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

V. Fernández-Quiruelas[a], C. Blanco[a], A.S. Cofiño[a], J. Fernández[a]

[a] *Grupo de Meteorología, Dpto. Matemática Aplicada y CC. Computación. Universidad de Cantabria. Santander, Spain*

Resumen

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 modelling 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.

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

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 and homogeneity [41]. 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 [16, 23, 31]. 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 [11, 27, 42]. These works do not solve the issues related to the heterogeneity of resources nor to the data management. Instead, 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. [14] 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. [4] 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 of scalability and scheduling capabilities.

Other groups have explored new possibilities to run applications on HDCIs without a common middleware. Bretherton et al. [5] 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 [37]. 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 [36] 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 [37]. 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 modelling 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 [39] 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 [31, 33], 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

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

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 [20], SGE [19], LSF [51], SLURM [50], LoadLeveler [24] and Condor [29] are examples of Cluster middlewares; gLite [28], Globus Toolkit [15], ARC [12] and UNICORE [1] are examples of Grid middlewares; and Eucalyptus [35], OpenNebula [40], OpenStack [10] and Nimbus [44] 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 [45], gUSE/WS-PGRADE [13], GridWay [21] or Condor-G [17]. 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

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

Table 1: Comparison of software frameworks.

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 [17]. DIRAC [7] relies on Pilot Jobs the job scheduling. 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 [26]. 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 [49], 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 [22] 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. And 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 transfered. 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 [25, 43, 30] 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. [43] 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 [38, 2], to date, there are no solutions that provide a smart replica allocation. gLite LFC [3] and Globus RLS [8], 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 [39]) 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 reproduce the experiment fully or partially. WRF4G is an open-source and public 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 check-pointing/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 in 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 in 50 chunks, a total of 5,000 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

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

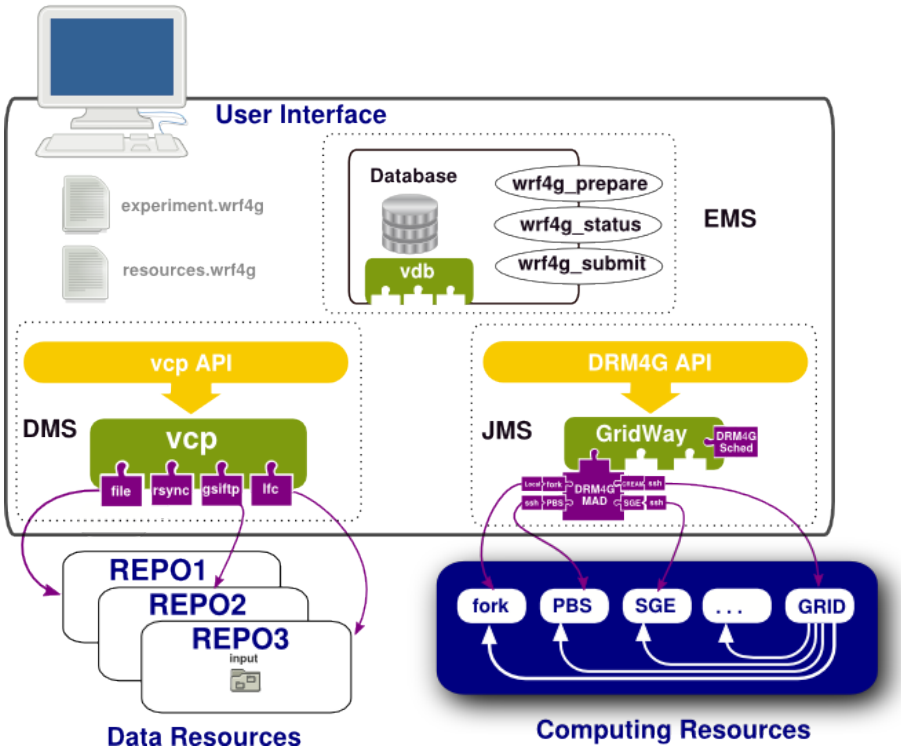

Figure 1: Schema of the WRF4G components: Experiment Management Service (WRF4G commands, database and vdb), Data Management Service (vcp) and Job Management Service (DRM4G).

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, ...) were 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[3].

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 Figure 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 transfered 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] `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

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) [22]. GridWay's MAD architecture is flexible enough to enable the interoperability among the components of an HDCI. For instance, it has been tested on different Grid [48, 6] and Cloud [46, 47] infrastructures.

Although GridWay's approach eases the management of HDCIs, there are no plugins to access Cluster middlewares. DRM4G [9] is at the core of WRF4G and bridges the gap between GridWay and Cluster resources. It also provides an API(Figure 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 following strong scalability requirements, which are an essential feature for running large-scale experiments, as those requested by some climate modeling experiments (Menéndez et al. [32] show an example with more than 7,000 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. [32].

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 GB to 14 GB.

The reference execution in our local cluster (cluster1, see Table 2) took ∼60 hours using 18 nodes (144 cores, Intel Xeon E5620). The execution time for each realization was around 175 minutes 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.

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 (cluster1) and it matches exactly the configuration of Menéndez et al. [32] 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 (cluster1), 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.

Name	**Type**	**Nodes**	**Cores**	**Comm.**	**RM**	**OS**	**Repo.**
cluster1	Cluster	18	144	Local	PBS	Centos 6	Local
cluster2	Cluster	8	64	Local	PBS	Centos 6	Local
cluster3	Cluster	4	32	SSH	SGE	Debian 7	rsync
cloud	Cloud	4	32	SSH	PBS	Centos 7	rsync
EGI	Grid	14246	136352	Local	CREAM	SL 6	gsiftp/lfn

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.

The computing resources *cluster1* and *cluster2* 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[4] from EGI, which provides access to 136352 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. *cluster3* is the cluster of other research group from our university, included in the HDCI for the purpose of illutrating 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, *cluster1* and *cluster2* have the repository directly mounted in the nodes (local) and *cluster3* 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 hours.

As the output data were transfered as they were being produced, and the total output size was much smaller than the input, in all experiments the output data were stored in-house in a common location (data.unican.es).

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

[4]http://www.eueearthsciencegrid.org/

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.

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 Figure 2a 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 Figure 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 Figure 2b, Experiment 3.2 used less 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.

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

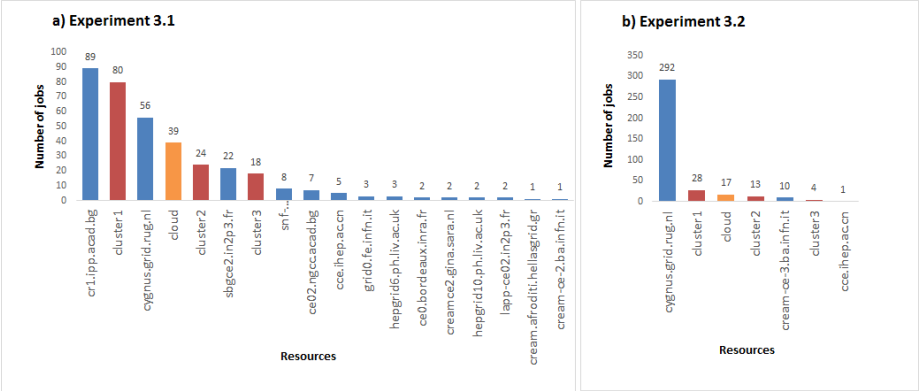

Figure 2: Distribution of the jobs among the different computing resources. a) first trial on the HDCI (Experiment 3.1) b) second trial, after the scheduling policy was updated (Experiment 3.2).

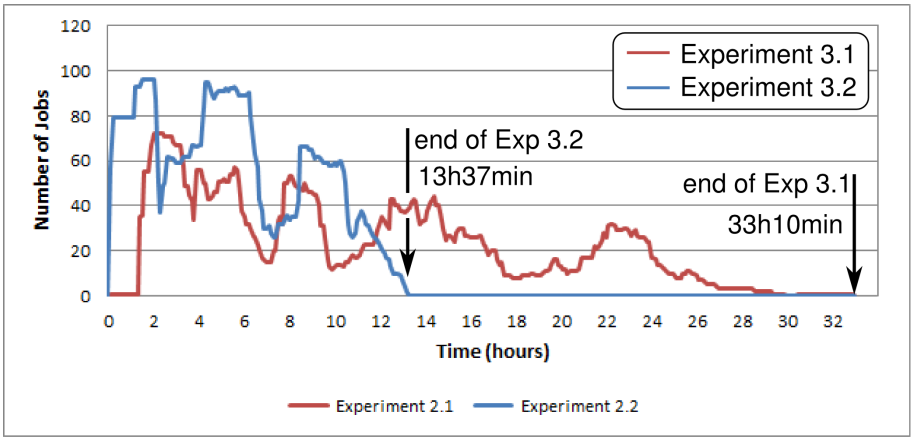

Figure 3: Number of jobs running vs. time (in hours) since the experiment submission.

Resource	Jobs	Wait	Download	WRF
cygnus.grid.rug.nl	292	262 (212)	4.8 (0.8)	119 (6)
cluster1	24	290 (100)	0.8 (0.1)	169 (17)
cloud	17	239 (164)	1.3 (0.2)	79 (1)
cluster2	11	231 (102)	0.6 (0.1)	171 (24)
cream-ce-3.ba.infn.it	7	123 (81)	4.0 (0.3)	111 (22)
cluster3	3	461 (1)	1.2 (0.1)	103 (1)
cce.ihep.ac.cn	1	10	12.2	194

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

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 [34, 18, 32].

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. [43] show how the algorithm they propose can be easily integrated in the GridWay scheduler policies.

Acknowledgements

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] Almond, J., Snelling, D., 1999. UNICORE: uniform access to supercomputing as an element of electronic commerce. Future Generation Computer Systems 15 (5–6) 539–548.

[2] Amjad, T., Sher, M., Daud, A., 2012. A survey of dynamic replication strategies for improving data availability in data grids. Future Generation Computer Systems 28 (2) 337–349.

[3] Baud, J.-P., Casey, J., Lemaitre, S., Nicholson, C., 2005. Performance analysis of a file catalog for the LHC computing grid. In: 14th IEEE International Symposium on High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 91–99.

[4] Blanco, C., Cofiño, A., Fernandez-Quiruelas, V., 2013. WRF4SG: A scientific gateway for the Weather Research and Forecasting model. In: 2013 36th International Convention on Information Communication Technology Electronics Microelectronics (MIPRO). 172–176.

[5] Bretherton, D. A., Blower, J. D., Haines, K., Smith, G. C., 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.

[6] Carrión, I. M., Huedo, E., Llorente, I. M., 2012. Interoperating grid infrastructures with the GridWay metascheduler. Concurrency and Computation: Practice and Experience .

[7] Casajus, A., Graciani, R., Paterson, S., Tsaregorodtsev, A., Team, t. L. D., 2010. DIRAC pilot framework and the DIRAC Workload Management System. Journal of Physics: Conference Series 219 (6) 062049.

[8] Chervenak, A., Schuler, R., Ripeanu, M., Amer, M., Bharathi, S., Foster, I., Iamnitchi, A., Kesselman, C., 2009. The Globus Replica Location Service: Design and Experience. IEEE Transactions on Parallel and Distributed Systems 20 (9) 1260–1272.

[9] Cofiño, A., Blanco, C., Fernández-Quiruelas, V., 2011. Aggregation of Grid and HPC resources for running huge experiments in climate and weather prediction. In: EGU General Assembly 2011, volume 13. 13194.

[10] Corradi, A., Fanelli, M., Foschini, L., 2014. VM consolidation: A real case based on OpenStack Cloud. Future Generation Computer Systems 32 118–127.

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

[12] Ellert, M., Grønager, M., Konstantinov, A., Kónya, B., Lindemann, J., Livenson, I., Nielsen, J. L., Niinimäki, M., Smirnova, O., Wäänänen, A., 2007. Advanced resource connector middleware for lightweight computational grids. Future Generation Computer Systems 23 (2) 219–240.

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

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

[15] Foster, I., Kesselman, C., 1996. Globus: A metacomputing infrastructure toolkit. International Journal of Supercomputer Applications 11 115–128.

[16] Fox, A., 2011. Cloud Computing—What's in it for me as a scientist? Science 331 (6016) 406–407.

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

[18] García-Díez, M., Fernández, J., Fita, L., Yagüe, C., 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.

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

[20] Henderson, R. L., 1995. Job scheduling under the portable batch system. In: Feitelson, D. G., Rudolph, L., (Eds.) Job Scheduling Strategies for Parallel Processing, number 949 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 279–294.

[21] Huedo, E., Montero, R. S., Llorente, I. M., 2001. The GridWay framework for adaptive scheduling and execution on grids. Scalable Computing: Practice and Experience 6 (3).

[22] Huedo, E., Montero, R. S., Llorente, I. M., 2007. A modular meta-scheduling architecture for interfacing with pre-WS and WS grid resource management services. Future Generation Computer Systems 23 (2) 252–261.

[23] Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B. P., Maechling, P., 2009. Scientific workflow applications on amazon EC2. In: 2009 5th IEEE International Conference on E-Science Workshops. IEEE, 59–66.

[24] Kannan, S., Roberts, M., Mayes, P., Brelsford, D., Skovira, J., 2001. Workload management with loadleveler. IBM Redbooks 2 2.

[25] Kosar, T., Balman, M., 2009. A new paradigm: Data-aware scheduling in grid computing. Future Generation Computer Systems 25 (4) 406–413.

[26] Kozlovszky, M., Karóczkai, K., Márton, I., Kacsuk, P., Gottdank, T., 2014. DCI bridge: Executing WS-PGRADE workflows in distributed computing infrastructures. In: Kacsuk, P., (Ed.) Science Gateways for Distributed Computing Infrastructures. Springer International Publishing, 51–67.

[27] 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.

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

[29] Litzkow, M., Livny, M., Mutka, M., 1988. Condor-a hunter of idle workstations. In: , 8th International Conference on Distributed Computing Systems, 1988. 104–111.

[30] Mansouri, N., Dastghaibyfard, G. H., Mansouri, E., 2013. Combination of data replication and scheduling algorithm for improving data availability in data grids. Journal of Network and Computer Applications 36 (2) 711–722.

[31] Mateescu, G., Gentzsch, W., Ribbens, C. J., 2011. Hybrid Computing—Where HPC meets grid and cloud computing. Future Generation Computer Systems 27 (5) 440–453.

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

[33] Montella, R., Foster, I., 2010. Using hybrid Grid/Cloud computing technologies for environmental data elastic storage, processing, and provisioning. In: Furht, B., Escalante, A., (Eds.) Handbook of Cloud Computing. Springer US, Boston, MA, 595–618.

[34] Nikulin, G., Jones, C., Giorgi, F., Asrar, G., Büchner, M., Cerezo-Mota, R., Christensen, O. B., Déqué, M., Fernandez, J., Hänsler, A., van Meijgaard, E., Samuelsson, P., Sylla, M. B., Sushama, L., 2012. Precipitation climatology in an ensemble of CORDEX-africa regional climate simulations. Journal of Climate 25 (18) 6057–6078.

[35] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D., 2009. Eucalyptus: an open-source cloud computing infrastructure. Journal of Physics: Conference Series 180 (1) 012051.

[36] 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.

[37] Redler, R., Budich, R., Ford, R., Riley, G., 2012. Earth System Modelling - Volume 5: Tools for Configuring, Building and Running Models. Springer, 1 edition.

[38] Saadat, N., Rahmani, A. M., 2012. PDDRA: a new pre-fetching based dynamic data replication algorithm in data grids. Future Generation Computer Systems 28 (4) 666–681.

[39] Skamarock, W., Klemp, J., Dudhia, J., Gill, D., Barker, D., Duda, M., Wang, W., Powers, J., 2008. A description of the advanced research wrf version 3. Technical report, NCAR.

[40] Sotomayor, B., Montero, R. S., Llorente, I., Foster, I., 2009. Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing 13 (5) 14–22.

[41] Strohmaier, E., Dongarra, J. J., Meuer, H. W., Simon, H. D., 2005. Recent trends in the marketplace of high performance computing. Parallel Computing 31 (3–4) 261–273.

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

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

[44] Tran Van Lang, N. T. D., 2012. Deploying business virtual appliances on open source cloud computing. International Journal of Computer Science and Telecommunications, ISSN: 2047-3338 3 (4) pp.26–30.

[45] Tsaregorodtsev, A., Bargiotti, M., Brook, N., Ramo, A. C., Castellani, G., Charpentier, P., Cioffi, C., Closier, J., Diaz, R. G., Kuznetsov, G., Li, Y. Y., Nandakumar, R., Paterson, S., Santinelli, R., Smith, A. C., Miguelez, M. S., Jimenez, S. G., 2008. DIRAC: a community grid solution. Journal of Physics: Conference Series 119 (6) 062048.

[46] Vazquez, C., Huedo, E., Montero, R. S., Llorente, I. M., 2009. Dynamic provision of computing resources from grid infrastructures and cloud providers. In: Grid and Pervasive Computing Conference, 2009. GPC '09. Workshops at the. IEEE, 113–120.

[47] Vázquez, C., Huedo, E., Montero, R. S., Llorente, I. M., 2011. On the use of clouds for grid resource provisioning. Future Generation Computer Systems 27 (5) 600–605.

[48] Vázquez-Poletti, J., Huedo, E., Montero, R., Llorente, I., 2006. Coordinated harnessing of the IRISGrid and EGEE testbeds with GridWay. Journal of Parallel and Distributed Computing 66 (5) 763–771.

[49] Vázquez-Poletti, J., Huedo, E., Montero, R., Llorente, I., 2007. A comparison between two grid scheduling philosophies: EGEE WMS and Grid Way. Multiagent and Grid Systems 3 (4) 429–439.

[50] Yoo, A. B., Jette, M. A., Grondona, M., 2003. SLURM: Simple linux utility for resource management. In: Feitelson, D., Rudolph, L., Schwiegelshohn, U., (Eds.) Job Scheduling Strategies for Parallel Processing, number 2862 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 44–60.

[51] Zhou, S., 1992. LSF: Load sharing in large-scale heterogeneous distributed systems. In: Proceedings of the Workshop on Cluster Computing.