# Biomedical and Civil Engineering Experiences Using Grid Computing Technologies

J.M. Alonso, V. Hernández, G. Moltó

published in

*Parallel Computing:*
*Current & Future Issues of High-End Computing*,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 647-654, 2006.

# Biomedical and Civil Engineering Experiences Using Grid Computing Technologies

J.M. Alonso[a], V. Hernandez[a], G. Molto[a]

[a]Departamento de Sistemas Informaticos y Computacion, Universidad Politecnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain

## 1. Introduction

In the mid 90s, the Grid concept appeared as a new trend in distributed computing. Grid Computing technology [6] emerged as the solution for some of the computational problems of organisations, enabling the collaborative use of remote resources to execute computationally expensive tasks across the world. The main aim of a computational Grid is to provide a coherent view of distant heterogeneous machines so that they act as a single, huge, powerful and self-managed computer. This fact has caused a tremendous impact for computationally intensive applications, which can bene t from the power that a Grid infrastructure aims at delivering.

Nowadays, important research efforts are being dedicated to the development of the software infrastructures that enable to achieve this vision. However, Grid Computing is still in the early stages and the users face a lot of complexity as well as a steep learning curve when applying this technology to their own applications.

In this paper we describe the usage of Grid Computing technologies in two applications belonging to different scienti c elds. The rst one involves the 3D dynamic analysis of large-scale buildings. The second one performs electrical simulations of cardiac tissues. In both applications, an iterative simulation procedure gives place to a computationally and memory-intensive process. Two previously implemented applications have been ported to a Grid infrastructure by developing and using later, a generic software layer called GMarte, which simpli es the process of executing a scienti c software on a Grid-based distributed infrastructure.

To test the bene ts of Grid Computing in the applications considered, executions have been performed both on a local and a large-scale Grid infrastructure. On the one hand, the local infrastructure represents a Globus-based Grid composed of machines from our research group. On the other hand, we have also employed part of the resources available in the framework of the EGEE project, the largest distributed deployment available for e-science.

The remainder of the paper is structured as follows: Section 2 presents the two target applications considered. Then, section 3 details the GMarte software layer employed to develop the Grid applications that enable the target applications to be executed on a Grid. Next, section 4 describes the structural case study that has been executed, detailing the computational infrastructure employed and the results. Later, section 5 reports the execution of a cardiac case study on the large Grid deployment. A discussion of the principal problems that arise when moving from a local to a global Grid is performed in section 6. Finally, section 7 summarises the main achievements and concludes the paper.

## 2. Target Applications

This chapter brie y describes the two applications chosen for their execution on a Grid. They have been selected because of their large computational and memory requirements.

### 2.1. Structural Dynamic Analysis of Buildings

3D dynamic analysis of large-scale buildings has been considered by engineers as a challenging problem, owing to its high computational demand. Most of the existing commercial codes are composed of questionable simpli cations, reducing the number nodes to be considered and the number of degrees of freedom associated, because the computational and memory requirements involved in a realistic simulation may be too intensive for a traditional computer. Notwithstanding, these simpli- cations, although appropriate for single structures, have demonstrated to be completely inadequate for complex buildings. Previous research has shown that results from seismic analysis are model dependent and it indicates the need for an adequate and realistic 3D analytical model [11].

In the preliminary stage of a building, structural engineers usually work with different initial designs. Each design can be a different layout, composed of distinct materials (concrete, steel, etc.), where different dimensions are assigned to the members, or even where several external loads, which could occur during the lifetime of the structure, are applied. The large number of resulting structural solutions must be analysed rapidly.

In some cases, all these designs are rejected, thus returning to the initial design stage. Nevertheless, a selection among these solutions has to be made before proceeding to the detailed design phase. Now, an iterative trial-error process takes places by the structural engineer to achieve the most ef cient solution where, varying the member dimensions, the whole structure is analysed and the results are interpreted. Obviously, calculations must be performed accurately, complying criteria of safety, cost limitations and construction constraints. Moreover, this number of simulations required is notably increased when dealing with dynamic analysis owing to the need to work with several dynamic loads. As an example, the Spanish Earthquake-Resistant Construction Standards (NCSE-02) requires that a building is analysed with at least ve different representative earthquakes. This gives place to a large amount of different structural alternatives to be computed (a structural case study), where a huge computational power is demanded.

In a previous work, an application was developed to perform 3D dynamic analysis of buildings, where all the nodes of the structure are considered and 6 degrees of freedom per node are taken into account [3]. Simulations are carried out by means of 8 well-known direct time integration methods [8].

### 2.2. Simulation of the Electrical Activity in Cardiac Tissues

According to the European Heart Network[1], cardiovascular disease causes nearly half of all deaths in Europe. In fact, cardiac arrhythmias are one of the rst causes of mortality in developed countries. However, despite the intense research in this area, the generation of ventricular tachycardias and ventricular brillation (the most mortal arrhythmias) are not clearly understood.

Simulation is an excellent tool that enables to formulate multiple hypotheses *in silico* prior to clinical experimentation. However, the simulation of the electrical propagation in cardiac tissues (a key indicator of the state of the heart) represents a computational and data intensive problem, where the electrical study of a medium-sized tissue can last for several days on a sequential platform. This is mainly due to the usage of comprehensive ionic models which detail the internal behaviour of each cardiac cell being simulated.

A MPI-based parallel simulator was developed for the simulation of the action potential propagation in cardiac tissues. It enabled a substantial reduction in the execution time of a single simulation on a cluster of PCs [1]. However, in addition to this computational cost, there are many research studies that involve the execution of parametric simulations. For example, to test the effects of new

[1]http://www.ehnheart.org

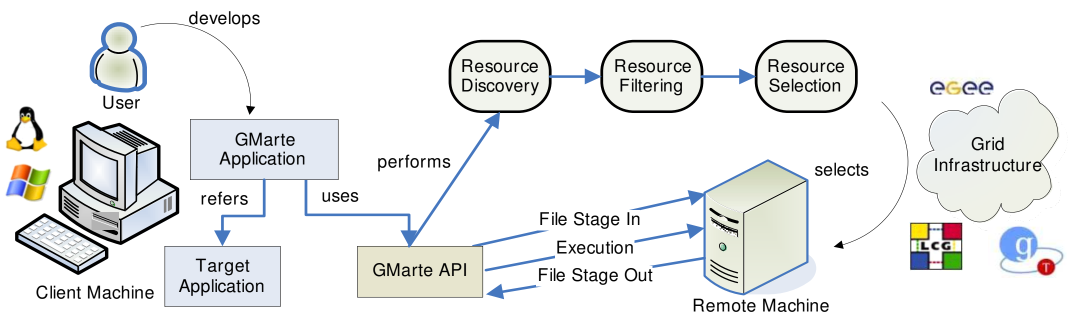

Figure 1. Principal diagram of usage of GMarte.

medicines, it is required to perform multiple simulations varying the drug concentration. As another example, to study the effects of late ischemia it is necessary to vary the junctional resistances in a given interval and investigate the evolution of the electrical activity.

These cardiac case studies, composed of multiple independent parametric executions, enlarge the total computing cost by the number of simulations to be run, what turns the global study procedure into an even larger computational problem.

## 3. The GMarte Framework

These two different scienti c elds face common problems. They involve the execution of multiple independent parametric simulations. Given that a Grid infrastructure aims at offering a large computing power, a Grid-based solution, that enables to perform multiple concurrent simulations on these machines, would be an ideal solution to accelerate the execution of these problems, which will be denoted as *case studies*.

To support our executions, we have focused on the Globus Toolkit 2.4.3 [7], the current *de facto* standard middleware for computational Grids. This middleware provides protocols and services for connecting resources and deploying large-scale computational global Grids. However, its inherent complexity often discourages scientists from porting their applications to a Grid infrastructure. To overcome this handicap, GMarte (Grid Middleware to Abstract Remote Task Execution) [5] was developed. It consists of an easy-to-use middleware, developed on top of the Java CoG 1.2 [10], exposing an Object-Oriented view of the Grid that enables fault-tolerant metascheduling in a Globus-based framework. GMarte has also been extended to support executions on the Computing Elements of the LCG testbed, which is the largest Grid deployment available for scienti c computation. A more detailed description of this infrastructure will be provided later.

Using the high level API provided by GMarte, the user no longer interacts with the low-level services provided by Globus. Therefore, instead of detailing *how* to perform the executions, the user declares *what* to execute by using the high-level object abstractions offered by the Java API provided (i.e. GridTask, GridResource, etc). An XML approach for the de nition of the case studies is currently under development to enable the case study de nition without depending on the Java API. It is important noting that GMarte is a client-side middleware, only installed on the client machine, which will interact with the computational resources of the Grid infrastructure. In addition, as it is developed in Java, the client is functional on different operating systems, such Windows and Linux. Being out of the scope of this paper, further information about GMarte can be found elsewhere [4].

Figure 1 describes the corresponding usage diagram for a user that wants to execute a particular case study on a Grid deployment. First of all, the user writes a small *GMarte Application* which describes the tasks to be executed (i.e. instances of the *Target Application*) along with their computing requirements in terms of processors, needed RAM, etc. This application also speci es the set of machines to be employed for execution, either by an explicit enumeration or by delegating into a resource discovery component. Finally, this application may also specify a certain metascheduler (the component that performs the task allocation) from those already implemented in GMarte. Once the case study simulation is launched, the GMarte middleware performs a sequence of steps in order to achieve successful execution of the tasks. First of all, if required, the *Resource Discovery* phase obtains a list of potential machines candidates for execution. This is obtained by querying standard Index Information Services for Globus or LCG. Then, *Resource Filtering* discards those resources that are unavailable for execution, that can not be accessed with the user credentials supplied or that do not ful ll the minimum application-dependent execution requirements. These two phases are only executed once in the scheduling procedure.

Then, for each task, the following phases are performed to achieve remote task execution: First of all, *Resource Selection* involves choosing the current best resource for the execution of the task. The implemented policy in GMarte considers the application requirements as well as an estimate of the data transfer cost to each resource. Besides, a workload component prevents from the allocation of all the tasks to a single resource, what would be critical if the resource fails. This policy can easily be altered by the user in order to guide the scheduling procedure. Once the machine has been selected, the *File Stage In* phase transfers the *Target Application* as well as all its dependent input les from the *Client Machine* to the *Remote Machine*. Next, the *Execution* phase starts the application in the remote machine. Finally, when the execution nishes, the *File Stage Out* phase retrieves the selected output data les back to the local machine, erasing the remote data les.

## 4. Structural Case Study

To assess the effectiveness of Grid Computing technologies in the 3D structural analysis of buildings, a hotel was simulated. These singular buildings must be carefully designed, because an appropriate solution can be the differential factor for economical pro tability in countries where the tourism is one of the biggest industry. The design of hotel facilities and security requirements imply to consider distinct relevant factors, leading to different structural solutions for the same problem.

In our case, the hotel geometric model was composed of about 100,000 degrees of freedom and two alternatives were considered for the construction material: reinforced concrete and steel-concrete composite frame. For each material, ve combinations of different structural member dimensions were assigned. Finally, and taking into account the NCSE-02, six representative earthquakes were applied to each structural solution. Therefore, a total of 60 dynamic simulations were executed, before selecting the most suitable structural solution. HHT-$\alpha$ was chosen as the direct time integration method to carry out the simulations. The behaviour of the different structural alternatives were analysed during 7 seconds. The accelerograms used presented values of ground acceleration at every 0.01 seconds and this time was chosen as the integration time step.

Each simulation demanded a total of 350 MBytes of RAM and generated 126 KBytes of output results. For each structural member, the output les indicated if its stresses and deformations obtained overcame the maximum values according to the construction standard employed, and therefore showing if the member dimension must be changed or not.

### 4.1. Computational Infrastructure and Execution Results

The Grid deployment that we used for the execution of this structural case study is composed of three clusters of PCs belonging to our research group. The Globus Toolkit 2.4.3 was installed in all

Table 1
Summary of the local Grid deployment and distribution of the simulations in the testbed.

| **Machine** | **Computational Nodes** | **Memory** | **Avail. Nodes** | **Simulations** |
|---|---|---|---|---|
| RAMSES | 10 Dual Pentium III 866 Mhz | 512 MB | 10 | 10 |
| KEFREN | 20 Dual Xeon 2.0 Ghz | 1 GB | 20 | 20 |
| ODIN | 55 Dual Xeon 2.8 Ghz | 2 GB | 52 | 30 |

the machines except Ramses, which runs the LCG-2 middleware and is part of the Biomed Virtual Organisation which will be explained later on.

Table 1 summarises the testbed (i.e., the set of computational resources) employed and the task allocation result. The table indicates, for each machine involved, its principal capabilities, the number of computing nodes that were available just before that scheduling procedure started, together with the number of simulations allocated to it.

The global execution time of the whole structural case study on this Grid was 33 minutes, since the scheduling procedure started until the output data of the last task was retrieved. Later analysis revealed that the executions at Ramses machine lasted 3 times longer than those executed at Odin. This caused the scheduling procedure to wait a few minutes for the executions at Ramses to  nish, what could have probably been shortened with a different task allocation policy that had assigned more simulations to the free nodes of Odin cluster. A typical execution model in an engineer studio would involve the sequential simulation of the case study on just one computer. This kind of execution, on one node of Odin cluster, required a total 495 minutes (8.25 hours), what represents a speedup of 15 when executing the 60 structural simulations.

## 5. Cardiac Case Study

Myocardial ischemia is a condition caused by oxygen deprivation to the heart that can result in an angina. During myocardial ischemia, the extracellular potassium concentration increases in a triphasic pattern, an initial early increase, a constant phase, and a late increasing stage [9]. Other cellular characteristics such as the intracellular concentration of ATP and ADP, and the sodium and the calcium currents that traverse the cell membrane, also vary during this pathology.

In order to study the effects of various degrees of ischemia, a cardiac case study composed of multiple parametric simulations was executed. It analysed the in uence, in the electrical propagation, of different degrees of ischemic conditions that take place during the  rst 10 minutes from the onset of a myocardial ischemia. This case study was originally executed on a restricted Grid deployment, composed of machines from our research group and from another university at Spain, with a naive Globus-based Grid prototype. A fully detailed description of it, which requires the execution of 21 parametric simulations, can be found in a related paper [2]. However, the simulation time has been reduced from 80 ms., in the original case study, to 20 ms, as previous executions revealed that the important information for this particular study appeared within the  rst 20 ms.

### 5.1. Computational Infrastructure

The execution of this case study has been carried out in a large deployment. The LCG[2] project aims at the development of the computing infrastructure for the simulation, processing and analysis of the data provided by the Large Hadron Collider (LHC), the most powerful particle accelerator

[2]LHC Computing Grid Project. http://lcg.web.cern.ch

Table 2
Initial state of the resources and distribution of the cardiac simulations in the testbed, for each machine. The number in parentheses indicates the number of nodes involved in the execution.

| Machine | Country | Nodes | Memory | Av. Nodes | Simulations |
|---|---|---|---|---|---|
| RAMSES | Spain | 10 Dual Xeon 866 Mhz | 512 MB | 8 | 2 (7 p.), 1 (2 p.) |
| CNAF-INFN | Italy | 8 Dual Xeon 2.4 Ghz | 512 MB | 7 | 7 (1 p.) |
| IN2P3-1 | France | 112 Dual Xeon 3 Ghz | 4 GB | 42 | 2 (1 p.) |
| IN2P3-2 | France | 58 Dual Pentium IV 1 Ghz | 512 MB | 28 | 8 (1 p.) |
| BA-INFN | Italy | 84 Dual Xeon 2.4 Ghz | 881 MB | 32 | 1 (1 p.) |

which is being constructed at CERN. The LCG-2 middleware, developed in the mentioned project, was the starting point of EGEE[3], a European Union funded project that aims at developing a service Grid infrastructure available to scientists 24 hours a day.

The LCG-2 middleware is based on the Globus Toolkit 2, providing additional enhancements and functionalities to support the execution of the experiments to be performed in the eld of physics. Within the framework of both the LCG and the EGEE projects, a large testbed composed of machines is available for execution. This testbed currently consists of more than 9500 CPUs, from 112 sites, across 31 countries, with a total storage capacity of 4 Petabytes, representing the largest computational deployments devoted to Grid Computing in science.

Although the LCG project was originally devoted to physics, the EGEE project fosters the migration of biomedical applications to the Grid. In fact, the *Biomed* Virtual Organisation (VO) has been created with those computational resources belonging to project partners that support biomedical research. It currently consists of more than 2500 processors, from 35 sites, across 13 countries.

## 5.2. Execution Results

Currently, the submission of parallelised applications with the MPI standard is not supported in the EGEE testbed. Therefore, even though GMarte allows the execution of parallel applications on multiprocessor or clusters of PCs, the simulations on machines belonging to the EGEE testbed will always be sequential, that is, with just one processor. However, the Ramses cluster, although belonging to the EGEE testbed, has been specially con gured to support MPI-based executions.

Table 2 summarises the task allocation process. The table shows, for each machine involved in the scheduling procedure, its corresponding country and its features, the number of computing nodes that were available just before the scheduling process started and the number of simulations allocated. The parentheses indicate the number of processors employed in each execution.

When the executions were performed, the resource discovery listed a set of potential remote machines. The resource ltering phase discarded some of them that were unavailable or stated authorisation problems with the user credentials supplied. Also, the machines that did not satisfy the minimum execution requirements were also discarded. This notably reduced the available computing resources. In addition, several hosts were never selected for execution, mainly due to the large data transfer cost estimated (for example, one machine located in Taiwan).

The total execution time of the case study was 102 minutes. Performing a sequential execution of the case study, one simulation after another on a Pentium IV machine, required a total of 1569 minutes. Using a cluster of such PCs, performing executions with 8 processors one after another, lasted

[3]Enabling Grids for E-sciencE. http://eu-egee.org

for a total 221 minutes. This represents a speedup of 15.38 and 7.1 respectively when executing the 21 simulations on the Grid infrastructure. In terms of global execution time, the Grid computing approach enabled to reduce the simulation time of the case study from more than one day to less than two hours. This increases the research productivity as more simulations per day can be achieved.

## 6. Discussion: From Local to Global Grid

As shown, the solution adopted to accelerate the execution of the case studies have been to perform multiple concurrent simulations on the machines of a Grid infrastructure. Under an ideal situation, we could assume that the Grid would offer enough computing resources for all the executions to proceed simultaneously. Moreover, the underlying middleware would have a negligible cost and data movements among the resources of the Grid would be instantaneous. With these assumptions, the global time of executing a case study would be reduced to the time required by the slowest task. However, as ideal conditions can not be applied to distributed infrastructures such as a Grid, we have to cope with these drawbacks.

When moving from a local Grid to a global Grid, many problems arise. First of all, the lower bandwidth and the increased latency of the communication networks make the data transfer cost a crucial parameter that needs to be considered when selecting a machine for execution. GMarte implements a resource selection policy that considers resource proximity by computing the bandwidth available, for each data transfer, from the local to the remote machine, which is averaged among all the data movemements with this host to provide valuable feedback to the metascheduler.

Second, the resource selection phase, involves a huge amount of communications to gather all the information required from each candidate host in order to select the most appropriate one for the execution of a given task. GMarte implements a *cache* component that stores the static attributes of each machine (i.e. Operating System, CPU architecture, etc.) so that no unnecessary queries and communication are performed. We have also uncoupled the most important stages of remote task execution (resource selection, le stage in, execution and le stage out), allowing multiple threads to independently perform these phases for different tasks. This approach enables to concurrently perform multiple task execution, thus accelerating the global procedure of scheduling.

Finally, a Global Grid subject to different management policies, and an obvious computational diversity, is very prone to distinct kind of failures. In GMarte, failures are managed at different levels. A data transfer failure is retried several times before notifying the error to the upper software layers. Also, failures during the execution of a task are detected by the scheduler. In both cases, the application is re-scheduled for execution in a different machine. A failure in a resource is also managed by re-scheduling the tasks as well as marking the resource as failed. A *resource resurrector* component periodically investigates if the resource is available again for execution.

## 7. Conclusion

In this paper, we have described the usage of Grid Computing in two different applications, one performing the 3D structural dynamic analysis of buildings and the other simulating the electrical activity in cardiac tissues. A common software layer has been employed, called GMarte, which enables to perform fault-tolerant metascheduling over Globus-based Grid infrastructures.

We have performed the execution of a structural case study on a local Grid and a cardiac case study on part of a large-scale Grid available in the framework of the EGEE project. Substantial reductions in the global execution times of both case studies have been achieved by performing multiple concurrent executions on the distributed infrastructures. Finally, we have also discussed some important aspects that must be considered when employing a large Grid. As said in the introduction, the Grid is still in the early stages, and much research and developments have to be performed in order to turn the Grid into an easy-to-use technology for its application in different scienti c  elds. However, important bene ts can be obtained, as shown, for computationally intensive applications.

## Acknowledgements

The authors wish to thank the  nancial support received from the Spanish Ministry of Science and Technology to develop the GRID-IT project (TIC2003-01318), the Conselleria de Empresa, Universidad y Ciencia - Generalitat Valenciana for the GRID4BUILD project (GV04B-424), and the Commission of the European Communities, Directorate-General Information Society, Research Infrastructures, for the EGEE project (IST-2003-508833). The authors would also like to thank the Instituto Nazionale di Fisica Nucleare (INFN), the Institute National de Physique Nucleaire et de Physique des Particles (IN2P3) for providing the computional resources, in the framework of the EGEE project, on which some executions were performed.

## References

[1] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz. Computer simulation of action potential propagation on cardiac tissues: An efficient and scalable parallel approach. *Parallel Computing: Software Technology, Algorithms, Architectures and Applications, included in series: Advances in Parallel Computing*, 13:339–346, 2004.

[2] J. M. Alonso, J. M. Ferrero (Jr.), V. Hernández, G. Moltó, M. Monserrat, and J. Saiz. Three-dimensional cardiac electrical activity simulation on cluster and grid platforms. In *Proceedings of the Vecpar 2004 International Conference*, pages 485–498, 2004.

[3] J. M. Alonso and V. Hernández. 3D structural dynamic analysis with parallel direct time integration methods. In *Proceedings of the The Tenth International Conference on Civil, Structural and Environmental Engineering Computing*, 2005.

[4] J. M. Alonso, V. Hernández, and G. Moltó. Enabling high level access to grid computing services. *ERCIM News. Special: Grids: The Next Generation*, 59:42–43, 2004.

[5] J.M. Alonso, V. Hernández, and G. Moltó. An object-oriented view of grid computing technologies to abstract remote task execution. In *Proceedings of the Euromicro 2005 International Conference*, pages 235–242, 2005.

[6] V. Berstis. *Fundamentals of Grid Computing*. IBM Redbooks Paper, 2002.

[7] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[8] T.C. Fung. Numerical dissipation in time-step integration algorithms for structural dynamic analysis. *Progress in Structural Engineering and Materials*, 5:167–180, 2003.

[9] K. Sakamoto, J. Yamazaki, and T. Nagao. Diltiazem inhibits the late increase in extracellular potassium by maintaining glycolytic atp synthesis during myocardial ischemia. *Journal of Cardiovascular Pharmacology*, 30(4):424–430, October 1997.

[10] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. A Java commodity grid kit. *Concurrency and Computation-Practice & Experience*, 13(8-9):645–662, July 2001.

[11] S. Wilkinson and D. Thambiratnam. Simplified procedure for seismic analysis of asymmetric buildings. *Computers and Structures*, 79:2833–2845, July 2001.