A Grid Computing Based Virtual Laboratory for Environmental Simulations

I. Ascione[1], G. Giunta[1], P. Mariani[2], R. Montella[1], and A. Riccio[1]

[1] Dept. of Applied Sciences at University of Naples "Parthenope" - Italy
[2] Danish Inst. for Fisheries Research,
Dept. of Marine Ecology and Aquaculture - Denmark

Abstract. The grid computing technology permits the coordinate, efficient and effective use of (geographically spread) computational and storage resources with the aim to achieve high performance throughputs for intensive CPU load applications.

In this paper we describe the development of a virtual laboratory for environmental applications. The software infrastructure, and the related interface, are developed for the straightforward use of shared and distributed observations, software, computing and storage resources. The user can design and execute his experiments building up and assembling data acquisition procedures, numerical models, and applications for the rendering of output data, with limited knowledge of grid computing, thereby focusing his attention to the application.

Our solution aims at the goal of developing black-box grid applications for earth observation, marine and environmental sciences.

1 Introduction

Numerical modeling plays a main role in the earth sciences, filling in the gap between experimental and theoretical approach. Now, the computational approach is widely recognized as the complement to the today scientific analysis. Meanwhile, the huge amount of observed/modeled data, and the need to store, process and refine them, often makes the use of high performance parallel computing the only effective solution to ensure the real usability of numerical applications, as in the case of the atmospheric/oceanography field, where the development of the Earth Simulator supercomputer is just the edge [1].

The grid computing is a key technology in the field of the computational sciences, allowing the use of inhomogeneous and geographically-spread computational resources, shared across a virtual laboratory. Moreover, this technology offers several invaluable tools, ensuring the security, the performance and the availability of applications [2].

A great amount of simulation models have been successfully developed in the past, but a lot of them are poorly engineered and built following a monolithic programming approach, unsuitable for a distributed computing environment. The use of the grid computing technologies is limited to domain specialists, because of the complexity of grid itself and of its middleware complexity. Another

W.E. Nagel et al. (Eds.): Euro-Par 2006, LNCS 4128, pp. 1085–1094, 2006.

source of complexity resides on the use of coupled models, as, for example, in the case of atmosphere/sea-wave/ocean dynamics. The grid enabling approach could be hampered by the grid software and hardware infrastructure complexity. In this context, the buildup of a grid-aware virtual laboratory for environmental applications is a "grand challenge" for computer scientists.

In this paper we describe the implementation and application of a grid-enable virtual laboratory for environmental simulations. This application is built on the componentization of different environmental models: atmospheric circulation, air quality and ocean related models. The grid-enabling approach is described in the next section, while in section 3, we give an example of a grid application providing on-demand or operational weather and sea forecasts.

2 The Grid-Enabling Approach

For our grid infrastructure development, we use the middleware Globus Toolkit [3] version 4.x (GT4), developed within the Globus Alliance and the Global Grid Forum (GGF) with a wide support of institutions belonging to the academia, the government and the business area. The GT4 has been chosen because it exposes its features via web services using common W3C standards as the Web Service Description Language (WSDL), the Standard Object Access Protocol (SOAP), and the Hyper Text Transfer Protocol (HTTP). Complex features, as the service persistence, the state and stateless behavior, the event notification, the data element management and the index services tools are implemented in the respect of this standards. The GT4 also offers support to pre-web services features as the GridFTP protocol, an FTP enhanced version, capable of massive parallel striping and reliable file transfer.

In our grid virtual laboratory we coupled several environmental models: the MM5 (Mesoscale Model 5) [4], the STdEM (Spatio-temporal distribution Emission Model) [5], and the PNAM (Parallel Naples Airshed Model) air quality model [6]. Our grid enabling approach also integrates marine-related environmental models, such as the POM (Princeton Ocean Model) [7], and the WW3 (WaveWatch III) sea-wave propagation model [8]. We enhanced the computational capabilities of the POM model, developing a parallel version with nesting capabilities (POMpn) [9]. Moreover, we recently integrated the WRF (Weather and Research Forecasting model) [10] and the CAMx (Comprehensive Air quality Model with eXtensions) air quality models [11], while we are working on the integration of the sea-wave propagation model SWAN [12].

The grid-enabled version of each model is based on three files: the model package, the launching script and the RSL job description file.

We configured and packaged each model, in order to be independent on the software and hardware configuration of the local machine. A framework approach was used to abstract different model configuration, by exploiting an object oriented programming-like methodology. We standardized the model packing/unpacking, configuring and setup defining which methods, implemented as shell scripts or Java class code, have to be called to perform operations as

namelist placeholder, processor configuration and packing and unpacking of input/output data. Each model runs in a private custom environment, so that several instances of the same software from the same or different user can be concurrently executed. This approach is based on a repository where model packages are stored and from where they can be retrieved as instance template at each run.

A job launcher, invoked as a grid job on a remote machine set up the virtual private environment doing all needed data file stage in and stage out operations. In this way all implementation details are hidden and a Resource Specification Language (RSL) file can be used to describe each grid operation to the globusrun-ws job submitter. The launching script is deployed with a stage in file transfer operation on the target machine and represents the job executable file to be run on the remote machine. This scripts unpack the model package eventually downloaded from a repository or copied from a local directory, unpack and inflate the input data, run the model and the pack and deflate results. The script communicates with the job submitter via the standard output and the standard error.

A RSL (Resource Specification Language) file [13] describes the job to be submitted in a very detailed way, specifying the executable path, the current working directory, the files to be staged in before the execution and staged out after the job run, and any additional argument. Job submission is managed through the Grid Resource Allocation Manager (GRAM) tool. All files are named using URL, with protocol details from the target machine point of view and specifying the gridFTP high performance parallel striping transfer protocol when referring to a remote machine.

In the RSL evolution from the Globus Toolkit version 3 to version 4 some operations were simplified, making the RSL less verbose and more expressive; on the other hand, some features, as the automatic management of scratch directories, disappeared, so that we implemented a custom RSL pre-processor for the easy and straightforward definition of jobs, introducing an advanced method of labeling and placeholders parsing and evaluation, macro-based code explosion and late binding capabilities.

The Globus Toolkit 4.x grid middleware provides job submission tools via web-services and pre-webservices infrastructure without any kind of support for job flow scheduling and resource broking, while different grid technologies, such as the Condor [14] and Unicore [15] middleware offer a full support of direct acyclic graph job workflow with conditional branches, recovery features and graphical user interfaces. Our custom software solution was developed with the aim to provide domain scientists of a full configurable, really straightforward grid computing tool minimizing the impact of the grid infrastructure.

As in many grid applications, the final result is obtained by assembling different components executed as jobs on remote machines. Each component could be related to its previous/next component as data producer or data consumer, defining the so-called computational pipeline in which we have one job for one component. For example, consider this simple application: a regional-scale atmospheric

model waits until data can be downloaded from a specified service, then acquires the boundary and initial conditions from a global-scale forecast, runs for a specified time period; when data are ready to be processed, another job, encapsulating an ocean circulation model, uses the atmospheric data as boundary conditions; when this second job finishes, the produced data is consumed by another job simulating the wind-driven sea wave propagation and forecasts the wave height/period and direction fields. At last, the user retrieves all pipeline outputs produced by all models. This simple grid application can be implemented via shell scripts and RSL files specifying the target submitting machine in the script itself.

This approach, though operatively correct, presents many disadvantages. The user needs to know the details about the script programming language, the job submission technical details related to a specific middleware and the to system environment setup. The developed code is tightly coupled to its application: any change to the job behavior or model configuration affects the entire application. In case of complex job fluxes, like in a concurrent ramification context, for example when the weather simulation model forces both wave propagation and oceanic circulation models, control code grows in complexity and data consuming/production relationships could be hard to implement, since synchronization issues may arise. Moreover, this kind of approach is potentially insecure because the user must be logged-in to the system to run a script, and this scenario is not applicable in the case of an interactive application on web portal.

In order to enhance the flexibility and to minimize the impact on the grid configuration, we implemented a custom job flow scheduler (JFS). Using this tool the entire complex, multi branch, grid application could be configured through a XML file. The JFS takes care of submitting jobs to computing nodes. JFS integrates itself in the Globus Toolkit environment both as a web service and a command tool with very few configuration needs. It uses a customized version of the Job Description Language (JDL), developed under the Condor project. In this way, every job is described through its RSL file and built in a XML file, which describes the activation order and relationships between jobs. The description language implemented has been defined as Job Flow Definition Language (JFDL) with a suitable XML schema. The following JFDL file implements the coupled use of the MM5/WW3 models:

```
<jfdl:jfs project="experiment01">
  <!-- Job definition -->
  <jfdl:jobs>
    <jfdl:job name="downloadConditions"
         target="dgric.uniparthenope.it"
         rsl="downloadConditions.rsl"/>

    <jfdl:job name="runMM5"
         target="dgbeobi.uniparthenope.it"
         rsl="runMM5.rsl"/>
```

```
    <jfdl:job name="runWW3"
         target="dgbeobe.uniparthenope.it"
         rsl="runWW3.rsl"/>
   </jfdl:jobs>

   <!-- Job Relationship Definition -->
   <jfdl:nodes>
     <jfdl:node job="downloadConditions">
       <jfdl:next>runMM5</jfdl:next>
     </jfdl:node>
     <jfdl:node job="runMM5">
       <jfdl:prev>downloadConditions</jfdl:prev>
       <jfdl:next>runWW3</jfdl:next>
     </jfdl:node>
     <jfdl:node job="runWW3">
       <jfdl:prev>runMM5</jfdl:prev>
     </jfdl:node>
   </jfdl:nodes>
</jfdl:jfs>
```

The file describing the experiment could be divided in two parts: inside the element ⟨`jfdl:jobs`⟩ each job belonging to the grid application is described specifying its symbolic name, the computing node where it will be submitted and the name of the RSL file specifying all needed resources. Inside the element ⟨`jfdl:nodes`⟩ the jobs activation order is described using a direct acyclic graph. In this section, each job node is characterized by the reference to all previous jobs, the ⟨`jfdl:prev`⟩ element, by the way the jobs that have to be finished before the start of the current job, and by a reference to all next jobs which will be submitted after the current job finishes using the ⟨`jfdl:next`⟩ element.

The described experiment is a typical example of a simple virtual laboratory grid application, but our JFS could submit very complex application graphs, thanks to its Java multithread implementation (Fig. 1).

The Jobflow Scheduler was implemented using the Java language using a class framework encapsulating all described features including XML file parsing based on the StaX [16] package, graph setup and application runtime support. The most interesting class is Job derived from Thread, implementing the job submission in its run method using a clear, effective and efficient algorithm: if the job is to be started, make a join to each thread-related jobs using the previously defined dependence graph. In this way the thread waits until all prerequisite data are successfully produced. Then the job is submitted to the grid using the globusrun-ws service specifying the target factory and the job RSL file. The class Job is an item of the collection Jobs composing the JobFlow class, providing methods for graph setup, management and run. The Jobs run method starts all jobs belonging to the collection with no previous job dependence. For example, more data providers have to download initialization data to feed a consuming job.

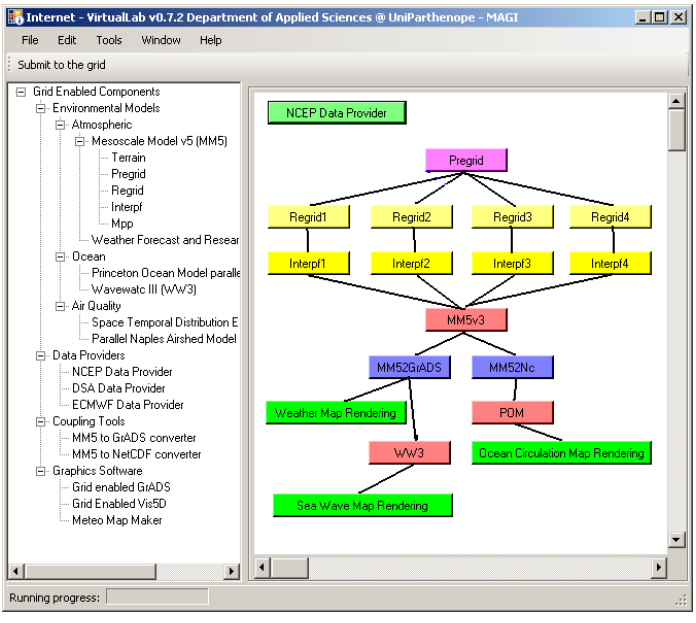

Fig. 1. A GUI for interactive JFDL files editing with direct grid interfacing capabilities via MyProxy

The described grid application is classifiable as a grid-enabled application because it uses the grid to submit a job to the best computing node, but this association is statically performed at the design time. On the other hand a grid-aware application could be adapted in relation to the grid status using a Resource Broker [17] component, designed to submit a job to the best fitting node, based on needed computational and storage requirements. Our JFS automatically activates this feature if no target machine is specified in the ⟨`jfdl:job`⟩ element. The Resource Broker algorithm is straightforwardly configurable, changing the behavior of the implementation class in a properties file.

3 Laboratory Components

The Jobflow scheduler and the Resource Broker implement the core of the grid based virtual laboratory. The domain scientist can configure and run his experiments using the JFDL and RSL files, or through a web portal, or an under development Java user interface, selecting and assembling each component from a palette.

Actually our virtual laboratory provides several grid components for data acquisition: the NCEPDataProvider performs the data download from the NOAA-NCEP [18] for the initialization of the meteorological model, thanks to a daemon component, completely decoupled from the grid; the ECMWFDataProvider [19]

performs the on-demand download of historical data for scenario and "what if " analysis; the DSADataProvider performs the on-demand download of processed data.

Numerical models are grouped in atmospheric circulation models, such as the gWRF and the gMM5 suites, whose components (gTERRAIN, gPREGRID, gREGRID, gINTERPF and gMPP) have been ported to our grid environment; air quality related models as gSTdEM, gPNAM, gCAMx and ocean related models as the gPOMpn, gWW3, gSWAN. We provided our virtual laboratory with a suite of tools for model coupling, data conversion, classification and graphics rendering software. Thanks to our packaged framework for grid enabling legacy software components, adding more grid components is straightforward.

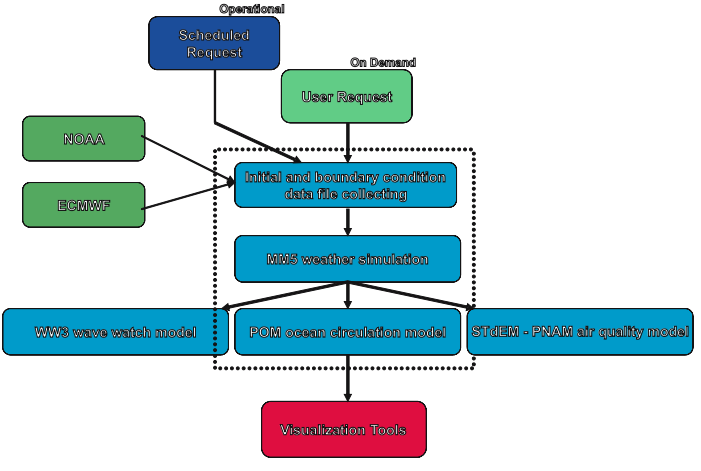

Fig. 2. The grid application building blocks

We used the Jobflow Scheduler (JS) and the Resource Broker (RB) to develop a grid application aiming at producing weather and marine forecasts in both operational and on demand mode, by coupling several simulation models, data acquisition, conversion, and visualization software (Fig. 2). The application workflow is easy to understand: the starting event is produced by the on demand user request, or by the availability of initial data in the case of an operational production environment. Then, the weather forecast model is initialized, and the output data is rendered by a presentation software and concurrently consumed by other models, as ocean dynamics, sea wave propagation or air quality models. Each application branch proceeds on separate thread. This workflow could be represented by an acyclic direct graph into a JFDL file, while each job to be submitted is described by the RSL file and its launching script. Our JS permits the implementation using a single XML self describing file, while the RB makes grid-aware the application with any kind of constrain and without the need to use a storage element as intermediate files repository because of our late binding reference approach. This application run in operational mode with a few

maintenance operations, except components or grid middleware upgrades. All performed results are interactively published at the Department web portal and used by several scientists, local institutions and citizens [20] (Fig. 3).

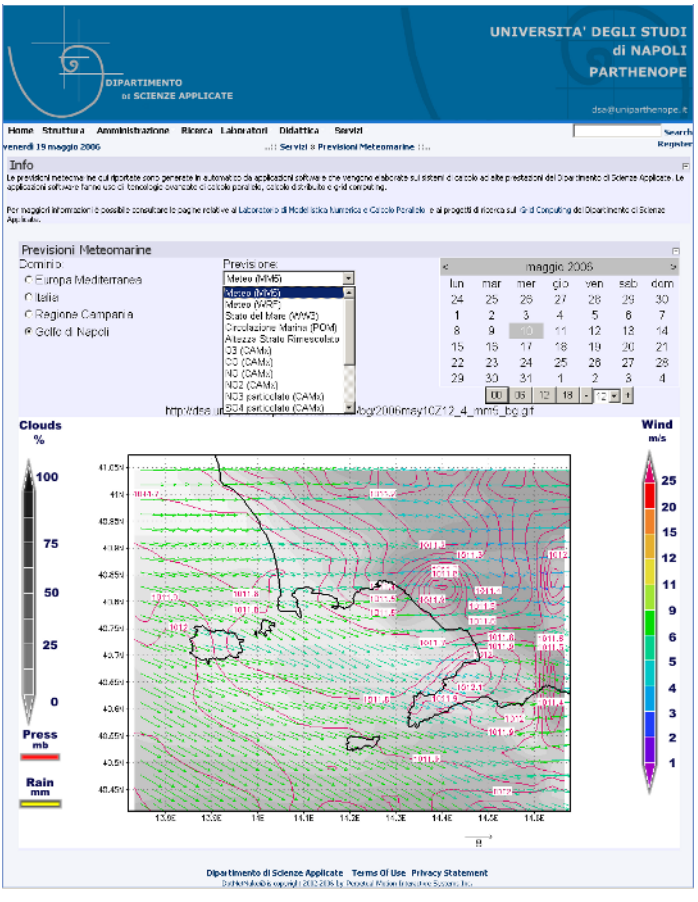

Fig. 3. Weather forecast grid application in operational mode: an output example

4 Conclusions and Future Development

In this paper we described some of our results in the field of grid computing research. The virtual laboratory for earth observation and computational environmental sciences based on the grid computing technology is a tool used both for research and application-oriented uses, running a complex grid application dedicated to operational weather, marine and air quality forecasts on nested domains from the Mediterranean Europe to the Bay of Naples area. Comparison tests between a grid and non-grid implementation, performed using a simple benchmark

weather forecast application, affected by networking capabilities, demonstrates that with the number of simulated hours increasing from 72 to 144 the efficiency of the grid implementation rise with clear evidence.

The JS and the RB realized the primary goal of our research providing the power of the computing grids and the high performance computing with the simplicity and the flexibility of a local XML configurable application demonstrating the grid technology features.

The Globus Tooklit middeware version 4 is stable enough to perform production activities in the range of our needs, but some points have to be improved. The JS engine works very well, but it have to be enhanced offering more expression power to the JFDL especially regarding conditional branches and resume features. The RB algorithm have to be well tested and improved.

References

1. Lin, S., Atlas, R., Yeh K.: Global weather prediction and high-end computing at Nasa. Computing in Science & Engineering, **6** (2004)
2. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, **15** (2001) – 200-222
3. The Globus Toolkit, The Globus Alliance, http://www.globus.org
4. Michalakes, J., Canfield, T., Nanjundiah, R., Hammond S., Grell, G.: Parallel Implementation, Validation, and Performance of MM5. Parallel Supercomputing in Atmospheric Science. World Scientific, River Edge, NJ 07661 (1994)
5. Barone, G., D'Ambra, P., di Serafino, D., Giunta, G., Montella, R., Murli, A., Riccio, A.: An Operational Mesoscale Air Quality Model for the Campania Region. Annali Istituto Universitario Navale (2000) – 179-189
6. Barone, G., D'Ambra, P., di Serafino, D., Giunta, G., Murli, A., Riccio, A.: Parallel software for air quality simulation in Naples area. J. Environ. Manag. & Health (2000) 209–215
7. Blumberg, A.F., Mellor, G. L.: A description of a three-dimensional coastal ocean circulation model. Three-Dimensional Coastal ocean Models, edited by N. Heaps, American Geophysical Union. (1987)
8. Tolman. H.L.: A third-generation model for wind waves on slowly varying, unsteady and inhomogeneous depths and currents. J. Phys. Oceanogr. , **21** (1991) 782–797
9. Giunta, G., Montella, R., Mariani P., Riccio, A.: pPOM: A nested, scalable, parallel and Fortran 90 implementation of the Princeton Ocean Model, accepted by Environmental Modelling & Software
10. Michalakes, J., Dudhia, J., Gill, D., Henderson, T., Klemp, J., Skamarock, W., Wang, W.: The Weather Research and Forecast Model: Software Architecture and Performance. 11th ECMWF Workshop on the Use of High Performance Computing in Meteorology. 25–29 October 2004, Reading, U.K.
11. CAMx Comprehensive Air Quality Model with eXtensions. Version 4.20. ENVIRON International Corporation (2005)
12. Booij, N., Holthuijsen, L.H., Ris, R.C.: The SWAN wave model for shallow water. Int. Conf. on Coastal Engineering., Orlando, USA (1996) 668–676
13. Resource Specification Language (RSL). http://globus.org/toolkit/docs/4.0/execution/wsgram/

14. Thain, D., Tannenbaum, T., Livny, M.: Distributed Computing in Practice: The Condor Experience. Concurrency and Computation: Practice and Experience, **17** (2005) 323–356
15. Streit, A., Erwin, D., Lippert, T., Mallmann, D., Menday, R., Rambadt, M., Riedel, M., Romberg, M., Schuller, B., Wieder, P.: UNICORE – from Project Results to Production Grids
16. Stream API for XML, http://dev2dev.bea.com/xml/stax.html
17. Mavilio, C., Montella, R.: A resource broking algorithm for grid computing applications, Technical Report 2005/11, Dept. of Applied Sciences – University of Naples "Parthenope"
18. National Centre for Environmental Prediction. http://www.ncep.noaa.gov
19. European Centre for Medium-Range Weather Forecasts. http://www.ecmwf.int
20. Giunta, G., Montella, R., Mariani P., Riccio, A.: Modeling and computational issues for air/water quality problems: A grid computing approach. Il Nuovo Cimento, **28** 2005