:

- Experiment: When preparing the experiment, this table is filled with the type of perturbation used (multiplicative, random, etc), the number of realizations and a description and simulation dates of start and end.
- Realization: Each realization can be executed in many different nodes. This table keeps track of current time step, restart files, id of the current job executing the realization, etc.
- Job: This table is used to keep track of the different jobs used in an experiment. It stores the timing information, the WN and the realization it contributed to. Most of this information is stored for statistical purposes.
- Outputfile: Each realization generates a number of files as it runs. This table stores metadata and access information for the files stored in the catalog. This speeds up the data discovery process.

Scientific experiment: design and results

In our current status, the application allows us to run a variety of scientific experiments using CAM. As a first example, we ran an experiment of 50 CAM realizations with boundary conditions (sea surface temperature) as observed during the period from January 1997 until March 1998 (15 months). This period includes the strongest El Niño event observed to date. We investigated the sensitivity of the precipitation over south-American countries to modifications of the sea surface temperature (SST) by means of an ensemble of CAM realizations.

To create the ensemble, we perturbed the observed SST by adding a given spatial pattern scaled by a random number. The spatial pattern used to perturb the SST is shown in Fig. 4. It is the SST anomaly of the last two strongest El Niño events with respect to the climatological mean. For the 1997 event, this anomaly reached a maximum value of 2.5 K. We normalized this anomalous pattern by dividing by this maximum value. Then, we selected random scale factors in the range from −2.5 (normal conditions, since the perturbation is opposite and of similar intensity to the anomaly generated by El Niño) to 2.5 (double anomaly than in the 1997 event). A zero value, mean a forcing similar to that observed during 1997 year.

The initial conditions for the experiment correspond to a previous CAM run which started from climatological conditions on 1st January 1990, but was forced by the observed SST for one decade. The atmospheric and soil conditions on the 1st January 1997 from this run were used as initial conditions for the perturbed SST experiment.

The 10-year simulation showed a slight but clear precipitation response to El Niño conditions. However, the perturbed runs from our Grid experiment did not clearly

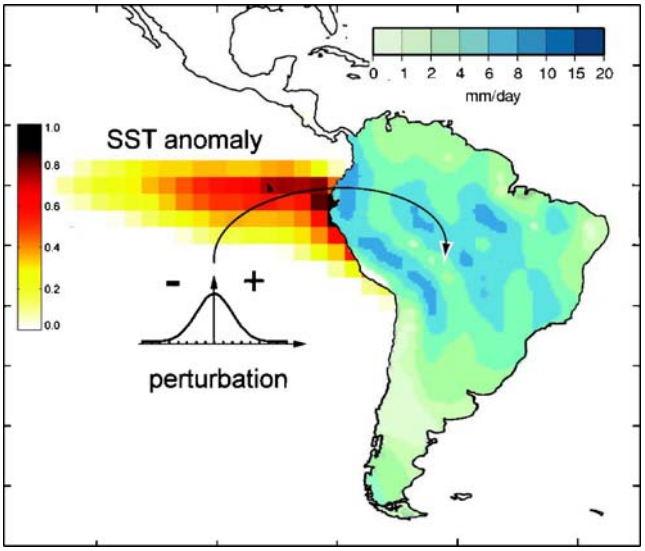


Fig. 4 Scheme of the sensitivity experiment. The normalized pattern over the sea is scaled with a random number and added as a perturbation to the observed SST. The resulting SST is used as lower boundary condition for a CAM realization, giving rise to a precipitation distribution over land

respond to higher than observed SST anomalies (Fig. 5a). Figure 5 represents the mean precipitation for the period October through January, when the most intense anomalies are registered. When the precipitation at a single sensitive point is analyzed against the scale of the perturbation applied (Fig. 5a), the response to increased El Niño conditions is clear until a value 0 of the perturbation scale (Niño'97 conditions) but seems to remain stable to stronger perturbations. These can also be seen in Fig. 5b and c where the whole South American distribution of precipitation is presented for a Niño'97-like realization and a stronger (double) anomaly realization. These are preliminary results which still need to be analyzed in a more systematic manner, using a larger number of simulations for statistical significance, and probably require the use of a regional model to generate more reasonably the precipitation response over mountainous areas.

Technical results

The scientific results showed on the previous section were obtained using the EELA testbed during 1 week. The testbed was composed by seven sites distributed in Europe and Latin America making a total of 150 CPUs available.

After one week, 41 realizations concluded successfully, nine were still running. In order to complete all the experiments 510 independent jobs were necessary. In the next table we can see the output status (listed in a previous section) of the 510 independent jobs submitted to complete the experiment:

- 143 SUC and WALL
- 158 ERR
- 87 SUSP
- 41 PERF
- 81 MID.ERR

The big amount of ERR status (error) was caused by the misconfiguration of two sites where the all the jobs failed systematically. The amount of SUSP status is also higher than expected because during the experiment some sites didn't accept jobs because there were a lot of jobs waiting in their queues.

Another source of errors, mainly MID.ERR, is because some of the middleware services are overseas (Europe and Latin America). After testing some other testbeds the previous results can be considered as an example of the performance of the GRID.

Conclusions and future work

We presented the successful port of an atmospheric Global Climate Model to the GRID by using existing middleware solutions plus newly developed tools (Grid Enabling Layer and Workflow Management Layer) to account for specific requirements posed by this application. In doing so, several

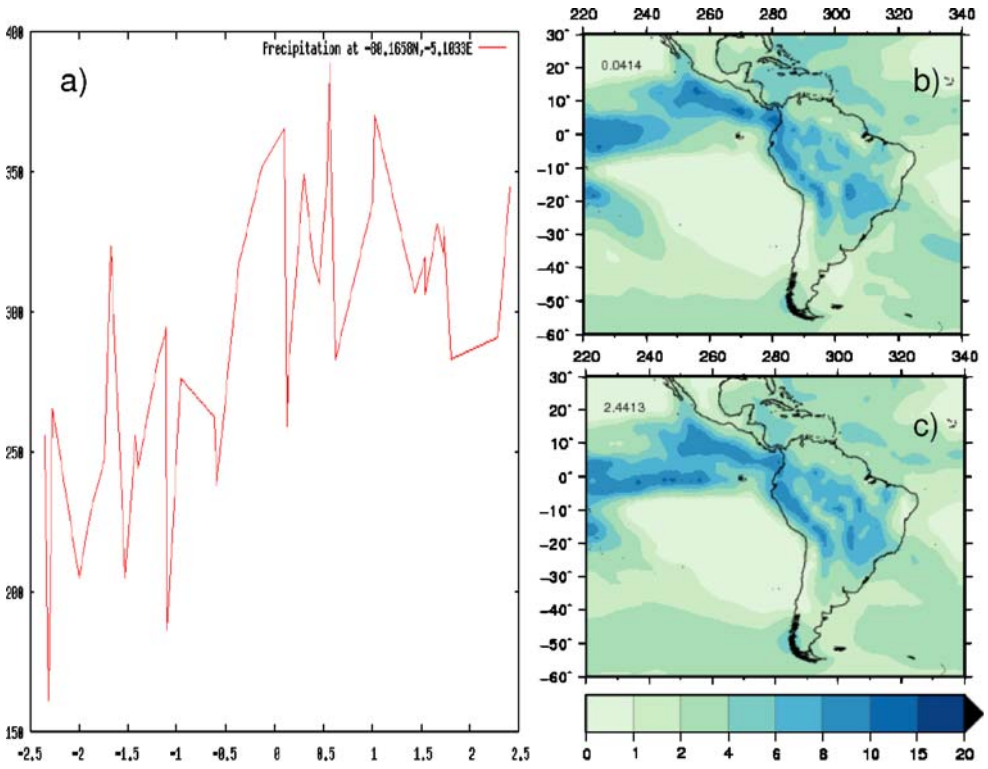


Fig. 5 **a** Accumulated precipitation interpolated to the location of Piura vs. the perturbation scale applied. **b**, **c** Mean precipitation (mm/day) for the period October 1997 through January 1998. The random scale factor used is shown on the *upper left corner*

weaknesses of current middleware were identified. We showed how current GRID technology is immature and not completely well suited for the Earth Science community. Specifically, for climate models, it has several weaknesses such as a high failure rate and time simulation limits, which require the creation of new middleware able to handle these problems.

In spite of these weaknesses, there are some fields in earth science, including recent trends in ensemble forecasting or any other involving independent simulations, which could greatly benefit from the GRID.

In our case, the achievements made thanks to the GRID-enabled application could not have been reached with the means usually at out hand. Unlike local clusters or supercomputers, optimized for MPI and OpenMP jobs, the GRID is perfectly suited for independent simulations. In our case, it would have been impossible accessing a cluster with the number of CPUs available in the GRID. An even more important topic is the storage capacity; climate models manage huge files that need a lot of storage capacity. Thus, even though many jobs failed and needed to be rescheduled, the benefits from accessing a GRID composed of a large number of processors and disk space outweigh the drawbacks. We completed around 50 simulated years in 1 week.

The next steps in our work are porting the regional WRF model to the GRID and create a new application combining CAM and WRF. The idea is that the users specify a region to study and then run a cascade of CAM and WRF simulations for this region. In order to do this, improvements to the workflow will be necessary to manage the dependencies between the CAM and WRF jobs.

As a final remark, although the effort of developing an application in GRID is very high, the results are worth. Once the initial effort is done, many other applications with similar requirements can be much easily ported. This successful experience is promising and we will continue working in this line.

Acknowledgements This work has been partially funded by the EELA project under the 6th Framework Program of the European Commission (contract no. 026409) and the Spanish Ministry of Education and Science through the Juan de la Cierva program.

References

Allen M (1999) Do it yourself climate prediction. Nature 401:642

Burke S, Campana S, Delgado Peris A, Donno F, Méndez Lorenzo P, Santinelli R, Sciabà A (2007) gLite 3.0 users guide. https://edms.cern.ch/file/722398/1.1/gLite-3-UserGuide.pdf

Collins WD, Rasch PJ et al (2004) Description of the NCAR Comunity Atmospheric Model (CAM 3.0). Technical Report NCAR/TN-464+STR, National Center for Atmospheric Research. http://www.ccsm.ucar.edu/models/atm-cam/docs/description/description.pdf

Foster I, Kesselman C (1999) The grid. Blueprint for a new computing infrastructure. Kaufmann, San Francisco

Hagedorn R, Doblas-Reyes FJ, Palmer T (2005) The rationale behind the success of multi-model ensembles in seasonal forecasting—I. Basic concept. Tellus A 57:219–233

Koblitz B, Santos N, Pose V (2007) The AMGA metadata service. J Grid Comput 6:61–76

Palmer TN (2002) The economic value of ensemble forecasts as a tool for risk assessment: from days to decades. Quart J Royal Meteor Soc 128:747–774

2.2. Benefits and Requirements of Grid Computing for Climate Applications. An Example with the Community Atmospheric Model

Cita completa

Fernández-Quiruelas, V.; Fernández, J.; Cofiño, A.S.; Fita, L.; Gutiérrez, J.M. Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model. *Environmental Modelling & Software*. 26 - 9, pp. 1057 - 1069. 2011. doi:10.1016/j.envsoft.2011.03.006

Resumen

Este artículo, orientado a los investigadores que ejecutan los modelos climáticos en infraestructuras tipo *cluster*, presenta los beneficios de la utilización de infraestructuras *Grid* para llevar a cabo experimentos de predicción por conjuntos con modelos climáticos. Para ello, se describen los componentes básicos del *Grid* y se muestran las infraestructuras *Grid* actuales. También se hace una revisión del trabajo de la comunidad climática en *Grid* y un análisis de los requerimientos de los modelos. A continuación, se describe CAM4G, un nuevo *middleware* utilizado para ejecutar el modelo CAM en *Grid*. CAM4G es una versión mejorada de la aplicación GRID-CAM presentada en el artículo anterior, cuyos componentes han sido modificados siguiendo criterios de robustez y escalabilidad. Para demostrar el rendimiento de CAM4G, se ha realizado un experimento de predicción por conjuntos compuesto de 750 simulaciones en la infraestructura *Grid* de EGEE.

Environmental Modelling & Software 26 (2011) 1057–1069

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Featured Article

Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model

V. Fernández-Quiruelas [a,*], J. Fernández [a], A.S. Cofiño [a], L. Fita [a], J.M. Gutiérrez [b]

[a] Dpto. Matemática Aplicada y C.C. Universidad de Cantabria, Santander, Spain
[b] Instituto de Física de Cantabria, CSIC-UC, Santander, Spain

ARTICLE INFO

Article history:
Received 10 June 2010
Received in revised form
2 March 2011
Accepted 18 March 2011
Available online 17 April 2011

Keywords:
Grid computing
Community Atmospheric Model (CAM)
El Niño
Sensitivity analysis
Workflow management

ABSTRACT

Grid computing is nowadays an established technology in fields such as High Energy Physics and Biomedicine, offering an alternative to traditional HPC for several problems; however, it is still an emerging discipline for the climate community and only a few climate applications have been adapted to the Grid to solve particular problems. In this paper we present an up-to-date description of the advantages and limitations of the Grid for climate applications (in particular global circulation models), analyzing the requirements and the new challenges posed to the Grid. In particular, we focus on production-like problems such as sensitivity analysis or ensemble prediction, where a single model is run several times with different parameters, forcing and/or initial conditions. As an illustrative example, we consider the Community Atmospheric Model (CAM) and analyze the advantages and shortcomings of the Grid to perform a sensitivity study of precipitation with SST perturbations in El Niño area, reporting the results obtained with traditional (local cluster) and Grid infrastructures. We conclude that new specific middleware (execution workflow managers) is needed to meet the particular requirements of climate applications (long simulations, checkpointing, etc.). This requires the side-by-side collaboration of IT and climate groups to deploy fully ported applications, such as the CAM for Grid (CAM4G) introduced in this paper.

1. Introduction

Earth Science (ES) applications – in particular weather and climate models – are among the most computer power and storage demanding disciplines; thus, they are key users of High Performance Computing (HPC) infrastructures, favoring their continuous growth and improvement. For instance, ES-dedicated supercomputers such as the Earth Simulator (www.es.jamstec.go.jp) rank at the top of the list of the world's most powerful computers.[1] However, during the last two decades new computing paradigms have emerged, such as Grid computing (Foster and Kesselman, 1999) and volunteer computing (Anderson and Fedak, 2006). They provide an alternative to HPC for different problems facilitating the access to high capacity production-quality computing infrastructures to small groups or institutions.

Grid computing consists of a geographically distributed infrastructure gathering computer resources around the world in a transparent way (Foster and Kesselman, 1999). Unlike volunteer computing projects, such as climate-prediction.net (Allen, 1999), where the applications (a global climate model in this case) need to be simplified and most of the results thrown away to avoid the overloading of the volunteer hosts, the Grid allows running a full state-of-the-art model and store the regular output information. This is done through a software layer, referred to as middleware, which allows for the transparent use of the distributed computing and storage resources which are seen as a single infrastructure. Thus, the most complex tasks of the Grid (security, authentication, resource discovery and allocation, storage, job execution) are managed by the middleware built on top of the infrastructure providing a simple and transparent interface for users.

In the last two decades, a number of computer-demanding applications in fields such as High Energy Physics (HEP) and Biomedicine have migrated toward Grid technologies as a complementary way to fulfill their increasing CPU power and storage requirements. Most of the problems and applications in these fields correspond to the so-called *production tasks*, where a single application is run many times with different parameters and/or input files. In those cases, parallel capabilities are used for the different realizations of a serial application, instead of the parallel execution

* Corresponding author.
[1] The Earth Simulator ranked first of the world since its creation in 2002 until 2004. Moreover, computers at different national weather services can often be found at the top 10; see www.top500.org.

1364-8152/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2011.03.006

of a single application. Many challenges have been achieved using Grid infrastructures to run production tasks; (see, e.g. Jacq et al., 2008) in Biomedicine or the LHCb computing data challenge (Nandakumar et al., 2008) in HEP. Although the Grid was initially though for both production and parallel tasks, nowadays parallel execution is still dependent on the specific Grid infrastructure. This makes the process of migrating a parallel application to the Grid harder than migrating a serial one.

The ES Grid community, unlike the above mentioned fields, has been mainly concerned with data access and management. There are efforts aiming to develop Grid services for transparent discovery and access to heterogeneous data such as satellite data, model simulations or observations (see Cossu et al., 2010, and the documents of the DEGREE project www.degree-eu.org). However, less effort has been devoted to the deployment and execution of applications such as a global climate model either for parallel or production tasks. Note that although the main need of a climate science user would be the parallel execution of a climate model, modern problems that involve large amounts of independent simulations such as ensemble prediction (Palmer, 2002) and sensitivity analysis experiments (Murphy et al., 2004; Board, 2009) correspond to *production* tasks appropriate to be deployed and run in Grid infrastructures. These problems have received increasing attention in the last decades due to their connections with the study of uncertainties, such as those related to seasonal prediction or climate change and its impacts on the different socio-economic sectors (Parry et al., 2007).

In this paper we give an up-to-date and user-oriented view of the Grid for the Climate community where the different applications have common needs. As an illustrative application, we describe an experiment with the popular Community Atmospheric Model (CAM; Collins et al., 2004, 2006a) to test the sensitivity of the precipitation simulated in South America to sea surface temperature variations over areas affected by the El Niño phenomenon. As shown in Fernández-Quiruelas et al. (2009a), unlike other areas of research, the particular characteristics and requirements of climate applications become a challenge for actual Grid middlewares, posing new problems to the Grid: long execution times, multiple jobs with complex interdependencies, huge input files, etc. These particular applications need to be managed in terms of ad hoc implementations of execution workflow frameworks, building on the available middleware. For instance, in this paper we describe CAM for Grid (CAM4G), a port of CAM to the Grid including an execution workflow manager implemented using existing middleware services to organize and manage the execution of the climate model. This workflow manager has been devised as a generic wrapper that allows the execution of this climate model in a Grid infrastructure without modifying its source code. This paper extends the capabilities of the prototype port of the CAM model to the Grid presented in Fernández-Quiruelas et al. (2009a) and provides a successful proof-of-concept experiment solving the problems they identified.

This paper is structured as follows. Section 2 describes the Grid including its main components, different solutions and the most important infrastructures available. It is intended for a potential user from the climate community and only covers the most basic concepts from the user's point of view. Section 3 provides an overview of both the benefits of the Grid for the climate community and the special requirements that a climate application poses on the existing Grid solutions. As an example, Section 4 describes CAM for Grid (CAM4G), a port of CAM to the Grid solving the special requirements of the climate model on the Grid. Finally, Section 5 presents a sample experiment using CAM4G to perform a sensitivity test consisting of 750 simulations successfully run on the Grid and summarizes the statistics of the execution in the Grid environment compared to the execution on local resources.

2. The Grid technology

Grid computing has recently emerged (Foster and Kesselman, 1999) as an alternative for flexible and secure access to heterogeneous and geographically distributed resources (computing clusters, storage units, etc.). Thus, for instance, in order to create a collaborative virtual community, several institutions that collaborate in a project with different resources (a computing cluster, storage units or databases) could agree to share them, granting access to users from other institutions. The way of optimizing these synergies could be the creation of a Grid infrastructure that aggregates all the resources allowing the users to transparently access to a macro-system composed by all the processors and storage units of all the associated centers.

The analogy for this infrastructure is the power grid, where users plug their equipment obtaining energy in a transparent form, regardless of where and how it is produced.

This approach has several advantages for the users:

- Users take advantage of resources not fully used. In some institutions, clusters are used just a few hours per day or during some months in the year. Sharing the resources among several institutions will improve the usage capacity of the system. Institutions that have access to Grid will not have to be sized on peak load but can cleverly share the burden.
- Users are provided access to an enormous amount of storage space and computing resources difficult to reach by a single institution. This allows the research community to face new challenges that could not be achieved with traditional computing paradigms.
- Accessing geographically distributed heterogeneous resources in a homogeneous way make it easier for the user working with data or computing resources of other institutions. As we will see in section 2.1, Grid technology provides security mechanisms that manage the access to shared resources. System administrators find Grid technology helpful because they can rely on its security mechanisms to grant access to users. On the other hand, users can discover and access a vast amount of data sets distributed in several locations as if they were stored on a single computer.

2.1. Main components of the Grid

In this paper we describe the Grid from a user's point of view. Technical details about Grid can be found in Foster and Kesselman (1999). A typical user from ES is accustomed to local cluster environments, where all resources are homogeneous and access to them is done through a unique account. In a Grid environment, each resource has its own users and may have different policies and systems. In order to provide the users transparent access to these distributed resources, Grid technology uses some services called middleware, that aggregates heterogeneous resources and present them as a single homogeneous system.

The most important part of Grid middleware is the core services, in charge of centralizing the management of all the resources (see Fig. 1). There are two basic services, authentication and authorization (AUTH) and information (manages resource characteristics and status, INFO). These basic services are used by other core services in charge of centralizing the access to each kind of resource or service (e.g. the data and execution services, labeled as DATA and EXEC, respectively, in Fig. 1). These resource-specific core services rely on the authentication and information services to make their decisions. In order to communicate with the core services some middleware has to be installed in the resources. Finally, some

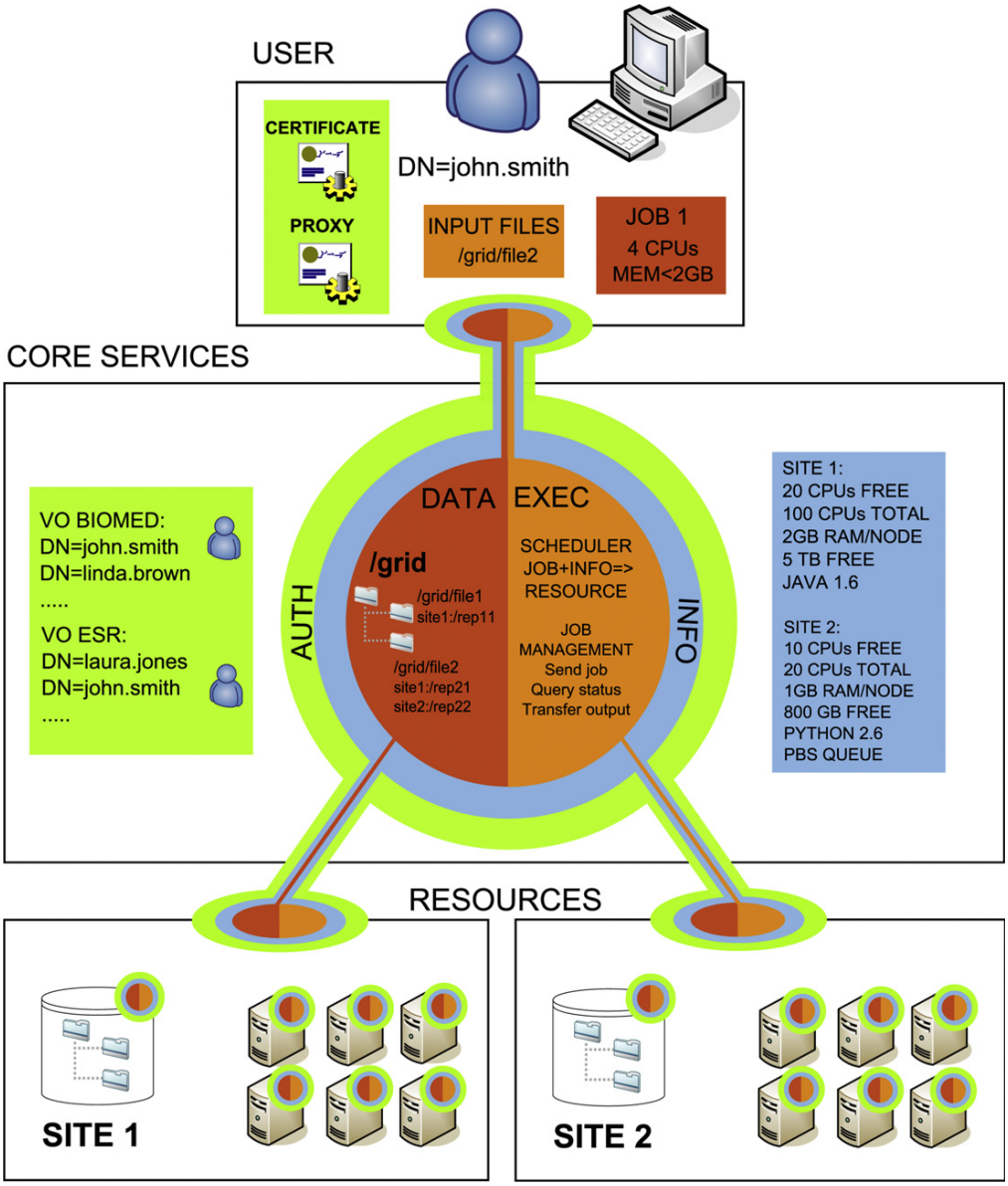


Fig. 1. Schematic representation of the Grid. There are 3 main layers: resources, core services and user environment. All of them make use of a given middleware in order to communicate with the others (see Section 2.1).

middleware user tools need to be installed in the user interface, in order to access the Grid services and infrastructures (as schematically depicted in Fig. 1).

One of the main differences between working in Grid and in a traditional computing system is the authentication method. Grid users have a personal *certificate* instead of the traditional *user name* and *password*. This aspect of Grid often constitutes the task most difficult to understand by a non-experimented user, but because of it, all processes developed within a Grid infrastructure are highly secured. Personal *certificates* are X509 certificates (Tuecke et al., 2004) signed by Certification Authorities (CA) that have previously checked that the user belongs to the institution he claims to be part of. This certificate is password protected to ensure that only the owner of the certificate can use it to access the resources. To avoid typing the password every time the user carries out a transaction, a time-limited proxy – which is a self-signed copy of the certificate (Welch et al., 2004) – is used automatically in all the transactions for a limited time period. This security infrastructure is known as Grid Security infrastructure (GSI; Foster et al., 1998).

Grid users are organized in so-called Virtual Organizations (VO; Foster et al., 2001), where they register their *certificates*. A VO is just an entity that maintains a list with the *certificates* of all the users that belong to it along with their roles and groups. The VO is queried by the resources in order to determine if a user can access it or not. Usually, VO members share something in common (work in the same project, organization or research topic) regardless their physical location. In large Grid infrastructures, where there are many VOs and institutions, not all the resources are shared among all the VOs (e.g. a meteorological center may only share its resources among the Earth Science VO). In several cases, such as when confidential data sets are shared, other VO features such as groups and roles may be used for fine-grained access to the resources.

From the user's point of view, the job submission to a Grid infrastructure works the same as in a local cluster or a supercomputer: the user fills a template with the job requirements and the executable to be run and submits the job to a queue using the middleware user tools. The storage and access to data is done through a virtual file system which maintains a relationship between the logical names in a virtual structure and the sites where the data are stored (multiple copies). This way, the data is replicated and distributed through the different sites and the Grid middleware selects the particular copy to be used for a particular execution according to, for instance, proximity to the execution node. Furthermore, to avoid data loss and to improve efficiency, the virtual file system can automatically manage replicas of the files. The user can transfer/download files to/from the Grid though GridFTP, a new protocol based on FTP (Postel and Reynolds, 1985) and GSI (Allcock et al., 2005) for the Grid.

The technical requirements to take advantage of a Grid infrastructure depend on the level of involvement. There are at least 3 levels of involvement in a Grid infrastructure:

1. The minimum requirement for a new user to start using a Grid infrastructure is to have a personal certificate and join a VO. If the infrastructure provides (traditional) access to a user interface (a machine with the user middleware installed) this would be enough to start using the infrastructure. Otherwise, the user must install the user middleware and configure it to use this infrastructure (see Fig. 1, top).
2. If an institution wanted to share their resources in the Grid infrastructure, they would have to install the resource middleware in their resources and configure them to interact with the core services of the infrastructure (see Fig. 1, bottom).
3. If the institution wanted to create a new Grid infrastructure (e.g. joining all the resources from all the departments), in addition to installing the user and resource middleware, they would have to install the core services in charge of giving transparent access to the resources (see Fig. 1, middle).

New users interested in using Grid resources may start by contacting the national Grid initiative of their respective countries.

2.2. Middleware implementations

Nowadays, there are several Grid middleware implementations that provide seamless access to distributed resources.

The first Grid middleware, Globus Toolkit (www.globus.org; Foster, 2006), was developed in the 90's in the United States and it is currently one of the most used implementations among the academia and industry. A middleware based on Globus Toolkit, gLite (glite.web.cern.ch), was created under the scope of the EGEE project in Europe (Enabling Grids for E-sciencE, www.eu-egee.eu). It is the middleware used in most of the European Grid initiatives. The application workflow presented in this study has been deployed using gLite.

gLite defines middleware packages or roles for each service. It provides 4 different roles for the core services. The Berkeley Database Information Index (BDII) is the information core service, Virtual Organizations Management system (VOMS) is the authorisation service and the Large Hadron Collider Grid File Catalog (LFC) and Workload Management System (WMS) are the data and execution core services respectively. The users interact with them through a computer where the User Interface (UI) role has been installed. Users can install their own UI (usually UIs can be downloaded as a virtual machine) or access the UI of the infrastructure. Each institution can join its computing cluster to a Grid infrastructure by installing the Computing Element (CE) and Worker Node (WN) roles in the head (the single point of management and job scheduling for the cluster) and computing nodes of their cluster respectively. Note that in order to ensure an easy installation and configuration, gLite only supports certain platforms and Operating Systems for each role. Currently, the WN middleware can only be installed on x86_64 computing nodes with the Scientific Linux 5 or Debian 4. In order to interface with the local storage system the Storage Element (SE) role can be used.

There are many other special-purpose middleware implementations, such as UNICORE (www.unicore.eu), which was initially developed to join German supercomputing centers.

2.3. Grid infrastructures

Although Grid middleware can be used in several scenarios to join different resources (in some cases just 2 or 3), in this paper we focus on large heterogeneous Grid infrastructures that join several institutions geographically distributed.

The largest Grid infrastructure in the world is the one created under the European project Enabling Grids for E-sciencE (EGEE, www.eu-egee.eu). It started in 2004 with the goal of aggregating as many as possible computing and storage resources from different organizations in order to face the challenge of storing and analyzing the data produced by the CERN's Large Hadron Collider (LHC). Nowadays, it aggregates 150.000 processors and 41 PB of storage distributed in 260 sites all over the world using the gLite middleware. The use of the EGEE infrastructure is not only limited to the HEP community. Today, there are thousands of users distributed in more than 200 VOs that comprise several disciplines (Biomedicine, Earth Sciences, Astrophysics, etc …).

As EGEE, other EU-funded projects have aggregated European resources within Latin America (EELA projects, www.eu-eela.eu), Asia (EUAsiaGrid project, www.euasiagrid.org), South Eastern Europe (SEE-Grid, www.see-grid.org), etc.

Apart from EGEE, that joins commodity data and execution resources, there are other large infrastructures more focused on joining supercomputing centers. For instance, DEISA (Distributed European Infrastructure for Supercomputing Applications, www.deisa.eu) puts together 11 of the most important supercomputing centers in Europe using the UNICORE middleware. As DEISA, TeraGrid (www.teragrid.org; Catlett, 2002) interconnects 11 American institutions using high performance networks and has, nowadays, a computing capacity over 1 PetaFlop and 30 PB of storage.

With respect to the climate science community the most representative infrastructure has been the Earth System Grid [ESG 44]. ESG is focused on facilitating the access to more data for climate scientists. This data comprise more than 200 TB of climate data and is distributed to more than 10,000 users registered in the ESG portal.

3. Grid for the climate modeling community

Climate science community already benefits from technologies like the Web and is starting to benefit from the Grid to manage the increasing amount of data produced. For instance, Web services were rapidly adopted and nowadays provide data from many international climate initiatives. Successful examples are ESA G-POD (Fusco et al., 2008) and ESG (Williams et al., 2009, earthsystemgrid.org). Renard et al. (2009) and Cossu et al. (2010) offer recent reviews mainly focused on data. However, the use of Grid infrastructures to perform large experiments that make intensive use of the computer power is in a more incipient status. Only a few efforts have been reported to adopt the Grid technology to execute applications (Lagouvardos et al., 2010; Mineter et al., 2003; Sulis, 2009). An updated overview of this problem has

been analyzed in the DEGREE project (Dissemination and Exploitation of GRids in Earth sciencE, www.eu-degree.eu).

The computer power and storage provided by a huge Grid infrastructure such as EGEE allows the climate science community to face new challenges. This is particularly important for emerging countries (e.g. in South America and Asia) which could easily use the existing Grid infrastructures, such as those of EELA (www.eu-eela.eu) and EUAsiaGrid (www.euasiagrid.org). Moreover, due to the complexity of the climate model applications there is an inherent difficulty of migrating these applications to other computing infrastructure. One benefit of Grid technology is that once an application has been migrated to a Grid infrastructure, the user will find very easy running it in every computing element of this Grid infrastructure or the new ones joining in the future.

However, further research is necessary in order to adopt the applications from the climate modeling community due to their high productivity and high performance requirements. The specific characteristics and requirements of climate modeling applications pose new challenges to the Grid. Today, the existing Grid middleware does not meet many of the requirements climate models demand to properly run in Grid infrastructures. To overcome this situation, particular ad hoc solutions are developed to adapt each experiment to run in Grid (Sulis, 2009; Davidovic and Skala, 2010).

Considering that most climate models face the same problems to run in Grid, the development of a generic framework that meets these requirements would be desirable. With this aim, Fernández-Quiruelas et al. (2009a) devised a first prototype of the framework and performed some experiments using the CAM model. This helped us to detect the weaknesses of our prototype and to establish the requirements the framework had to fit. The following Section summarizes these requirements.

3.1. Requirements for climate modeling

One of the main issues of the Grid is the heterogeneity of computing resources, which may be a critical fact in order to properly run long executions managing large amounts of memory and data (see, e.g. Fernández-Quiruelas et al., 2009a). Moreover, most clusters in the Grid have limitations regarding: CPU time (the processor time spent, not counting the time waiting for input/output operations or for the availability of resources), wall time (the real time spent running in the queue), disk usage, memory usage, etc. These limitations may force the premature end of a job. Furthermore, it is also common to find misconfigured resources, due to the large number of sites and administrators involved. Regarding data transfer, when sites are scattered all over the world, network bandwidth becomes critical.

Some typical applications from disciplines such as biomedicine or HEP are short-time simulations that do not manage large data sets nor need a huge amount of memory or disk space to be run. Thus, if a simulation fails, it is sent again to the infrastructure with minimum impact on the whole experiment. By contrast, ES applications usually require running complex models during days, consuming a lot of memory and generating large amounts of data. If these simulations were sent directly to the Grid, it may happen that none of them finished due to the limitations explained before (memory, CPU, disk limits). Moreover, climate models highly interact with data resources requiring the data sets to be intelligently replicated; otherwise, models may expend more time downloading and uploading data than running. This is why it is necessary to do some changes in the workflow of the applications in order to adapt them to overcome these limitations.

The most important requirements for a successful climate Grid application are (Fernández-Quiruelas et al., 2009a):

- Failure awareness: The application has to foresee all the possible sources of failure (including wall time and CPU time limitations) being able to face them or at least detect them and act in consequence.
- Checkpointing for restart: In case of failure, due to the computational cost of climate applications, one would want to restart the simulation in a different working site from the point it was interrupted (or as close as possible). This is done by writing intermediate recovery files to disk at a given frequency.
- Monitoring: Since climate simulations last for a long time, the user requires to know the current status of the experiment and their associated simulations: which percentage of the experiment is complete, whether there are simulations running, which time step is being calculated by a simulation, which data sets have been produced and in which storage elements are they, which is the last checkpointing/restarting point, etc.
- Data and Metadata storage: The goal of the climate model experiments is the generation of (large amounts of) simulated climatic information. This information needs to be post-processed and analyzed by the different tools used by the climate researcher. Therefore, the data has to be easily accessed by users. A data and metadata management system has to be developed to handle all the information generated.

The above requirements made necessary the development of a goal-oriented workflow manager in order to run the experiments with a minimum of human intervention.

As mentioned, the current Grid middleware does not fulfill these requirements. Therefore, the development of a new framework is necessary to use the current Grid resources and infrastructures by climate modeling applications. This framework has to address all the previous requirements which, at the same time, must be transparent and easy to use for the end user (usually not a Grid expert). With these ideas in mind, the CAM4G application has been developed, which is a Grid workflow management layer for the climate simulation with CAM. CAM4G is described next as an illustrative example of how a state-of-the-art climate application has been ported to the Grid.

4. CAM for Grid: CAM4G

The Community Atmospheric Model (CAM; Collins et al., 2004, 2006a) is the atmospheric component of the Community Climate System Model (CCSM, Collins et al., 2006b), which is a coupled atmosphere-ocean global climate model (AOGCM). CCSM3 is a state-of-the-art climate model developed at the National Center for Atmospheric Research (NCAR) of the U.S. and used e.g. to simulate future scenarios in the latest (4th) assessment report of the Intergovernmental Panel of Climate Change (IPCC; Randall et al., 2007). We deal only with the atmospheric component (CAM3) coupled with the land surface model (CLM3). A relatively coarse T42 (approx. $2.8 \times 2.8^{\circ}$) resolution is used in order to simulate our experiment in a reasonable time and to be able to use the largest amount of grid resources. The CAM3 model is open-source, it is coded in Fortran and is available from http://www.cesm.ucar.edu/models/atm-cam.

Fernández-Quiruelas et al. (2009a) (hereafter referred to as FQ09) presented an initial prototype of a framework to run the CAM model on a Grid environment. In this first attempt, the gLite middleware was used to build the framework. The data management was controlled by the LFC server, and the monitoring system was handled by AMGA (gLite Grid Metadata Catalog). With this prototype FQ09 discovered that the implementation had a bottleneck in the data management and that the monitoring system had to be improved. Also, they slightly modified the source code of the model to interact with the Grid middleware.

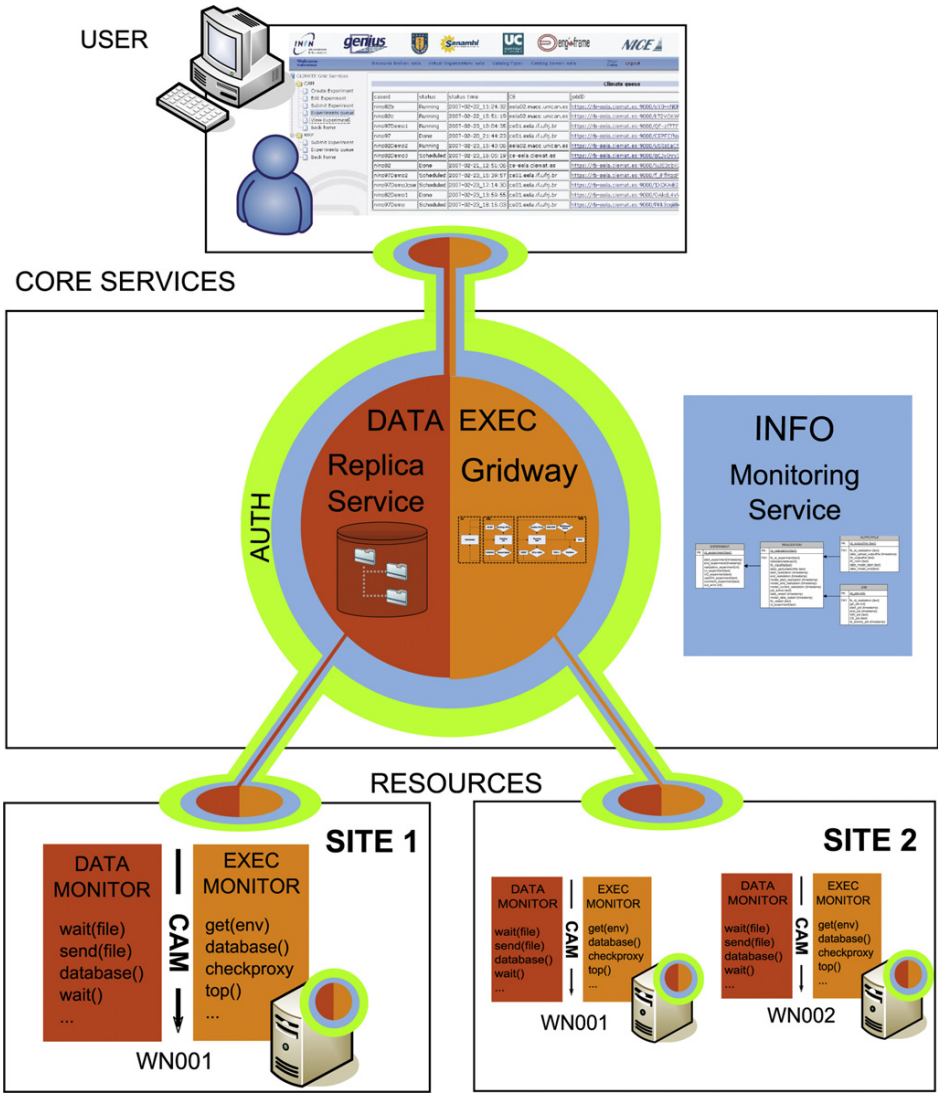


Fig. 2. CAM4G framework components. From the user interface the user manages the data and jobs and monitors the experiment. This is done thanks to the core services in charge of managing the jobs (GridWay), data replicas (Replica Service) and the experiments information (Monitoring Service). Jobs are executed in the WN wrapped by 2 monitors (execution and data monitors).

CAM4G is a new implementation of CAM for Grid, improving the FQ09 execution workflow by adding new data management and monitoring capabilities, as described in this section. Additionally, the execution workflow framework developed for CAM4G should be flexible enough to be applied to other climate models. In order to meet this goal, we avoided changing the model source code and we rely on the use of monitors that wrap the model execution by searching patterns in the output and log files generated by the model. When new files are ready or other model events occur, the monitor triggers the tasks needed to abstract the execution environment. The procedure of keeping the original source code has two clear advantages: the framework is more easily ported to other models regardless the internal software complexity and it is more easily upgraded as new versions of the code are made available by the center developing the model. As an example of the portability, the framework presented here has been applied to other models such as the Weather Research and Forecasting limited area atmospheric model (WRF4G; Fernández-Quiruelas et al., 2009b).

From the user's point of view, CAM4G has 3 hierarchical components: (Allcock et al., 2005) The experiment to be carried out with the model, designed to answer some scientific question, usually by means of an ensemble of (Allen, 1999) realizations, that will be carried out in a single or, most probably, several (Anderson and Fedak, 2006) Grid jobs. The term *realization* refers to the independent pieces an experiment can be divided into. A Grid job cannot be related one to one with a *realization* since realizations cannot be guaranteed to finish in a single job. In general, a *realization* requires several Grid jobs to be completed, each one restarted from the previous one. Thus, from the point of view of the workflow, the *realizations* are independent tasks to be carried out on the Grid and the jobs spanning a *realization* are dependent tasks.

In order to submit an experiment with CAM4G, the user only has to fill the experiment details in a configuration file, prepare the input data and submit the experiment to the Grid using the CAM4G user tools or the web portal. During the execution of the experiment, the user can check the status of the realizations conforming

the experiment and access the output data while they are produced by the running jobs. Failing jobs are restarted in an unattended way until each realization is completed. To achieve this transparency for the user, a complex execution framework has been designed. This framework has been built from scratch by adapting well-known Grid services to our needs and creating new modules for the tasks that the existing middleware could not manage (see Fig. 2). In order to provide the monitoring capability, we retrieved all the events in the workflow and consolidated them in a self-developed monitoring system based on MySQL (the database system used by FQ09 did not fulfill the scalability, reliability and time performance requirements of CAM4G). Regarding the new data management in CAM4G, after analyzing the middleware solutions provided by gLite, we decided to create a replica service with the aim of optimizing the data transfers (using a system that finds the nearest replica of a file). The job execution is managed by GridWay (Huedo et al., 2005), a flexible job meta-scheduler.

The monitoring system is fed with the information retrieved by two monitors (execution and data) that are started with the job in the computing node. Apart from giving the realization status to the monitoring system, the execution monitor interacts with GridWay to overcome all the possible job failures and reschedule the jobs. The data monitor detects when new output or restart data are created and uploads it using the replica service.

Although CAM4G provides a precompiled serial version of CAM3, thanks to this framework, users could run their own compiled code in the Grid. It is important to note that although there are some production experiments using this framework, currently, it is just a prototype. At the moment, CAM4G has been tested in Globus and gLite infrastructures, and it supports ×86 and x86_64 systems running Linux (tested on Scientific Linux, CentOS, RedHat and Debian). Further efforts are being made in order to adapt the framework to other arquitectures and operating systems. As soon as CAM4G is fully documented, it will be launched under an open-source license. All the components of the framework, including the model itself, are open-source.

Fig. 2 shows an schematic illustration of the CAM4G components, using the Grid representation shown in Fig. 1. The top of the figure shows the web environment from where the user can submit and monitor the realizations and manage the data. In order to carry out these tasks, the user's web environment make use of the CAM4G core services (see the middle part of the figure): The job execution workflow is managed by GridWay, the data by the replica service and the job status is retrieved from the monitoring service. For instance, in the example, the user has submitted an experiment composed by 3 realizations that have been scheduled and sent by GridWay to 3 worker nodes in 2 different sites. Each site shares a cluster with one or more worker nodes. Each worker node may have several cores (processing units) able to execute a serial job such as ours. The bottom of the figure shows how the jobs run in the computing resources (WN001, WN002, …) wrapped by the data and execution monitors. These monitors transfer the relevant information to the monitoring core service, upload the output and restart data produced by the job and interact with GridWay to overcome the possible job failures. If a job fails in a computing node, the execution monitor will detect it and will notify GridWay and the monitoring service. Then, GridWay will send the job to another site and download the data for restarting the job from the nearest replica.

5. An illustrative experiment with CAM4G

In order to illustrate the performance of state-of-the-art Grid computing for the climate community, in this section we present the results obtained using the EGEE Grid infrastructure (Section 2.3) to run a sensitivity experiment involving the execution of 750 19-month simulations of the CAM model (T42 resolution) with varying prescribed sea ice and sea surface temperature (SST). The goal is to analyze the effect of El Niño SST forcing in the accumulated precipitation. The El Niño phenomenon consists of an anomalous heating of the eastern pacific ocean, which has an associated atmospheric circulation counterpart known as the Southern Oscillation (both oceanic and atmospheric components are referred to as El Niño/Southern Oscillation or ENSO). ENSO events occur every 2–7 years and affect the global circulation, changing e.g. the rainfall patterns in distant regions. This phenomenon has huge social impact since it is related to flood and drought events in different regions (e.g. in several south American countries). CAM3 has already been used in previous works to study El Niño responses with the same T42 resolution uncoupled version (Jin and Kirtman, 2010) and also comparing different resolutions (Zhang and Sun, 2006) or the coupled and uncoupled versions (Hack et al., 2006).

5.1. Description of the experiment

As a first step, we computed an El Niño SST perturbation pattern using the mean SST anomaly in the tropical Pacific ocean given by the two strongest events recorded (1982 and 1997), with respect to the long term SST climatology. This SST pattern was scaled by its maximum grid value (2.5 K). The resulting normalized anomalous SST pattern (hereafter NAS pattern, Fig. 3) was applied to generate perturbed SST fields that were used as boundary conditions in our CAM4G ensemble. For instance, if the NAS pattern is multiplied by −2.5 and added to the observed SST, the El Niño anomalous signal will be removed. If it is multiplied by a negative scaling parameter $-2.5 < s_n < 0$, the El Niño signal will be weakened. Values above zero intensify the SST anomaly producing record-breaking ENSO events. We generated 750 perturbed SST distributions by randomly

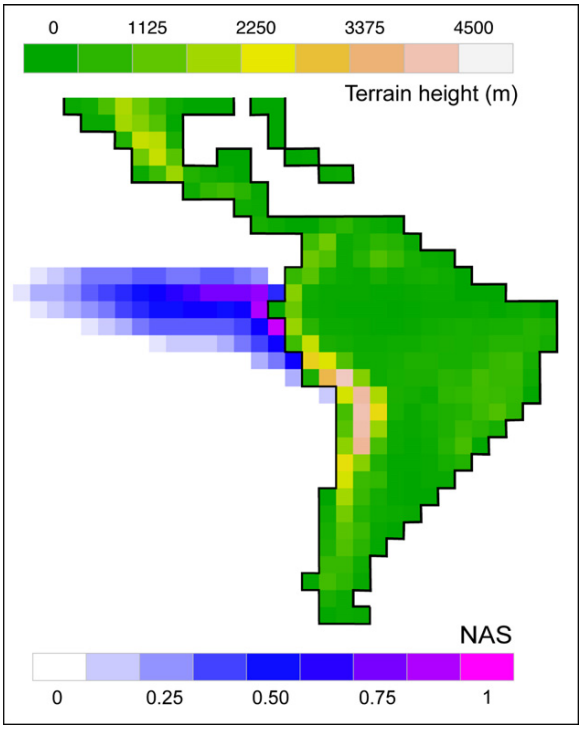


Fig. 3. Terrain elevation as seen by the CAM model (over land) and the normalised anomalous SST (NAS) pattern used to perturb the SST (over sea). The insets show the sensitivity of precipitation to the SST perturbation at different grid points.

selecting scaling parameters s_n in the range [−2.5, 2.5] from a uniform distribution:

$$\mathrm{SST}_n(t,x) = \mathrm{SST}_{\mathrm{obs}}(t,x) + s_n \mathrm{NAS}(x), \quad n = 1 \ldots 750.$$

That is, we are sampling SST distributions from normal conditions ($s_n \approx -2.5$) to an ENSO-like SST anomaly around twice as strong as that observed in 1997–98 ($s_n \approx 2.5$). The large number of simulations allows a quantification of the internal variability of the model (sensitivity to small variations in the boundary conditions) as a reference for the changes observed as the SST changes. We focused on the eastern tropical pacific, where most of the circulation variability can be explained by the SST variability in AMIP-type simulations (Kushnir et al., 2002).

The atmospheric and soil initial conditions for all the ensemble realizations were the same. They were obtained from a previous model run which started from climatological conditions on 1st January 1990, and was forced by the observed SST and sea ice for one decade, in order to properly spinup the soil component (AMIP-type simulation, Gates, 1992). The resulting SST-assimilated atmospheric and soil state on January 1st, 1997 was used as initial condition in our experiment. The simulations run for the 19-month period, up to July 1998. This period includes one of the strongest El Niño events observed to date (McPhaden, 1999).

As a sample analysis we focused only on precipitation averaged from January through April, when the largest precipitation in coastal Peru occurred (Takahashi, 2004). This region is specially sensitive to ENSO events, carrying floods to places where usually there is few or no rainfall (Waylen and Caviedes, 1986). Fig. 4a and b compare the observed precipitation according to TRMM (Rasmusson et al., 2006) data with the precipitation simulated by the model when the SST is as observed (not perturbed). CAM simulations underestimate the observed mean precipitation in the period considered. Although the main precipitation pattern is well reproduced, there are several deviations. The tropical rainbelt associated with the ITCZ appears in the simulation split into two. This is a recurrent problem in coupled and uncoupled GCMs (Zhang et al., 2007; Dai, 2006; Hurrell et al., 2006) which is not related to their execution on the Grid. Also, the precipitation maximum north of Paraguay was not reproduced in the simulations.

In order to analyze the changes produced by the intensity of the SST ENSO perturbation (s_n) on the simulated precipitation in this region, we chose four illustrative locations (labeled as A, B, C and D in Fig. 4b). For each of these grid points we considered the ensemble of 750 simulations performed using the Grid and displayed the scatter plots showing the precipitation value vs. the perturbation (see Fig. 4A–D). These figures exhibit a linear trend in most of the cases, although there are also some nonlinear responses (see Panel A). Therefore, in order to test the sensitivity of the results we did not consider the slope of the corresponding fitted regression line, but the Spearman correlation coefficient, as shown in Fig. 5 for the global domain. Thus, positive values indicate increasing precipitations with higher perturbations, whereas the negative values indicate decreasing trends. This figure shows a complex

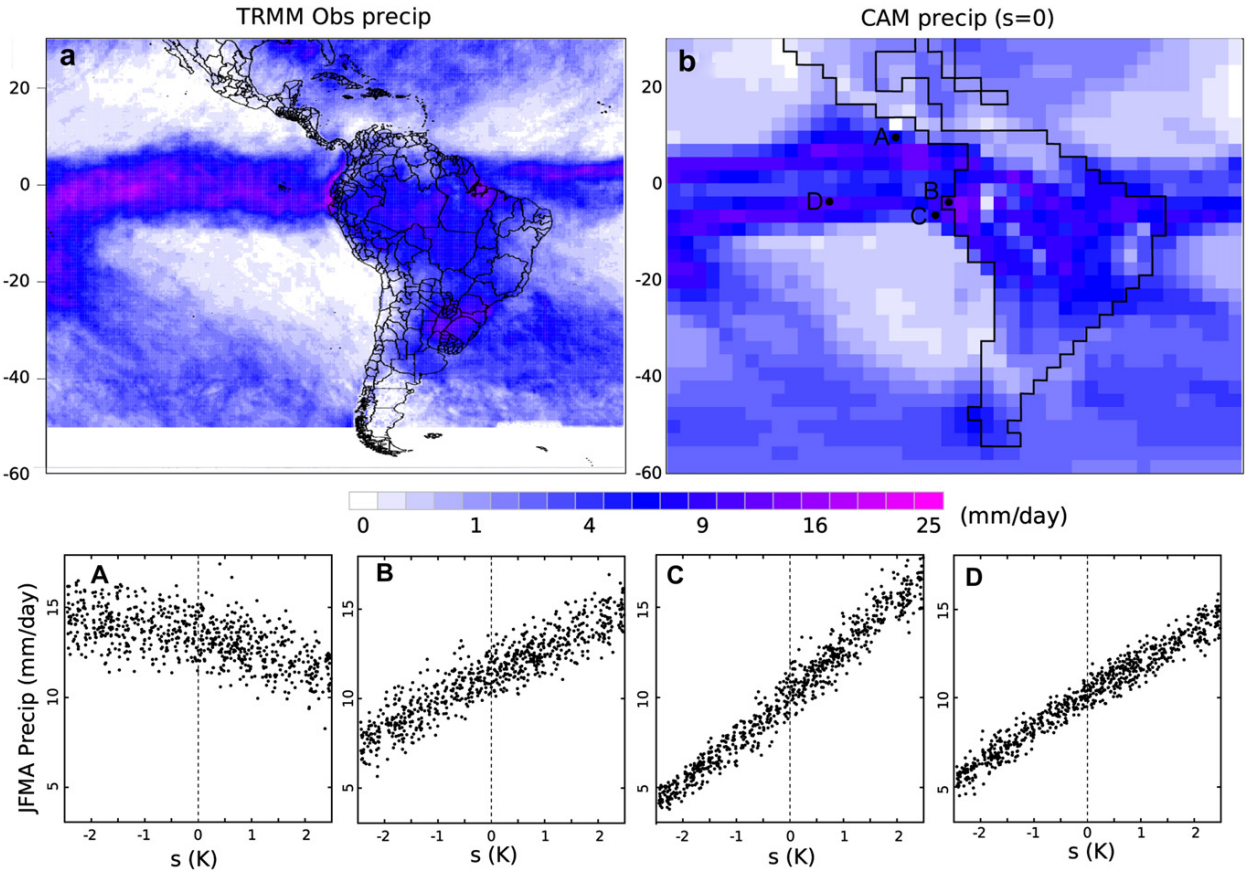


Fig. 4. (a) Mean observed precipitation (mm/day) from January through April 1998 according to TRMM data vs. (b) CAM simulated precipitation for a realization with $s = 0$ (unperturbed simulation). The scale is square-root, to appreciate low precipitation areas. Panels A–D show scatter plots of the precipitation vs. the perturbation for the four different grid points shown in the figure.

sensitivity pattern with highly correlated regions (both negative and positive; see the scatter plots) as well as intermediate positive ones particularly in midlatitudes. The highest responses are mainly located over the region where we added the SST perturbation. However, there is also a significant response over southwestern Africa; this ENSO sensitivity over Africa was documented long ago (Ropelewski and Halpert, 1987), and has been related to SST variations over the Atlantic and western Indian ocean (Nicholson and Kim, 1997). In the model, this sensitivity appears even though only the eastern pacific SST was modified.

Fig. 5 also shows the scatter plots of precipitation change vs. the perturbation intensity for those grid points with largest positive (panels E and F) and negative (G and H) correlation values. From this figure it can be clearly shown the existence of nonlinear responses (panel F). Further analysis is needed for a detailed comprehension of this pattern, but this work is out of the scope of this paper.

The internal variability of the model due to small variations of the SST is related to the thickness of the scatter plots. A measure of this thickness was obtained both globally, by removing the linear trend at each grid point and computing the standard deviation, and locally, computing the standard deviation of those points in a window of ± 0.1 K around the zero perturbation value ($s = 0$), obtaining similar results. Fig. 6 shows the variability obtained with the later approach. Again, three scatter plots with the grid points

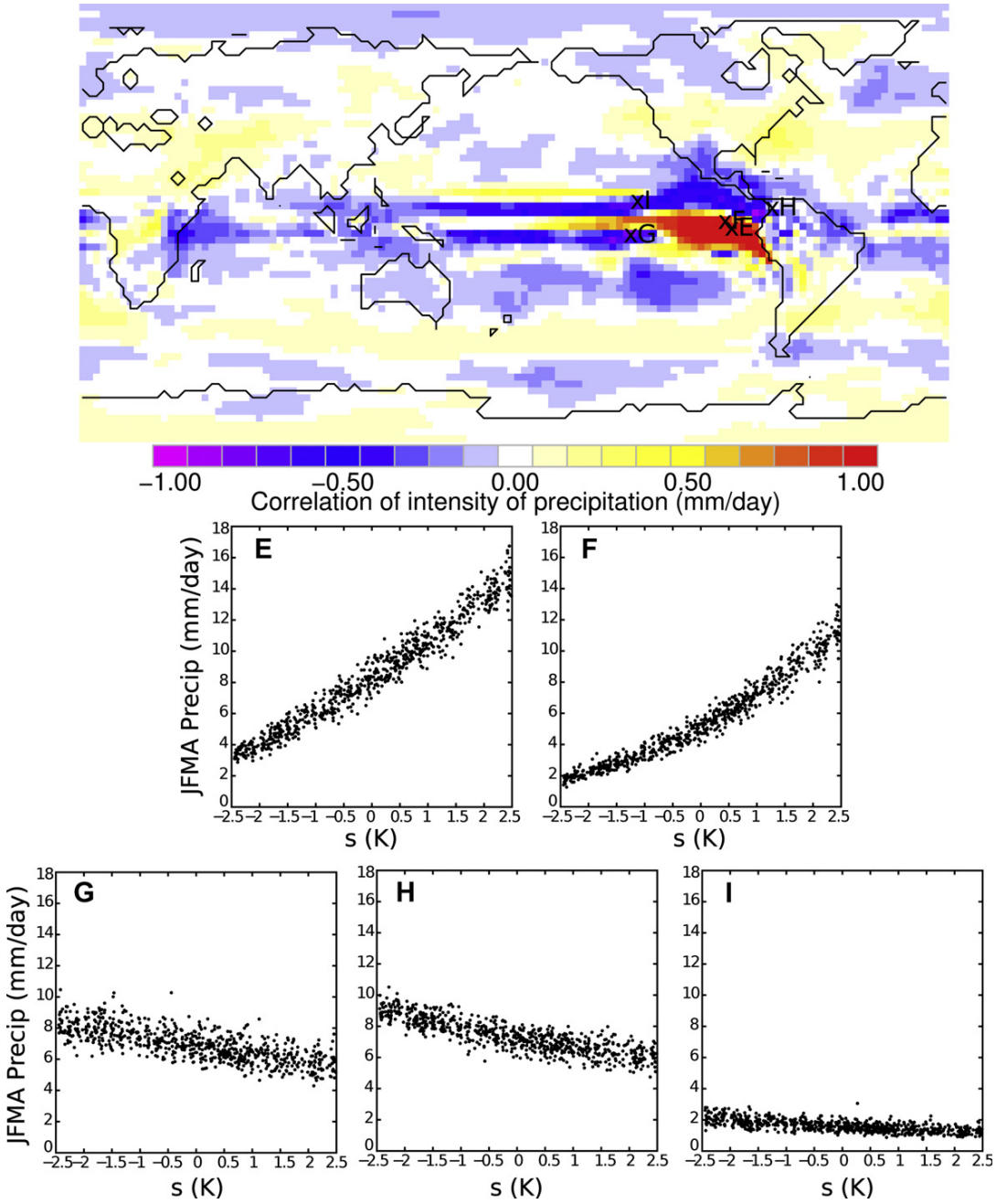


Fig. 5. Spearman correlation between January–April mean precipitation (mm/day) and the perturbation intensity (K) for each model grid point (see text). The scatter plots correspond to the 750 simulated values for five illustrative locations shown in the map.

with largest variabilities are also shown. Note that the variability is not directly related to the precipitation intensity. For instance, Fig. 6J–L correspond to grid points with very different precipitation amounts, but exhibiting similar variability.

Therefore, Figs. 5 and 6 provide different sensitivity information about the relationship of precipitation amount and SST perturbation intensity. The former provides an estimation of the increasing or decreasing trends associated with large perturbation values (±2.5 K), whereas the later provides information about the variability of the result for a small perturbation (±0.1 K). Note that a high number of simulations (750 in this example) is required to appropriately estimate both quantities. In a previous attempt (Fernández-Quiruelas et al., 2009a), the number of successful simulations was not enough to distinguish the signal of the response to the SST perturbation from the noisy internal variability. This stresses the need for a large number of simulations and the benefits of Grid computing for this kind of experiments.

5.2. Job execution statistics

The above example consisting of 750 realizations was run on the EGEE Grid infrastructure (Section 2.3). The serial version of CAM was used to perform this experiment (each realization only required one core to execute). In order to compare the efficiency of the Grid with traditional computing resources, we run a realization using one core of our local cluster (16 nodes with 2 Intel Xeon E5410 CPUs –totaling 128 cores– and 8 GB of memory). This realization took 44 h to complete and, thus, if we would have send the 750 realization to our cluster, the whole experiment would have taken all of our computing resources for 11 days.

A pre-screening of the available sites was done to meet the requirements of the experiment. Each realization runs for about 44 h and requires around 200 MB of input data. Sites with short job wall time should be avoided since, in that case, most of the time would be spent by the download of the input data and resubmitting the jobs instead of running. We established a requirement on the wall time to be over 12 h. Given the large amount of input data, we selected sites with large bandwidth and, among those, we chose the 9 sites with the faster cores.

Once the sites were chosen, the input data were replicated in 6 European sites. In this way, the network load would be distributed when the 750 realizations started to run. One of the main advantages of the CAM4G framework is that it is prepared to locate the nearest replica from a given location. This feature is very useful also when the output data and restart files are uploaded to the storage elements.

The 750 realizations were submitted at the same time. As shown in Fig. 7, in half an hour the first 400 jobs had started to run (i.e. the input files were already downloaded) in the computing nodes. The rest of the jobs started to run as resources were available. Some of them were queued for some time in the clusters and others spent a long time to download the input files. After 6 h, all the 750 jobs were running.

The first set of realizations finished in 42 h. Each of the realizations that finished before 48 h was run in a single job (the

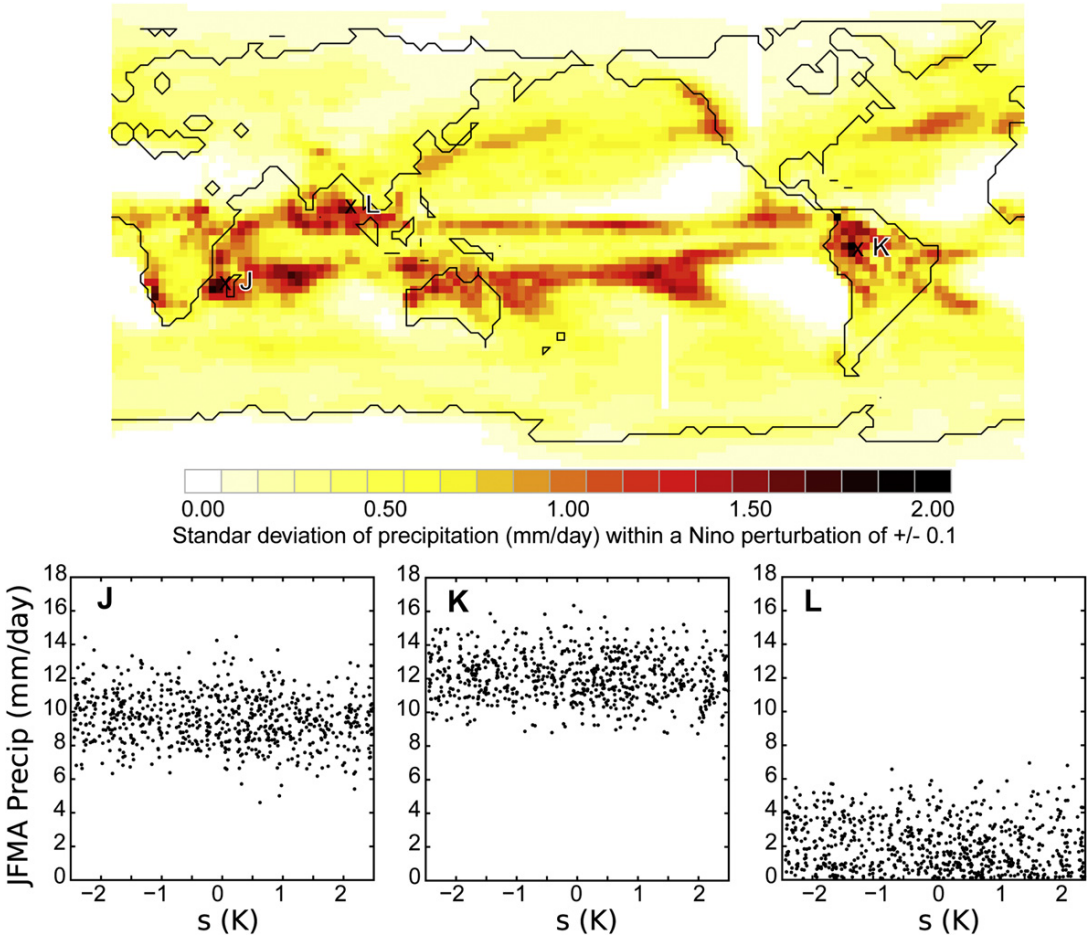


Fig. 6. Internal model variability, obtained as the standard deviation of the points in a window of ±0.1 K around the zero perturbation value ($s = 0$).

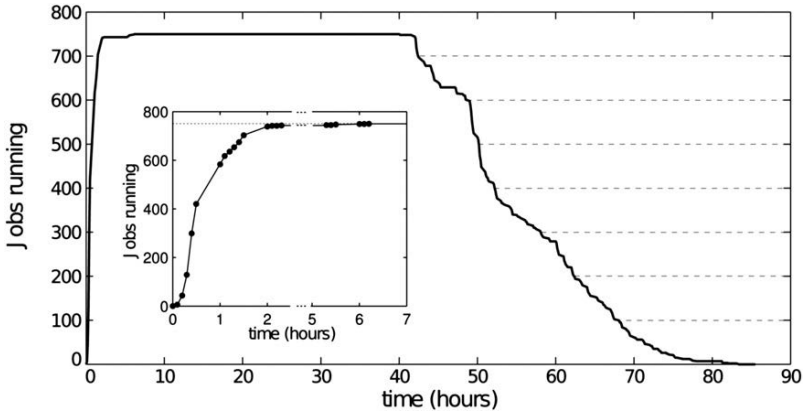


Fig. 7. Number of jobs running at a given time (in hours from the experiment submission). The inset shows a zoom of the initial 7 h.

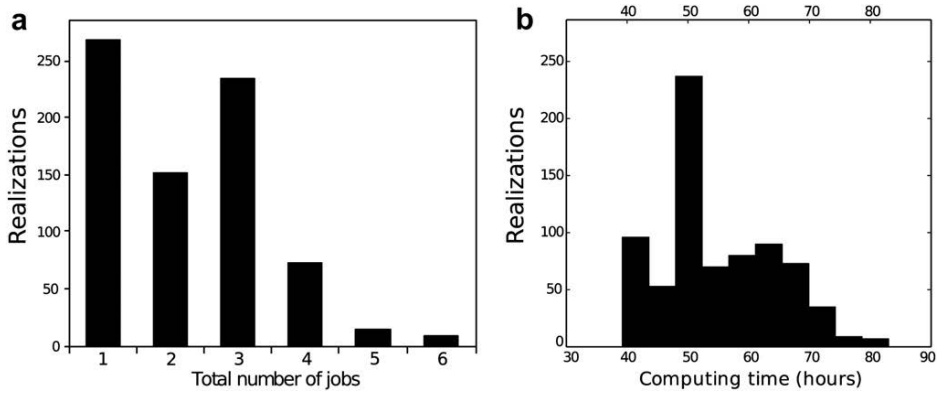


Fig. 8. (a) Number of jobs required by each realization to complete. (b) Histogram of the time (in hours) required by each realization to complete.

realization did not need to be restarted and ran in a single attempt). This implies that the sites where they were running had a wall time larger than 42 h and no problem was found during the execution. The rest of the realizations had to be restarted at least once and spanned at least 2 Grid jobs. In these cases, the realization started to run, and before finishing, the job failed (usually the wall time limit had been exceeded). Then, the framework detected it and submitted another job that continued the simulation from the last restart point stored by the failed job. In the worst case, a realization required 6 jobs to complete, but most of the realizations spanned less than 4 jobs (Fig. 8a). The realizations took between 40 and 85 h to complete (Fig. 8b). The computing time differences among realizations were due to several factors including the CPU speed, queue wall time (increases the number of restarts required), errors during the execution and bandwidth differences.

The experiment was finished in 3.5 days and all the 750 realizations were successfully completed.

6. Conclusions

In this article, a wide and general introduction to Grid computing is presented having in mind a climate science researcher used to work with local clusters. Thus, we focus on those aspects which are different when using the Grid than when using a local cluster (e.g. login, transferring data, submitting jobs, etc.). For this purpose, the three main layers of the Grid (user, core services and resources) have been presented and the new protocols of security and authentication services to access a Grid infrastructure have been described (certificates, virtual organizations, etc.). It has also been introduced the new concept of *middleware*, which provides services for a transparent and clear access to the heterogeneity of resources of a Grid, and *GridFTP*, as a protocol to distribute and access data in the Grid.

Moreover, it has also been described the existing state-of-the-art Grid infrastructures (such as the European EGEE initiative, with more than 10,000 CPUs from 50 different sites) and the different ways to aggregate them through Virtual Organizations (VOs). A review of the work done within the climate community (included in the Earth Science VO) has been done, pointing out that most of the attention has been focused on data management. Thus, further work is needed to deploy and run climate applications (such as climate models), which is still a challenge for actual Grid infrastructures.

This paper has focused on this problem, analyzing the main requirements of climate applications (failure awareness, checkpointing for restart, monitoring and data and metadata management, etc.). As a result of this analysis it was concluded that new middleware components (execution workflow managers) are needed to cope with the particular requirements of these applications to run efficiently on the Grid. For instance, as an illustrative example, we described a new Grid execution workflow for the Community Atmospheric Model (CAM) wrapping the model and allowing to restart interrupted jobs, to manage the data and to monitor the running experiments. Moreover, in order to

demonstrate the performance of this Grid-deployed model, a real computing challenge of a climate research experiment has been designed. The experiment consisted in a sensitivity analysis of global precipitation to perturbations in the sea surface temperature (SST) in the Niño region, considering a total of 750 perturbed simulations (realizations). Results show that precipitation sensitivity is higher in the areas where SST was modified. However, world-wide teleconnections of El Niño signal are found, since some sensitivity is also found in some other places like in West Equatorial Africa. It is also illustrated that precipitation shows both linear and nonlinear responses to the strength of the Niño signal.

To quantify the benefits of Grid computing, statistics of Grid execution of the experiment are given and compared to the computational cost on a local cluster. It was shown how the 11 days required to finish the experiment (all 750 simulations) in the authors' cluster is reduced just to 3.5 days in the Grid. At the same time, statistics of the different realizations are also shown describing that in general individual realizations require about 2 or 3 Grid jobs to be finished, consuming an average of 50 h, a 114% of the 44 h spent in the local cluster. As an example of the technical realism of our experiment, in a study published last year, Jin and Kirtman (2010) performed with this same model and resolution, four 72-year simulations totaling 288 simulated years. Our simulations, carried out in less than 4 days in the Grid, are the equivalent of 1187 simulated years (750 $\times$ 19 months).

For the sake of illustration and fast deployment, we ported only the atmospheric and land components of a state-of-the-art coupled GCM. The experiment was designed to finish in a reasonable time by choosing a relatively coarse resolution. The state-of-the-art resolution of a GCM varies wildly with the particular experiment to be carried out. For example, for century-long simulations in the last IPCC assessment report, they used this model with T85 resolution. This is twice the resolution used in our experiment (i.e 4 times more memory demand and 8 times slower). Moreover, the model was coupled to the ocean component and ran for a 100-times-larger period. Such an application would necessarily need to take advantage of the parallel capabilities of the resources contributed to the Grid. We also tested successfully the parallel execution in the Grid, but it restricts the number of usable sites and requires a more specific treatment for each site, unlike the serial execution example shown in this work.

Due to the complexity and high demanding computational resources of climate modeling, Grid infrastructures have some aspects that should be improved in order to be completely useful for the climate modeling community. Some of these lacks have been found during the development and use of the CAM4G middleware presented in this paper. The main issues are related to data storage and access. Climate models need a large amount of data in order to be run and at the same time, produce a large amount of data. That fact reduces the Grid resources where a climate application such as CAM can be used. An improvement on bandwidth on Grid infrastructure would be desired. At the same time fast and stable management and replication of the data is also needed. Finally, an effort to allow parallel execution on Grid infrastructures should also be done, allowing the design of more demanding experiments, in terms of memory and CPU resources, than the one presented in this article.

Acknowledgements

This work has benefited from the ESR VO infrastructure of the EU FP7 EGEE-III. This work has been partially supported by the EU FP7 project EELA-2 (Contract number 223797) and the Spanish Ministry of Science and Innovation through project CORWES (CGL2010-22158-C02-01).

References

Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., Foster, I., 2005. The globus striped Gridftp framework and server. In: Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference. IEEE Computer Society, p. 54.

Allen, M., 1999. Do it yourself climate prediction. Nature 401, 642.

Anderson, D.P., Fedak, G., 2006. The computational and storage potential of volunteer computing. In: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid. IEEE Computer Society, pp. 73–80. http://portal.acm.org/citation.cfm?id=1134996.

Board, I.A., 2009. Impact assessment guidelines. In: Technical Report, vol. 92. European Commission.

Catlett, C., 2002. The philosophy of TeraGrid: building an open, extensible, distributed TeraScale facility. In: 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 8.

Collins, W., Rasch, P., Boville, B., Hack, J., McCaa, J., Williamson, D., Briegleb, B., Bitz, C., Lin, S., Zhang, M., 2006a. The formulation and atmospheric simulation of the Community Atmosphere Model version 3 (CAM3). Journal of Climate 19 (11), 2144–2161.

Collins, W.D., Bitz, C.M., Blackmon, M.L., Bonan, G.B., Bretherton, C.S., Carton, J.A., Chang, P., Doney, S.C., Hack, J.J., Henderson, T.B., et al., 2006b. The community climate system model: CCSM3. Journal of Climate 19 (11), 2122–2143.

Collins, W.D., Rasch, P.J., Boville, B.A., Hack, J.J., McCaa, J.R., Williamson, D.L., Kiehl, J.T., Briegleb, B., Bitz, C., Lin, S.-J., Zhang, M., Dai, Y., 2004. Description of the NCAR Comunity Atmospheric Model (CAM 3.0). Technical Report NCAR/TN-464+STR. National Center for Atmospheric Research. www.cesm.ucar.edu/models/atm-cam/docs/description/description.pdf.

Cossu, R., Petitdidier, M., Linford, J., Badoux, V., Fusco, L., Gotab, B., Hluchy, L., Lecca, G., Murgia, F., Plevier, C., Renard, P., Schwichtenberg, H., de Cerff, W., Tran, V., Vetois, G., 2010. A roadmap for a dedicated earth science grid platform. Earth Science Informatics 3 (3), 135–148.

Dai, A., 2006. Precipitation characteristics in eighteen coupled climate models. Journal of Climate 19, 4605.

Davidovic, D., Skala, K., 2010. Implementation of the WRF-ARW prognostic model on the Grid. In: MIPRO, 2010 Proceedings of the 33rd International Convention. IEEE, pp. 220–225.

Fernández-Quiruelas, V., Fernández, J., Baeza, C., Cofiño, A.S., Gutiérrez, J.M., 2009a. Execution management in the GRID, for sensitivity studies of global climate simulations. Earth Science Informatics 2 (1), 75–82.

Fernández-Quiruelas, V., Cofiño, A.S., Fernández, J., Fita, L., 2009b. WRF4G: enabling WRF on the Grid. In: Proceedings of the Second EELA-2 Conference. Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, Madrid, pp. 149–154. http://www.ciemat.es/recursos/doc/Redes_Cientifico_Tecnicas/422103760_17112009151715.pdf.

Foster, I., 2006. Globus Toolkit version 4: software for service-oriented systems. In: International Conference on Network and Parallel Computing LNCS 3779, vol. 15. Springer-Verlag, pp. 2–13.

Foster, I., Kesselman, C., 1999. The Grid. Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco.

Foster, I., Kesselman, C., Tsudik, G., Tuecke, S., 1998. A security architecture for computational grids. In: Proceedings of the 5th ACM Conference on Computer and Communications Security. ACM New York, NY, USA, pp. 83–92.

Foster, I., Kesselman, C., Tuecke, S., 2001. The anatomy of the grid: enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15 (3), 222.

Fusco, L., Cossu, R., Retscher, C., 2008. Open Grid services for Envisat and Earth observation applications. High Performance Computing in Remote Sensing, 237–280.

Gates, W.L., 1992. AMIP: the atmospheric model intercomparison project. Bulletin of the American Meteorological Society 73 (12), 1962–1970.

Hack, J.J., Caron, J.M., Yeager, S.G., Oleson, K.W., Holland, M.M., Truesdale, J.E., Rasch, P.J., 2006. Simulation of the global hydrological cycle in the ccsm community atmosphere model version 3 (cam3): mean features. Journal of Climate 19 (11), 2199–2221.

Huedo, E., Montero, R., Llorente, I., 2005. The GridWay framework for adaptive scheduling and execution on grids. SCPE 6 (8).

Hurrell, J.W., Hack, J.J., Phillips, A.S., Caron, J., Yin, J., 2006. The dynamical simulation of the community atmosphere model version 3 (CAM3). Journal of Climate 19 (11), 2162–2183.

Jacq, N., Salzemann, J., Jacq, F., Legré, Y., Medernach, E., Montagnat, J., Maass, A., Reichstadt, M., Schwichtenberg, H., Sridhar, M., et al., 2008. Grid-enabled virtual screening against malaria. Journal of Grid Computing 6 (1), 29–43.

Jin, D., Kirtman, B., 2010. The extratropical sensitivity to the meridional extent of tropical ENSO forcing. Climate Dynamics 34 (7), 935–951.

Kushnir, Y., Robinson, W.A., Bladé, I., Hall, N.M.J., Peng, S., Sutton, R., 2002. Atmospheric GCM response to extratropical SST anomalies: synthesis and evaluation. Journal of Climate 15, 2233–2256.

Lagouvardos, K., Floros, E., Kotroni, V., 2010. A grid-enabled regional-scale ensemble forecasting system in the Mediterranean area. Journal of Grid Computing 8 (2), 181–197.

McPhaden, M.J., 1999. Genesis and evolution of the 1997–98 El Niño. Science 283, 950–954.

Mineter, M., Jarvis, C., Dowers, S., 2003. From stand-alone programs towards grid-aware services and components: a case study in agricultural modelling with

interpolated climate data. Environmental Modelling & Software 18 (4), 379–391.
Murphy, J., Sexton, D., Barnett, D., Jones, G., Webb, M., Collins, M., Stainforth, D., 2004. Quantification of modelling uncertainties in a large ensemble of climate change simulations. Nature 430 (7001), 768–772.
Nandakumar, R., Jimenez, S., Adinolfi, M., Bernet, R., Blouw, J., Bortolotti, D., Carbone, A., M'Charek, B., Perego, D., Pickford, A., et al., 2008. The LHCb computing data challenge DC06. In: Journal of Physics: Conference Series, vol. 119. Institute of Physics Publishing 072023.
Nicholson, S.E., Kim, J., 1997. The relationship of the El Niño–Southern Oscillation to African rainfall. International Journal of Climatology 17, 117–135.
Palmer, T.N., 2002. The economic value of ensemble forecasts as a tool for risk assessment: from days to decades. Quarterly Journal of the Royal Meteorological Society 128 (581), 747–774.
Parry, M., Canziani, O., Palutikof, J., van der Linden, P., Hanson, C. (Eds.), 2007. Climate Change 2007: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge.
Postel, J., Reynolds, J., 1985. File transfer protocol.
Randall, D.A., Wood, R.A., Bony, S., Colman, R., Fichefet, T., Fyfe, J., Kattsov, V., Pitman, A., Shukla, J., Srinivasan, J., Stouffer, R.J., Sumi, A., Taylor, K.E., 2007. Climate models and their evaluation. In: Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, pp. 589–662.
Rasmusson, E., Chandrasekar, V., Clayson, C.A., Hawkins, J.D., Katsaros, K.B., McCormick, M.P., Steiner, M., Stephens, G.L., Velden, C.S., Williamson, R.A., 2006. Assessment of the Benefits of Extending the Tropical Rainfall Measuring Mission: a Perspective from the Research and Operations Communities. The National Academies Press. Available on-line at: http://www.nap.edu/catalog/11195.html.
Renard, P., Badoux, V., Petitdidier, M., Cossu, R., 2009. Grid computing for earth science. EOS 90 Trans. AGU.
Ropelewski, C.F., Halpert, M.S., 1987. Global and regional scale precipitation patterns associated with the El Niño/Southern Oscilation. Monthly Weather Review 115, 1606–1626.
Sulis, A., 2009. GRID computing approach for multireservoir operating rules with uncertainty. Environmental Modelling & Software 24 (7), 859–864.
Takahashi, K., 2004. The atmospheric circulation associated with extreme rainfall events in Piura, Peru, during the 1997–1998 and 2002 El Niño events. Annals of Geophysics 22, 3917–3926.
Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M., 2004. Rfc 3820: Internet X.509 public key infrastructure (PKI) proxy certificate profile.
Waylen, P., Caviedes, C., 1986. El Niño and annual floods on the north peruvian littoral. Journal of Hydrology 89 (1–2), 141–156. http://www.sciencedirect.com/science/article/B6V6C-487D1S0-D/2/5729b7151f79430fe2966e17ab7f3649.
Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S., Siebenlist, F., 2004. X.509 p.oxy certificates for dynamic delegation. In: 3rd Annual PKI R&D Workshop, vol. 14.
Williams, D.N., Drach, R., Ananthakrishnan, R., Foster, I.T., Fraser, D., Siebenlist, F., Bernholdt, D.E., Chen, M., Schwidder, J., Bharathi, S., Chervenak, A.L., Schuler, R., Su, M., Brown, D., Cinquini, L., Fox, P., Garcia, J., Middleton, D.E., Strand, W.G., Wilhelmi, N., Hankin, S., Schweitzer, R., Jones, P., Shoshani, A., Sim, A., 2009. The earth system grid: enabling access to multimodel climate simulation data. Bulletin of the American Meteorological Society 90 (2), 195–205. http://journals.ametsoc.org/doi/abs/10.1175/2008BAMS2459.1.
Zhang, T., Sun, D., 2006. Response of water vapor and clouds to El Niño warming in three National Center for Atmospheric Research atmospheric models. Journal of Geophysical Research 111 (D17), D17103.
Zhang, X., Lin, W., Zhang, M., 2007. Toward understanding the double Intertropical Convergence Zone pathology in coupled ocean-atmosphere general circulation models. Journal of Geophysical Research 112 (D12), D12102.

2.3. Large-scale climate simulations harnessing Clusters, Grid and Cloud infrastructures

Cita Completa

Fernández-Quiruelas, V; Blanco, C.; Cofiño, A.S.; Fernández, J. Large-scale climate simulations harnessing clusters, grid and cloud infrastructures. *Future Generation Computer Systems*. 51, pp.36 - 44. 2015. doi:10.1016/j.future.2015.04.009

Resumen

En este artículo se muestra una versión del entorno de trabajo presentado en los dos artículos anteriores. Esta nueva versión, llamada WRF4G, ha sido optimizada para ejecutar el modelo atmosférico regional WRF en infraestructuras HDCI. También se muestran los retos que las HDCI imponen a las aplicaciones y se describen los principales componentes utilizados en WRF4G para afrontar estos retos. Para ilustrar la utilidad del nuevo entorno de trabajo en HDCI, se ha llevado a cabo un experimento real, compuesto por 365 simulaciones ejecutadas usando recursos *cluster*, *Grid* y *Cloud* simultáneamente. Los resultados se comparan con los obtenidos en una ejecución previa de este experimento (Menéndez et al., 2014) en un *cluster*.

Future Generation Computer Systems 51 (2015) 36–44

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Large-scale climate simulations harnessing clusters, grid and cloud infrastructures

V. Fernández-Quiruelas *, C. Blanco, A.S. Cofiño, J. Fernández

Grupo de Meteorología, Dpto. Matemática Aplicada y CC. Computación, Universidad de Cantabria, Santander, Spain

HIGHLIGHTS

- Hybrid DCI (HDCI) is a common computing paradigm in earth science.
- The key problems found by standard applications to be run in HDCI are presented.
- The components of a new execution framework to run climate models are proposed.
- WRF4G is a successful implementation of the framework to run the WRF model in HDCI.

ARTICLE INFO

Article history:
Received 24 January 2014
Received in revised form
24 March 2015
Accepted 23 April 2015
Available online 5 May 2015

Keywords:
Grid computing
Cloud computing
HPC
Regional climate model
WRF
Hybrid distributed computing infrastructures

ABSTRACT

The current availability of a variety of computing infrastructures including HPC, Grid and Cloud resources provides great computer power for many fields of science, but their common profit to accomplish large scientific experiments is still a challenge. In this work, we use the paradigm of climate modeling to present the key problems found by standard applications to be run in hybrid distributed computing infrastructures and propose a framework to allow a climate model to take advantage of these resources in a transparent and user-friendly way. Furthermore, an implementation of this framework, using the Weather Research and Forecasting system, is presented as a working example. In order to illustrate the usefulness of this framework, a realistic climate experiment leveraging Cluster, Grid and Cloud resources simultaneously has been performed. This test experiment saved more than 75% of the execution time, compared to local resources. The framework and tools introduced in this work can be easily ported to other models and are probably useful in other scientific areas employing data- and CPU-intensive applications.

1. Introduction

The improvements achieved on commodity computers during the last two decades have changed the accessibility and availability of computing resources for research. Although supercomputers still play an important role for the research community, clusters and other infrastructures based on commodity computers such as Grid and Cloud infrastructures are widely used due to their low cost. This situation has promoted the spread of new computing facilities and, as a consequence, researchers can simulate in a wide range of computing resources. Today, most researchers have access to several clusters and Grid infrastructures and can rent on-demand Cloud resources to temporarily solve peak workloads [1–3]. The aggregation of these resources as a single Hybrid Distributed Computing Infrastructure (HDCI) can provide a great computing potential. This work introduces a framework for performing large experiments with climate models on HDCIs. The framework has been designed to access transparently these heterogeneous distributed environments providing a uniform interface to run simulations.

The heterogeneous and distributed nature of HDCIs poses new challenges to the applications willing to exploit them. Although there are several pieces of middleware that facilitate the use of HDCIs, there are still unsolved aspects. Moreover, applications with special requirements, such as data-intensive ones or those with long-term runs, require a workflow modification in order to make an efficient use of resources. Section 2 of this work describes the main issues a user finds when trying to port an application to an HDCI.

So far, some efforts have been made to run climate models on HDCIs. Several works have been devoted to adapt a climate model to perform a given experiment in a given Grid infrastructure [4–6]. These works do not solve the issues related to the heterogeneity of resources nor to the data management. Instead,

* Corresponding author.
E-mail address: valvanuz.fernandez@unican.es (V. Fernández-Quiruelas).

http://dx.doi.org/10.1016/j.future.2015.04.009
0167-739X/

they provide ad-hoc solutions that only work in a very limited set of computing resources. The main disadvantage of this approach is that the solution is not scalable nor reusable. In a previous work, Fernández-Quiruelas et al. [7] developed a framework for executing a climate model on a Grid infrastructure (the EGEE testbed). This first prototype was scalable and was able to exploit all the available Grid resources. However, it could not manage other computing infrastructures such as Clusters or Cloud resources and only allowed the execution of a given experiment. Blanco et al. [8] presented a scientific gateway focused on running climate workflows on Grid resources. This solution is useful to handle experiments with complex workflows among different Grid virtual organizations, but it lacks scalability and scheduling capabilities.

Other groups have explored new possibilities to run applications on HDCIs without a common middleware. Bretherton et al. [9] developed a framework based on web services that can be used to run different climate models by slightly modifying the job configuration. This solution provides some advanced features like the abstraction that allows the execution of different models. However, some important issues, such as job scheduling, efficient data management or recovery of failed simulations, were not handled.

The complexity of climate models poses challenges not only to the HDCI middleware, but also to the standard software used in a single cluster or supercomputer [10]. The execution of climate models usually involves managing complex workflows that produce large volumes of data and occasionally long-term runs lasting for weeks or even months. Furthermore, new trends in climate modeling employ ensemble prediction [11] to sample the uncertainties inherent to the simulations. This technique requires running multiple times the same simulation with varying parameters and thus complicates the experiment management even further. The growth in the number of independent simulations required to perform ensemble prediction has forced the community to find new sources of computing power. The independent nature of these simulations makes them suitable to run on HDCIs. Adapting these models to efficiently take advantage of HDCIs can enormously enlarge the computing power accessible for climate researchers.

In order to simplify the execution of climate and weather experiments, some institutions have created their own frameworks that allow the users to easily perform experiments on their computing facilities. These frameworks provide a set of commands and services that hide the complexity of configuring, running and monitoring all the simulations involved in an experiment. Among other features, these frameworks usually provide means for running the whole experiment workflow unattended and restarting part or the whole experiment in case of failure. The FMS Runtime environment (FRE), the ECMWF Supervisor monitor scheduler (SMS) or the IC3's Autosubmit[1] are some examples [10]. These frameworks have been designed to work with a given model and a single batch-queuing system (usually the one used in the developers institution). Adapting them to use different resource managers or computing configurations might involve a lot of work.

The framework shown in this paper encompasses many features already available in other institutional climate modeling frameworks (e.g. FRE, SMS and Autosubmit), facilitating the management and execution of climate experiments. The main contribution of our framework is the ability to combine heterogeneous, distributed resources to run the simulations. Additionally, its layered design allows to take advantage of most of the developments and easily port the framework to other climate models. Section 3, describes WRF4G, the framework created for running the WRF [12] atmospheric modeling system on HDCIs. This section describes the framework architecture and shows how other applications could also benefit from it to run on HDCIs. It is important to note that this work is not only useful for the climate community, but could also be of interest to other disciplines. Applications that require long running times, large data transfers or complex workflows could take advantage of this work.

In order to illustrate the usefulness of the framework, Section 4 shows how using WRF4G to access Cluster, Grid and Cloud resources simultaneously, the time spent in running a real climate experiment has been reduced 4 times. The experiment performed consisted of 365 independent simulations, which were executed using computing and data resources from our institutional Cluster, a remote Cloud and an EGI Grid infrastructure. Finally, Section 5 presents some conclusions and future work.

2. Hybrid distributed computing infrastructures challenges

Although the combination of Cluster, Grid and Cloud resources on a single HDCI can offer a great computing potential [3,13], leveraging these heterogeneous resources poses several challenges. The distributed nature of these infrastructures complicates tasks such as the monitoring and debugging of applications. Furthermore, combining Cluster, Grid and Cloud resources poses an additional challenge: providing a uniform interface that allows interoperability among different job managers. Interoperability of data resources is also an issue on these infrastructures. Below, we show the main difficulties users might find when executing their applications on an HDCI.

2.1. Application monitoring and debugging

In traditional computing infrastructures, users have direct access to the simulation working environment. They can track the simulation status or find errors inspecting the files generated by the application. Moreover, they can debug errors by re-running the simulation from the last checkpoint file or just stopping the job and running it again with debugging parameters. When the computing nodes belong to an HDCI, very often it is not possible to have direct access to the working directory as the simulations run or even when they have finished. The adaptation of the simulation environment to the policies of each computing resource (i.e. disk, memory and CPU quotas, scratch directory, interconnection among nodes) is another issue that has to be faced when running on HDCIs.

Therefore, a framework to run simulations on HDCIs must allow users to track and control their simulations, and to adapt the simulation environment to each site policy. To provide the framework with such capabilities, it is necessary to orchestrate and monitor the simulation workflow. One approach to application monitoring on HDCIs is the use of a wrapper that registers in a central database all the events produced by the simulation. This wrapper can also transfer the output and checkpoint datasets to the data repositories as they are being produced. Thanks to the checkpoint datasets, in case of crash, the simulations can continue from their last checkpoint, with minimal data loss. This wrapper will also prepare the execution environment (application paths, location of the parallel libraries or scratch file systems...).

Section 3 describes how the central database and WRF wrapper have been implemented in WRF4G.

2.2. Executing jobs on heterogeneous resources

Running computational jobs on heterogeneous infrastructures can be difficult due to the different middleware available. The variety of such middlewares is quite large, even on each HDCI: PBS [14], SGE [15], LSF [16], SLURM [17], LoadLeveler [18] and Condor [19] are examples of Cluster middlewares; gLite [20], Globus

[1] https://redmine.dkrz.de/collaboration/attachments/194/autosubmit.pdf.

Table 1
Comparison of software frameworks.

Framework	Resource support	Scheduler type
Condor-G	Grid, Cloud and Cluster	Matchmaking metascheduler
DIRAC	Grid, Cloud and Cluster	Pilot Jobs
gUSE/WS-PGRADE	Grid, Cloud and Cluster	Metabroker
GridWay	Grid and Cloud	Metascheduler

Toolkit [21], ARC [22] and UNICORE [23] are examples of Grid middlewares; and Eucalyptus [24], OpenNebula [25], OpenStack [26] and Nimbus [27] are examples of Cloud middlewares. These middlewares with different interfaces are seldom compatible with each other, creating substantial barriers to users. Fortunately, there are several successful software frameworks which aim at solving the job interoperability among HDCIs, such as DIRAC [28], gUSE/WS-PGRADE [29], GridWay [30] or Condor-G [31]. These frameworks provide generic solutions to simplify the user access to heterogeneous resources.

Table 1 summarizes the features of the mentioned frameworks, focusing on supported resources and scheduler type. Although these frameworks are aimed at facilitating the use of HDCIs, none of them offers a final solution. All frameworks, except GridWay, support running jobs on Grid, Cloud and Cluster infrastructures. However, the Local Resource Manager Systems (LRMS) supported for Clusters are usually PBS, LSF and Condor. Only DIRAC offers a larger variety in terms of these middlewares by enabling SGE as well.

There are different approaches to perform the job scheduling (Table 1). Condor-G supplies mechanisms to match job requirements against resource features (matchmaking process). Unfortunately, it does not support scheduling policies [31]. In DIRAC [32], the job scheduling relies on Pilot Jobs. Once on the computing node, Pilot Jobs contact a central server to get tasks to run. The main limitation of this approach is the requirement of an internet connection. Most Cluster policies do not allow these kind of connections, which make Pilot Jobs unusable for most HPC infrastructures. For these situations DIRAC provides a matchmaking scheduling. The metabroker concept posed by gUSE/WS-PGRADE aims at supporting interoperability among resources at workflow level [33]. Thus each component of the workflow can be run on a different resource. gUSE/WS-PGRADE does not provide a scheduling policy by delegating the job scheduling on the middlewares of the infrastructure chosen to run each workflow component.

To conclude, none of the above mentioned approaches implements scheduling policies to optimize the use of an HDCI. Only GridWay provides an adaptive scheduling, with fault recovery mechanisms and on-request and opportunistic job migration [34], which can operate independently from the infrastructure.

In order to provide efficient access to HDCIs, we decided to develop our own framework but, rather than developing from scratch, we took advantage of some capabilities of the GridWay metascheduler. We selected GridWay because of its unique adaptive scheduling [35] and its customization support, which allows to add new resource managers easily without modifying the code. This development will be described in detail in Section 3.3.

2.3. Data intensive applications

The increasing data processing demand in many scientific fields has made the management of storage resources and data transfers one of the biggest issues that users have to face when running applications on HDCIs. In HPC environments, data transfer time is negligible compared to the overall job execution time. Thus, most application workflows are optimized to reduce the CPU time. When these workflows are run in an HDCI, very often the data transfers become the bottleneck.

In climate modeling, for example, an experiment run in an HDCI using the traditional workflow would be inefficient because most of the time will be wasted transferring input and output data across the Internet. The typical workflow of a climate modeling experiment is composed of 3 tasks. First, the input data is transformed to fit the model requirements (preprocess). Second, the model simulations are submitted to the computing resource. These simulations will read the preprocessed data, and will store the raw output in the data storage. Third, the raw output is filtered (postprocessed), significantly reducing its size if the experiment focuses in a particular field of application.

To avoid the excessive data transfer, the experiment workflow in HDCIs has to be modified to integrate the postprocessing step inside the simulation execution in the computing node. An additional preprocessing has also to be performed locally, before the submission, to reduce the input data for each simulation. In a local cluster, the input for several simulations is often stored in a single file. In a remote resource, this extra data transfer would be a waste of time. Therefore, only the minimum required input data should be transferred. Performing these tasks is crucial in Cloud infrastructures, where the pay-as-you-go pricing model not only charges for the use of the instances, but also for long-term storage and network transfers.

Apart from modifying the execution workflow, the data management efficiency can be improved by using data-aware job schedulers. These take into consideration the location of the data required by the jobs. When data and computational resources are geographically dispersed, the use of a data-aware job scheduler can enormously reduce the job execution time. There are several works [36–38] that propose algorithms to develop this kind of job schedulers. These works relay on data replicas to optimize the scheduling strategy. Among the solutions proposed, the approach of Taheri et al. [37] best fits the characteristics of climate models; these authors present a scheduling methodology where "the overall makespan of executing all jobs as well as the overall delivery time of all data files to their dependent jobs is concurrently minimized". To date, none of these proposals have been implemented.

The use of file replicas can also strongly improve the speed and reliability of data transfers. Although many studies have been performed in order to optimize transfers using scheduling algorithms based on the server loads or bandwidth [39,40], to date, there are no solutions that provide a smart replica allocation. gLite LFC [41] and Globus RLS [42], two of the most commonly used replica services, let the user choose the replica data resource or let the system use one at random. It is important to note that both services only support replicas among GridFTP servers.

Although GridFTP is a *de facto* standard for data transfers in HDCIs, there are still several data repositories that do not support it. Instead, they provide less efficient protocols for transferring data such as RSYNC, SFTP or HTTP. This lack of homogeneity makes it difficult for the users to access them. Section 3 shows how this issue has been partially solved in the WRF4G framework.

3. WRF4G framework

WRF4G is a tool that simplifies the execution of atmospheric numerical simulation experiments with the Weather Research and Forecasting model (WRF [12]) in HDCIs. WRF is a limited area model, widely used due to its flexibility and modularity. It has been used in a variety of research and application areas, from weather forecasting to climate change projection.

WRF4G allows the execution of large-scale experiments efficiently combining heterogeneous, distributed resources. It provides full control of the simulations and means for restarting part or the whole experiment in case of failure. It also provides the ability of reproducing the experiment fully or partially. WRF4G is

an open-source and publicly available software that can be downloaded from the Santander Meteorology Group website[2] and installed in any Unix-like system.

3.1. Statement of the problem

Performing a climate experiment very often requires running multiple times the same simulation with varying parameters. These parameters can be different dates, input data, model configurations, etc. The computing requirements of a given simulation do not always fit the resources capabilities. In some cases, the input boundary data exceed the computing resource disk capacity and, in others, the simulation wall-time is longer than the queue limit. To avoid these restrictions, it is necessary to use the checkpointing/restart capabilities of climate models to split the simulations in a cascade of smaller chunks.

In this framework, each experiment comprises a set of independent *realizations* (at least one) that can be split into several *chunks* executed as dependent simulations (one chunk does not start until the previous chunk has finished). Thus, if an experiment consists of an ensemble of 100 independent realizations and each realization is split into 50 chunks, a total of 5000 chunks will have to be handled. Each chunk is submitted as a job to the HDCI. In case of failure, several jobs will be required to finish a chunk, and they will probably run on different resources.

The WRF4G framework is layered to separate the experiment design from the execution environment. An atmospheric simulation experiment is defined through two configuration files: one contains the scientific configuration of the experiment (start and end dates, model configuration, experiment setup, input data, postprocess to apply, etc.) and the other, the execution environment (number of MPI process to run, memory required, data repositories, etc.).

The WRF4G framework is composed of three services. The Experiment Management service (EMS) creates, monitors and manages the experiments according to the user requests. In order to provide transparent and unified access to HDCIs, the EMS interacts with the Job Management Service (JMS) and the Data Management Service (DMS). JMS and DMS are in charge of performing an efficient management of the computational and data resources, respectively.

3.2. Experiment management service

WRF4G incorporates a set of tools that facilitate the experiment management. All these tools relay on a python library (WRF4Glib.py) that provides functions to interact with the different components of the framework.

To manage the experiment, it is necessary to persist all the experiment information and status (Section 2.1) in a database. The WRF4G database is embedded in the framework and hidden to the users. In order to facilitate the interaction with the database, an API called *vdb* (virtual database) has been created. The WRF4G library uses *vdb* to provide high level functions that interact with the database. Although, the database used by default is the MySQL installed with the framework, other instances of MySQL can be used and, with a few modifications in the vdb API, any other relational database could be accessed. The database contains several tables (Experiment, Realization, Chunk, Job, …) where the configuration and status of the different components are recorded. With the information stored in the database, the status of the experiment execution can be monitored in real-time. More information about the schema and relationships among the database components can be found in the WRF4G website.

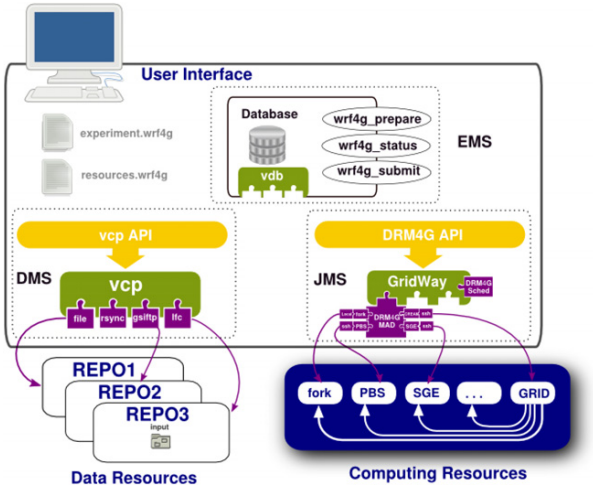


Fig. 1. Schema of the WRF4G components: Experiment Management Service (WRF4G commands, database and vdb), Data Management Service (vcp) and Job Management Service (DRM4G).

The EMS comprises the user tools, the WRF4G library and the database. The user tools requests are handled by the WRF4G library, which interacts with the database, the JMS and the DMS (see Fig. 1). To run an experiment, WRF4G goes through 3 phases:

Preparation The experiment configuration is analyzed and the details about the resulting realizations and chunks are recorded in the database. Configuration files for each of the chunks are also created and transferred to the data repositories using the DMS.

Submission The database is queried to obtain a list of the chunks that need to be run. These chunks are efficiently scheduled to the computing resources taking into consideration the dependencies among them. To do so, the EMS prepares the jobs templates and submits them to the JMS.

Execution The chunk execution in the computing node is driven by a wrapper script in charge of preparing the environment and orchestrating the run. The wrapper contains a monitor that tracks the events occurred during the model execution, updates them in the central database and manage the output files. The algorithm of the WRF4G monitor is shown in Algorithm 1. The location of the executables and libraries required to run WRF can be customized in the experiment configuration file for each computing resource. The serial, MPI, OpenMP and Hybrid (MPI–OpenMP) execution environments of WRF are supported. If the location of the executables and libraries required to run WRF is not supplied, a precompiled binary is transferred to the computing nodes during the environment preparation step. The precompiled binary includes OpenMPI, an MPI implementation characterized by its modularity and its adaptability to many LRMS.

3.3. Job management service

As mentioned above (Section 2.2), there is not a complete framework to manage HDCIs. Considering the options available, the GridWay metascheduler is well-suited to our purpose thanks to its modular design, with plugins named Middleware Access Drivers (MAD) [35]. GridWay's MAD architecture is flexible enough to enable the interoperability among the components of an HDCI.

[2] http://www.meteo.unican.es/software/wrf4g.

```
Data: Process ID
Result: exit code
while workflow still running do
    update chunk status;
    if new dataset then
        postprocess dataset;
        upload postprocessed dataset;
        set event in database;
        remove dataset;
    end
end
Clean up and upload logging info;
update chunk status in database;
exit(exit code);
```

Algorithm 1: WRF Monitor in charge of sending the events occurred during the execution to the central database

For instance, it has been tested on different Grid [43,44] and Cloud [45,46] infrastructures.

Although GridWay's approach eases the management of HDCIs, there are no plugins to access Cluster middlewares. DRM4G [47] is at the core of WRF4G and bridges the gap between GridWay and Cluster resources. It also provides an API (Fig. 1). DRM4G can define, submit, and manage computational jobs among Cluster, Grid and Cloud resources. It also provides a single point of control for these resources without installing any middlewares. Furthermore, DRM4G improves the GridWay adaptive scheduling enabling new parameters such as the maximum queued jobs or the maximum running jobs on each HDCI component.

Following a modular design like that of GridWay, DRM4G's MAD consists of two components: the protocol used to access resources (*communicator*) and the middleware used to manage jobs (*resource manager*). As a result, it can be easily customized by combining a *communicator* (*ssh*, *gsissh* or *local*) with a *resource manager* (*fork*, *sge*, *pbs*, *slurm*, *lsf*, *loadleveler*, *globus* or *cream*). DRM4G treats all *resource managers* and *communicators* equally, thus a user can even add Grid resources over SSH and GSISSH protocols. This unique approach makes DRM4G access many kinds of remote resources.

DRM4G has been designed to efficiently manage thousands of jobs, which is an essential feature for running large-scale experiments, as those requested by some climate modeling experiments (Menéndez et al. [48] show an example with more than 7000 jobs). To provide scalability, several management queues have been implemented inside DRM4G to handle job requests. These requests can now be dispatched without overloading the framework. DRM4G has also the ability to perform SSH channel multiplexing. This technique allows to handle each resource by using a single SSH connection. Otherwise, several concurrent SSH connections would be established between each resource and DRM4G server. This could lead to stability issues or even be considered an attack to the resource.

In summary, the combination DRM4G–GridWay solves the issue of efficiently handling different HDCI components by providing a homogeneous access to Cluster, Grid and Cloud resources. Both are embedded in the WRF4G framework and their use is transparent for users.

3.4. Data management service

The heterogeneity of data resources and its geographical distribution increases the difficulty of managing efficiently the data on an HDCI (see Section 2.3). The combination of different infrastructures involves the use of different data transfer protocols. Thus, the data repository used by a job depends on the infrastructure or resource where it will run. For instance, a job dispatched to a Grid resource will have to transfer the data using GridFTP. A job dispatched to a Cluster will use rsync, or just a local copy.

In order to handle this heterogeneous situation, WRF4G provides a tool called *vcp* (virtual copy). The aim is to hide the complexity of managing different transfer protocols. To date, vcp supports the following URLs: rsync, lfn (gLite LFC), gsiftp, sftp and http. Although vcp does not implement any replica algorithms, it is able to handle file replicas using the LFC (only GridFTP transfers).

Additionally, WRF4G has a fine-grained data management which allows to indicate the location of data repositories on a per-resource basis. The user can easily customize the input and output data repositories and, when a job starts to run on a resource, the data repository is selected depending on the infrastructure where the job is running.

3.5. Leveraging the framework to run other applications

The modularity of the framework proposed allows other applications to benefit from the WRF4G components to leverage HDCIs. Applications with no special requirements, such as long-term runs or detailed monitoring capabilities, can be easily adapted to HDCIs using DRM4G and vcp. Only a couple of commands to prepare the job templates and to submit the jobs should be created.

The framework deployment would be more complex for an application requiring monitoring. First, it would be necessary to create a new database schema. Some database components, such as the Jobs or Events tables, could be copied from the WRF4G schema. Then, the user tools and the WRF4G library should be modified to fit the new schema. Most of the WRF4G library functions will be reusable. The highest level part of the library should be rebuilt, but the functions that interact with the database, DRM4G and vcp, that are the most complex part, could be exploited.

We have leveraged the framework presented in this paper to port the Community Atmospheric Model (CAM) to HDCIs and we are currently adapting the framework to work with other two regional climate models: COSMO-CLM and RegCM4. As the characteristics of these models are very similar to WRF, only a few changes in the database and in the library are required.

4. An illustrative example of WRF4G

In order to illustrate the usefulness of the framework proposed, we rerun on an HDCI resource a wind simulation experiment that had been previously executed in a single cluster. The use of HDCI significantly increased the computational capacity available for the simulation. In this section, we show how, leveraging different computing infrastructures, the overall execution time was reduced 4 times. This was done without additional effort in porting the application to the different infrastructures and with the output data gathered in a common place at the end of the simulation.

The experiment performed for this example aims at simulating the wind behavior over the Mediterranean basin during one year. To do so, an ensemble of 365 independent realizations was run. Scientific details of the experiment can be found in Menéndez et al. [48].

The input for each realization were 230 MB of global reanalysis data (84 GB for the whole experiment). For each realization, the WRF model produced 2 GB of meteorological variables at 15-km resolution and hourly time step. As the main goal of the study focused on wind, we selected just this variable, reducing the output produced to 40 MB. Thus, the data output for the whole experiment was reduced from 730 to 14 GB.

The reference execution in our local cluster (c1_intel, see Table 2) took ∼60 h using 18 nodes (144 cores, Intel Xeon E5620). The execution time for each realization was around 175 min using 8 MPI parallel processes. Given that the realization execution time was short, in this particular example it was not necessary to split the realizations into chunks.

Table 2
Characteristics of the resources used for running the Experiments 2 and 3, showing for each resource the name, type, number of nodes, number of cores available, Communicator used to access it (Comm.), Resource Manager (RM), operating system installed in the computing nodes (OS) and data repository (Repo.) used for input and output data in Experiment 2.

Name	Type	Nodes	Cores	Comm.	RM	OS	Repo.
c1_intel	Cluster	18	144	Local	PBS	Centos 6	Local
c1_amd	Cluster	8	64	Local	PBS	Centos 6	Local
c2_long	Cluster	4	32	SSH	SGE	Debian 7	rsync
cloud	Cloud	4	32	SSH	PBS	Centos 7	rsync
EGI	Grid	14,246	136,352	Local	CREAM	SL 6	gsiftp/lfn

4.1. Experimental setup

Three different experiments were performed to test the framework. All of them were performed using the WRF4G framework and were run in jobs using 8 MPI parallel processes.

The first experiment was run in a single cluster (c1_intel) and it matches exactly the configuration of Menéndez et al. [48] SeaWindI experiment. This configuration was preserved in all experiments, except for the list of computing and data resources.

The second and third experiments were run on an HDCI. In order to create a realistic scenario, we combined different computing infrastructures supported by the framework. It is important to note that all the computing infrastructures, except for the Cloud resource, were shared with other researchers, and no resource reservation was made. The WRF4G framework was installed in the submission machine of the research group cluster (c1_intel), which also acts as a Grid user interface. The list of resources available for running the experiments (i.e. the HDCI) is shown in Table 2.

The computing resources *c1_intel* and *c1_amd* are two queues of our research group cluster, featuring processors from different makers (Intel and AMD, respectively). *EGI* is the European Grid Infrastructure. We used the Earth Science Virtual Organization[3] from EGI, which provides access to 136,352 cores from 61 different sites distributed all over the world. *cloud* is a private Cloud infrastructure managed with OpenNebula where we deployed 4 computing nodes. *c2_long* is the cluster of other research group from our university, included in the HDCI for the purpose of illustrating the access to an additional local resource manager (SGE) and OS (debian 7) via SSH.

In order to show that the use of replica file services is essential when running large-scale experiments in HDCIs, we performed two different experiments on this HDCI (Experiment number 2 and 3). In Experiment 2, a single data repository was used (data.unican.es). The *EGI* resources accessed this repository using the GridFTP protocol, *c1_intel* and *c1_amd* have the repository directly mounted in the nodes (local) and *c2_long* and *cloud* accessed through rsync. In Experiment 3 the input data accessed by jobs running in EGI were replicated in 5 different GridFTP data repositories using the LFC service (under the logical file name lfn://grid/esr/seawind). This process replicated 365 files from data.unican.es to 5 different EGI storage repositories, transferring a total of 420 GB of data. This operation took 6 h.

The output data were stored in-house in a common location (data.unican.es). There was no bottleneck for these data, because they were transferred as they were being created. Furthermore, the total output size was much smaller than the input.

During the execution of Experiment 2, the data repository crashed because the server run out of memory (80 GridFTP processes were downloading data simultaneously). After tuning the configuration of the GridFTP service, the realizations were able to run. However, the jobs executed in EGI resources hanged while downloading the data. The problem was caused by a bottleneck in the Internet connection of the data repository that resulted in a very slow transfer rate in the GridFTP connections. Under these circumstances, we canceled Experiment 2 and no statistics are shown. This illustrates the essential role of replica services in deploying data intensive applications on HDCIs.

[3] http://www.euearthsciencegrid.org/.

Table 3
Job statistics for Experiment 3.2, showing for each resource the number of jobs run (Jobs) and the average and standard deviation (in parenthesis) of the time spent waiting to run (Wait), Downloading input data (Download) and running WRF (WRF). Times are expressed in minutes.

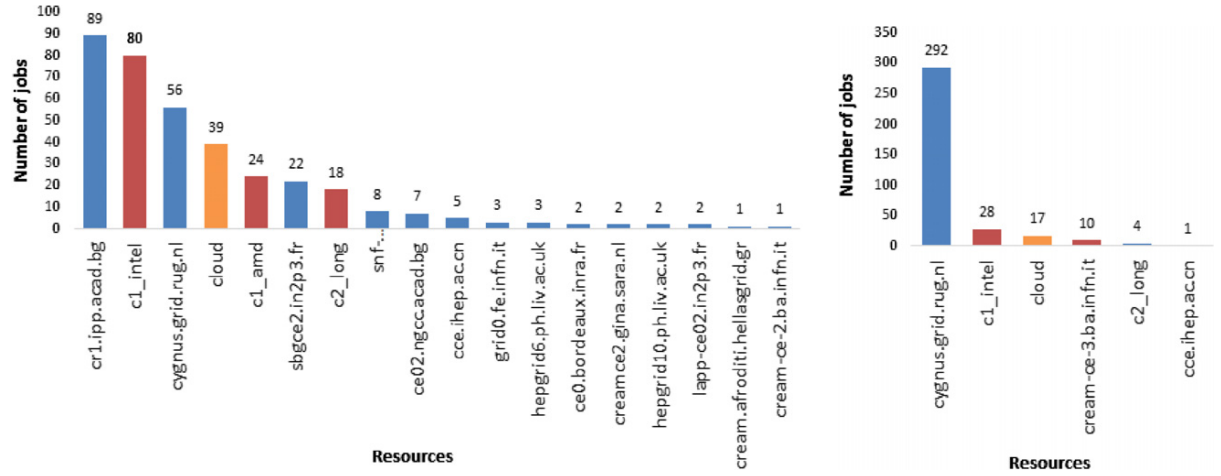
Resource	Jobs	Wait	Download	WRF
cygnus.grid.rug.nl	292	262 (212)	4.8 (0.8)	119 (6)
c1_intel	24	290 (100)	0.8 (0.1)	169 (17)
cloud	17	239 (164)	1.3 (0.2)	79 (1)
c1_amd	11	231 (102)	0.6 (0.1)	171 (24)
cream-ce-3.ba.infn.it	7	123 (81)	4.0 (0.3)	111 (22)
c2_long	3	461 (1)	1.2 (0.1)	103 (1)
cce.ihep.ac.cn	1	10	12.2	194

4.2. Job execution statistics

Experiment 3 was executed twice (Experiments 3.1 and 3.2) to show the importance of the scheduling policy when running in heterogeneous environments. The DRM4G scheduling policy penalizes the resources when they do not behave as expected. Thus, when a site is publishing available CPUs but jobs queued on those resources do not start to run after a given time (one hour in the configuration used to run these experiments), the scheduler penalizes these sites and migrates the jobs to other resources. The same happens when jobs crash without a reason.

In the execution of Experiment 3.1, the scheduler did not have previous information about the resources. Experiment 3.2 was launched afterwards, once the scheduler was "trained". The default behavior of DRM4G scheduler is dispatching the jobs to the resources with fastest cores. As the fastest computer resource was publishing more than 1920 free cores (365 jobs x 8 MPI processes), all the jobs were submitted to that resource. Unfortunately, only one job started to run. After one hour, DRM4G migrated all those queued jobs to other resources. This situation was repeated with other EGI resources; most resources with 3 jobs or less in Fig. 2(a) suffered the same problem. Some jobs run in EGI failed because they reached the resource storage quotas. The waste of time while jobs were queuing or running before they die in Experiment 3.1, together with a temporary better availability of free resources in Experiment 3.2 are the main reasons why the Experiment 3.2 was twice faster than the 3.1 (see Fig. 3). It is important to note that even the inefficient execution in the HDCI (Experiment 3.1) was twice faster than the execution on a single cluster (Experiment 1). As shown in Fig. 2(b), Experiment 3.2 used less computing resources as a result of the change in the scheduling priority, which banned some sites.

Table 3 shows some job statistics for Experiment 3.2. Data download times are significantly larger in the Grid resources. However, the input data retrieval time is still negligible compared to the computing time (WRF) in the node, so the use of Grid resources is still worth.



(a) Experiment 3.1. (b) Experiment 3.2.

Fig. 2. Distribution of the jobs among the different computing resources. (a) first trial on the HDCI (Experiment 3.1) (b) second trial, after the scheduling policy was updated (Experiment 3.2).

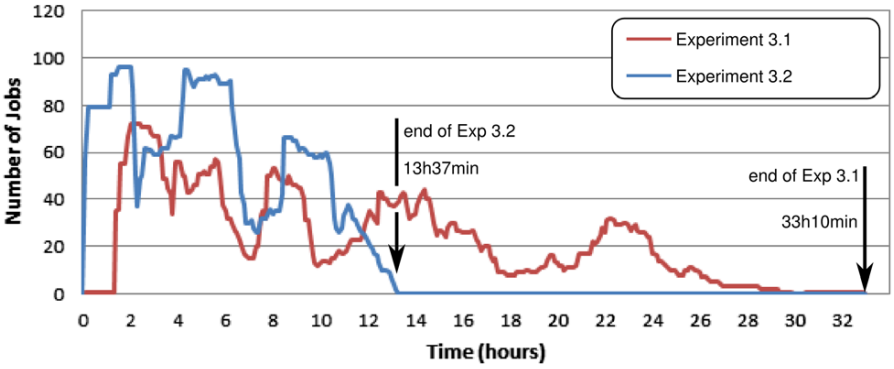


Fig. 3. Number of jobs running vs. time (in hours) since the experiment submission.

5. Summary and conclusions

This work has introduced a new framework for running complex applications in HDCIs. The framework frees the user from tedious tasks such as the experiment workflow management, the monitoring or the data transfers, providing at the same time a homogeneous access to the computing and storage resources. It fits the purpose of climate modeling, which is characterized by large data transfers and long execution times. WRF4G is an implementation of the above framework to run a regional climate model (WRF) taking advantage of an HDCI. The WRF4G framework has already been used for scientific production runs, contributing to several works [49,50,48].

The new framework has been tested on an HDCI using a realistic experiment consisting of 365 simulations of wind speed over the Mediterranean area. The efficient use of the resources led to saving more than 75% of the execution time, as compared to the same simulations run on local resources. WRF4G benefited from the additional heterogeneous resources without increasing the complexity for the user. Moreover, the ability to add on-demand resources is very useful to solve peak-loads or to guarantee service-level agreements by renting Cloud resources.

All output data were transparently sent back to a common local repository as if local computing resources would have been used. Input data replication has been shown crucial to sustain a large scale experiment. Otherwise (Experiment 2), the initial concurrent access to a single data resource acts as a bottleneck.

Also, the training of the scheduling policy (Experiment 3.1 vs 3.2) seems to introduce strong differences in the final execution time. It should be noted that, due to the intermittent availability of some of the resources in the HDCI used in the test (mainly the Grid component), the results may vary strongly if repeated. In fact, the differences between Experiments 3.1 and 3.2 include not only the training of the scheduler, but also these differences in the resource availability. One of the features of HDCIs is this variability of resources, which prevents a detailed evaluation of the impact of different approaches.

Although WRF4G has been adapted to the WRF workflow, the modularity of the framework allows other climate models with similar characteristics to be easily ported. Moreover, applications from other disciplines can also benefit from most of the framework components shown in this work (DRM4G, vcp, wrapper services, etc.).

WRF4G is an active project and it is continuously adding new features. An interesting addition to the current framework would be to provide GridWay with data-aware scheduling. This capability would enormously improve the overall experiment execution time. In this regard, Taheri et al. [37] show how the algorithm they propose can be easily integrated in the GridWay scheduler policies.

Acknowledgments

This work has been supported by the Spanish National R&D Plan under projects WRF4G (CGL2011-28864, co-funded by

the European Regional Development Fund—ERDF) and CORWES (CGL2010-22158-C02-01) and the IS-ENES2 project from the 7FP of the European Commission (grant agreement no. 312979). C.B. acknowledges financial support from Programa de Personal Investigador en Formación Predoctoral from Universidad de Cantabria, co-funded by the regional government of Cantabria. The authors are thankful to the developers of third party software (e.g. GridWay, WRFV3, python and netcdf), which was intensively used in this work. the authors are also thankful to the reviewers who contributed to improve the final manuscript.

References

[1] A. Fox, Cloud computing—What's in it for me as a scientist? Science 331 (6016) (2011) 406–407.
[2] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B.P. Berman, P. Maechling, Scientific workflow applications on amazon EC2, in: 2009 5th IEEE International Conference on E-Science Workshops, IEEE, 2009, pp. 59–66.
[3] G. Mateescu, W. Gentzsch, C.J. Ribbens, Hybrid computing—Where HPC meets grid and cloud computing, Future Gener. Comput. Syst. 27 (5) (2011) 440–453.
[4] D. Davidović, K. Skala, D. Belušić, M. Telišman Prtenjak, Grid implementation of the weather research and forecasting model, Earth Sci. Inform. 3 (4) (2010) 199–208.
[5] K. Lagouvardos, E. Floros, V. Kotroni, A grid-enabled regional-scale ensemble forecasting system in the mediterranean area, J. Grid. Comput. 8 (2) (2010) 181–197.
[6] A. Sulis, GRID computing approach for multireservoir operating rules with uncertainty, Environ. Model. Softw. 24 (7) (2009) 859–864.
[7] V. Fernández-Quiruelas, J. Fernández, A. Cofiño, L. Fita, J. Gutiérrez, Benefits and requirements of grid computing for climate applications. an example with the community atmospheric model, Environ. Model. Softw. 26 (9) (2011) 1057–1069.
[8] C. Blanco, A. Cofiño, V. Fernández-Quiruelas, 2013, WRF4SG: A scientific gateway for the weather research and forecasting model, in: 36th International Convention on Information Communication Technology Electronics Microelectronics, MIPRO, 2013, pp. 172–176.
[9] D.A. Bretherton, J.D. Blower, K. Haines, G.C. Smith, Running climate models on grids using G-Rex, Phil. Trans. R. Soc. A 367 (1890) 847–853.
[10] R. Redler, R. Budich, R. Ford, G. Riley, Earth System Modelling, first ed., in: Tools for Configuring, Building and Running Models, vol. 5, Springer, 2012.
[11] T.N. Palmer, The economic value of ensemble forecasts as a tool for risk assessment: From days to decades, Q. J. R. Meteorol. Soc. 128 (581) (2002) 747–774.
[12] W. Skamarock, J. Klemp, J. Dudhia, D. Gill, D. Barker, M. Duda, W. Wang, J. Powers, A Description of the Advanced Research wrf version 3, Technical Report, NCAR, 2008.
[13] R. Montella, I. Foster, Using hybrid Grid/Cloud computing technologies for environmental data elastic storage, processing, and provisioning, in: B. Furht, A. Escalante (Eds.), in: Handbook of Cloud Computing, Springer, US, Boston, MA, 2010, pp. 595–618.
[14] R.L. Henderson, Job scheduling under the portable batch system, in: D.G. Feitelson, L. Rudolph (Eds.), Job Scheduling Strategies for Parallel Processing, in: Lecture Notes in Computer Science, vol. 949, Springer, Berlin Heidelberg, 1995, pp. 279–294.
[15] W. Gentzsch, 2001, Sun grid engine: towards creating a compute power grid, in: First IEEE/ACM International Symposium on Cluster Computing and the Grid, Proceedings, 2001, pp. 35–36.
[16] S. Zhou, LSF: Load sharing in large-scale heterogeneous distributed systems, in: Proceedings of the Workshop on Cluster Computing, 1992.
[17] A.B. Yoo, M.A. Jette, M. Grondona, SLURM: Simple linux utility for resource management, in: D. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing, in: Lecture Notes in Computer Science, vol. 2862, Springer, Berlin, Heidelberg, 2003, pp. 44–60.
[18] S. Kannan, M. Roberts, P. Mayes, D. Brelsford, J. Skovira, Workload management with loadleveler, IBM Redbooks 2 (2001) 2.
[19] M. Litzkow, M. Livny, M. Mutka, 1988, Condor-a hunter of idle workstations, in: 8th International Conference on Distributed Computing Systems, 1988, pp. 104–111.
[20] E. Laure, C. Gr, S. Fisher, A. Frohner, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, R. Byrom, L. Cornwall, M. Craig, A.D. Meglio, A. Djaoui, F. Giacomini, J. Hahkala, F. Hemmer, S. Hicks, A. Edlund, A. Maraschini, R. Middleton, M. Sgaravatto, M. Steenbakkers, J. Walk, A. Wilson, Programming the grid with gLite, in: Computational Methods in Science and Technology, 2006.
[21] I. Foster, C. Kesselman, Globus: A metacomputing infrastructure toolkit, Int. J. Supercomput. Appl. 11 (1996) 115–128.
[22] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J.L. Nielsen, M. Niinimäki, O. Smirnova, A. Wäänänen, Advanced resource connector middleware for lightweight computational grids, Future Gener. Comput. Syst. 23 (2) (2007) 219–240.
[23] J. Almond, D. Snelling, UNICORE: uniform access to supercomputing as an element of electronic commerce, Future Gener. Comput. Syst. 15 (5–6) (1999) 539–548.
[24] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, Eucalyptus: an open-source cloud computing infrastructure, J. Phys. Conf. Ser. 180 (1) (2009) 012051.
[25] B. Sotomayor, R.S. Montero, I. Llorente, I. Foster, Virtual infrastructure management in private and hybrid clouds, IEEE Internet Comput. 13 (5) (2009) 14–22.
[26] A. Corradi, M. Fanelli, L. Foschini, VM consolidation: A real case based on OpenStack Cloud, Future Gener. Comput. Syst. 32 (2014) 118–127.
[27] N.T.D. Tran Van Lang, Deploying business virtual appliances on open source cloud computing, Int. J. Comput. Sci. Telecommun. (ISSN: 2047-3338) 3 (4) (2012) 26–30.
[28] A. Tsaregorodtsev, M. Bargiotti, N. Brook, A.C. Ramo, G. Castellani, P. Charpentier, C. Cioffi, J. Closier, R.G. Diaz, G. Kuznetsov, Y.Y. Li, R. Nandakumar, S. Paterson, R. Santinelli, A.C. Smith, M.S. Miguelez, S.G. Jimenez, DIRAC: a community grid solution, J. Phys. Conf. Ser. 119 (6) (2008) 062048.
[29] Z. Farkas, P. Kacsuk, P-GRADE portal: A generic workflow system to support user communities, Future Gener. Comput. Syst. 27 (5) (2011) 454–465.
[30] E. Huedo, R.S. Montero, I.M. Llorente, The GridWay framework for adaptive scheduling and execution on grids, Scalable Comput.: Pract. Exp. 6 (3) (2001).
[31] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-g: A computation management agent for multi-institutional grids, Clust. Comput. 5 (3) (2002) 237–246.
[32] A. Casajus, R. Graciani, S. Paterson, A. Tsaregorodtsev, t.L.D. Team, DIRAC pilot framework and the DIRAC workload management system, J. Phys. Conf. Ser. 219 (6) (2010) 062049.
[33] M. Kozlovszky, K. Karóczkai, I. Márton, P. Kacsuk, T. Gottdank, DCI bridge: Executing WS-PGRADE workflows in distributed computing infrastructures, in: P. Kacsuk (Ed.), Science Gateways for Distributed Computing Infrastructures, Springer International Publishing, 2014, pp. 51–67.
[34] J. Vázquez-Poletti, E. Huedo, R. Montero, I. Llorente, A comparison between two grid scheduling philosophies: EGEE WMS and Grid Way, Multiagent Grid Syst. 3 (4) (2007) 429–439.
[35] E. Huedo, R.S. Montero, I.M. Llorente, A modular meta-scheduling architecture for interfacing with pre-WS and WS grid resource management services, Future Gener. Comput. Syst. 23 (2) (2007) 252–261.
[36] T. Kosar, M. Balman, A new paradigm: Data-aware scheduling in grid computing, Future Gener. Comput. Syst. 25 (4) (2009) 406–413.
[37] J. Taheri, A.Y. Zomaya, P. Bouvry, S.U. Khan, Hopfield neural network for simultaneous job scheduling and data replication in grids, Future Gener. Comput. Syst. 29 (8) (2013) 1885–1900.
[38] N. Mansouri, G.H. Dastghaibyfard, E. Mansouri, Combination of data replication and scheduling algorithm for improving data availability in data grids, J. Netw. Comput. Appl. 36 (2) (2013) 711–722.
[39] N. Saadat, A.M. Rahmani, PDDRA: a new pre-fetching based dynamic data replication algorithm in data grids, Future Gener. Comput. Syst. 28 (4) (2012) 666–681.
[40] T. Amjad, M. Sher, A. Daud, A survey of dynamic replication strategies for improving data availability in data grids, Future Gener. Comput. Syst. 28 (2) (2012) 337–349.
[41] J.-P. Baud, J. Casey, S. Lemaitre, C. Nicholson, Performance analysis of a file catalog for the LHC computing grid, in: 14th IEEE International Symposium on High Performance Distributed Computing, HPDC-14, Proceedings, 2005, pp. 91–99.
[42] A. Chervenak, R. Schuler, M. Ripeanu, M. Amer, S. Bharathi, I. Foster, A. Iamnitchi, C. Kesselman, The globus replica location service: design and experience, IEEE Trans. Parallel and Distrib. Syst. 20 (9) (2009) 1260–1272.
[43] J. Vázquez-Poletti, E. Huedo, R. Montero, I. Llorente, Coordinated harnessing of the IRISGrid and EGEE testbeds with GridWay, J. Parallel Distrib. Comput. 66 (5) (2006) 763–771.
[44] I.M. Carrión, E. Huedo, I.M. Llorente, Interoperating grid infrastructures with the GridWay metascheduler, Concurrency Comput.: Pract. Exp. (2012).
[45] C. Vazquez, E. Huedo, R.S. Montero, I.M. Llorente, Dynamic provision of computing resources from grid infrastructures and cloud providers, in: Grid and Pervasive Computing Conference, GPC '09, Workshops at the. IEEE, 2009, pp. 113–120.
[46] C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente, On the use of clouds for grid resource provisioning, Future Gener. Comput. Syst. 27 (5) (2011) 600–605.
[47] A. Cofiño, C. Blanco, V. Fernández-Quiruelas, Aggregation of Grid and HPC resources for running huge experiments in climate and weather prediction, in: EGU General Assembly, vol. 13, 2011, p. 13194.
[48] M. Menéndez, M. García-Díez, L. Fita, J. Fernández, F.J. Méndez, J.M. Gutiérrez, High-resolution sea wind hindcasts over the mediterranean area, Clim. Dynam. 42 (7–8) (2014) 1857–1872.
[49] G. Nikulin, C. Jones, F. Giorgi, G. Asrar, M. Büchner, R. Cerezo-Mota, O.B. Christensen, M. Déqué, J. Fernandez, A. Hänsler, E. van Meijgaard, P. Samuelsson, M.B. Sylla, L. Sushama, Precipitation climatology in an ensemble of CORDEX-africa regional climate simulations, J. Clim. 25 (18) (2012) 6057–6078.
[50] M. García-Díez, J. Fernández, L. Fita, C. Yagüe, Seasonal dependence of WRF model biases and sensitivity to PBL schemes over europe, Q. J. R. Meteorol. Soc. 139 (671) (2013) 501–514.

Ms. Valvanuz Fernández is Ph.D. candidate at Dep. of Applied Mathematics and Computer Sciences of the University of Cantabria. She holds a Master degree in Mathematics and Computer Science. Her main research activity is the adaptation of climate models to heterogeneous distributed infrastructures.

Mr. Carlos Blanco is a research fellow at the Santander Meteorology Group in the University of Cantabria since 2009. He received his M.Sc. from the University of Cantabria in 2010. He started Ph.D. studies at the University of Cantabria in 2011. His research interest includes grid computing, interoperability solutions, and compute and data meta-scheduling on DCIs. He is lead developer of the open-source DRM4G framework.

Dr. Antonio S. Cofiño is Associate Professor at Dep. of Applied Mathematics and Computer Science of the University of Cantabria and Head of Distributed Computing in Earth Science Research in the Santander Meteorology Group. His main research areas are the development and deployment of earth science applications (e.g., global and regional climate models) to run in geographically distributed data and computing environments (GRID computing) with research contributions in both Earth Science and GRID computing. He has experience in regional modeling, high-performance computing and data management.

Dr. Jesús Fernández is Associate Professor at Dep. of Applied Mathematics and Computer Science of the University of Cantabria and Head of Numerical Modeling of the Atmosphere in the Santander Meteorology Group. His research interests include regional climate modeling, its sensitivity to physical parameterizations and uncertainty estimation through ensemble techniques.

CAPÍTULO 3

Discusión final y conclusiones

3.1. Discusión y resultados de la tesis

El objetivo final de esta tesis es mostrar a la comunidad del modelado climático las posibilidades ofrecidas por las e-infraestructuras *Grid* y, en general, por la agregación de recursos computacionales distribuidos geográficamente. Para ello, se han identificado los requerimientos de los modelos climáticos para su ejecución en infraestructuras distribuidas y se ha puesto de manifiesto que es necesario el uso de un entorno de trabajo que permita automatizar la preparación y ejecución de los experimentos. Para demostrarlo, se ha diseñado e implementado un nuevo entorno de trabajo que ha sido utilizado por dos modelos climáticos para su ejecución en *Grid* y HDCI. En el presente capítulo se ponen en común los resultados más importantes, se discuten, y se extraen conclusiones de la línea de investigación seguida en esta tesis.

Se puede decir que esta línea de investigación comenzó en el momento que lanzamos las primeras simulaciones con un modelo global de clima a la infraestructura *Grid* de EELA y se puso de manifiesto el esfuerzo que iba a suponer conseguir que acabasen de manera satisfactoria. Al no existir estudios previos sobre la ejecución de modelos climáticos en *Grid*, se siguió la misma metodología usada para la ejecución de aplicaciones en biomedicina y física de altas energías, que habían encontrado en la tecnología *Grid* una forma de satisfacer sus crecientes requerimientos de cálculo (Kasam et al., 2009; Aad et al., 2010), consistente en utilizar solamente las herramientas proporcionadas por el *middleware Grid*.

En las primeras pruebas, se ejecutaron en la infraestructura de EELA las simulaciones que habitualmente se ejecutaban en el *cluster* del grupo de investigación y los resultados no fueron los esperados. El límite en el tiempo de ejecución de los trabajos fue un factor crítico, y ninguna de las simulaciones finalizó correctamente. Reduciendo el tiempo de ejecución de las pruebas, los resultados mejoraron sensiblemente, aunque continuaron sin ser satisfactorios, ya que sólo el 25 % de los trabajos terminó sin errores.

Las fuentes de fallo detectadas: configuraciones erróneas de los sitios, fallos hardware y software en los nodos de cálculo y errores de *middleware*, fueron descritas posteriormente en Fernández-Quiruelas et al. (2009a). Esta caracterización de los errores para infraestructuras *Grid*, coincide con otras publicaciones de la época y posteriores (Neocleous et al., 2006; Dabrowski, 2009; De Almeida et al., 2013). Aunque la tasa de fallos fue más elevada de lo habitual si se compara con datos similares de la época (Jacq et al., 2008), el origen de los errores se debe a la naturaleza heterogénea y distribuida de la infraestructura, por lo que afecta, en mayor o menor medida, a todo tipo infraestructuras *Grid*. Por esto, los esfuerzos realizados para controlar estas fuentes de error son válidos para cualquier tipo de infraestructura y *middleware*.

Tras este análisis de fallos, se concluyó que era necesario el desarrollo de una herramienta que gestionase el flujo de ejecución de las simulaciones, permitiendo el uso de ficheros de control para reiniciarlas en caso de fallo. Para demostrarlo, se creó GRID-CAM, un prototipo de entorno de ejecución para realizar simulaciones con el modelo CAM en *Grid*. El desarrollo de este entorno de ejecución y de sus implementaciones posteriores (CAM4G y WRF4G), se ha basado en metodologías ágiles de desarrollo de software (Martin, 2003), en las que cada iteración del ciclo de vida de la aplicación incluye las siguientes fases: planificación, análisis de requisitos, diseño, codificación, pruebas y documentación.

En la primera iteración de este desarrollo, se realizó un análisis de los requerimientos de los modelos climáticos y se detectó que, además de la gestión desatendida del flujo de ejecución, era necesario proporcionar mecanismos de monitorización y de gestión de las transferencias de datos. Aunque en Bretherton et al. (2009) también se pusieron de manifiesto los requerimientos de gestión de transferencias de datos y monitorización, hasta el año 2013, en el que se publicaron Yalew et al. (2013); Zhao et al. (2013), nadie había considerado los requerimientos de gestión desatendida del flujo de ejecución. Este aspecto es fundamental para la adaptación de los experimentos a características de los recursos y para su correcta finalización en entornos no controlados con altas tasas de fallo.

La primera implementación del entorno de ejecución se apoyó en los servicios del *middleware* gLite. Tras un primer análisis, se decidió sustituir el gestor de trabajos

de gLite (WMS) por Gridway, por adaptarse mejor a los requerimientos de planificación y de control y detección de errores. Para facilitar el despliegue de CAM, en lugar de seguir la literatura y práctica de la época (Lagouvardos et al., 2010; Davidović et al., 2010), en la que se había sustituido la técnica de compilar los programas en los nodos de cálculo por su instalación a demanda, se decidió transferir, al comienzo de cada trabajo, un ejecutable compilado previamente. El código fuente había sido modificado para realizar llamadas al sistema que se encargaban de las tareas de monitorización y gestión de datos. La técnica de utilizar un ejecutable compilado estáticamente que se transfiere al nodo de cálculo como datos de entrada, presenta varias ventajas frente a la instalación previa del software: facilita la actualización a nuevas versiones y proporciona acceso a todos los recursos computacionales disponibles sin depender de su instalación en cada sitio. A pesar de que el uso de esta técnica es común en áreas como la biomedicina (Kasam et al., 2009), en la que las aplicaciones no tienen dependencias de otros software o librerías, Fernández-Quiruelas et al. (2009a) fue el primer trabajo en utilizar esta técnica para una aplicación compleja.

La prueba de este prototipo se realizó con un experimento compuesto por 50 simulaciones del fenómeno *El Niño* ejecutadas con el modelo CAM en la infraestructura *Grid* de EELA (experimento 1). Los resultados, publicados en Fernández-Quiruelas et al. (2009a), fueron los primeros en mostrar ejecuciones de modelos climáticos en e-infraestructuras *Grid*. Hasta el día de hoy, ningún grupo ha publicado resultados similares en este tipo de infraestructuras.

Los primeros experimentos con modelos climáticos llevados a cabo usando *middleware Grid* fueron ejecutados en dos supercomputadores de TeraGrid y publicados en Nefedova et al. (2006). Si bien, parte del trabajo realizado puede ser aprovechado en e-infraestructuras *Grid*, al no tener que resolver problemas propios de estas infraestructuras, como los derivados del bajo ancho de banda de la red o del uso de múltiples recursos heterogéneos, la aproximación presentada en este trabajo no cumple los requerimientos de esta tesis. Lo mismo sucede con otros trabajos (Lagouvardos et al., 2010; Davidović et al., 2010) en los que sólo se seleccionó una pequeña parte de los recursos de una e-infraestructura. El resto de publicaciones sobre el tema (Sulis, 2009; Davidović et al., 2013; Danovaro et al., 2014), describen la implementación desarrollada, pero no muestran resultados sobre la ejecución de las simulaciones.

A pesar de la cantidad de errores ocurridos durante la ejecución de las simulaciones, el nuevo entorno de trabajo GRID-CAM fue capaz de detectarlos y sobreponerse a ellos, consiguiendo terminar las simulaciones en un tiempo similar al que se hubiese tardado usando todos los recursos del *cluster* del grupo de investigación. Debido al bajo ancho de banda de la red, el tiempo destinado a la transferencia de datos supuso una parte significativa del tiempo total de ejecución del experimento. También se

detectó que los componentes de gLite AMGA y LFC no eran robustos (dejaban de funcionar sin causa aparente) y no escalaban adecuadamente, siendo causa de gran parte de los errores producidos durante la ejecución.

Para proporcionar robustez y escalabilidad, en la siguiente iteración del desarrollo (CAM4G), se sustituyó AMGA por una base de datos MySQL y, tras estudiar alternativas y no encontrar ninguna que se adaptase a los requerimientos, se diseñó e implementó un gestor de réplicas de datos. También se alteró el mecanismo de control y monitorización de trabajos en los nodos, sustituyendo las modificaciones en el código fuente por *scripts* que controlaban la evolución de la ejecución a través de búsquedas de patrones en los ficheros de registro (*logs*). La necesidad de buscar en el código fuente los puntos en los que insertar las llamadas al sistema, dificultaba la actualización a nuevas versiones del modelo y la reutilización de este entorno de trabajo para ejecutar otros modelos. En esta fase, se probó la ejecución de la versión MPI de CAM. Los resultados, pioneros en la ejecución de modelos climáticos en paralelo en infraestructuras gLite, coincidieron con otras publicaciones de la época (Childs, 2007; Dichev et al., 2008) en que la ejecución en MPI restringe el número de sitios disponibles y requiere un tratamiento específico para cada sitio.

Para realizar las pruebas de esta iteración de CAM4G, se ejecutó en la infraestructura *Grid* de EGEE el experimento 2, compuesto por 750 simulaciones de iguales características a las del experimento 1. La tabla 3.1 muestra un resumen de los principales resultados de ambos experimentos.

Nombre	**Infr.**	**Año**	**Num**	**T. cluster**	**T. Grid**	**Fin**	**Jobs/sim.**
Experimento 1	EELA	2008	50	7 días	7 días	82 %	10,2
Experimento 2	EGEE	2010	750	11 días	3,5 días	100 %	2,2

Tabla 3.1: Resultados obtenidos en los experimentos, en los que se muestra la infraestructura (Infr.) y año (Año) de ejecución, el número de simulaciones (Num), el tiempo requerido para su finalización en *cluster* (T.Cluster) y Grid (T.Grid), el porcentaje de simulaciones finalizadas (Fin) y el número de medio de trabajos requeridos para terminar una simulación (Jobs/sim).

Durante la ejecución del experimento 2 se detectaron menor cantidad de errores que en el experimento 1 y se consiguió que todas las simulaciones finalizasen (en el experimento 1 sólo terminaron el 82 %). Esta mejora se puede atribuir a 2 factores: la sustitución de AMGA y LFC por software más robusto y el perfeccionamiento, debido a la experiencia ganada con el tiempo, del *middleware* y las infraestructuras *Grid*. Es importante destacar que el principal objetivo del proyecto EELA era realizar tareas de diseminación para crear una nueva infraestructura *Grid* en América latina y que el experimento 1 se llevó a cabo al comienzo del despliegue de

esta infraestructura, por lo que todavía no se podía considerar en producción. Las estadísticas de errores y la caracterización de los requerimientos de los modelos climáticos presentados en Fernández-Quiruelas et al. (2009a) y Fernández-Quiruelas et al. (2011) contribuyeron a la mejora de las infraestructuras sucesoras de EELA (EELA-2 y GISELA).

En el experimento 2, hubo una gran diferencia de tiempo de ejecución entre las simulaciones. La heterogeneidad de procesadores y anchos de banda influyeron respectivamente en los tiempos de simulación y de transferencia de datos, pero no fueron la causa principal. Cada interrupción de un trabajo, supuso una penalización debido al tiempo requerido para reubicarlo en otro nodo de cálculo y para realizar la descarga de datos. Por ello, las simulaciones más rápidas en el experimento 2 fueron aquellas que se llevaron a cabo en un solo trabajo. Estas causas, también explican la diferencia entre el tiempo total de ejecución del experimento 2, frente al del experimento 1. En el experimento 1 cada simulación se realizó ejecutando de media 10,2 trabajos, con la penalización que ello supone, mientras que en el experimento 2, la media fue de 2,2. Estos resultados pusieron de manifiesto la importancia de elegir una política adecuada a la hora de adaptar los trabajos a las características de los recursos (lo que en inglés se conoce como *task partitioning*), algo que también fue mostrado posteriormente en Zhao et al. (2013) y Yalew et al. (2013).

El experimento 2 terminó en 3,5 días, frente a los 11 días que hubiese tardado en los recursos del *cluster* del grupo de investigación. Estos resultados supusieron una mejora significativa frente al experimento 1, en el que no todas las simulaciones habían concluido y el tiempo de ejecución fue igual que el del *cluster* (en el año 2008 el *cluster* contaba con 18 cores. En 2010 aumentó hasta los 128 cores).

Las conclusiones obtenidas de este segundo experimento fueron publicadas en Fernández-Quiruelas et al. (2011), mostrando por primera vez los beneficios y requerimientos de la utilización de e-infraestructuras *Grid* para la ejecución de aplicaciones climáticas.

Para demostrar que la arquitectura modular de CAM4G permitía la ejecución de otras aplicaciones en *Grid*, se utilizaron dos modelos de características muy diferentes. Por un lado, y como parte de las tareas de diseminación del proyecto EELA, adapté el modelo oceánico de predicción de olas WAM (WAMDI, 1988) para su ejecución en EELA. El *framework* fue creado de acuerdo a las necesidades de investigadores del *Coastal and Marine Research Centre* del *University College Cork* y utilizado para predecir las condiciones de oleaje en la costa irlandesa (Lassoued, 2009).

Por otro lado, se utilizó el mismo entorno de trabajo para ejecutar el modelo WRF en *Grid* (WRF4G). Dada la complejidad de los distintos tipos de experimentos llevados a cabo con WRF (ver sección 1.1.3), fue necesario añadir nuevas

características al gestor de flujo de ejecución usado en CAM4G. El resultado fue un nuevo gestor que automatiza la preparación y ejecución de experimentos complejos, garantiza su reproducibilidad y permite la división de las simulaciones en tareas más pequeñas para adaptarlas a las restricciones de los recursos. Para homogeneizar la preparación y gestión de experimentos independientemente de la infraestructura computacional en la que se vayan a ejecutar, se decidió ampliar el *framework* para habilitar el soporte a equipos individuales y *clusters* PBS. Al no encontrar soluciones que lo facilitasen, se desarrolló un gestor de trabajos sencillo, en *Python*, que adaptaba los comandos utilizados para enviar los trabajos al tipo de infraestructura seleccionada por el usuario.

El experimento utilizado en la fase de pruebas de la nueva implementación se ejecutó en el *cluster* del grupo de investigación y en la infraestructura *Grid* de EGEE. Los resultados mostraron que el uso de WRF4G simplificaba enormemente la gestión, ejecución y monitorización de experimentos en ambas infraestructuras. En la ejecución del experimento en el *cluster*, la única desventaja de utilizar WRF4G en lugar de *scripts ad-hoc* fue el uso de los ejecutables descargados desde el repositorio. Esta característica del *framework* presentó dos problemas: una penalización de 2 minutos en el tiempo total de ejecución de cada simulación (empleado en descargar los ejecutables) y la imposibilidad de elegir entre las versiones de WRF instaladas en el sistema. En cuanto a la ejecución en *Grid*, el experimento empleó más tiempo transfiriendo datos que ejecutando las simulaciones. Las características de esta primera versión de WRF4G fueron descritas en Fernández-Quiruelas et al. (2009b) y mostradas a la comunidad de WRF en Fernández-Quiruelas et al. (2010), en donde también se hacía una descripción de los principales estudios llevados a cabo con WRF4G hasta la fecha.

Para incorporar las necesidades de los usuarios en el desarrollo, se organizó un workshop de WRF4G[1] y se creó un portal de soporte[2]. Uno de los principales resultados de la interacción con los usuarios fue descubrir que las mayores dificultades encontradas durante el aprendizaje de WRF eran la compilación del modelo y el lanzamiento de las primeras simulaciones.

En la siguiente iteración de WRF4G se llevaron a cabo 4 grandes mejoras:

- La incorporación de pre y post-procesadores para optimizar las transferencias de datos.

- La recodificación de los todos componentes del *framework* siguiendo criterios de usabilidad, robustez y escalabilidad.

[1] http://corwes.webs.ull.es/workshop
[2] https://meteo.unican.es/trac/wiki/WRF4G

- La creación de tutoriales y documentación, así como de ejemplos para facilitar la capacitación de los usuarios.
- El cambio de gestor de trabajos anterior por DRM4G para proporcionar un acceso único y transparente a las diferentes infraestructuras computacionales. Como se ha puesto de manifiesto en la sección 1.2.4, tras analizar las opciones disponibles, se llegó a la conclusión de que la única forma de aprovechar eficientemente los recursos HDCI era creando DRM4G.

Para demostrar la capacidad de WRF4G para llevar a cabo estudios a gran escala, se realizó un experimento compuesto por 365 simulaciones que fueron ejecutadas utilizando 8 procesos MPI en una HDCI. Esta HDCI estaba compuesta por 3 *clusters*, la infraestructura *Grid* de EGI y un *Cloud* privado. Los resultados más importantes se resumen a continuación:

- El entrenamiento de la política de planificación redujo en un 60 % el tiempo de ejecución del experimento.
- Gracias al uso eficiente y combinado de los recursos, se ahorró un 75 % de tiempo respecto a la ejecución utilizando únicamente el *cluster* del grupo de investigación.
- El uso de mecanismos de replicación de datos fue imprescindible para evitar la sobrecarga de los repositorios. Con todo, la descarga de datos de entrada en los recursos *Grid* fue, de media, 4 veces más lento que en los recursos *Cloud* y *cluster*, aunque el impacto en el tiempo total de ejecución fue bajo.
- Para poder seleccionar los sitios de EGI con soporte MPI, fue necesario realizar previamente tareas de exploración lanzando trabajos de prueba.

De los resultados obtenidos, se puede afirmar que el *framework* facilita y optimiza la ejecución de modelos climáticos en *Grid*. Si comparamos WRF4G con otros *frameworks* similares (Redler et al., 2012), comprobaremos que, además de disponer de las mismas funcionalidades en cuanto a diseño y manejo de experimentos, WRF4G es más modular, facilitando el aprovechamiento en múltiples infraestructuras computacionales. Con su utilización para llevar a cabo simulaciones con 3 modelos diferentes (CAM, WAM y WRF) se ha demostrado su capacidad de abstracción para poder ser utilizado por cualquier modelo con requerimientos similares: largos tiempos de ejecución y manejo de grandes cantidades de datos. La incorporación de DRM4G para facilitar el acceso único y transparente a otras infraestructuras, ha demostrado a la comunidad del clima cómo, utilizando todos los recursos disponibles simultáneamente, es posible afrontar nuevos retos científicos.

3.2. Conclusiones principales

En la introducción se plantea como objetivo de esta tesis el análisis de requerimientos, diseño, implementación y verificación de una metodología para la ejecución de experimentos con modelos climáticos en infraestructuras heterogéneas y distribuidas. Este objetivo puede darse por cumplido ya que se ha hecho un análisis novedoso sobre los requerimientos de las simulaciones climáticas y se ha diseñado una metodología para facilitar la preparación, ejecución y monitorización de los experimentos en cualquier infraestructura de cálculo, incluyendo HDCI. Siguiendo esta metodología, se ha creado el *framework* utilizado en CAM4G y WRF4G, que constituye una importante contribución no solo a la comunidad climática, si no a la comunidad investigadora en general. A continuación se enumeran brevemente las conclusiones a las que se ha llegado en esta línea de investigación:

- El *middleware Grid* no está adaptado para la comunidad climática, por lo que ha sido necesario desarrollar un *framework* que cumpla con los requerimientos de los modelos climáticos: monitorización, gestión desatendida del flujo de ejecución y gestión óptima de las transferencias de datos.

- La tecnología *Grid* es adecuada para la ejecución de experimentos paramétricos como los de predicción por conjuntos, permitiendo a la comunidad aprovechar nuevas fuentes de cálculo. Sería recomendable mejorar el soporte a la ejecución de aplicaciones paralelas y la gestión de réplicas de datos de las infraestructuras *Grid* actuales.

- El uso del *framework* para aprovechar las infraestructuras *Grid* puede reducir notablemente el tiempo de ejecución de los experimentos si lo comparamos con lo que tardarían en un *cluster* de un grupo de investigación de tamaño medio. Para optimizar el tiempo de ejecución, es recomendable implementar una buena política de planificación que además soporte réplicas de datos.

- Tanto el *middleware* como las configuraciones de las infraestructuras *Grid* han mejorado a lo largo del tiempo en que se ha realizado esta tesis, disminuyendo considerablemente la cantidad de errores producidos durante la ejecución de los trabajos.

- WRF4G se ajusta a los requerimientos del modelado climático. Ha sido usado para llevar a cabo experimentos de producción científica, contribuyendo a varios trabajos en el ámbito exclusivo del clima: Nikulin et al. (2012); García-Díez et al. (2013); Menéndez et al. (2014); Argüeso et al. (2012a,b); Cardoso et al. (2012); Jiménez-Guerrero et al. (2013); Vautard et al. (2013); Expósito et al. (2015); Díaz et al. (2015).

- WRF4G se puede beneficiar de recursos heterogéneos y distribuidos sin aumentar la complejidad para el usuario, permitiendo la agregación de recursos bajo demanda. Al mismo tiempo, acelera el aprendizaje, permitiendo ejecutar las primeras simulaciones de WRF en minutos [3].

- Por su modularidad, el entorno de ejecución presentado en esta tesis es fácilmente adaptable para ser aprovechado por otros modelos y disciplinas con requerimientos similares (grandes transferencias de datos y largos tiempos de ejecución).

3.3. Perspectivas abiertas por esta tesis

WRF4G es un proyecto activo que está continuamente añadiendo nuevas funcionalidades, muchas de ellas solicitadas por los investigadores que lo utilizan. Además de seguir mejorando WRF4G con las aportaciones de nuestros usuarios, a medio plazo sería interesante incorporar las siguientes funcionalidades:

- Dotar a DRM4G con capacidades de planificación basadas en la ubicación de los datos y en el uso de réplicas. De esta forma, solventaríamos la optimización de las transferencias de datos, que es el último gran problema a resolver por las HDCI. De los estudios existentes en la literatura, el presentado por Taheri et al. (2013) propone una metodología de planificación que encaja perfectamente con la arquitectura de DRM4G.

- Añadir soporte a nuevos tipos de almacenamiento como la infraestructura ESGF o los repositorios *Cloud* S3[4], Swift[5] y StorPool[6].

- Utilizar la arquitectura y componentes presentados en esta tesis para implementar *framework*s de ejecución para otros modelos climáticos con una amplia comunidad de usuarios (e.g. RegCM (Giorgi et al., 2012) o CCLM (Rockel et al., 2008)).

- Dado que muchos de los componentes presentados en esta tesis no son exclusivos de la comunidad climática, exportar este modelo de trabajo a otras disciplinas con requerimientos similares.

[3] https://meteo.unican.es/trac/wiki/WRF4GTutorial
[4] https://aws.amazon.com/s3
[5] https://docs.openstack.org/developer/swift
[6] https://storpool.com

Apéndices

En cumplimiento de la normativa, este apéndice recoge un resumen de los artículos que componen esta tesis:

- Fernández-Quiruelas, V.; Fernández, J.; Baeza, C.; Cofiño, A.S.; Gutiérrez, J.M. Execution management in the GRID for sensitivity studies of global climate simulations. *Earth Science Informatics*. 2, pp. 75 - 82. 2009. doi:10.1007/s12145-008-0018-z

- Fernández-Quiruelas, V.; Fernández, J.; Cofiño, A.S.; Fita, L.; Gutiérrez, J.M. Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model. *Environmental Modelling & Software*. 26 - 9, pp. 1057 - 1069. 2011. doi:10.1016/j.envsoft.2011.03.006

- Fernández-Quiruelas, V; Blanco, C.; Cofiño, A.S.; Fernández, J. Large-scale climate simulations harnessing clusters, grid and cloud infrastructures. *Future Generation Computer Systems*. 51, pp.36 - 44. 2015. doi:10.1016/j.future.2015.04.009

A. Gestión de la ejecución de estudios de sensibilidad con modelos climáticos globales en el Grid

Técnicas recientes en el modelado del clima pueden encontrar en el paradigma de computación *Grid* una forma de lograr resultados científicos compartiendo recursos de computación y almacenamiento distribuidos geográficamente. Los experimentos de predicción por conjuntos (Hagedorn et al., 2005; Palmer, 2002) se basan en la generación de múltiples simulaciones con modelos para explorar estadísticamente las incertidumbres existentes en las predicciones del tiempo y del clima. En este trabajo, presentamos una aplicación *Grid* que utiliza un modelo climático de última generación. El objetivo principal de esta aplicación es proporcionar una herramienta que pueda ser utilizada por un investigador del clima para realizar estudios de sensibilidad ejecutando experimentos de predicción por conjuntos en el *Grid*.

Las simulaciones llevadas a cabo con modelos climáticos, a menudo requieren largos tiempos de ejecución. Debido a la alta tasa de fallos y a las limitaciones de tiempo de uso de procesador en los nodos de cálculo, la ejecución de trabajos de larga duración en *Grid* es poco común. Así, a diferencia de otras aplicaciones migradas al *Grid*, los modelos climáticos necesitan hacer uso de técnicas avanzadas en la gestión de flujos de trabajo que permitan ejecutar los experimentos de forma desatendida con un mínimo de intervención humana.

Con el fin de demostrar los beneficios del uso de infraestructuras *Grid* para llevar a cabo experimentos de predicción por conjuntos con modelos climáticos, se ha seleccionado el modelo atmosférico global CAM (Collins et al., 2004) y se ha creado una nueva aplicación llamada GRID-CAM, que es un gestor de flujo de trabajo que cumple con los principales requerimientos que los modelos climáticos demandan para su ejecución en *Grid*: tolerancia a fallos, capacidad de reinicio de las simulaciones, monitorización y gestión de datos y metadatos.

GRID-CAM utiliza el *middleware* gLite (Burke et al., 2006) y varios componentes desarrollados *ad-hoc*. Para realizar la interacción con el *Grid* se ha modificado el código fuente de CAM introduciendo llamadas al sistema que se encargan de realizar las tareas de monitorización y gestión de datos (incluyendo la generación y gestión de los ficheros de reinicio). Con el fin de crear un flujo de trabajo robusto, se han identificado las fuentes de fallo más comunes: errores de configuración en los sitios, bajo rendimiento o terminación prematura de los trabajos en los nodos de cálculo (incluyendo las finalizaciones por alcanzar el límite de tiempo de ejecución) y fallos de *middleware*. Tras analizar varios gestores de trabajos para *Grid*, se ha encontrado que el meta-planificador GridWay es el que mejor detecta y gestiona estos problemas, por lo que ha sido seleccionado para realizar la planificación y gestión de trabajos en GRID-CAM. Para almacenar la información relativa a los experimentos (inicio y

fin de las simulaciones, ficheros de salida y reinicio, etc.) y consolidar los eventos de monitorización de los trabajos (nodo de cálculo en el que se está ejecutando, paso de tiempo simulado, etc.) se ha utilizado la base de datos AMGA (Koblitz et al., 2008), que proporciona mecanismos de autenticación y autorización compatibles con el *Grid*.

Para demostrar la utilidad de GRID-CAM, se ha realizado un experimento de predicción por conjuntos que analiza el fenómeno *El Niño* en América Latina utilizando la infraestructura de EELA, distribuida por 11 países. El experimento, compuesto por 50 simulaciones que abarcan el periodo comprendido entre enero de 1997 y marzo de 1998, incluye el evento más fuerte de *El Niño* observado hasta la fecha. Se ha investigado la sensibilidad de la precipitación a las modificaciones en la temperatura de la superficie del océano en los países de latinoamericanos mediante un conjunto de simulaciones llevadas a cabo con el modelo CAM.

A pesar de la cantidad de errores ocurridos durante la ejecución de las simulaciones, la aplicación GRID-CAM ha sido capaz de detectarlos y sobreponerse a ellos, consiguiendo terminar las simulaciones en un tiempo similar al que se hubiese tardado usando todos los recursos del *cluster* de cálculo de los autores.

Una de las principales conclusiones de este trabajo es que la tecnología *Grid* actual se encuentra en un estado inmaduro y no está adaptada para la comunidad de Ciencias de la Tierra. Para los modelos climáticos tiene varias carencias, como la alta tasa de fallos o los límites de tiempo de ejecución de los trabajos, lo que requiere la creación de nuevo *middleware* capaz de manejar estos problemas. A pesar de estas debilidades, las aplicaciones que utilizan nuevas técnicas de predicción por conjuntos o cualquier otra que implique simulaciones independientes, podrían beneficiarse enormemente del *Grid*.

En el caso del grupo de investigación en el que se ha realizado este trabajo, los logros obtenidos gracias a disponer de una aplicación migrada al *Grid* no se habrían alcanzado con los medios habitualmente disponibles. A diferencia de los *clusters* o supercomputadores, optimizados para trabajos MPI y OpenMP, el *Grid* es perfectamente adecuado para simulaciones independientes. Por lo tanto, a pesar de que muchos trabajos fallaron y fue necesario volver a ejecutarlos, los beneficios de acceder a un *Grid* compuesto por un gran número de procesadores y espacio de almacenamiento superan los inconvenientes. Gracias a GRID-CAM ha sido posible completar alrededor de 50 años simulados en una semana.

En resumen, aunque el esfuerzo de desarrollar una aplicación en *Grid* es muy alto, los resultados valen la pena. Una vez realizado el esfuerzo inicial, muchas otras aplicaciones con requisitos similares podrían ser fácilmente migradas.

B. Beneficios y requerimientos de la computación Grid para las aplicaciones climáticas. Un ejemplo con el Community Atmospheric Model

Este artículo presenta una introducción a la computación *Grid* orientada a los investigadores que habitualmente trabajan con modelos climáticos en *cluster*, y se centra en las diferencias de uso entre ambas infraestructuras (inicio de sesión, transferencia de datos, envío de trabajos, etc.)

Para ello, se describen los tres componentes principales de una infraestructura *Grid* (herramientas de usuario, servicios básicos y recursos), y se muestran los nuevos protocolos de seguridad y autenticación para acceder a estas infraestructuras (certificados digitales, organizaciones virtuales, etc.). También se introduce el nuevo concepto de *middleware*, como una capa de software que proporciona servicios para dar un acceso transparente a recursos heterogéneos, y se describe GridFTP, protocolo *de facto* para la distribución y acceso a datos en el *Grid*.

Además, se hace un recorrido por las principales infraestructuras *Grid* actuales (como la iniciativa europea EGEE, que cuenta con más de 10.000 procesadores repartidos en 50 sitios) y las diferentes formas de autorizar su uso a través de organizaciones virtuales. También se ha hecho una revisión del trabajo realizado dentro de la comunidad climática hasta el momento, señalando que la mayor parte de la atención se ha centrado en la gestión de datos y apenas se han invertido esfuerzos en desplegar y ejecutar aplicaciones, lo que todavía supone un desafío para las infraestructuras *Grid* reales.

Este trabajo se ha centrado en identificar los principales requerimientos de las aplicaciones climáticas para su ejecución en infraestructuras *Grid*: tolerancia a fallos, capacidad de reinicio de las simulaciones, monitorización y gestión de datos y metadatos. Como resultado de este análisis, se ha concluido que es necesario desarrollar nuevos componentes de *middleware* (gestores del flujo de trabajo) para hacer frente a los requisitos exigidos por estas aplicaciones.

Como ejemplo ilustrativo se describe CAM4G, una nueva aplicación que controla la simulación de CAM en *Grid* y permite reiniciar trabajos interrumpidos, administrar datos y monitorizar los experimentos en ejecución. Además, para mostrar la capacidad de CAM4G, se ha llevado a cabo un desafío computacional utilizando la infraestructura *Grid* de EGEE. Este desafío consiste en la ejecución de 750 simulaciones con el modelo CAM con las que se analiza la sensibilidad de la precipitación global a perturbaciones en la temperatura superficial del océano (SST) en la región *El Niño* durante un periodo de 15 meses.

Los resultados muestran que la sensibilidad a la precipitación es mayor en las

áreas en las que se modificó la SST. Sin embargo, se encuentran también teleconexiones mundiales de la señal de *El Niño*, ya que también se puede apreciar cierta sensibilidad en otros lugares como el África Ecuatorial Occidental. También se ilustra que la precipitación muestra respuestas lineales y no lineales a la intensidad de la señal *El Niño* aplicada.

Para cuantificar los beneficios de la computación *Grid*, se proporcionan las estadísticas de ejecución del experimento y se comparan con el coste computacional en un *cluster*. Se muestra cómo los 11 días necesarios para finalizar el experimento en el *cluster* de cálculo de los autores, se reducen a sólo 3,5 días en el *Grid*. Al mismo tiempo, se presentan estadísticas de las diferentes realizaciones que muestran que, en general, las realizaciones individuales ejecutadas en *Grid* requirieron 2 o 3 trabajos para terminar, empleando un promedio de 50 horas, un 114 % de las 44 horas empleadas en el *cluster*.

También se ha probado con éxito la ejecución de código paralelo en el *Grid*, pero, a diferencia del ejemplo de ejecución en serie mostrado en este trabajo, se ha detectado que la ejecución de código en paralelo restringe el número de sitios utilizables y requiere un tratamiento más específico para cada sitio.

Debido a la complejidad y a los requerimientos computacionales del modelado climático, las infraestructuras *Grid* necesitan mejorar algunos aspectos para ser completamente útiles para la comunidad que trabaja con modelos climáticos. Algunas de estas carencias se han encontrado durante el desarrollo y uso del *middleware* CAM4G presentado en este artículo. Los principales problemas están relacionados con el almacenamiento y el acceso a los datos. Los modelos climáticos necesitan y producen una gran cantidad de datos y, debido a las limitaciones del ancho de banda de las infraestructuras *Grid*, las transferencias son, a menudo, el cuello de botella de las simulaciones.

Para reducir los tiempos empleados en las transferencias de datos, sería recomendable una mejora en el ancho de banda de las infraestructuras *Grid*. Al mismo tiempo, también sería aconsejable el uso de sistemas de replicación eficientes y robustos, que aceleren las transferencias. Finalmente, se debe realizar un esfuerzo para permitir la ejecución de código paralelo en las infraestructuras *Grid*, de forma que sea posible diseñar experimentos más exigentes, en términos de memoria y recursos de procesamiento, que el presentado en este artículo.

C. Ejecución de simulaciones a gran escala uniendo infrastructuras Cluster, Grid y Cloud

La variedad de infraestructuras de computación disponibles hoy en día (HPC, *Grid* y *Cloud*) proporciona una gran potencia computacional en muchos campos de la ciencia. Sin embargo, su aprovechamiento para llevar a cabo grandes experimentos científicos sigue siendo un reto. En este trabajo, utilizamos el paradigma del modelado climático para presentar los problemas encontrados cuando se ejecutan aplicaciones estándar en infraestructuras híbridas de computación distribuida (HDCI) y proponemos un *framework* que permite a un modelo climático aprovechar estos recursos fácilmente y de manera transparente.

La naturaleza distribuida de las HDCI complica tareas como la monitorización y depuración de aplicaciones o la gestión de datos. Además, la combinación de recursos HPC, *Cloud* y *Grid* plantea un reto adicional: proporcionar una interfaz uniforme que permita la interoperabilidad entre los diferentes gestores de trabajos. Para poder hacer un uso eficiente de las HDCI, las aplicaciones que requieren la ejecución de trabajos de larga duración y la gestión de grandes transferencias de datos, necesitan modificar su flujo de trabajo para adaptarse a las características de los recursos.

Este trabajo propone un *framework* que facilita la gestión y ejecución de experimentos con modelos climáticos en HDCI. Como ejemplo, se muestra WRF4G, una implementación de este entorno de trabajo para el modelo WRF (Skamarock et al., 2008), que permite llevar a cabo experimentos a gran escala combinando eficientemente recursos heterogéneos y distribuidos. También proporciona control total sobre las simulaciones y medios para reiniciar de forma parcial o total el experimento en caso de fallo.

WRF4G se compone de tres servicios que se encargan de la gestión de experimentos, trabajos y datos respectivamente. La interacción con el usuario se realiza a través de un conjunto de herramientas que facilitan la creación, gestión y monitorización de experimentos. Estas herramientas forman parte del servicio de gestión de experimentos (EMS). Para consolidar la información y estado de las simulaciones, el EMS incorpora una base de datos oculta a los usuarios.

Con el fin de proporcionar un acceso transparente y unificado a la HDCI, el EMS interactúa con los servicio de gestión de trabajos (JMS) y datos (DMS), que, a su vez, actúan como interfaz único de acceso a los recursos. Para realizar una gestión eficiente, el JMS utiliza la herramienta DRM4G, que ofrece una solución robusta y escalable para el acceso transparente a recursos de tipo *cluster*, *Grid* y *Cloud*.

Para gestionar la heterogeneidad de protocolos de acceso a datos, el DMS incorpora la herramienta *vcp*. Además, proporciona una gestión de datos descentralizada que permite indicar la ubicación de los repositorios de datos para cada recurso de

cálculo.

Para ilustrar la utilidad del entorno de trabajo propuesto, se ha vuelto a ejecutar en un recurso HDCI un experimento de simulación de viento que había sido ejecutado previamente en un solo *cluster*. Este experimento tiene como objetivo simular el comportamiento del viento en la cuenca mediterránea durante un año. Para ello, se ha ejecutado un conjunto de 365 realizaciones independientes. Los detalles científicos del experimento se pueden encontrar en Menéndez et al. (2014). La entrada para cada realización fue de 230 MB de datos de reanálisis global (84 GB para todo el experimento). Para cada realización, el modelo WRF produjo 2 GB de variables meteorológicas con una resolución de 15 km y un paso de tiempo de una hora. Como el objetivo principal del estudio se centró en el viento, seleccionamos sólo esta variable, reduciendo la salida a 40 MB. La salida de datos para todo el experimento se redujo de 730 GB a 14 GB.

Este experimento de prueba ahorró más de un 75 % del tiempo, si lo comparamos con su ejecución en el *cluster*. WRF4G se benefició de recursos heterogéneos adicionales sin aumentar la complejidad para el usuario. Además, se constató que la capacidad de agregar recursos bajo demanda es muy útil para resolver picos de carga o para garantizar acuerdos de nivel de servicio mediante la utilización de recursos *Cloud*.

Todos los datos de salida fueron enviados, de forma transparente, a un repositorio local común como si se hubieran utilizado recursos informáticos locales. Los resultados mostraron que la replicación de datos de entrada y el entrenamiento de la política de planificación son cruciales para llevar a cabo un experimento a gran escala.

Aunque WRF4G ha sido adaptado al flujo de trabajo del modelo WRF, la modularidad del *framework* permite que otros modelos climáticos con características similares sean fácilmente adaptados. Además, aplicaciones de otras disciplinas también podrían beneficiarse de la mayoría de los componentes de este entorno de trabajo: DRM4G, vcp, herramientas de usuario, etc.

WRF4G es un proyecto activo y está continuamente añadiendo nuevas funcionalidades. Para mejorar el rendimiento de las transferencias de datos, sería interesante mejorar la política de planificación del entorno de trabajo actual para que tenga en cuenta la ubicación de lo datos. Esta capacidad mejoraría enormemente el tiempo total de ejecución del experimento. En este sentido, Taheri et al. (2013) propone un algoritmo de planificación basado en el uso de réplicas de datos que podría ser integrado en las políticas de DRM4G.

Bibliografía

Aad, G., et al., 2010: The ATLAS simulation infrastructure. *The European Physical Journal C*, 70 (3), 823–874.

Adams, C. y S. Lloyd, 1999: *Understanding public-key infrastructure: concepts, standards, and deployment considerations*. Macmillan Technical Publishing.

Alfieri, R., R. Cecchini, V. Ciaschini, L. Dell'Agnello, A. Frohner, A. Gianoli, K. Lorentey, y F. Spataro, 2004: VOMS, an authorization system for virtual organizations. *Lecture notes in computer science*, 33–40.

Allcock, W., J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, y I. Foster, 2005: The Globus Striped GridFTP Framework and Server. *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, IEEE Computer Society, 54.

Allen, M., 1999: Do-it-yourself climate prediction. *Nature*, 401 (6754), 642–642.

Anderson, D. P. y G. Fedak, 2006: The computational and storage potential of volunteer computing. *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, IEEE, Vol. 1, 73–80.

Andreozzi, S., et al., 2009: GLUE Specification v. 2.0. *Open Grid Forum Recommendation Documents*, Open Grid Forum.

Anjomshoaa, A., F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, y A. Savva, 2005: Job submission description language (jsdl) specification, version 1.0. *Open Grid Forum, GFD*, Vol. 56.

Argüeso, D., J. Hidalgo-Muñoz, S. Gámiz-Fortis, M. Esteban-Parra, y Y. Castro-Díez, 2012a: Evaluation of WRF Mean and Extreme Precipitation over Spain: Present Climate (1970-99). *Journal of Climate*, 25 (14), 4883–4897.

Argüeso, D., J. Hidalgo-Muñoz, S. Gámiz-Fortis, M. Esteban-Parra, y Y. Castro-Díez, 2012b: High-resolution projections of mean and extreme precipitation over Spain using the WRF model (2070–2099 versus 1970–1999). *Journal of Geophysical Research*, 117 (D12).

Armbrust, M., et al., 2010: A view of cloud computing. *Communications of the ACM*, 53 (4), 50–58.

Awan, N., H. Truhetz, y A. Gobiet, 2011: Parameterization induced error-characteristics of MM5 and WRF operated in climate mode over the Alpine Region: An ensemble based analysis. *Journal of Climate*, 24 (12), 3107–3123.

Balasko, A., Z. Farkas, y P. Kacsuk, 2013: Building science gateways by utilizing the generic WS-PGRADE/gUSE workflow system. *Computer Science*, 14.

Birrittella, M. S., M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Underwood, y R. C. Zak, 2015: Intel® Omni-path Architecture: Enabling Scalable, High Performance Fabrics. *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*, IEEE, 1–9.

Blanco Real, J. C., 2017: Agregación de infraestructuras computacionales usando técnicas de meta-planificación centradas en el usuario. Ph.D. thesis, Universidad de Cantabria.

Board, I. A., 2009: Impact Assessment Guidelines. Technical Report 92, European Commission.

Boden, N. J., D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, y W.-K. Su, 1995: Myrinet: A gigabit-per-second local area network. *IEEE micro*, 15 (1), 29–36.

Bretherton, D. A., J. D. Blower, K. Haines, y G. C. Smith, 2009: Running climate models on grids using G-Rex. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367 (1890), 847–853.

Burke, S., S. Campana, A. D. Peris, F. Donno, P. M. Lorenzo, R. Santinelli, y A. Sciaba, 2006: glite 3.1 user guide. *Manuals Series*.

Cardoso, R., P. Soares, P. Miranda, y M. Belo-Pereira, 2012: WRF high resolution simulation of Iberian mean and extreme precipitation climate. *International Journal of Climatology*.

Casavant, T. L. y J. G. Kuhl, 1988: A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on software engineering*, 14 (2), 141–154.

Childs, S., 2007: Supporting MPI applications on the EGEE Grid. *2nd EGEE User Forum*.

Collins, W. D., et al., 2004: Description of the NCAR Comunity Atmospheric Model (CAM 3.0). Tech. Rep. NCAR/TN-464+STR, National Center for Atmospheric Research, 226 pp.

Collins, W. D., et al., 2006: The community climate system model: CCSM3. *Journal of Climate*, 19 (11), 2122–2143.

Cossu, R., et al., 2010: A roadmap for a dedicated Earth Science Grid platform. *Earth Science Informatics*, 3 (3), 135–148.

Dabrowski, C., 2009: Reliability in grid computing systems. *Concurrency and Computation: Practice and Experience*, 21 (8), 927–959.

Danovaro, E., et al., 2014: Setting up an hydro-meteo experiment in minutes: the DRIHM e-Infrastructure for HM research. *e-Science (e-Science), 2014 IEEE 10th International Conference on*, IEEE, Vol. 1, 47–54.

Davidović, D., K. Skala, D. Belušić, y M. Telišman Prtenjak, 2010: Grid implementation of the weather research and forecasting model. *Earth Science Informatics*, 3 (4), 199–208.

Davidović, D., T. Lipić, y K. Skala, 2013: AdriaScience gateway: Application specific gateway for advanced meteorological predictions on croatian distributed computing infrastructures. *Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on*, IEEE, 217–221.

De Almeida, E. S., H. F. de Campos Velho, y A. J. Preto, 2013: Challenges for mesoscale climatology execution on experimental grid computing systems. *Journal of the Brazilian Computer Society*, 19 (3), 279–290.

Díaz, J., A. González, F. Expósito, J. Pérez, J. Fernández, M. García-Díez, y D. Taima, 2015: WRF multi-physics simulation of clouds in the African region. *Quarterly Journal of the Royal Meteorological Society*, 141 (692), 2737–2749.

Dichev, K., S. Stork, R. Keller, y E. Fernández, 2008: MPI Support on the Grid. *Computing and Informatics*, 27 (2), 213–222.

Díez, E., B. Orfila, M. Frías, J. Fernández, A. Cofiño, y J. Gutiérrez, 2011: Downscaling ECMWF seasonal precipitation forecasts in Europe using the RCA model. *Tellus A*, 63 (4), 757–762.

Ellert, M., et al., 2007: Advanced Resource Connector middleware for lightweight computational Grids. *Future Generation computer systems*, 23 (2), 219–240.

Erwin, D. y D. Snelling, 2002: UNICORE-a Grid computing environment. *Concurrency and Computation: Practice and Experience*, 14 (13-15), 1395–1410.

Evans, J., M. Ekström, y F. Ji, 2011: Evaluating the performance of a WRF physics ensemble over South-East Australia. *Climate Dynamics*, 1–18.

Expósito, F. J., A. González, J. C. Pérez, J. P. Díaz, y D. Taima, 2015: High-Resolution Future Projections of Temperature and Precipitation in the Canary Islands. *Journal of Climate*, 28 (19), 7846–7856.

Farkas, Z. y P. Kacsuk, 2011: P-GRADE Portal: A generic workflow system to support user communities. *Future Generation Computer Systems*, 27 (5), 454–465.

Fernández, J., J. P. Montávez, J. Sáenz, J. F. González-Rouco, y E. Zorita, 2007: Sensitivity of MM5 Mesoscale Model to Physical Parameterizations for Regional Climate Studies: Annual Cycle. *JGR*, 112, D04 101.

Fernández-Quiruelas, V., C. Blanco, A. Cofiño, y J. Fernández, 2015: Large-scale climate simulations harnessing clusters, grid and cloud infrastructures. *Future Generation Computer Systems*, 51, 36–44.

Fernández-Quiruelas, V., J. Fernández, C. Baeza, A. S. Cofiño, y J. Gutiérrez, 2009a: Execution management in the GRID, for sensitivity studies of global climate simulations. *Earth Science Informatics*, 2 (1), 75–82.

Fernández-Quiruelas, V., J. Fernández, A. S. Cofiño, C. Blanco, M. García-Díez, M. Magariño, L. Fita, y J. M. Gutiérrez, 2015: WRF4G: WRF experiment management made simple. *Geoscientific Model Development Discussions*, 8, 6551–6582.

Fernández-Quiruelas, V., J. Fernández, A. S. Cofiño, y L. Fita, 2009b: WRF4G: enabling WRF on the Grid. *Proceedings of the Second EELA-2 Conference*, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, Madrid, 149–154.

Fernández-Quiruelas, V., L. Fita, J. Fernández, y A. Cofiño, 2010: WRF workflow on the Grid with WRF4G. *11th WRF users' workshop. Boulder (CO), USA*.

Fernández-Quiruelas, V., J. Fernández, A. Cofiño, L. Fita, y J. Gutiérrez, 2011: Benefits and requirements of grid computing for climate applications. An example with the community atmospheric model. *Environmental Modelling & Software*, 26 (9), 1057–1069.

Foster, I., 2006: Globus Toolkit Version 4: Software for Service-Oriented Systems. *International Conference on Network and Parallel Computing LNCS 3779*, Springer-Verlag, Vol. 15, 2–13.

Foster, I. y C. Kesselman, 1999: *The grid. Blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers, San Francisco, 677 pp.

Foster, I., C. Kesselman, G. Tsudik, y S. Tuecke, 1998: A security architecture for computational grids. *Proceedings of the 5th ACM Conference on Computer and Communications Security*, ACM New York, NY, USA, 83–92.

Foster, I., C. Kesselman, y S. Tuecke, 2001: The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15 (3), 222.

Fox, A., et al., 2009: Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28 (13), 2009.

Frey, J., T. Tannenbaum, M. Livny, I. Foster, y S. Tuecke, 2002: Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5 (3), 237–246.

Gagliardi, F., B. Jones, F. Grey, M.-E. Bégin, y M. Heikkurinen, 2005: Building an infrastructure for scientific Grid computing: status and goals of the EGEE project. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 363 (1833), 1729–1742.

García-Díez, M., J. Fernández, L. Fita, y C. Yagüe, 2013: Seasonal dependence of WRF model biases and sensitivity to PBL schemes over Europe. *Quarterly Journal of the Royal Meteorological Society*, 139 (671), 501–514.

García-Díez, M., J. Fernández, y R. Vautard, 2015: An RCM multi-physics ensemble over Europe: multi-variable evaluation to avoid error compensation. *Climate Dynamics*, 45 (11-12), 3141–3156.

Gentzsch, W., 2001: Sun Grid Engine: towards creating a compute power grid. *First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001. Proceedings*, 35–36.

Giorgi, F., et al., 2012: RegCM4: model description and preliminary tests over multiple CORDEX domains. *Climate Research*, 52, 7–29.

Goodale, T., S. Jha, H. Kaiser, T. Kielmann, et al., 2006: SAGA: A Simple API for Grid Applications. High-level application programming on the Grid. *Computational Methods in Science and Technology*, 12 (1), 7–20.

Grama, A., G. Karypis, V. Kumar, y A. Gupta, 2003: *Introduction to Parallel Computing (2nd Edition)*. Addison Wesley.

Hagedorn, R., F. J. Doblas-Reyes, y T. Palmer, 2005: The rationale behind the success of multi–model ensembles in seasonal forecasting - I. Basic concept. *Tellus*, 57A, 219–233.

Hammer-Lahav, E., D. Recordon, y D. Hardt, 2011: The OAuth 2.0 authorization protocol. *Network Working Group Internet-Draft*.

Henderson, R. L., 1995: Job scheduling under the portable batch system. *Workshop on Job Scheduling Strategies for Parallel Processing*, Springer, 279–294.

Herbordt, M. C., T. VanCourt, Y. Gu, B. Sukhwani, A. Conti, J. Model, y D. DiSabello, 2007: Achieving high performance with FPGA-based computing. *Computer*, 40 (3).

Hey, T. y A. E. Trefethen, 2005: Cyberinfrastructure for e-Science. *Science*, 308 (5723), 817–821.

Holton, J., 2004: *An introduction to dynamic meteorology*. Academic press.

Howes, T. A., M. C. Smith, y G. S. Good, 2003: *Understanding and deploying LDAP directory services*. Addison-Wesley Longman Publishing Co., Inc.

Huedo, E., R. S. Montero, y I. M. Llorente, 2001: The GridWay Framework for Adaptive Scheduling and Execution on Grids. *Scalable Computing: Practice and Experience*, 6 (3).

Hwang, E., S. Kim, J.-S. Kim, S. Hwang, y Y.-r. Choi, 2016: On the role of application and resource characterizations in heterogeneous distributed computing systems. *Cluster Computing*, 1–16.

Jacob, R., C. Schafer, I. Foster, M. Tobis, y J. Anderson, 2001: Computational design and performance of the Fast Ocean Atmosphere Model, version one. *Computational Science—ICCS 2001*, 175–184.

Jacq, N., et al., 2008: Grid-enabled virtual screening against malaria. *Journal of Grid Computing*, 6 (1), 29–43.

Jeffers, J. y J. Reinders, 2013: *Intel Xeon Phi coprocessor high-performance programming*. Newnes.

Jerez, S., J. P. Montavez, P. Jimenez-Guerrero, J. J. Gomez-Navarro, R. Lorente-Plazas, y E. Zorita, 2013: A multi-physics ensemble of present-day climate regional simulations over the Iberian Peninsula. *Climate dynamics*, 40 (11-12), 3023–3046.

Jiménez, P. A. y J. Dudhia, 2012: Improving the representation of resolved and unresolved topographic effects on surface wind in the WRF model. *Journal of Applied Meteorology and Climatology*, 51 (2), 300–316.

Jiménez-Guerrero, P., et al., 2013: Mean fields and interannual variability in RCM simulations over Spain: the ESCENA project. *Climate Research*, 57 (3), 201–220.

Kasam, V., et al., 2009: WISDOM-II: Screening against multiple targets implicated in malaria using computational grid infrastructures. *Malaria journal*, 8 (1), 88.

Koblitz, B., N. Santos, y V. Pose, 2008: The AMGA Metadata Service. *Journal of Grid Computing*, 6 (1), 61–76.

Kricheff, D. N., C. Constable, y C. T. Gambetta, 2001: Virtual data storage (VDS) system. Google Patents, uS Patent 6,324,627.

Kshemkalyani, A. D. y M. Singhal, 2011: *Distributed computing: principles, algorithms, and systems*. Cambridge University Press.

Lagouvardos, K., E. Floros, y V. Kotroni, 2010: A Grid-Enabled Regional-Scale Ensemble Forecasting System in the Mediterranean Area. *Journal of Grid Computing*, 8 (2), 181–197.

Lassoued, Y., 2009: WAM (WAve prediction Model). *EELA-2 User Forum, Montevideo*.

Laure, E., et al., 2006: Programming the Grid with gLite. *Computational Methods in Science and Technology*, 2006.

Lewis, K. D., 2009: Web single sign-on authentication using SAML. *arXiv preprint arXiv:0909.2368*.

Lindholm, E., J. Nickolls, S. Oberman, y J. Montrym, 2008: NVIDIA Tesla: A unified graphics and computing architecture. *IEEE micro*, 28 (2).

Livny, M., J. Basney, R. Raman, y T. Tannenbaum, 1997: Mechanisms for high throughput computing. *SPEEDUP journal*, 11 (1), 36–40.

Lossau, N., 2012: An overview of research infrastructures in Europe-and recommendations to LIBER. *Liber Quarterly*, 21 (3-4).

Lynch, P., 2008: The origins of computer weather prediction and climate modeling. *Journal of Computational Physics*, 227 (7), 3431–3444.

Manubens-Gil, D., R. Mula-Valls, y J. Vegas-Regidor, 2016: *AUTOSUBMIT and EC-EARTH configuration management*.

Marechal, B., P. R. Bello, y D. Carvalho, 2007: Building a grid in Latin America: The EELA project e-infrastructure. *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, IEEE, 835–839.

Martin, R. C., 2003: *Agile software development: principles, patterns, and practices*. Prentice Hall PTR.

Mateescu, G., W. Gentzsch, y C. J. Ribbens, 2011: Hybrid Computing—Where HPC meets grid and Cloud Computing. *Future Generation Computer Systems*, 27 (5), 440–453.

Menéndez, M., M. García-Díez, L. Fita, J. Fernández, F. J. Méndez, y J. M. Gutiérrez, 2014: High-resolution sea wind hindcasts over the Mediterranean area. *Climate Dynamics*, 42 (7-8), 1857–1872.

Milojičić, D., I. M. Llorente, y R. S. Montero, 2011: Opennebula: A cloud management tool. *IEEE Internet Computing*, 15 (2), 11–14.

Mohammadi, M. y T. Bazhirov, 2017: Comparative benchmarking of cloud computing vendors with High Performance Linpack. *arXiv preprint arXiv:1702.02968*.

Mooney, P., F. Mulligan, y R. Fealy, 2013: Evaluation of the sensitivity of the Weather Research and Forecasting model to parameterization schemes for regional climates of Europe over the period 1990-1995. *Journal of Climate*, 26, 1002–1017.

Müller, P. y H. von Storch, 2004: *Computer Modelling in Atmospheric and Oceanic Sciences. On the Building of Knowlegde*. Springer Verlag, Berlin.

Murphy, J., D. Sexton, D. Barnett, G. Jones, M. Webb, M. Collins, y D. Stainforth, 2004: Quantification of modelling uncertainties in a large ensemble of climate change simulations. *Nature*, 430 (7001), 768–772.

Narayan, S., 2009: Supercomputers: past, present and the future. *Crossroads*, 15 (4), 7–10.

Nefedova, V., et al., 2006: Automating climate science: Large ensemble simulations on the TeraGrid with the GriPhyN Virtual Data System. *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*, IEEE, 32–32.

Neocleous, K., M. D. Dikaiakos, P. Fragopoulou, y E. Markatos, 2006: Grid reliability: A study of failures on the egee infrastructure. *Proceedings of the CoreGRID Integration Workshop*, 165–176.

Neuman, B. C. y T. Ts'o, 1994: Kerberos: An authentication service for computer networks. *IEEE Communications magazine*, 32 (9), 33–38.

Nikulin, G., et al., 2012: Precipitation Climatology in an Ensemble of CORDEX-Africa Regional Climate Simulations. *Journal of Climate*, 25, 6057–6078.

Palmer, T. N., 2002: The economic value of ensemble forecasts as a tool for risk assessment: From days to decades. *Quarterly Journal of the Royal Meteorological Society*, 128 (581), 747–774.

Parry, M., O. Canziani, J. Palutikof, P. van der Linden, y C. Hanson, (Eds.) , 2007: *Climate Change 2007: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge: Cambridge University Press.

Pordes, R., et al., 2007: The open science grid. *Journal of Physics: Conference Series*, 78 (1), 012 057.

Postel, J. y J. Reynolds, 1985: File transfer protocol. STD 9, RFC 959, October 1985.

Recordon, D. y D. Reed, 2006: OpenID 2.0: A Platform for User-centric Identity Management. *Proceedings of the Second ACM Workshop on Digital Identity Management*, ACM, 11–16.

Redler, R., R. Budich, R. Ford, y G. Riley, 2012: *Earth System Modelling - Volume 5: Tools for Configuring, Building and Running Models*. 1st ed., Springer.

Rew, R. y G. Davis, 1990: NetCDF: an interface for scientific data access. *IEEE computer graphics and applications*, 10 (4), 76–82.

Rockel, B., A. Will, y A. Hense, 2008: The regional climate model COSMO-CLM (CCLM). *Meteorologische Zeitschrift*, 17 (4), 347–348.

Sefraoui, O., M. Aissaoui, y M. Eleuldj, 2012: OpenStack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55 (3).

Shanley, T. y J. Winkles, 2003: *InfiniBand Network Architecture*. Addison-Wesley Professional.

Skamarock, W., J. Klemp, J. Dudhia, D. Gill, D. Barker, M. Duda, W. Wang, y J. Powers, 2008: A description of the Advanced Research WRF Version 3. Tech. rep., NCAR.

Stensrud, D., 2009: *Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models*. Cambridge University Press, 480 pp.

Sulis, A., 2009: GRID computing approach for multireservoir operating rules with uncertainty. *Environmental Modelling & Software*, 24 (7), 859–864.

Taheri, J., A. Y. Zomaya, P. Bouvry, y S. U. Khan, 2013: Hopfield neural network for simultaneous job scheduling and data replication in grids. *Future Generation Computer Systems*, 29 (8), 1885–1900.

Thain, D., T. Tannenbaum, y M. Livny, 2005: Distributed computing in practice: the Condor experience. *Concurrency and Computation: Practice and Experience*, 17 (2-4), 323–356, doi:10.1002/cpe.938, URL http://dx.doi.org/10.1002/cpe.938.

Tröger, P., H. Rajic, P. Domagalski, y A. Haas, 2007: Standardization of an API for Distributed Resource Management Systems. *In proceedings of CCGrid'07*.

Tsaregorodtsev, A., et al., 2008: DIRAC: a community grid solution. *Journal of Physics: Conference Series*, 119 (6), 062048.

Tuecke, S., V. Welch, D. Engert, L. Pearlman, y M. Thompson, 2004: RFC 3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. Internet Engineering Task Force.

Vautard, R., et al., 2013: The simulation of European heat waves from an ensemble of regional climate models within the EURO-CORDEX project. *Climate Dynamics*, 41 (9-10), 2555–2575.

Vázquez Blanco, C., 2013: Arquitectura para el aprovisionamiento dinámico de recursos computacionales. Ph.D. thesis, Universidad Complutense de Madrid.

WAMDI, 1988: The WAM model—A third generation ocean wave prediction model. *Journal of Physical Oceanography*, 18 (12), 1775–1810.

Wang, F., S. Oral, G. Shipman, O. Drokin, T. Wang, y I. Huang, 2009: Understanding lustre filesystem internals. *Oak Ridge National Laboratory, National Center for Computational Sciences, Tech. Rep.*

Weil, S. A., S. A. Brandt, E. L. Miller, D. D. Long, y C. Maltzahn, 2006: Ceph: A scalable, high-performance distributed file system. *Proceedings of the 7th symposium on Operating systems design and implementation*, USENIX Association, 307–320.

Welch, V., et al., 2004: X.509 proxy certificates for dynamic delegation. *3rd annual PKI R&D workshop*, Vol. 14.

Williams, D. N., et al., 2011: The Earth System Grid Federation: Software framework supporting CMIP5 data analysis and dissemination. *CLIVAR Exchanges*, 56 (2), 40–42.

Yalew, S., A. van Griensven, N. Ray, L. Kokoszkiewicz, y G. D. Betrie, 2013: Distributed computation of large scale SWAT models on the Grid. *Environmental Modelling & Software*, 41, 223–230.

Ylonen, T., 1996: SSH–secure login connections over the Internet. *Proceedings of the 6th USENIX Security Symposium*, Vol. 37.

Yoo, A. B., M. A. Jette, y M. Grondona, 2003: Slurm: Simple linux utility for resource management. *Workshop on Job Scheduling Strategies for Parallel Processing*, Springer, 44–60.

Zhao, G., B. A. Bryan, D. King, Z. Luo, E. Wang, U. Bende-Michl, X. Song, y Q. Yu, 2013: Large-scale, high-resolution agricultural systems modeling using a hybrid approach combining grid computing and parallel processing. *Environmental Modelling & Software*, 41, 231–238.