Centro Euro-Mediterraneo
per i Cambiamenti Climatici

Issue RP0098
July 2010

Scientific Computing and Operation (SCO)

A DISTRIBUTED INFRASTRUCTURE FOR ENSEMBLE EXPERIMENTS

By **Italo Epicoco**
University of Salento, Italy
italo.epicoco@unisalento.it

Maria Mirto
CMCC
maria.mirto@cmcc.it

Silvia Mocavero
CMCC
silvia.mocavero@cmcc.it

and **Giovanni Aloisio**
CMCC
University of Salento, Italy
giovanni.aloisio@unisalento.it

SUMMARY The WP3/NA2 of the EU IS-ENES project aims at the set up and deploy an e-infrastructure providing climate scientists with the needed virtual proximity to distributed data and distributed compute resources. The access point of this infrastructure is represented by the v.E.R.C. portal: it will allow the ESM scientists to run complex distributed workflows for running ESM experiments and accessing to ESM data. The focus of this report is on the deployment of a grid prototype for running ensembles of multi-model experiments. Considering existing grid infrastructures and services, the design of this grid prototype has been lead by the necessity to build a framework that will leverage the external services offered within the European HPC ecosystem, e.g., today by DEISA2 and in the future by PRACE. The prototype allows exploiting advanced grid services, namely GRB services, developed at the University of Salento, and basic grid services offered by the Globus Toolkit middleware in order for submitting and monitoring of ensemble runs. The prototype has been deployed involving two sites composed of the CMCC and DKRZ nodes. A third node, represented by the BSC, has been considered but its deployment is yet an on going activity. A case study related to the HRT159, a global coupled ocean-atmosphere general circulation model (AOGCM) developed by CMCC-INGV, has been considered and preliminary tests carried out on CMCC and DKRZ sites are reported.

Keywords: Distributed environment;Grid infrastructure; Grid portal; ensemble experiments

JEL: C63

The work here presented, carried out with support of the European Commission funding under the FP7 within IS-ENES project.

INTRODUCTION

Users as well as developers of Earth System Models rely on an infrastructure, consisting of high-end computing, data storage and network resources to perform complex and demanding simulations. In the past, mainly local resources and infrastructures were used. However, the increasing requirements on computing capability and capacity as well as on data storage facilities often exceed the possibilities of single centers. An ensemble simulation consists of many individual runs, which require intensive computational power and produce huge amount of data to be post-processed and archived. The processing phase is often integrated in a complex workflow including also pre and post-processing steps that are performed on different machines, potentially at different sites, and often by different scientists. In Europe, there is the need to deploy and, when needed, to develop technologies in order to provide climate scientists with virtual proximity to distributed computing resources and data. The WP3/NA2 work package of the European IS-ENES project is aimed to set up and foster the deployment of an e-Infrastructure within the ESM community. This will take advantage of its own distributed service infrastructure and it will leverage the external services offered within the European HPC ecosystem, e.g. today by DEISA2 and in the near future by PRACE. The infrastructure is also named “virtual Earth-System modeling Resource Centre (v.E.R.C.)" and it will consist of:

- The ENES v.E.R.C. Portal: An information and collaboration portal to present all the services, tools and data available to the community in a coherent way to foster the exchange of information and the collaboration within the community.
- A unified HPC environment for Earth System Modeling to ease and improve the utilization of existing and upcoming HPC environments by the ESM community.
- A prototype grid infrastructure for training, designing and testing complex distributed workflows, e.g., in ensemble experiments.

This technical report describes the deployment of a grid prototype used for verifying if complex workflows, defined within the ESM climate community, can take advantage of the use of a distributed environment through the adoption of Grid technologies. This will concern running ensembles of multi-model experiments. The activities carried out in this task also aim at looking for making the interaction with and configuration of Grid environments straightforward and thus at improving the uptake of Grid technology on a larger scale. The ESM Grid Environment prototype allows exploiting enhanced grid services, namely GRB [6], developed by the University of Salento, and basic services offered by the Globus Toolkit [13] middleware. It has been designed to be compliant with the DEISA2 [18] and PRACE infrastructures. A workbench framework, based on web technologies, provides the users with the access to the computational power of the infrastructure. The GRB grid services, here employed, also support interaction with different underlying Grid middleware (such as Globus, gLite, Unicore or directly through SSH/SCP). The framework will provide tools to customize Grid users’ applications, to manage Grid resources and to support the development cycle of new Grid applications. With the help of this workbench not only Grid application users but also resource providers and application developers will be supported in their interactions with the Grid environment.

The report is organized as follows: after introduced the motivations for designing a grid prototype, the related work about the middleware, infrastructure and grid portal is described. Hence, the GRB services are presented and an overview of the whole architecture and its components is provided. Finally, the workbench framework is presented with description of the user interface and the services available to users today. The report ends with the description of the test case used for validating the infrastructure and the roadmap of the future work.

REQUIREMENTS FOR A CLIMATE GRID

One of the main features of the Grid environment is that it is based on the integration of heterogeneous computing resources. But the heterogeneity can also be a critical factor whenever we consider applications based on the adoption of numerical algorithms for problem solving. The heterogeneity can introduce differences on the execution of the same model on different machines, this because different compilers, or different level of code optimization can introduce a different order on the evaluation of the floating-point expressions. The ESM applications are very sensible to this kind of issues. For this reason, the climate scientist, even when possible, does not migrate a running experiment from a machine to another. An exception is represented by ensemble experiments. For this kind of experiments, composed by different members, the execution of each member is allowed on a different cluster. Moreover, an ESM job is typically a very long job that requires large amount of memory and data. Most clusters in the currently available Grid infrastructure have wall-time and CPU-time limitations or disk and memory quotas that may limit the execution of jobs, producing their premature end. Furthermore, there are also more chances to find miss-configured resources, due to the large number of sites and administrators involved. Regarding data transfers, when sites are scattered all over the world, network bandwidth becomes critical. Some disciplines, such as Bioinformatics or HEP, foresee high throughput jobs in which each short-time simulation does not manage large datasets nor need a huge amount of memory or disk space to be run. If such a simulation fails it is resubmitted again with minimum impact. The ES applications usually require running complex models during days, consuming a lot of memory and generating large amounts of data. A job failure can have a deep impact in terms of time and resource wasted. For these reasons, it is important to consider a grid infrastructure for ESM applications that overcome the limitations of the current grid infrastructures, as well as it can be necessary to do some changes in the workflow of the applications in order to adapt them to these limitations. The most important requirements for a successful grid-based climate application are [12]:

- 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 to detect them and act accordingly.
- Check-pointing for restart: the automatic creation of checkpoints allows managing a multitude of shorter jobs instead of a single long job. Thus, in case of failure we can restart a simulation from the most closely point it was interrupted. This is done by the creation of intermediate recovery simulation files written on 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 is the estimated time for completion, etc.

- Fast access to file server from both: computing server and post-processing server.

As mentioned, the current Grid middleware does not fully meet these requirements. Therefore, the development of a new framework is necessary to use a distributed Grid environment by climate modeling applications. This framework has to address all the previous requirements, and, at the same time, must be transparent and easy to use for the end user (usually a non Grid expert). With this intention, the grid architecture has been designed leveraging on the current grid services.

RELATED WORK

This section presents an overview of related portal frameworks and Grid Portals influencing GRB design and development.

GRID MIDDLEWARE

Nowadays, there are several Grid middleware implementations that provide seamless access to distributed resources. The first Grid middleware, Globus Toolkit [13], was developed in the late 90's in the United States and it is currently one of the most used implementations among the academia and industry. It offers core services such as resource allocation and process management (GRAM) for job submission and monitoring, monitoring and discovering (MDS) for distributed access to structure and state information and authentication and related security mechanisms based on mutual authentication (GSI). A middleware named gLite (glite.web.cern.ch) was created under the scope of the EGEE project in Europe (www.eu-egee.eu). It is the middleware used in most of the European Grid initiatives. gLite provides four different core services: the Berkeley Database Information Index (BDII) is the information core service, the Virtual Organizations Management system (VOMS) is the authorization service and the Large Hadron Collider Grid File Catalog (LFC) and the Workload Management System (WMS) are the data and execution core services respectively. Other services such as Logging and Bookkeeping keeps fresh information about the status of jobs processed by associated WMS and CREAM is a simple, lightweight service for job management operation at the CE level. UNICORE (www.unicore.eu) was initially developed to join German supercomputer centers. It offers core services such as the Gateway and Network Job Supervisor (NJS) for authenticating user requests and job submission based on SSL connections.

GRID INFRASTRUCTURES

Some of the current deployed grid infrastructures are presented in the following. The largest Grid infrastructure in the world is the one created under the European project Enabling Grids for E-sciencE. 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. 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. The DEISA (Distributed European Infrastructure for Supercomputing Applications, www.deisa.eu) grid infrastructure was born to interconnect supercomputing centers and mainly includes parallel clusters. Nodaway it puts together 11 of the most important supercomputing centers in Europe using the UNICORE middleware. TeraGrid [10] interconnects 11 American institutions using high performance networks and has, nowadays, a computing capacity over 1 PetaFlop and 30 PB of storage. It is based on the Globus Toolkit grid middleware. With respect to the climate science community, the most representative infrastructure is the Earth System Grid [24]. The ESG aims at easy the access to large amount of data for climate scientists. This data comprise more than 200TB distributed to more than 10,000 registered users in the ESG portal.

GRID PORTALS

One of the most relevant parts of a grid infrastructure is represented by the user's front end and access point to the infrastructure. The interface is mainly developed using the web and grid portals technologies. Due to space limitations, we only briefly describe some representative projects. Among Grid Portals for scientific computing we recall here the National Grid Service Portal [5], designed to access the core grid services available in the UK e-Science program and the P-GRADE portal [2], a workflow-oriented Grid portal developed at SZTAKI, Hungary, that enables the creation, execution and monitoring workflows in grid environments through high-level, graphical Web interfaces. Components of the workflows can be sequential and parallel (MPI, PVM) jobs. The SURA Coastal Ocean Observing and Prediction (SCOOP) [3] is a grid portal providing community access to data, models, and the resources of the SCOOP Grid. The portal provides tools for querying the data archive and downloading files, monitoring the status of the SCOOP Grid and file transfer. Capabilities for deploying model scenarios, such as wave/surge ensembles are currently being added, along with functionality for visualization. The Cactus Portal [1] developed by the Center for Computation and Technology at Louisiana State University evolved out of demands in the numerical relativity community where often large computational resources are required for the modeling of the complicated Einstein's equations. These numerical simulations can produce data sets in the order of terabytes and use thousands of processors, making them an ideal use-case for grid computing. The Telescience Portal [4] provides access and control of remote instruments, and allows managing data and submitting batch jobs. The Portal walks the user through the complex process of remote data acquisition via Telemicroscopy; Globus-enabled parallel tomographic reconstruction; advanced visualization, segmentation, and data processing tools; and transparent deposition of data products into federated libraries of cellular structure. Key features of the Portal include personalized user information; collaboration tools such as chat and shared white boards, automatic storage of data with the Storage Resource Broker, and job tracking tools. 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 [14] and ESG [24].

[22] and [11] offer recent reviews mainly focused on data. However, although some efforts have been made in order to adopt the Grid technology to execute applications [17] [21] [23] this issue is in a more incipient status. 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). Although the aim and scope of GRB and these Grid Portals are quite similar, there are some differences, because GRB:

- Has been designed to be user-centric instead of Virtual Organization-centric. The Portal does not manage a static, predefined set of grid resources, but allows each user to create her own grid, composed of computational resources belonging to several VOs if needed.
- Can interact with multiple Information Services and resource managers.
- Can use multiple credentials for a parameter sweep or workflow job.
- Uses an improved, extended version of the JSDL (Job Submission Description Language) to describe jobs.

GRB ARCHITECTURE

The portal architecture follows a standard three-tier model. The first tier is a client browser that can securely communicate with a web server on the second tier over a HTTPS connection. The web server exploits the GRB Scheduler, a GSI enabled web service on the second tier, which interacts on behalf of the users with grid services deployed on the third tier, the computational grid. In the following, the main components of GRB are described.

THE GRB PORTAL

GRB portal is made up of several components providing the following functionalities: User's Profile Manager: it handles the user's profile, to allow inserting, updating and removing information about grid resources.

User's Credential Manager: it allows configuring the credentials to be used for a given set of resources, retrieving them from a MyProxy server [9]. After this initial configuration step, the portal transparently retrieves the credentials needed to access specific resources.

Resource Finder: it is responsible for resource discovery querying an Information Service like the Globus Toolkit MDS or iGrid [7]. Other services can be added using the modular plug-in architecture of the portal. Indeed, we have developed many APIs that provide useful high-level abstractions for tasks such as information retrieval from an Information Service or job submission to different schedulers. This way, querying a new Information Service simply requires writing a low-level driver library to interface with the service being added and initializing, as needed the high-level API so that it can use the appropriate low-level driver.

Job Assistant: this component provides support for job submission. The portal contacts the GRB Scheduler providing it with all of the information needed for successful job submission. The user can specify the machines she wants to use or even let the Scheduler discover the pool of resources best suited for the execution.

Job Supervisor: it is responsible for monitoring job submissions, querying the GRB Scheduler to retrieve job status and results upon job completion.

THE GRB SCHEDULER

The GRB Scheduler acts as a metascheduler among the available grid resources. It has been designed to be fully compliant with respect to the JSDL specification. The JSDL language has been however extended in order to provide better support for: batch job definition, parameter sweep job definition and workflow graphs. Moreover, the extended JSDL allows defining multiple VOs access, managing multiple user credentials and defining a set of candidate hosts for resource brokering. The GRB scheduler has also been designed to meet the following requirements:

- Independence from a specific and predefined Virtual Organization. The GRB scheduler can act on behalf of the user among all of the specified computational resources; this means that if a user gains access to resources belonging to different VOs, the GRB will be able to use all of the user's credentials matching the remote resources security policies.
- Support for multiple and heterogeneous grid services. Due to the GRB libraries, the GRB scheduler can contact different remote resource managers such as Globus Toolkit GRAM and batch systems such as PBS and LSF.
- Modularity. The GRB scheduler has been designed to support different scheduling algorithms; new algorithms can be easily plugged in.
- Security. The GRB scheduler supports GSI and exploits the user's delegated credentials to act on grid resources on behalf of the user.

The scheduling process uses simple heuristics such as: Min-Min, Max-Min [20] [16], and Sufferage [19].

Min-Min is a simple algorithm which runs fast and delivers the satisfactory performance. The algorithm iteratively assign tasks to hosts by considering tasks i not yet scheduled with minimum of the computing predicted Completion Times (CTij) on host j. In most situations, Min-Min maps as many tasks as possible to their first choice of resources. However, the Min-Min algorithm is unable to balance the load well since it usually schedules small tasks first.

Max-Min algorithm is very similar to Min-Min, but assigns task with maximum expected completion time to the corresponding resource. The Max-Min algorithm may give a mapping with more balanced loads across the resources.

The Sufferage heuristic is based on the idea that better mappings can be generated by assigning a machine to a task that would “suffer” most in terms of expected completion time if that particular machine is not assigned to it. Let *diff* be the difference between the second minimum CTij and the minimum CTij, then the best is defined as the maximum *diff* over all i and j subscripts.

The scheduler support also the Workqueue [15] heuristic. It arbitrarily gives priorities to tasks and arbitrarily breaks the ties. The idea behind Workqueue is that faster processors will be allocated more tasks than slower processors. It does not use any prediction information on processor speeds and task lengths.

THE GRB LIBRARIES

The GRB software tools includes also the following production libraries:

- *grb_gram* for job execution using the Globus GRAM. We added the non-blocking versions of the functions in order to asynchronously submit, check and cancel a job.
- *grb_gridftp* for secure file and directory transfer.

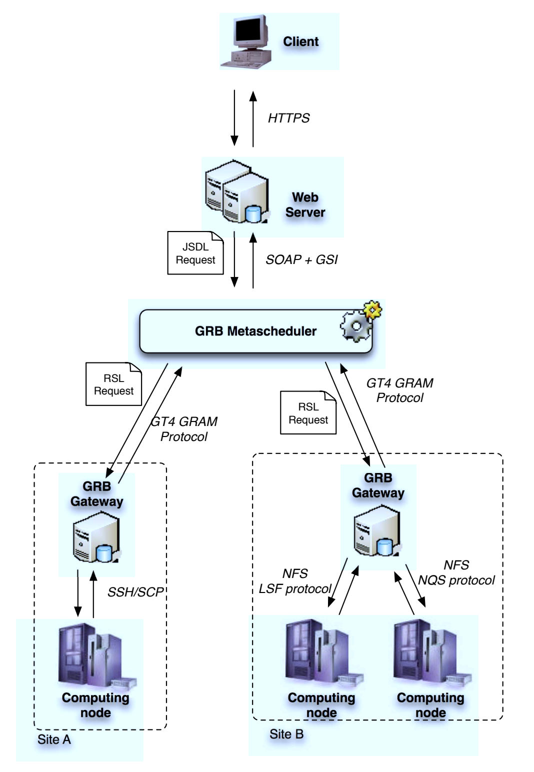


Figure 1:
Grid infrastructure architecture

- *grb_igrid* for querying the iGrid Information Service.
- *grb_mds* for querying the Globus MDS2 information service.
- *grb_lsf* for job submission with the LSF batch system.
- *grb_myproxy* for the retrieval of the user's proxy credential.

Other libraries covering additional grid services are actively being developed.

GRID PROTOTYPE INFRASTRUCTURE OVERVIEW

DESIGN OVERVIEW

The grid infrastructure is depicted in the Figure 1.

The main components of the architecture are:

Client: it represents the end-user terminal with a web browser. The client terminal can be a laptop, a workstation, a PDE or more in general a device with a web client supporting the HTTPS protocol.

Web Server: it provides the client with a front-end interface based on a web GUI, and implements all the mechanisms for contacting and using the GRB Metascheduler.

GRB Metascheduler: it is the core service for managing the distributed computing environment. It receives requests for job submissions in a JSDL format, according to a selected scheduling policy and forwards the request to the computational resources. It supports different submission protocols and namely: GT4 GRAM protocol, gLite WMS protocol and UNICORE NJS protocol.

GRB gateway: this component is mainly devoted to decouple the computational node from the rest of the grid architecture. The GRB gateway acts as a connector between the low level services (often not grid compliant), installed on the computational nodes, and the rest of the grid architecture. Due to the services provided by the GRB gateway, no kind of installation and reconfiguration are needed on the computational node. This aspect is very important considering that often the computational node is a parallel cluster used for production runs and tuned for best performance.

Computational node: it represents the target machine where the coupled model will be run. No specific requirements are needed for a computational node to be integrated in the grid infrastructure. The only requirements are: the

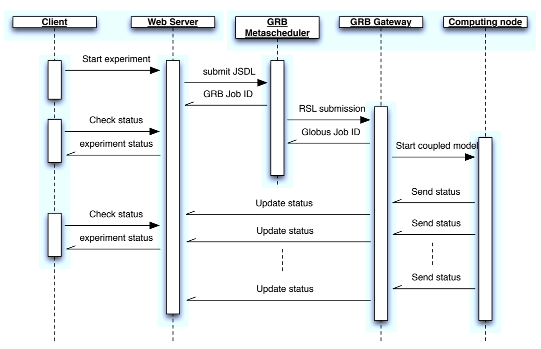


Figure 2:
Sequence diagram for submission and monitoring of experiment

coupled model must be installed; the GRB gateway must be able to remotely access to the local scheduler for starting the run and must be able to remotely access to the file system. Up to day the GRB gateway can remotely submit a job through an SSH connection or directly using an LSF client. It can access the file system through a SCP connection or through NFS.

The sequence diagram in Figure 2 illustrates the mechanism that takes place for the submission and monitoring of the experiment.

In order that a user can access to the grid infrastructure, an authentication and authorization phase is required. The user is authenticated via username and password provided through the HTTPS protocol between the web client and server. The web server stores some relevant information about the user like: the computational infrastructure she is allowed to use; the distinguished name and a x509v3 user proxy certificate for the authentication against the target computational resources; and a user work space. After the user provides a valid username and password, the web server sets a session through an unforgeable cookie (it contains digested information about the session). For submitting, the user has to specify the experiment parameters through the GUI on her web browser (the web interface is better described in section "Grid Portal"). Upon receiving the request for submission, the web server starts the following steps:

- It validates both the request and parameters specified by a form;
- It accesses the local DB for retrieving the computational resources the user is allowed to use;
- It creates the JSDL document that includes the submission parameters and information related to the underlying computational resources;
- It contacts the GRB Metascheduler via SOAP protocol over a GSI-enable connection, for sending a submission request described by the JSDL document. The mutual authentication occurs between the web server and the GRB Metascheduler. The GRB Metascheduler validates the request and provides back the grb-job-ID;
- It sends to the web client the grb-job-ID and notifies that the submission has been successfully completed;
- The GRB Metascheduler starts processing the submission request holding the request in the appropriate queue. Asynchronously and according to the GRB scheduler policy, the requests are extracted from queues and processed. Namely the GRB scheduler: i) Contacts the Information service of the computational resources defined in the JSDL document; ii) Splits the request into sub-jobs. If for example the request is related to an experiment with 30 members, 30 batch jobs will be identified; iii) According to

the scheduling policy, the GRB scheduler spreads the jobs on the available resources taking into account their actual computational load; iv) Retrieves the user proxy certificate from the MyProxy server using the credentials contained in the JSDL document; v) Creates the job submission request according to the low level grid middleware deployed on the target machine. Hence, the GRB scheduler creates a RSL string if the underlying grid service is GT4 GRAM, whether it creates a JDL document if gLite WMS must be contacted or an AJO binary file if Unicore NJS is contacted; vi) Contacts the remote resource manager using the user's proxy certificate and sends the job submission request.

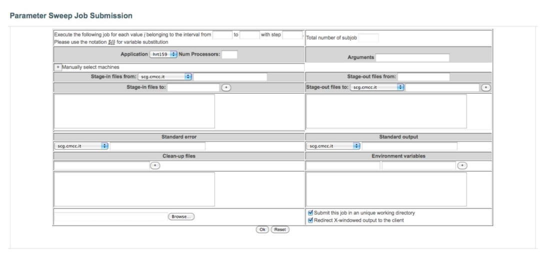
The GRB gateway introduces a further level of abstraction. It provides the low level grid services and hence a grid middleware must be installed. Since the current grid middleware tools are compliant with the parallel cluster used for running the coupled model, the GRB gateway seems needless in the architecture. It is worth noting here that actually the GRB gateway is not mandatory if the final computational resource is configured for a grid environment and if it includes all of the low level services available in the grid middleware toolkit. Even though, the deployment of grid services on a parallel cluster, commonly used for production runs, could require a deep revision of the cluster's configuration. New services require that some network ports must be opened for remote accesses, hence the security policy must be revised; a node of the cluster should be dedicated for such services in order to avoid some lost of performance, hence the cluster must be retuned. The GRB gateway allows the grid infrastructure to be not invasive with regards to the target computational resource. With the GRB gateway no any further service and any kind of reconfiguration must be performed on the target machine. The GRB gateway acts as an interface between the grid technologies and the technology used for managing parallel clusters. The grid infrastructure can be easily extended joining other climate centers using the software package for grid services and documentation that will be make available through the vERC portal.

GRID PORTAL

The submission in a distributed environment could require a considerable effort by the climate scientist; for this reason a high level web based interface for job submission and monitoring has been developed. The grid portal provides mechanisms to efficiently balance the workload over the available resources, a simple view of all of the experiment belonging to a user and an immediate overview of the status of all of the members belonging to a given ensemble with statistical information and an estimation of the time to complete.

EXPERIMENT SUBMISSION

An interface for the simulation of an ensemble experiment was designed and implemented. It allows the job submission on a set of computational resources involved into the grid prototype infrastructure. The interface allows specifying the starting date and prolongation of the experiment. Since an ensemble can have more members that start from a specific date, the interface takes as argument a configuration file that contains all of the input parameters used in the simulation. The interface allows specifying the input file expressed in a parametric form through an iterative index: each index identifies a member. The parameters include: the input



Parameter Sweep Job Submission

Execute the following job for each value *i* belonging to the interval from ___ to ___ with step ___
Please use the notation *${i}* for variable substitution

Total number of subjob ___

Application: hrt159 | Num Processors: ___

Arguments: ___

+ Manually select machines

Stage-in files from: scg.cmcc.it

Stage-in files to: ___ (+)

Stage-out files from: ___

Stage-out files to: scg.cmcc.it ___ (+)

Standard error: scg.cmcc.it

Standard output: scg.cmcc.it

Clean-up files: ___ (+)

Environment variables: ___ ___ (+)

Browse...

[✓] Submit this job in an unique working directory
[✓] Redirect X-windowed output to the client

Ok | Reset

Figure 3:
Submission Web Interface

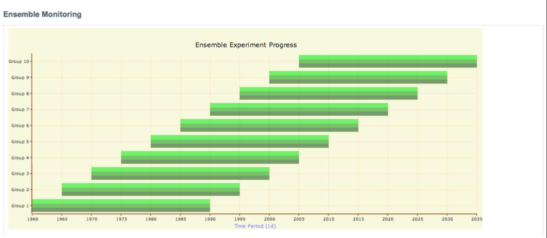


Figure 4:
Ensemble Monitoring Interface

files that need to be transferred from the storage node to the computing resource (the transfer is made during the run of the job using the GridFTP protocol); the output files that need to be copied from the computing resource to the machine specified by the user (the completion of the transfer is made through the GridFTP protocol). The interface can also specify the resource where transfer the files related to the standard error and output for the job. When a user submits a job, a description file in the JSDL format [8] is sent to the GRB scheduler that takes care of the job, submitting it by choosing from a set of computational resources previously configured through the GRB Grid Portal. The description file can be specified from the same interface to allow the repetition of the experiment and to guarantee the ability to load all of the submission parameters for potential change.

A snapshot of the interface is shown in Figure 3.

Summarizing the submission web interface allows:

- The submission of parameter sweep jobs.
- Stage in and stage out of input, standard error and output files.
- Automatic upload of ensemble parameters through a JSDL file.

RUNTIME MONITORING

Regarding the job monitoring, two interfaces have been designed and implemented. These interfaces are prototypal and aims at proposing to the climate community a starting point for further refinements. Future work will regard possible updates/adds following the feedback of the scientist. The first view gives a summary of the job status for both the whole experiment and each member of the ensemble; the second allows a graphical monitoring of the period simulated by the ensemble (expressed in days, months or years). The Figure 4 shows the interface for the monitoring of an experiment involving 10 mini-ensembles each one with a different starting date. 3 members with perturbed initial condition compose each mini-ensemble.

The user interface enables a real time view of the progress of the experiment. The interface offers a view as tabular summary of the experiment with some information such as the hostname where the job was done, the beginning and end of the ensemble, the start and final dates of the simulation, the simulated days/months/years and the total number of days/months/years. Other parameters are related to the SYPD (Simulation Per Years Day) and Estimated Time to Arrival (Figure 6). Moreover, details are provided on queuing, execution, waste and elapsed times of the host where a specific member of the ensemble was run.

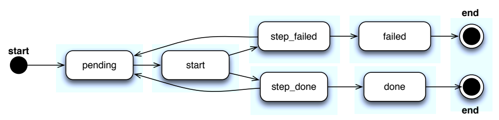
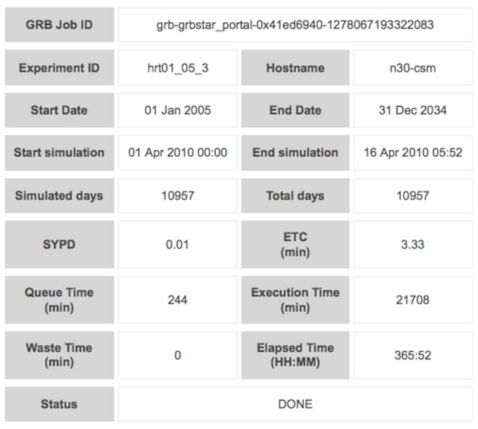


Figure 5:
State diagram for the execution of a member



GRB Job ID	grb-grbstar_portal-0x41ed6940-1278067193322083		
Experiment ID	hrt01_05_3	Hostname	n30-csm
Start Date	01 Jan 2005	End Date	31 Dec 2034
Start simulation	01 Apr 2010 00:00	End simulation	16 Apr 2010 05:52
Simulated days	10957	Total days	10957
SYPD	0.01	ETC (min)	3.33
Queue Time (min)	244	Execution Time (min)	21708
Waste Time (min)	0	Elapsed Time (HH:MM)	365:52
Status	DONE		

Figure 6:
Summary of the ensemble experiment in a tabular format

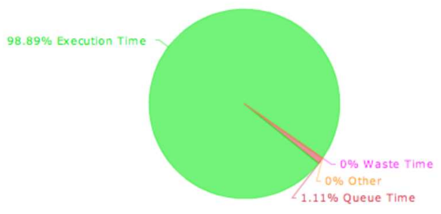
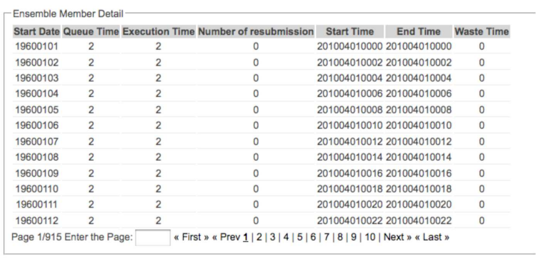


Figure 7:
Queuing, execution and waste times of the ensemble experiment

A pie chart shows the queuing, execution and waste times (Figure 7).

The Figure 5 shows the state diagram related to a member of an ensemble experiment. The states have the following means:



Ensemble Member Detail

Start Date	Queue Time	Execution Time	Number of resubmission	Start Time	End Time	Waste Time
19600101	2	2	0	201004010000	201004010000	0
19600102	2	2	0	201004010002	201004010002	0
19600103	2	2	0	201004010004	201004010004	0
19600104	2	2	0	201004010006	201004010006	0
19600105	2	2	0	201004010008	201004010008	0
19600106	2	2	0	201004010010	201004010010	0
19600107	2	2	0	201004010012	201004010012	0
19600108	2	2	0	201004010014	201004010014	0
19600109	2	2	0	201004010016	201004010016	0
19600110	2	2	0	201004010018	201004010018	0
19600111	2	2	0	201004010020	201004010020	0
19600112	2	2	0	201004010022	201004010022	0

Page 1/915 Enter the Page: « First » « Prev 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Next » « Last »

Figure 8:
Details of an ensemble member

"pending": the job is waiting for the availability of the requested resources;
"start": the job is executing and running;
"step_done": the simulation of one restart period has finished;
"done": the execution of the member is completed;
"step_failed": the execution of one restart period is finished with an error, the simulation will be restarted executing the same period;
"failed": the execution of the member is finished with error.

A tabular form shows the details of each member of the ensemble (start date, queuing time, execution time, number of resubmissions, start time and end of the simulation - Figure 8).

Finally a map allows monitoring the grid activity onto a 3D globe. Navigation around the globe is simple with the mouse's scroll wheel zooming in and out and left click with drag moving the globe. There is also a summary of the number of jobs running, pending and done across the globe on the screen. A marker, pointed at the location of the resources, represents each active site that is connected to the Grid. If the site is actively processing work then the site is a pulsing circle of magenta and green. This circle is a pie chart representing the proportion of jobs running at the site (the green segment) and jobs queued to be run (magenta segment).

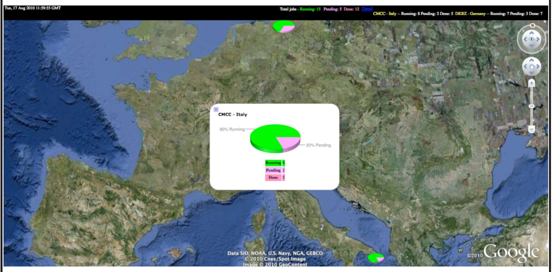


Figure 9:
Monitoring of the grid activity

The maximum size of the circle is related to the total number of jobs at a site (both running and queued). The site with more jobs has a bigger circle. The size is a logarithmic scale so the sites only grow as it reaches a new order of magnitude. In the Figure 9 the monitoring of the grid activity between CMCC and DKRZ sites during the execution of the ensemble experiment is showed.

This interface also keeps track of detailed information about each site on the Grid. Clicking on a hypertext link will result in a bottom window, which details the jobs that have been run on all of the sites. The monitoring interface has a refresh time of 30 sec. All information are stored in a relational database and periodically updated by an application that reads the log files produced during the experiment. The monitoring interface is regularly updated by reading the partial data from the database for the construction of the graph of the experiment and, on demand for a specific member, the data are visualized both in tabular and graphical format by using AJAX and JSON technologies. Regarding the map, the google earth API have been used for the development jointly with AJAX for adding the animation of the grid sites. In summary the monitoring interface offers the following features:

- Monitoring both the entire experiment and each individual member of the ensemble with advancement through a graphical display of each member and specific interaction with the object.
- Contents in tabular and graphic format of the experiment and detail of each individual member.
- Real time monitoring of the grid activity onto a 3D globe.

A CASE STUDY: THE ENSEMBLE EXPERIMENT

An ensemble experiment is characterized by a multitude of predictions (also named members), each of them obtained by perturbing some initial conditions. From a computational point of view, the members of an ensemble experiment are independent each other and they can be simultaneously executed using different input dataset (representing above mentioned initial conditions). An ensemble run can be considered as a high-throughput parallel application suitable for a distributed environment.

DESCRIPTION OF THE EXPERIMENT

The HRT159 is a global coupled ocean-atmosphere general circulation model (AOGCM) developed by CMCC-INGV. The atmospheric model component (ECHAM5) has a horizontal resolution of about 80 Km with 31 vertical levels, while the global ocean model (OPA8.2) has horizontal resolution of about 2° with an equatorial refinement (0.5°) and with 19 vertical levels. The communication between the atmospheric and the ocean model is performed through the CMCC parallel version of OASIS3 coupler, and the exchange of SST, surface momentum, heat, and water fluxes occurs every 2h40m. The total number of fields exchanged is 23.

The HRT159 model, such as many coupled models, is used to execute an ensemble experiment. This particular ensemble is composed by 10 members; for each of them 3 different run are executed perturbing initial conditions. Each member simulates 30 years and is characterized by a temporal phase-displacement of 5 years.

CONFIGURATION OF THE EXPERIMENT

Each component model is used with the spatial and temporal resolutions shown in Table 1, while the coupler OASIS3 has been configured as in Table 2.

Table 1
Spatial and temporal resolution of the component models

	OPA8.2	ECHAM5
time step	4800s	240s
grid points	182x149	480x240
vertical levels	19	31

Table 2
OASIS3 configuration

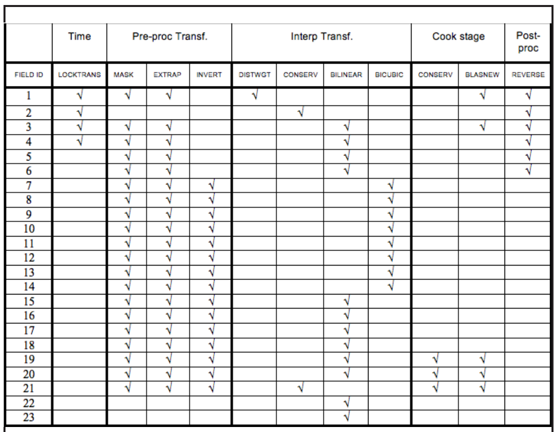
OASIS3 configuration	
Coupling period	9600s
Total number of fields to be transformed	23

	Number of fields exported to	Number of fields imported from	LAG
OPA8.2	17	6	4800s
ECHAM5	6	17	240s

In order to give an idea of the computational load, a detailed view of the transformations performed by OASIS3 on the exchanged fields is given in Figure 10.

THE RUN SCRIPT

If we consider a distributed environment, the monitoring of a run execution needs additional

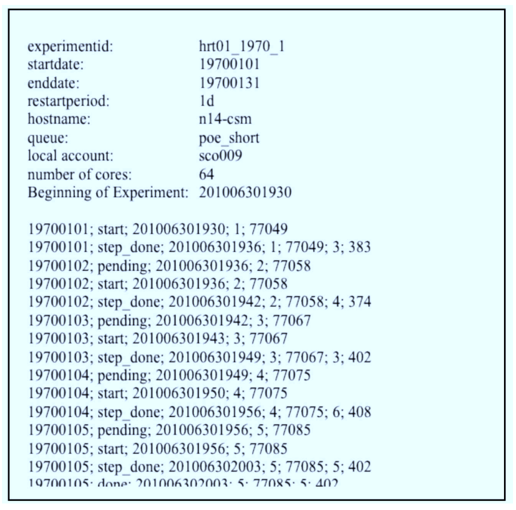


	Time	Pre-proc Transf.			Interp Transf.				Cook stage		Post-proc
FIELD ID	LOCKTRANS	MASK	EXTRAP	INVERT	DISTWGT	CONSERV	BILINEAR	BICUBIC	CONSERV	BLASNEW	REVERSE
1	√	√	√		√					√	√
2	√					√					√
3	√	√	√				√			√	√
4	√	√	√				√				√
5		√	√				√				√
6		√	√				√				√
7		√	√	√				√			
8		√	√	√				√			
9		√	√	√				√			
10		√	√	√				√			
11		√	√	√				√			
12		√	√	√				√			
13		√	√	√				√			
14		√	√	√				√			
15		√	√	√			√				
16		√	√	√			√				
17		√	√	√			√				
18		√	√	√			√				
19		√	√	√			√		√	√	
20		√	√	√			√		√	√	
21		√	√	√		√			√	√	
22							√				
23							√				

Figure 10:
OASIS3 transformations

information w.r.t. a classical run on a selected cluster. This way, we can obtain detailed information about a grid site and the ensemble members here executed. Besides setting up the environment and launch the coupled model, the run script has been modified in order to also store information about execution within a log file. An example of the produced log file has been reported in Figure 11. The example is related to the simulation of 5 days with a restart period of 1 day. The first section of the log file reports some general information about the experiment, such as the experiment ID, start and end dates of the simulated period, the restart period, the hostname of the resource allocated for the execution, the LSF queue name, the username used for submitting the job, the number of cores used and beginning date of the experiment expressed in the yyyymmddhhmm format. The second section is related to the execution of the subjobs, which number derives from both the simulated and the restart period. Each subjob has a number of entries equal to the number of status the job has been characterized. Suitable statuses are: start, pending, step_done, step_failed, done.

Each entry reports the following information



```
experimentid:              hrt01_1970_1
startdate:                 19700101
enddate:                   19700131
restartperiod:             1d
hostname:                  n14-csm
queue:                     poe_short
local account:             sco009
number of cores:           64
Beginning of Experiment:   201006301930

19700101; start; 201006301930; 1; 77049
19700101; step_done; 201006301936; 1; 77049; 3; 383
19700102; pending; 201006301936; 2; 77058
19700102; start; 201006301936; 2; 77058
19700102; step_done; 201006301942; 2; 77058; 4; 374
19700103; pending; 201006301942; 3; 77067
19700103; start; 201006301943; 3; 77067
19700103; step_done; 201006301949; 3; 77067; 3; 402
19700104; pending; 201006301949; 4; 77075
19700104; start; 201006301950; 4; 77075
19700104; step_done; 201006301956; 4; 77075; 6; 408
19700105; pending; 201006301956; 5; 77085
19700105; start; 201006301956; 5; 77085
19700105; step_done; 201006302003; 5; 77085; 5; 402
19700105; done; 201006302003; 5; 77085; 5; 402
```

Figure 11:
Experiment log file

split by semicolon:

- The date of the simulated period expressed in the yyyymmdd format.
- The job status.
- The time when the status has been reached expressed in the yyyymmddhhmm format.
- The subjob identifier.
- The LSF job ID.

For some status (step_done and step failed) we have other information extracted from the accounting system of LSF:

- The queue time;
- The execution time.

IS-ENES TESTBED

The distributed infrastructure above described has been tested on some of the IS-ENES nodes. In particular, the infrastructure has been used and tested within the prototype grid infrastructure built among some of the main climate centers under the ENES network, CMCC and DKRZ. BSC defined the computing resource to be integrated within the testbed as soon as possible.

CMCC NODE

The CMCC node includes a computing resource (an IBM cluster, named *Calypso*) and a gateway machine for submitting on *Calypso* nodes.

The IBM cluster has 30 IBM p575 nodes, each of them equipping with 16 Power6 dual-core CPUs at 4.7GHz (8MB L2/DCM, 32MB L3/DCM). With Simultaneous Multi Threading (SMT) support enabled, each node hosts 64 virtual cores. The whole cluster provides a computational power of 18 TFLOPS. Each node has 128GB of shared memory (4GB per core), two local SAS disks of 146,8GB at 10k RPM and two Infiniband network cards each one with four 4X IB galaxy-2 and four Gigabit network adapters. Some nodes are used as GPFS and TSM servers and have also two fibre channel adapters at 4Gb/s FC and two fibre channel adapters at 8Gb/s for interconnecting to the storage system. *Calypso* has 2 storage racks, each one equipped with 280 disks of 750GB, providing a total capacity of 210TB of raw storage. These disks are working with GPFS. *Calypso* interconnects also a tape library with 1280 cartridges LTO4 at 800GB (1PB total capacity) and Tivoli TSM for handling Hierarchical Storage Management. The default compilers are IBM XL C/C++, and IBM XL FORTRAN. The default resource scheduler manager is LSF.

Calypso is on a private network and can be accessed only using a gateway. The gateway is a virtual machine characterized by the CentOS v.5.5 x86_64 OS, 4GB of RAM, 30GB of HD and 1 core of a Xeon quadcore CPU Intel E5520. The virtual machine is running on a server IBM x3650M2 equipped with 2 processors Xeon quadcore Intel E5520 at 2.27 GHz, 48GB of RAM and 4 SAS disks of 500GB.

DKRZ NODE

The DKRZ node is composed of the computing backend (an IBM HPC Cluster, called *Blizzard*) and a frontend node providing access to the backend. The IBM cluster consists of 264 p575 nodes (16 dual core CPUs per node, 8448 cores), where 249 nodes act as computing nodes, 12 nodes are dedicated to I/O and 3 are dedicated to interactive access. A 3 PetaByte GPFS files system is attached and 20 TeraByte main memory is integrated. 8 Infiniband switches provide an overall bandwidth of 16 Gigabyte/s for node to node data trasfers (bidirectional). The actual ranking of the system in the top 500 list is 41 (as of June 2010) with a peak performance of 151 Teraflops.

Attached to the HPC cluster is a large HPSS High Performance Storage System with an overall capacity of 60 PetaByte and 500 TeraByte disk cache. The bidirectional bandwidth is 3 GigaByte/s (sustained) and 5 GigaByste/s (peak).

The initial deployment of the grid frontend was directly on a frontend node of the IBM HPC system, providing direct access to job submission functionality as well as gridftp based data transfer functionality.

BSC NODE

The MareNostrum supercomputer is based on processors PowerPC, architecture BladeCenter, Linux operating system and Myrinet interconnection. MareNostrum has 10240 IBM Power PC 970MP processors, 20 TB of main memory and 280 + 90 TB of disk storage. It uses two interconnection networks: Myrinet and Gigabit Ethernet. It is the first supercomputer that runs under a Linux operating system, a SuSe Distribution. The Peak Performance of the system is 94.21 Teraflops.

Marenostrum has 44 racks, 31 of them dedicated to computing tasks. The computing racks have a total of 10240 processors. Each rack is formed by 6 Blade Centers. In total, each rack has 336 processors and 672 Gb of memory; each one has a rough peak performance of 3.1 Tflops.

Each Blade Center has 14 server blades type JS21. Each of these nodes has 2 processors PowerPC 970MP at 2.3 GHz, 8 Gb of shared memory between both processors and a local SAS disk of 36 Gb. Each node has a network card Myrinet type M3S-PCIXD-2-I for its connection to the high speed interconnection and the two connections to the network Gigabit. Each node has a local disk of 36 Gb and works diskless, i.e., the operating system is in the storage racks instead of the local disk and it is loaded through a Gigabit network when each node is initialized.

In addition to the local disk of each node, MareNostrum has 20 storage servers arranged in 7 racks. These servers have a total of 560 disks of 512GB and each one provides a total capacity of 280TB external storage. These disks are working with Global Parallel File System (GPFS), which offers a global vision of the file system and also allows a parallel access.

PRELIMINARY RESULTS

Within the infrastructure, two kind of file transfers occur: the input files (grouped in a compressed tarball file) are sent from the storage node to each of the gateway nodes using gridftp protocol; the gateway nodes send the input files to the computing node through a SCP protocol. The infrastructure we have tested is composed of:

- The storage node is scg.cmcc.it.
- The DKRZ's computing node is blizzard.dkrz.de, accessible through the gateway scg.cmcc.it.
- The CMCC's computing node is calypso.cmcc.it, accessible through the gateway scg.cmcc.it.
- The GRB scheduler node is grb.cmcc.it.

It is worth noting here that scg.cmcc.it acts as gateway node for both blizzard and calypso. The following tests have been performed at different times during the day and with different file sizes:

- Grid-FTP transfer from grb.cmcc.it to scg.cmcc.it.
- SCP transfer from scg.cmcc.it to blizzard.dkrz.it and vice versa.
- SCP transfer from scg.cmcc.it to calypso.cmcc.it and vice versa.

The results are presented in Figure 12.

It is worth noting here that the members of an ensemble experiment use common input files. In order to obtain the best performance we have chosen to deploy the common input files to all of the computing nodes before running the ensemble, and leave to the grid management services to transfer the specific input dataset on the computing node where the memeber will be executed. For the considered ensemble experiment, the specific input file, for each member, is of 1.8 GB.

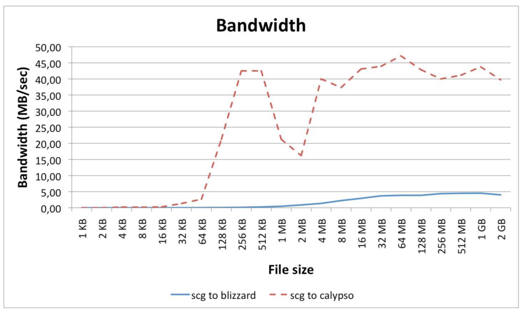


Figure 12:
File transfer tests among scg, blizzard and calypso

A third kind of file transfer should be also considered. The output file should, in some cases, be transferred to the post-processing node for data analysis and visualization. The infrastructure we have designed does not take into consideration the mechanism for transferring the output files because we are waiting from other IS-ENES WPs (namely from JRA4) the specifications, protocols and services interfaces for accessing data nodes. The output files for the considered ensemble experiment refers to a one-month simulation and have the following sizes (Table 3):

Table 3
Output and restart file sizes for a one-month simulation of the RT159 coupled model

	OPA8.2	ECHAM5	LIM	OASIS
Output	1,51	9,18	-	-
Restart	0,09	1,18	0,01	0,02

OVERALL SPEED-UP

The evaluation of the speed-up has been performed considering the case where the clusters are almost fully loaded and there is only one node available both on calypso and blizzard.

In order to reduce the elapsed time of the tests, we tailored the ensemble with 6 members each one with 31 simulation days and a restart period of one day. The members differ each other for the starting date, and namely we used input files referring to first of January of 1960, 1965, 1970,1975, 1980, 1985.

Each of the members has been run using 64 processes and hence allocating an entire node.

The results report the queue time, running time and elapsed time in different cases (see Table 4):

- Case A1: the grid infrastructure is not available, thus the ensemble experiment is run only on one cluster (calypso).
- Case A2: the grid infrastructure is not available; the ensemble experiment is run only on one cluster (blizzard).
- Case B: the experiment can exploit the grid infrastructure composed by blizzard and calypso and the computational load of both nodes are equivalent.
- Case C: the same as the previous case but supposing that calypso is more computationally loaded w.r.t. blizzard (we supposed that on average there are two nodes available on calypso and only one node on blizzard).

The queue time for case A1 and A2 refers to the time spent by the member waiting the availability of the resource. Being a member made of several resubmissions (each one simulates a time frame equal to the restart period parameter), even if at the beginning the node is available for running the member, we have no guarantee that the node will be available also for the execution of next periods. In this case the execution of the member can be waiting in queue between a restart period and the next one. The total runtime is the sum of the execution time (excluding the queuing time) of all of the simulations on all of the nodes. It expresses the total computing effort required for the entire experiment. The elapsed time measures the time from the start of the first job to the end of the last, it thus includes the queuing time. For case B and C, the queue time is the sum of the queue time measured on calypso and blizzard. The parallelism level is computed as the ratio between the total run time over the elapsed time. The load balance is the average of all of the times each member is executing over the maximum one (1 for an ideal perfect balanced configuration and for a totally unbalanced configuration).

Table 4
Execution time and speed-up of grid experiments

Case	Total Queue Time (min)	Total Run Time (min)	Elapsed time	Parallelism level	Load balance %
A1	12,32	1247,62	1259,94	1,00	100
A2	246,00	1212,00	1458,00	1,00	100
B	129,15	1230	729	1,86	93
C	90,08	1236	486	2,73	91

Considering the time for transfer the input files we got the results in Table 5.

Table 5
Overall elapsed time considering the input file transfer

Case	File Transfer (min)	Total Run Time (min)	Elapsed Time (min)
A1	0,78	1247,62	1260,72
A2	7,67	1212,00	1465,67
B	8,45	1230	736,67
C	8,45	1236	493,67

CONCLUSIONS

In this report, the first grid prototype system for the IS-ENES community has been described. The main goal of this work has been to demonstrate the system, as grid technology could be a valid mean for efficiently running complex experiments composed by ensemble simulations, inside the ESM climate community. The design of the infrastructure involved the development of several services for job submission and monitoring and interactive GUIs for controlling the ensemble progress.

The testbed involves three computing sites: the CMCC, DKRZ and BSC. Whereas the deployment of the infrastructure is complete for CMCC and DKRZ, the BSC node will be active in the next months. Hence the first prototype will be deployed also on its computing resources. As future work, the full integration of this system in the vERC portal will be performed. In particular, from the portal, after the registration and authentication, the user will access the interfaces for job submission and monitoring.

Integration with the services for data management is also required. The produced output file should be transferred to data nodes for archiving , data analysis and visualization. This task can be completed once the activity of JRA4 have been defined the protocol and service for interfacing to data bodes.

In the next phase, an activity will be to decide what are the models available for this system and for which users. The idea is to create several Virtual Organizations in which the users of the same group share the same models and we are considering some ÒtoyÓ models for guest users. Hence, when a user asks for registration to the vERC portal, a guest account for job submission will be automatically created. A user that would use an advanced model should request an account to a VO configured by administrator for the use of the model on the resources (as already adopted in the EGEE community). Next activities will require establishing these policies for using the grid platform.

Another activity will be to evaluate the possibility to integrate the services deployed in the GRB framework with the COMP Superscalar system, developed by BSC, which aims to ease the development of grid applications.

Bibliography

[1] The cactus portal. https://portal.cct.lsu.edu/gridsphere/grid_sphere.

[2] The p-grade portal. http://www.lpds.sztaki.hu/pgportal.

[3] The scoop portal. http://carmen.csi.lsu.edu/gridsphere/grid_sphere.

[4] The telescience portal. http://niblick.ucsd.edu:8080/gridsphere/gridsphere.

[5] The uk national grid service portal. https://portal.ngs.ac.uk.

[6] G. Aloisio, M. Cafaro, E. Blasi, and I. Epicoco. The Grid Resource Broker, a Ubiquitous Grid Computing Framework. *Special Issue on Grid Computing*, 10(2):113–119, 2002.

[7] G. Aloisio, M. Cafaro, I. Epicoco, S. Fiore, D. Lezzi, M. Mirto, and S. Mocavero. iGrid, a Novel Grid Information Service. In *Lecture Notes in Computer Science, Springer-Verlag*, volume 3470, pages 506–515. Proceedings of Advances in Grid Computing - EGC 2005, 2005.

[8] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job submission description language (jsdl) specification. Technical Report 1.0, Open Grid Forum, 2005. Tech. Rep. GFD-R.056.

[9] J. Basney, M. Humphrey, and V. Welch. The MyProxy Online Credential Repository. *Software: Practice and Experience*, 35(9):801–816, 2005.

[10] C. Catlett. The philosophy of teragrid: building an open, extensible, distributed terascale facility. In *2nd IEEE/ACM International Symposiumon Cluster Computing and the Grid*, pages 8–8, 2002.

[11] R. Cossu, M. Petitdidier, and V. Linford. A roadmap for a dedicated earth science grid platform. *Earth Science Informatics*, 3(3):135–148, 2010.

[12] V. Fernandez-Quiruelas, J. Fernandez, A. S. Cofino, J. M. Gutierrez, C. Baeza Retamal, R. Abarca del Rio, R. Miguel San Martin, and M. Carrillo. Climate modelling on the grid experiences in the eu-project eela. In *Proceedins of the Third EELA Conference*. R. Gavela and B. Marechal and R. Barbera and L.N. Ciuffo and R. Mayo., 2007.

[13] I. Foster. The globus toolkit for grid computing. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:2, 2001.

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

[15] T. Hagerup. Allocating Independent Tasks to Parallel Processors: An Experimental Study. *Journal of Parallel and Distributed Computing*, 47:185–197, 1997.

[16] O.H. Ibarra and C.E. Kim. Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors. *Journal of the ACM*, 24(2):280–289, 1977.

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

[18] H Lederer. Deisa2: supporting and developing a european high-performance computing ecosystem. *Journal of Physics: Conference Series*, 125, 2008.

[19] M. Maheswaran, S. Ali, and H.J. Siegel. Dynamic Mapping of a Class of Independent tasks onto Heterogeneous Computing Systems. *Journal of Parallel and Distributed Computing - Special issue on software support for distributed computing archive*, 52(2), 1999.

[20] M. Maheswaran, S. Ali, H.J. Siegel, and R. Freud. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneus Computing Systems. pages 30–44. 8th IEEE Heterogeneous Computing Workshop (HCW'99), 1999.

[21] M. Mineter, C. Jarvis, and S. Dowers. 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, 2003.

[22] P. Renard, V. Badoux, M. Petitdidier, and R. Cossu. Grid Computing for Earth Science. *Eos, Transactions American Geophysical Union*, 90(14):117–119, 2009.

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

[24] D. N. Williams, R. Drach, R. Ananthakrishnan, I. T. Foster, D. Fraser, F. Siebenlist, D. E. Bernholdt, M. Chen, J. Schwidder, S. Bharathi, A. L. Chervenak, R. Schuler, M. Su, D. Brown, L. Cinquini, P. Fox, J. Garcia, D. E. Middleton, W. G. Strand, N. Wilhelmi, S. Hankin, R. Schweitzer, P. Jones, A. Shoshani, and A. Sim. The earth system grid: Enabling access to multimodel climate simulation data. *Bulletin of the American Meteorological Society*, 90(2):195–205, 2009.