A High-Throughput Application for the Dynamic Analysis of Structures on a Grid Environment

J.M. Alonso, V. Hernández, G. Moltó

Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain

Abstract

This paper describes a Grid Computing application for the 3D dynamic analysis of large dimension buildings. A previously developed software composed of parallel implementations of eight direct integration methods has been integrated on this application, in order to perform structural simulations on a Grid deployment. The GMarte software abstraction layer has been employed to couple the parallel simulator with the Grid infrastructure. Performing distributed executions has enabled a considerable reduction in the global execution time of structural dynamic studies composed of different design alternatives.

Key words: Grid Computing technology, 3D structural dynamic analysis, large dimension buildings, HPC techniques, direct time integration methods, Grid middleware

1 Introduction

Three-dimensional dynamic analysis of large-scale buildings has been considered by engineers as a challenging problem, owing to its high computational demand [1]. Nowadays, an analyst that uses a structural analysis program can be waiting a considerable time before achieving the dynamic response of a medium-sized building. Besides, most of the existing commercial codes are composed of debatable simplifications, because the computation involved in a realistic simulation can be too intensive for a traditional computer. Notwithstanding, these simplifications, although appropriate for single structures, have demonstrated to be completely inadequate for complex buildings.

Moreover, a structural designer usually works with different preliminary designs of a building, and for each of them, a realistic 3D analysis is required. In

addition, it is usual that a structure has to be simulated under the influence of different dynamic loads. Thereby, for example, the Spanish Earthquake-Resistant Construction Standards (NCSE-02) demands that a building is analysed at least with five different representative earthquakes. Obviously, this number of simulations enlarges by several orders of magnitude the computational requirements of the problem.

Therefore, there is a pressing need to develop effective tools that allow us to simulate accurately and efficiently the dynamic behaviour of high-rise buildings. With the development of effective and reliable computing platforms, the application of High Performance Computing techniques allows tackling in a realistic way large-scale structural problems. However, although cost-effective clusters of PCs can simulate large dimension structures in reasonable times, in practice, studios for engineering rarely own parallel platforms. They normally employ overloaded standard PCs, that limits the size of the problems to be treated, and they are not interested in investing in clusters, mainly because of the physical space required and the associated maintenance problems, despite their excellent ratio productivity/price.

In this paper, a Grid Computing application for the 3D dynamic analysis of large dimension buildings is presented. The tool is based on the GMarte framework and makes use of a parallel software tool, composed of parallel implementations of eight well-known direct time integration methods, where all nodes of the structure are taken into account and 6 degrees of freedom per node are considered. Starting from one or more structural alternatives of a building (maybe subject to different dynamic loads), stored in a repository, and a set of available distributed computational resources, the Grid application developed carries out all the needed work to simulate, by means of the execution of this parallel software, the dynamic behaviour of the buildings on the available machines.

Nowadays, there is few existing work concerning the application of Grid Computing technologies to the structural dynamic analysis of buildings. The most important effort in this area is being carried out by the NEES (Network for Earthquake Engineering Simulation) Consortium [1] . It is devoted to reduce the impact of earthquake and tsunami disasters in the man-human infrastructures, such as roads, buildings, port facilities, and public utility systems, by linking research laboratories around the U.S. to ease the collaboration on experiments as well as for computational modelling.

The usage of a Grid-based infrastructure enables to maintain a central data repository with the information required for the simulations as well as the access to remote computational resources to perform the executions. NEESgrid [2]

[1] http://www.nees.org
[2] http://it.nees.org

represents the systems integration software of the NEES project. It links earthquake researchers with leading-edge computing resources and research equipment. NEESgrid is composed of a set of software applications devoted to earthquake engineering. Among them, OpenSEES and FEDEASlab are related to structural dynamic simulations. OpenSEES (Open System for Earthquake Engineering Simulation) is a software framework for developing applications to simulate the performance of structural and geotechnical systems subjected to earthquakes. The goal of OpenSEES is to improve the modelling and computational simulation in earthquake engineering through open-source development. FEDEASlab is a Matlab toolbox for nonlinear structural simulations under static or transient conditions for small structures or models.

On the other hand, InteliGrid [3] (Interoperability of Virtual Organisations on Complex Semantic Grid) is a European research project [2] that aims to provide a Grid-based integration and interoperability infrastructure for industries such as construction, automotive and aerospace. The main goal is to offer a flexible, secure, robust, ambient accessible, interoperable, pay-per-demand access to information, communication and processing infrastructure.

Finally, focusing on computational Grids, Alonso, et al. originally developed a Grid-based prototype based on shell-scripts which enabled to perform 3D linear static analysis of buildings on a Globus-based Grid deployment [3]. That work can be seen as a proof of concept about the applicability of Grid Computing to structural analysis. In this paper, we go one step beyond by using a much more comprehensive middleware for the execution of scientific applications on a Grid.

The remainder of the paper is structured as follows. First, section 2 describes the parallel structural simulator employed. Then, section 3 carries out a brief introduction to Grid Computing. Next, section 4 presents the main features of GMarte, the middleware that has made it possible the development of the Grid-based structural application. Implementation details of this application are shown in Section 5. Section 6 describes the case study executed, the computational resources employed and the task allocation performed. Finally, section 7 concludes the paper summarising the main achievements.

2 Parallel Structural Simulator

The parallel structural simulator employed carries out a 3D linear dynamic analysis of buildings [4]. Direct time integration algorithms, modal analysis and frequency domain analysis are different effective techniques widely em-

[3] http://www.inteligrid.com

ployed for the numerical solution of the high computational demanded equation that governs the motion of structural dynamic problems. Time integration methods have been traditionally used by analysts because of their inherent advantages. Whereas modal and frequency analysis can just be applied to linear problems, direct integration methods can be successfully applied to both of them. Numerous direct time-step integration approaches have been developed and they can be easily found in literature [5–7].

Among them, parallel implementations of the following eight integration methods compose the structural simulator employed in this Grid application: Newmark [8], Wilson-θ [9], Central Difference [10], Single-step Houbolt [11], HHT-α [12], WBZ-α [13], Generalized-α [14] and SDIRK [15]. Static condensation of nodes has not been assumed and all the elements and nodes of the structure are taken into account during the simulation process. Moreover, six degrees of freedom per each node are considered.

This simulator is based on the MPI library [16] and it employs MPI-2 I/O development of ROMIO [17] to provide good performance when accessing to disk. Therefore, the application is highly portable and it can be easily migrated to a wide variety of parallel architectures. Consistent mass matrix alternative has been employed, that supposes a more realistic approach than lumped mass matrix, although it requires a considerable increment in both the computational effort and the memory requirements. On the other hand, Rayleigh damping is employed, where the damping matrix is proportional to a combination of the mass and the stiffness matrices. All the phases that compose the dynamic simulation process have been parallelised: First, the stiffness, mass, damping and effective stiffness matrices are generated. Then, the joint displacements, velocities and accelerations are updated for each time step by means of the integration method employed. Next, the member end forces and reactions at the points attached to the rigid foundation, and the stress and deformations at any point of the structure are computed for each time step as well. Having in mind that implicit integration methods demand to solve a set of simultaneous equations at each time step, the following public domain parallel numerical libraries have been used: WSMP [18], MUMPS [19] and PETSc [20]. The simulator software is very versatile, and any combination of an integration method with any numerical library (and their corresponding different methods involved for solving the systems of linear equations) can be chosen.

3 Introduction to Grid Computing

Years ago, the computing environments were centralised, homogeneous, reliable and secure, as they had no kind of networking connectivity. With the

advent of recent increases in the bandwidth of communication networks, the idea of linking disparate machines across the world to provide a distributed computing infrastructure has been leveraged. This way, scientists and research groups fostered a collaborative work environment to address new challenges.

This collaboration started from the information sharing through the network, forums, and so on. Later they realised that there was a huge computing power distributed among the research groups and that it would be useful to coordinate it in order to face bigger challenges. The computers were shared by scientists, and a need to develop protocols and procedures arised to give place to a common, transparent, secure and coordinate way of using the resources. Therefore, the Grid Computing term was coined [21,22].

The Grid can be defined as a service for sharing not only the computational power and data storage capacity of resources, but even software applications, as much as the web is a service for sharing information over the Internet. The aim of the Grid is to provide a coherent view of distant heterogeneous computational resources so that they act as a single, huge and powerful computer.

In the traditional distributed environments, software developers used to divide their processes into independent and smaller tasks, trying to compute each of them in distinct computers. This way, Grid technology appeared to allow the resource sharing in research projects from distinct fields in which a great computing power is needed (i.e. physics or engineering). Grid technology tried to ease and coordinate the usage of the resources, owned by individuals or by research groups for sharing them with other researchers.

The users are organised into Virtual Organisations (VOs), which allow a global management for the usage of the resources. These VOs represent a virtualisation of physical organisations which establish common rules for sharing the resources (kind of resources, time intervals, etc.).

Using a Grid middleware, the user is provided with a huge and self-managed computer made up of heterogeneous systems and their associated resources. In this case, the need of computing power, storage, etc. is satisfied by the aggregation of new shared resources. The main features that are expected from a Grid middleware are:

- Local administrative control, because the resource provider maintains its control and decides whether to share it or not.
- Security, which is one of the main concerns when sharing the resources. Every user is authenticated an authorised (if proceeds) by means of the usage of personal certificates.
- Dynamic resource management, as resources may be easily removed or incorporated to the Grid environment.
- Access control and accounting, to determine which users or members of an

organisation can access the resources. Also, the accounting may be used for introducing some business models such as pay-per-usage.
- Reliability and redundancy, similar to hardware but in this case applied to software components which are distributed among the resources in the Grid.
- Load balancing and resource planning, which is provided by the introduction of dispatchers which take care of the ratio of usage of the resources.

Thus, Grid technology provides a framework which allows the exploitation of computational resources, resulting in an increase of their efficiency. In this way, there are many expectations about the Grid, in order to cope with, for example, computationally intensive applications.

4 GMarte: Simplifying the Grid for Remote Execution

Among all the available Grid middlewares, the Globus Toolkit [23,24] represents the *de facto* standard for deploying large-scale computational Grids. Globus provides software services and libraries such as: the *Monitoring and Discovery Service (MDS)* for resource discovery and monitoring; the *Grid Index Information Service (GIIS)* for resource information aggregation; the *Globus Resource Allocation Manager (GRAM)* for job execution; the *GridFTP* for reliable data transfer; and the *Grid Security Infrastructure (GSI)* for security and privacy, among others. All these services allow the deployment of large scale Grid infrastructures. This enables the users to access the remote resources as if they were available locally, while preserving the local control over who and when the resources can be accessed. By combining the usage of the client tools provided by Globus, which interact with the mentioned services, it is possible to achieve remote task execution, one of the main purposes for computational Grids.

However, the complexity of the Globus Toolkit, which only provides the basic services and capabilities to support Grid infrastructures, very often discourages scientists from porting their applications to a Grid deployment. To circumvent this issue, we developed GMarte [26,27], a software abstraction layer, on top of the Globus Toolkit and the Java CoG Kit 1.2 [25], which provides an object-oriented view of the Grid in order to simplify the remote task execution of scientific applications on Globus-based infrastructures.

Figure 1 summarises the principal layers that GMarte addresses in order to abstract the process of multiple remote task execution. First of all, a common interface is provided to access the information systems found on different versions of the Globus Toolkit (MDS2, MDS4) combined with several job managers (Portable Batch System (PBS), Torque, Sun Grid Engine, etc). On top of that layer, we introduce a new level that enables to perform the remote

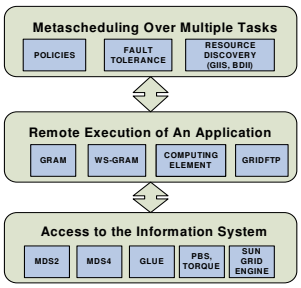

Fig. 1. Diagram of the three principal layers covered in GMarte and the Grid services involved.

execution of a single task in a computational resource by combining the usage of the underlying Grid services provided by the Globus Toolkit. Finally, above both layers we abstract the process of multiple task allocation, enabling to perform fault-tolerant metascheduling [28] combined with dynamic resource discovery.

GMarte is a client-side middleware developed in Java that offers a high level Application Programming Interface (API) to interact with remote computational resources in order to achieve remote task execution. This software addresses the execution of batch tasks that require a set of dependent input files, perform a computation, and generate a set of result data archives, which is the common behaviour of most of the non interactive applications, specially those devoted to the simulation of a physical phenomenon. These tasks are assumed to be independent and thus, they can be simultaneously executed in a Grid infrastructure. Currently, there is no support for tasks that depend on the output of other ones. However, GMarte supports the execution of parallel applications implemented with the MPI library, thus taking advantage of the multiprocessor machines of a Grid deployment.

The middleware offers task allocation functionality over multiple administrative domains. This enables the execution of a set of tasks in different computational resources of a Globus-based Grid infrastructure. The computational resources that can be employed are those with the Globus Toolkit 2.4.X and 4.X as well as the Computing Elements with the LCG-2 middleware [4]. Figure 2 summarises the principal and most important classes involved in the design of the object-oriented middleware. This class diagram (in UML notation) represents the core of the object-oriented high level API of GMarte. For the sake of brevity, only the relevant methods are exposed in the diagram.

In the figure, a *GridResource* stands as the abstraction for a computational resource, whereas a *GridTask* represents the object vision of a task that has to be executed in the Grid. The diagram shows that a *TestBed* is considered a collection of GridResources and a *GridTaskStudy* represents a set of GridTasks.

[4] http://www.cern.ch/lcg

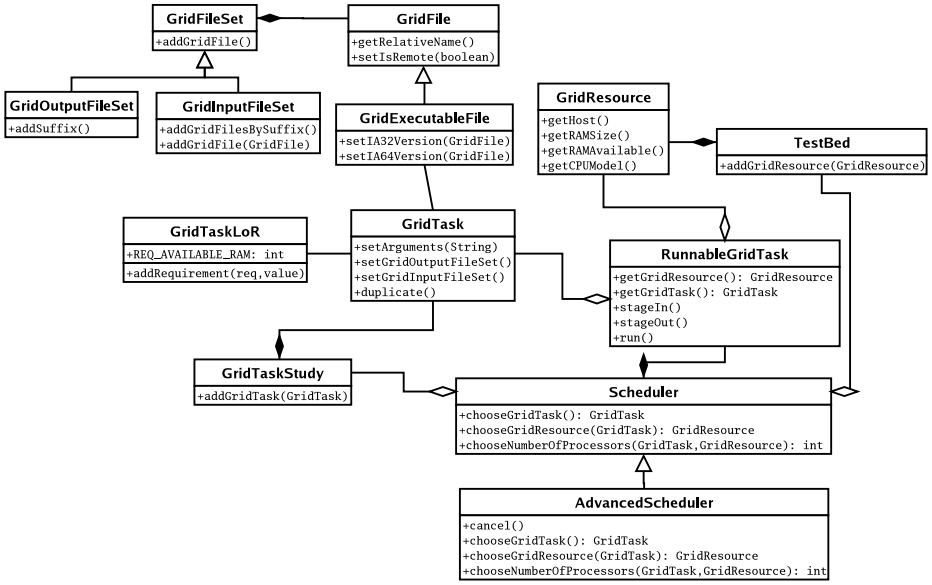

Fig. 2. Diagram of the main classes in the middleware.

A *RunnableGridTask* is a GridTask that has been assigned to a GridResource for execution. The *Scheduler* abstract class provides no implementation but a contract of the interface to be implemented by the schedulers. For example, the *AdvancedScheduler* supplies the implementation of the abstract methods defined by the Scheduler class, thus offering task allocation capabilities. This approach enables to implement different user-defined task allocation policies within the GMarte framework.

Using the API exposed by GMarte, the user no longer interacts with the Globus Toolkit services, but instead deals with instances (objects) of the classes in Figure 2. As an example, a scientific out of the field of Grid Computing may not know about the MDS or the GRAM services provided by the Globus Toolkit, but sure he/she can rapidly understand that a GridResource stands for a computational resource. In fact, introducing a user level interface dramatically reduces the *time to Grid*, or time spent on migrating an application to a Grid environment, because the user is provided with a natural high level tool that can easily be employed for non experts in Grid computing.

Additionally, GMarte also offers an XML (eXtensible Markup Language) interface so that the user does not have to develop a Java application to use the metascheduling services provided by this software. Currently, the user may specify, in XML language, the description of the computational tasks, the Grid resources to be employed and the configuration of the metascheduler. This way, we have developed components that automatically provide the bindings from the XML descriptions to GMarte objects. Thus, this approach enables the users to focus on their applications without having to deal with the development of a GMarte application.

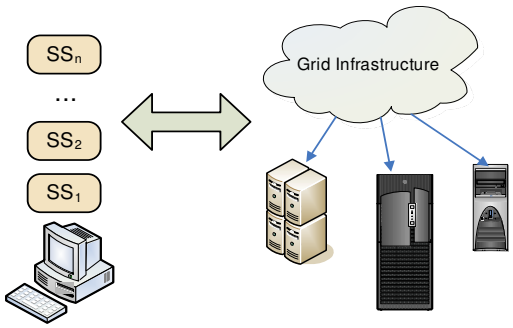

Fig. 3. Structural dynamic simulations in a Grid deployment.

In the field of abstracting the usage of Grid Computing technologies, we find several projects. First of all, the European Union GridLab project [5] aims at developing application tools and middlewares for Grid environments. Within this project, the Grid Application Toolkit (GAT) [29] also provides an object-oriented approach trying to provide a complete application oriented abstraction layer. From the GridLab documentation, it appears that the resource management is only available for the Globus Toolkit 2.4, thus being restricted to work with the Pre-Web Services components of Globus. Among others, we can also find Condor [30], a software system that effectively utilises the computing power of a set of resources. However, although Condor provides a gateway to interface with Globus (Condor-G) [31], it only offers a set of command-line applications without an API. Notwithstanding, developments are being made towards an API accessible via SOAP (Simple Object Access Protocol). In addition, the resource selection policy implemented in Condor-G does not consider the dynamic state of the computational resources, what imposes a serious limitation considering the highly dynamic behaviour of a Grid infrastructure.

5 Grid-based Structural Simulator

The Grid-based application developed aims at using a computational infrastructure in order to accelerate the simulation of case studies which are composed of one or more different structural solutions where one or more dynamic loads are applied to them. Figure 3 overviews the context of the problem. Consider a structural studio which intends to analyse the response of a building under the influence of different earthquakes. This problem requires the execution of multiple simulations (denoted by SS_i) which, depending on the magnitude of the building, may be time-consuming as well as memory-intensive for a traditional PC.

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

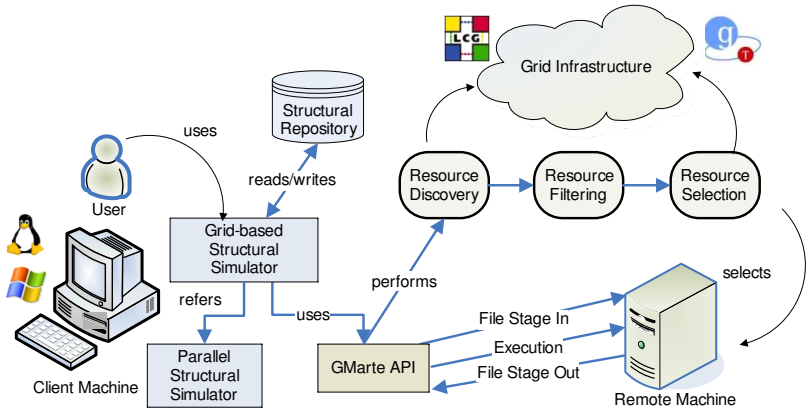

Fig. 4. Principal steps involved the execution on a Grid infrastructure.

If the studio could gain access to a distributed computing infrastructure, then it would have available distinct candidate computational machines to execute the different simulations. Thus, performing multiple concurrent executions on the computational resources of a Grid deployment could be an ideal situation to accelerate the execution of whole resource-starved structural studies. Anyway, this would potentially allow, at least, to tackle new structural problems with a dimension that overcomes the physical limitations of a traditional PC. Therefore, in order to harness the power that a Grid infrastructure aims to deliver, we have developed a software application that enables to easily perform 3D structural dynamic simulations on the computational nodes of a Globus-based Grid infrastructure. This application employs the API provided by the GMarte middleware.

5.1 Principal Steps Involved in the Execution of Case Studies

Figure 4 summarises the principal steps implemented in the Grid-based structural application in order to achieve remote task simulations. All these stages take place thanks to the corresponding methods belonging to the GMarte API and it is important to point out that the application works seamlessly for Windows and Linux platforms. First of all, a *resource discovery* phase is in charge of obtaining a list of potential execution machines, from a Grid infrastructure, where the different structural solutions will be dynamically simulated. Resource discovery is implemented by accessing the *Grid Index Information Service (GIIS)*, a service included in the Globus Toolkit which provides aggregate information about the resources within a site.

Once a preliminary list of machines has been obtained, the *resource filtering* phase enables to discard those resources that do not fulfill some basic capabilities, such as those computational requirements specified by the application. This step also includes to apply an authorisation filter to the original list of

resources discovered, discarding those that can not be accessed with the credentials supplied by the user, as well as those physically unavailable at this moment. This procedure results in a list of candidate machines for execution of simulations. It should be noticed that these two phases described are performed before the scheduling process starts and thus, they are executed only once for a structural case study. The following steps are performed for each structural solution to be dynamically analysed.

From this subset of computational resources, the *resource selection* phase is in charge of choosing one remote machine on which to schedule the task to be simulated. Resource selection is still considered an open research topic because the problem of organising as well as ranking resources having in mind the application requirements is difficult [32]. This procedure involves some *rank function* that attempts to quantify how good is a machine for the execution of a given task. In our case, the rank function makes use of a performance model that takes into account the cost of the data transfer, from the submission machine to the candidate resources, as well as a load distribution component that prevents from assigning all the simulations to a single machine. The number of processors involved in the parallel execution are selected according to the available computing nodes of the remote machine.

Once the resource on which the task being scheduled has been selected, several phases are performed in order to achieve remote task execution:

(1) *File Stage In*: The parallel structural simulator as well as all the input files describing the building and the external loads applied, which are obtained from the *structural repository*, are transferred to the remote machine.
(2) *Execution*: The application is started on the remote machine and periodically monitored to take into account failures as well as detecting when the execution finishes.
(3) *File Stage Out*: This phase is in charge of retrieving all the output files generated on the remote machine to the local computer and erasing all of them. This way, all these simulation results can be later and offline analysed by a structural specialist in the local machine.

The implementation makes use of three different threads, each one performing a distinct phase. This enables to concurrently handle an expensive stage in phase for a task, while performing the execution or the stage out phase for another simulation, what reduces the overhead introduced by the task allocation process.

The Grid-based software implements fault-tolerance at different levels. Failures occurred during the data movement phases are detected by the data transfer components and retried several times (three by default) until notifying the

failure and re-scheduling the task to another computational machine. Also, failures occurred during the execution of the application in the remote machine cause the re-scheduling of the task, which will be executed on a different resource.

5.2 Describing a Case Study via Object-Oriented Abstractions

It is very straightforward to modify the Grid-based application and define a new structural case study to be executed in a Grid Computing infrastructure. Of course, the description of any case study will be done using the API provided by GMarte. First of all, we have to create (recall Figure 2) a class which describes all the GridTasks (in our case the different structural simulations) that must be executed. Therefore, we create the *StructuralCaseStudy* class which inherits from the GridTaskStudy class and includes all the GridTasks that compose the case study. For each GridTask, we must specify:

(1) The *GridExecutableFile*, corresponding to the executable file of the parallel structural simulator that it will be run on the remote machines.
(2) The *GridInputFileSet*, which indicates the input data files that the application requires. This set of archives describes the geometry of the building, its properties and the external loads applied. They will be transferred to the remote resource before execution.
(3) The *GridOutputFileSet*, indicating the output data files that the application generates. They will be transferred to the local machine when the execution finishes. The parallel simulator employed can be configured to save periodically, according to a certain time step decided by the user, joint displacements, velocities and accelerations; member end forces; and stresses and deformations at any point of the structure.
(4) The *GridTaskLoR* object, that is, the list of computational requirements of the parallel application, in terms of the minimum number of processors employed (if required by the user) as well as the minimum required RAM for execution.

The following extract of Java code, belonging to the StructuralCaseStudy class within the Grid-based application, shows how a user would provide the definition of a GridTask:

```
GridTask gt =  new GridTask();
GridExecutableFile gef =
new GridExecutableFile("/home/user/hiperbuild");
gt.setGridExecutableFile(gef);
gt.setArguments(commandLineArguments);
GridInputFileSet gifs = new GridInputFileSet();
```

```
gifs.addGridFilesByPreffix("/home/user","ss1");
gt.setGridInputFileSet(gifs);
GridOutputFileSet gofs = new GridOutputFileSet();
gofs.addGridFilesBySuffix(".bin");
gt.setGridOutputFileSet(gofs);
GridTaskLoR gtl = new GridTaskLoR();
gtl.addRequirement(AVAILABLE_RAM, 512);
gt.addGridTaskLoR(gtl);
addGridTask(gt);
```

In the example, the executable file called *hiperbuild* as well as the input data files (a set of files starting with *ss1*) are assumed to reside in the /home/user directory. The variable *commandLineArguments* must previously contain the command-line arguments that are required to invoke the parallel structural simulator, such as the direct integration algorithm employed, with its appropriate parameters, and the method selected to solve the appearing systems of linear equations. The output data files that the simulator will generate are specified as a set of files ending in *.bin*. All of them will be sent back to the local machine. Besides, resources with at least 512 MBytes of available RAM memory will be selected. Finally, the GridTasks are assigned to the case study via the addGridTask method of the GridTaskStudy class. Notice that only one GridTask has been defined. To generate the other instances, we can duplicate an already created GridTask and only modify the command-line parameters, together with the instruction line where the input data files are specified. This is an easy way to create a full structural case study.

For the sake of completeness, we also provide the XML description of the case study containing one task:

```
<GridTaskStudy>
 <GridTask>
  <GridExecutableFile directory="/home/user"> hiperbuild
  </GridExecutableFile>
  <Arguments> commandLineArguments </Arguments>
  <GridInputFileSet directory="/home/user">
    <Preffix> ss1 </Preffix>
  </GridInputFileSet>
  <GridOutputFileSet>
    <Suffix> .bin </Suffix>
  </GridOutputFileSet>
  <GridTaskLoR>
    <Requirement type="AVAILABLE_RAM" value="512"/>
  </GridTaskLoR>
 </GridTask>
</GridTaskStudy>
```

To specify the computational resources involved in the execution of a case study, we can either use the resource discovery functionality provided by GMarte or explicitly enumerate the machines to be employed. The latter is implemented, for example, by creating an instance of the TestBed class and adding the GridResource objects that correspond to each machine, with this simple manner:

```
TestBed testBed = new TestBed();
GridResource gr =  new GridResource("machine.domain.com");
testBed.addGridResource(gr);
```

The corresponding XML syntax for the definition of the TesBed is:

```
<TestBed>
 <GridResource hostName="machine.domain.com"/>
</TestBed>
```

Once the case study has been created and described (the GridTaskStudy), as well as the computational resources to be employed (the TestBed), we invoke the scheduling capabilities of GMarte. For that, several steps must be accomplished:

(1) To instantiate the GridTaskStudy, that is, to create a new instance of the StructuralCaseStudy.
(2) To create the *TestBed*, that is, the set of computational machines that are going to provide the execution services.
(3) To create an instance of the *Scheduler*, which will perform the task allocation of the GridTaskStudy to the TestBed.
(4) To initiate the scheduling procedure. The procedure finishes when all the tasks have been successfully executed.

The following extract of Java code will perform all the mentioned steps:

```
GridTaskStudy scs =  new StructuralCaseStudy();
TestBed testBed =  new TestBed();
testBed.addGridResource(new GridResource("machine.domain.com"));
Scheduler scheduler =  new AdvancedScheduler(scs, testBed);
scheduler.start();
scheduler.waitUntilFinished();
```

In this example, we have assumed that the TestBed class includes all the computational resources that are going to be employed. Alternatively, we could have employed the resource discovery functionality provided by GMarte that queries a GIIS server to obtain a list of candidate machines for execution. In this particular study, we have determined which computational resources are going to be used for execution. This is why we decided to enumerate the list

of machines involved. Also, notice that in this example the XML description that corresponds to the scheduler configuration has not been included. Given that the default configuration has been applied, the XML definition is not required.

6 Case Study

In order to assess the benefits of using a distributed computing infrastructure in the structural dynamic analysis domain, we have performed the execution of a realistic structural case study on a Globus-based Grid deployment.

6.1 Description

An apartment building was proposed to test the performance of the Grid application. It is composed of two basements, one ground floor and thirty-three stories. The basements are dedicated to garages, the ground floor to a commercial establishment and the upper stories to apartments.

The building is composed of reinforced concrete, and eight combinations of different structural member dimensions were considered for its design. Each structural alternative was composed of about 40,000 structural elements and 22,000 nodes, i.e. 132,000 degrees of freedom. Taking into account the current Spanish-Resistant Construction Standards, each of the eight different preliminary solutions was linearly simulated under the influence of five different representative earthquakes, according to the geographical location of the building.

Therefore, this case study resulted in 40 independent structural dynamic simulations. The varying parameters for the definition of the case study were the input files which described the distinct structural alternatives. The accelerograms employed presented values of ground acceleration equally spaced at every 0.01 seconds and they had a duration between 5 and 10 seconds. However, simulation time was fixed to 5 seconds in all the analyses, with integration time steps equals to 0.01 seconds. Generalized-α method was chosen to carry out the simulations, because of its versatility. In this case study, the parallel simulator was configured to employed MPI-based PETSc and MUMPS numerical libraries. PETSc was used to matrix assembly and matrix-vector operations in parallel. Taking advantage of the interface to external numerical libraries provided by PETSc, the MUMPS package was employed for solving the resultant systems of linear equations by means of a parallel direct method based on Multifrontal techniques.

Table 1
Detailed machine characteristics of the Grid infrastructure.

Machine	Processors	Memory
Kefren	20 dual Intel Pentium Xeon @ 2.0 Ghz	1 GByte
Ramses	12 dual Intel Pentium III @ 866 Mhz	512 MBytes
Odin	55 dual Intel Pentium Xeon @ 2.8 Ghz	2 GBytes

Each execution required a total amount of 417 MBytes of RAM and generated 82.73 MBytes of raw binary data. The output data contains information about the stresses and deformations at multiple predefined intermediate points of all the structural elements that compose the building, saved on disk every 1 second. This resulted in a total output data set of 3.2 GBytes.

6.2 Computational Grid Deployment

For the execution of this case study, we have employed a Grid infrastructure composed of computational resources which belong to our research group. It consists of 3 clusters of PCs whose principal characteristics are detailed in Table 1. These machines are interconnected via a local area network delivering 100 Mbits/sec. The Globus Toolkit version 2.4.3 [23] was previously installed on each machine of the Grid deployment.

In order to assess the effectiveness of Grid Computing we decided to rely on our local resources without depending on machines from other organisations. Although employing a distributed testbed with resources from multiple organisations is at first glance more attractive, the only important difference would reside on the amount of time involved in the data transfers.

6.3 Execution Results of the Case Study

Executions were submitted from a machine belonging to the same local area network than the clusters. Table 2 summarises the task allocation process. It indicates the initial number of available computing nodes at each cluster during the case study execution, as well as the number of tasks allocated to each machine. It may be surprising that although Odin is the most powerful machine, it did not receive all the simulations, as it could be expected. However, the scheduling policy implemented in the Grid application tries to distribute the executions on different machines, implementing a two-fold strategy: First, it eliminates the dependence on a single resource thus reducing the impact caused if the machine fails. Second, this is a polite strategy that avoids overloading a single machine, possibly belonging to another organisation.

Table 2
Distribution of the simulations in the testbed, for each machine.

Machine	Initial Available Nodes	Tasks Allocated
Kefren	19	17
Ramses	10	10
Odin	53	13

On average, each execution required 11.37 seconds to perform the file stage in phase, 530.96 seconds to carry out execution and 88.36 seconds to accomplish the stage out phase. The simulation process of all the structural solutions required 34 minutes from the beginning of the scheduling until the data transfer for the last task finished.

A sequential execution of the whole case study on one node of cluster Odin lasted a total of 354 minutes (almost 6 hours). This represents a reduction factor of 10.41, that is, the Grid computing approach was more than 10 times faster than using a traditional sequential execution.

Consider an average 8-processor cluster of PCs, which may be suitable for a typical research centre. The execution of the case study performing two concurrent 4-processor executions on cluster Odin required a total of 63.73 minutes. Thus, the Grid Computing approach was almost 2 times faster than this High Performance Computing approach.

Caution should be taken when performing executions on distributed remote machines with such an amount of output data. It is clear that the generated output results should be reduced to the minimum in order to minimise the time invested in the data transfers. Obviously, we are aware that the time spent of these data movements supposes the main drawback of the application implemented. However, we have preferred for this case study to generate a large amount of data, considering the fact that we are working on a local area network.

7 Conclusions and Future Work

As a conclusion, this paper presents a Grid Computing application that allows members of a scientific Virtual Organisation and SMEs to carry out multiple and concurrent 3D dynamic simulations of singular buildings. This application has been developed on top of the GMarte middleware, which enables to perform fault-tolerant task allocation on Globus-based machines, and it makes use of a parallel simulator where different time integration methods have been implemented. The Grid application developed takes advantage of

different resources available in the network without the need of investing in High Performance Computing machines.

To assess the benefits of Grid Computing technology, we have executed a structural case study on a Grid infrastructure obtaining an important reduction in the global simulation time. It is clear that Grid Computing is a promising new technology which will have an important impact in many engineering fields. Therefore, the development of software applications that demonstrate the advantages of using these technology is of paramount importance for its adoption.

The future works involve the usage of a Grid service-oriented architecture on which the clients would connect to a machine offering the service of dynamically analysing a large dimension structure. This would require a reengineering of the application developed to move into an approach based on Grid services. However, it would provide important advantages for the potential users, since they could employ a High Performance Computing-based dynamic simulator without the need of purchasing this software and being worried about licences and new updates. Besides, different post-processing procedures can be carried out by means of the Grid service in order to reduce the time spent on the result file transfers. For example, the calculation results can be compressed before sending them to the local machine. On the other hand, stresses and deformations obtained when different earthquakes are applied to a same structural solution can be combined, adopting as result the average characteristic values generated by each of them, thus reducing considerable the amount of data to be recovered.

8 Acknowledgments

The authors wish to thank the financial support received from The Spanish Ministry of Science and Technology and the Generalitat Valenciana to develop the projects GRID-IT (TIC2003-0131) and Grid4Build (GV04B-424) respectively. The work developed under the GRID-IT project has been partially supported by the Structural Funds of the European Regional Development Fund (ERDF).

References

[1] Clough RW, Penzien J. Dynamics of Structures. Second Edition. Computers and Structures, Inc., 2004.

[2] Dolenc M, Turk Z, Katranuschkov P, Kurowski K, Hannus M. Towards a Grid enabled engineering collaboration environment. In: B.H.V. Topping (Editor), Civil-Comp Press. Proceedings of the Tenth International Conference on Civil, Structural and Environmental Engineering Computing (CC 2005), paper 69, 2005.

[3] Alonso JM, de Alfonso C, García G, Hernández V. Integrating HPC and Grid Computing for 3D structural analysis of large buildings. In: B.H.V. Topping (Editor), Civil-Comp Press. Proceedings of the Fourth International Conference on Engineering Computational Technology (ECT 2004), paper 92, 2004.

[4] Alonso JM, Hernández V. Three-dimensional structural dynamic analysis using parallel direct time integration methods. In: B.H.V. Topping (Editor), Civil-Comp Press. Proceedings of The Tenth International Conference on Civil, Structural and Environmental Engineering Computing (CC 2005), paper 232, 2005.

[5] Fung TC. Numerical dissipation in time-step integration algorithms for structural dynamic analysis. Progress in Structural Engineering and Materials 2003; 5:167–180.

[6] Dokainish MA, Subbaraj K. A survey of direct time-integration methods in computational structural dynamics-I. Explicit methods. Computers & Structures 1989; 32:1371–1386.

[7] Dokainish MA, Subbaraj K. A survey of direct time-integration methods in computational structural dynamics-II. Implicit methods. Computers & Structures 1989; 32:1387–1401.

[8] Newmark NM. A method of computation for structural dynamics. Journal of Engineering Mechanics Division, ASCE 1959; 85:67–94.

[9] Wilson EL. A computer program for the dynamic stress analysis of underground structures. SESM Report 68-1; Division of Structural Engineering and Structural Mechanics, University of California, Berkeley, 1968.

[10] Belytschko T, Mullen R. Explicit integration of structural problems. Finite Elements in Nonlinear Mechanics 1977; 2:697–720.

[11] Chung J, Hulbert GM. A family of single-step Houbolt time integration algorithms for structural dynamics. Computer Methods in Applied Mechanics and Engineering 1994; 118(1-2):1–11.

[12] Hilber HM, Hughes TJR, Taylor RL. Improved numerical dissipation for time integration algorithms in structural dynamics. Earthquake Engineering and Structural Dynamics 1977; 5:283–292.

[13] Wood WL, Bossak M, Zienkiewicz OC. An alpha modification of Newmark's method. International Journal for Numerical Methods in Engineering 1980; 15:1562–1566.

[14] Chung J, Hulbert GM. A time integration method for structural dynamics with improved numerical dissipation: the generalized-α method. Journal of Applied Mechanics 1993; 60:371–376.

[15] Owren B, Simonsen HH. Alternative integration methods for problems in structural dynamics. Computer Methods in Applied Mechanics and Engineering 1995; 22(1-2):1–10.

[16] Gropp WD, Lusk E. Users guide for MPICH, a portable implementation of MPI. Mathematics and Computer Science Division, Argonne National Laboratory, 1996.

[17] Thakur R, Gropp W, Lusk E. On implementing (MPI-IO) portably and with high performance. Proceedings of the Sixth Workshop of I/O in Parallel and Distributed Systems 1999; 23–32.

[18] Gupta A, Joshi M, Kumar A. WSMP: a high-performance shared-and distributed-memory parallel sparse linear equation solver. IBM Research Report RC 22038(98932), IBM, 2001.

[19] Amestoy P, Duff I, L'Excellent JY, Koster J. MUltifrontal Massively Parallel Solver (MUMPS version 4.6.3) users guide, 2006.

[20] Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley M, Curfman-McInnes L, Smith BF, Zhang H. PETSc users manual. Technical Report ANL-95/11 - Revision 2.3.2. Argonne National Laboratory, 2006.

[21] Foster I, Kesselman C. The GRID 2: blueprint for a new computing infrastructure. Second edition. Morgan-Kaufmann, 2004.

[22] von Laszewski G. The Grid-idea and its evolution. To be published, 2005. http://www-unix.mcs.anl.gov/~laszewsk/papers/vonLaszewski-grid-idea.pdf.

[23] Foster I, Kesselman C. Globus: a metacomputing infrastructure toolkit. International Journal of Supercomputer Applications 1997; 11:115–128.

[24] Foster I. Globus Toolkit version 4: Software for service-oriented systems. In: IFIP International Conference on Network and Parallel Computing, 2005, Springer-Verlag LNCS 3779, 2-13.

[25] von Laszewski G, Foster I, Gawor J, Lane P. A Java Commodity Grid kit. Concurrency and Computation: Practice and Experience 2001; 13:643–662.

[26] Alonso JM, Hernández V, Moltó G. An object-oriented view of Grid Computing technologies to abstract remote task execution. Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-based Processing 2005; 235–242.

[27] Alonso JM, Hernández V, Moltó G. GMarte: Grid Middleware to abstract remote task execution. Concurrency and Computation: Practice and Experience. To appear. http://www3.interscience.wiley.com/cgi-bin/fulltext/112606256/PDFSTART.

[28] Schopf JM. Ten actions when superscheduling. Scheduling Working Group, SchedWD 8.5, 2001.

[29] Seidel E, Allen G, Merzky A, Nabrzysky J. GridLab: A Grid application toolkit and testbed. Future Generation Computer Systems 2002; 18: 1143–1153.

[30] Thain D, Wright D, Miller K, Livny M. Condor - A distributed job scheduler. In: Beowulf Cluster Computing with Linux, T. Sterling (ed.), MIT Press, 2001.

[31] Frey J, Tannenbaum T, Foster I, Livny M, Tuecke S. Condor-G: A computation management agent for multi-institutional Grids. Journal of Cluster Computing 2002; 5:237–246.

[32] Angulo D, Foster I, Liu C, Yang L. Design and evaluation of a resource selection framework for Grid applications. IEEE International Symposium on High Performance Distributed Computing (HPDC-11), 2002.