A Service Oriented System for On Demand Dynamic Structural Analysis over Computational Grids [⋆]

J. M. Alonso, V. Hernández, R. López, and G. Moltó

Departamento de Sistemas Informáticos y Computación.
Universidad Politécnica de Valencia. Camino de Vera s/n 46022 Valencia, Spain
{jmalonso,vhernand,rolopez,gmolto}@dsic.upv.es
Tel. +34963877356, Fax +34963877359

Abstract. In this paper we describe the implementation of a service oriented environment that enables to couple a parallel application, which performs the 3D linear dynamic structural analysis of high-rise buildings, to a Grid Computing infrastructure. The Grid service, developed under Globus Toolkit 4, exposes the dynamic simulation as a service to the structural scientific community. It employs the GMarte middleware, a metascheduler that enables to perform the computationally intensive simulations on the distributed resources of a Grid-based infrastructure.

Topics. Parallel and Distributed Computing, Cluster and Grid Computing, Large Scale Simulations in All Areas of Engineering and Science.

1 Introduction

Traditionally, the dynamic analysis of large scale buildings has been limited to simplifications with the purpose of reducing the computational and memory requirements of the problem. Although these simplifications have been proved to be valid for simple and symmetric structures, they have demonstrated to be inappropriate for more complex buildings.

Nowadays, many buildings are asymmetric and the effects of torsion have been identified as one of the main reasons that make a building collapse when an earthquake occurs. Considering the dramatic effects of earthquakes, it is crucial to investigate their impact before a building gets constructed. However, the required memory and the computation involved in a 3D realistic analysis of a large dimension building can be too intensive for a traditional PC.

This way, the authors have developed an MPI-based application that performs the 3D linear dynamic analysis of structures using three different direct

[⋆] The authors wish to thank the financial support received from the Spanish Ministry of Science and Technology to develop the GRID-IT project (TIC2003-01318) and the Conselleria de Empresa, Universidad y Ciencia - Generalitat Valenciana for the GRID4BUILD project (GV04B-424). We wish also to thank Anshul Gupta for providing us with a trial license of WSMP library for 16 CPUs.

time integration schemes. Typically, a structural designer works with different preliminary alternatives when designing a building, considering distinct layouts or applying multiple sections or dimensions to its members, requiring their simulation under the influence of several dynamic loads. For example, the Spanish Earthquake-Resistant Construction Standards (NCSE-02) demands a building to be analysed with at least five different representative earthquakes. Obviously, this situation largely increases the computational cost of the problem. However, although the parallel application offers quite good parallel performance and carries out a 3D realistic analysis, studios for engineering rarely own parallel platforms to execute this software.

Therefore, we have implemented a service oriented system, based on Grid services, that enables to perform on demand dynamic analysis over computational Grids in a collaborative environment. It implies a two-fold strategy. Firstly, the main objective of Grid technology is to share and use different resources available in the network. Thus, it is possible to create a scientific and technical virtual organisation where most of the members do not need to invest in computational machines and software, and to be worried about licenses and new updates. It would be enough to establish agreements for their usage. Secondly, the service exploits, in a transparent way for the user, the computational capabilities of a distributed Grid infrastructure which delivers enough power to satisfy the computational requirements of the resource-starved dynamic structural simulations of high-rise buildings.

The reminder of this paper is structured as follows: First, section 2 describes the motion equation and the parallel application developed to simulate the behaviour of structures. One building has been also simulated to analyse the performance of this HPC application. Next, section 3 shows the Grid service implemented and the metascheduling approach to enable high-throughput when multiple user requests are concurrently received. Section 4 presents the structural case study that has been executed to test the performance of the Grid service, the computational resources employed and the task allocation performed. Finally, section 5 concludes the paper.

2 Parallel 3D Linear Dynamic Analysis of Buildings

The second order differential equations in time that governs the motion of structural dynamic problems can be written as follows [1]:

$$Ma(t) + Cv(t) + Kd(t) = f(t) \tag{1}$$

where M, C and K are the mass, damping and stiffness matrices respectively, $f(t)$ is the applied dynamic load vector, and $d(t)$, $v(t)$ and $a(t)$ represent the unknown displacement, velocity and acceleration vectors at the joints of the structure. The initial conditions at $t = 0$ are given by $d(0) = d0$ and $v(0) = v0$.

Because of their inherent advantages, direct time integration algorithms have been widely employed for the numerical solution of this computationally demanding equation [2]. In this way, an MPI-based parallel application for the 3D linear dynamic analysis of high-rise buildings has been implemented, where all the nodes of the structure are taking into account and six degrees of freedom per joint are considered. Node condensation techniques have not been assumed. All these resultant computational burden implies the need of using HPC strategies able to tackle large dimension problems and reduce the time spent on the analysis. In the application, the following three well-known time integration methods have been parallelised, providing comprehensive results in very reasonable response times: Newmark [3], Generalized-α [4] and SDIRK [5]. Consistent-mass matrix has been assumed, and Rayleigh damping has been employed, what means that $C = \alpha M + \beta K$. Besides, the standard implementation of MPI-2 I/O by ROMIO has been used to guarantee good performance on secondary storage device accesses. The application is highly portable and it can be easily migrated to a wide variety of parallel platforms.

Regarding the parallelisation of the problem, each processor is assigned firstly a group of N/p consecutive nodes and another one of B/p consecutive structural elements, being N and B the total number of nodes and beams in the building, respectively, and p the number of processors employed. Then, each processor generates and assemblies in parallel its local part of the stiffness, mass and damping matrices, according to their nodes assigned. In this way, all the matrices of the problem, together with the different resulting vectors, will be partitioned among the processors following a row-wise block stripped distribution. Consistent-mass matrices have been considered, a more realistic alternative than lumped (diagonal) mass matrix. However, the dynamic analysis of a consistent-mass system requires more considerable computational effort and memory requirements than a lumped-mass system does.

Next, the effective stiffness matrix $\hat{K}$, or coefficient matrix of the problem, is obtained in parallel by means of a linear combination of K, M and C matrices. Different functions for summing sparse matrices in parallel have been implemented in order to generate these C and $\hat{K}$ matrices. Finally, the initial conditions are imposed in the system. Displacements and velocities at $t = 0$ will be usually known, and initial accelerations will be computed by solving the resulting system of linear equations when the Equation (1) is evaluated at $t = 0$, where M matrix constitutes the coefficient matrix.

Then, for each time step ($t = \Delta t, 2\Delta t, 3\Delta t, \ldots, n\Delta t$) different numerical phases must be also carried out. Firstly, the movement, velocity and acceleration vectors at the joints of the structure are computed in parallel by means of the chosen time integration method. More in detail, movements are worked out by solving a system of linear equations where the $\hat{K}$ coefficient matrix is large, sparse, symmetric and positive definite. Fortunately, the K, M and C matrices are constant, along the time, in a linear analysis. Thus, the $\hat{K}$ coefficient matrix does not change during the simulation process, and it just need to be factorized once if a direct method is employed to compute the linear systems. In this way, one forward-backward substitution will be carried out, for each time step, for computing the nodal movements. Parallel direct and iterative methods implemented in WSMP [6], MUMPS [7] and PETSc [8] public domain numerical libraries have been used for solving these linear systems. These three libraries have been chosen due to its availability, good performance and state-of-the-art capabilities. WSMP and MUMPS are MPI-based numerical libraries for solving large sparse symmetric and non-symmetric systems of linear equations. The parallel symmetric numerical factorization implemented in WSMP is based on Cholesky Multifrontal algorithm. MUMPS uses a Multifrontal technique which is a direct method based on LU or LDLT factorization of the matrix. On the other hand, PETSc provides parallel matrix and vector assembly routines, basic linear algebra operations and parallel linear, nonlinear equation solvers and time integrators. The combination of a Krylov subspace method and a preconditioner is the heart of the parallel iterative methods implemented in PETSc. Besides, PETSc provides and efficient access to different external numerical libraries that implement direct methods, such as MUMPS, or preconditioners.

Before solving the linear system, the effective dynamic load vector, i.e. the right hand-size vector, must be evaluated in parallel. Again, each processor just computes and assembles the load vector corresponding to its group of nodes assigned. Sparse matrix-vector products, a constant times a vector and sums of vectors are the basic lineal algebra operations than take place in this phase. Therefore, different functions that carry out these mentioned linear algebra operations in parallel have been programmed and they will be used when employing WSMP, but not when using PETSc or MUMPS, since PETSc already provides routines for these functionalities.

Notice that parallel sparse matrix-vector product has a crucial importance for each time step, where the performance achieved could be severely degraded if an efficient implementation is not developed. Having in mind this consideration, communications have been tried to be minimised. For that, the processor i just sends the processor j those elements of its local vector that the processor j needs to carry out the matrix-vector product. Remember two things: (1) the vector is initially partitioned into the processors by means of a row-wise block-striped distribution and (2) the matrix is sparse and so not all the vector elements belonging to other processors will be needed. Considering the non-zero structure of problem matrices, each processor computes just once, at the beginning of the simulation and in a very fast way, which elements belonging to itself must be sent for each time step to every other processor. As a consequence, each processor just receives from the others the vector elements that it strictly needs during the simulation.

Once joint displacements have been computed, velocity and acceleration values are updated by taking advantage of the implemented routines of sum of vectors. In contradistinction to Newmark and Generalized-α methods, SDIRK procedure requires to solve two linear systems for each time step. The first one, for $\hat{K}$ coefficient matrix, is composed of s right hand-size vectors, being s the number of stages employed in the method. Solution vectors of this system will be employed for updating displacement and velocity vectors. In the second one, M represents the coefficient matrix and the acceleration vector is computed.

Obviously, both matrices will be factorized once if a direct method is used, and multiple forward-backwards substitutions will be required for each time step.

Finally, each processor evaluates in parallel, for its structural elements initially assigned, the member end forces and the reactions at the points attached to the rigid foundation. Bending moments and deformations at the predefined division points of the members will be evaluated in parallel, with the same data distribution, to check that they do not exceed the established design limits.

A building composes of 68,800 nodes (412,800 degrees of freedom) and 137,390 structural elements has been chosen to show the performance achieved in the parallel application. The behaviour of the building was dynamically analysed under the influence of an earthquake applied during 6 seconds, with time steps equals to 0.01 seconds.

Tables 1, 2 and 3 show the time (in minutes) and the efficiencies spent on the whole structural analysis, for the different integration methods parallelized, employing up to 16 processors, for WSMP, MUMPS and PETSc numerical libraries. MUMPS was employed thanks to the interface provided by PETSc. This time does not include the initial one corresponding to the generation of the stiffness, mass, damping and effective stiffness matrices, or the imposition of initial conditions or the factorization of effective stiffness matrices. The simulations have been run on a cluster of 20 dual Pentium Xeon@2GHz, with 1 GByte of RAM and interconnected by a SCI network.

Proc.	WSMP		MUMPS (QAMD)		MUMPS (MND)		PETSc	
1	-	-	53.94	100.00%	-	-	2087.06	100.00%
2	32.16	100.00%	27.33	98.28%	26.72	100.00%	1119.20	93.23%
4	17.91	89.80%	16.04	84.07%	14.82	90.15%	581.91	89.66%
8	11.32	71.00%	10.33	65.27%	9.24	72.29%	305.64	85.36%
16	7.97	50.41%	7.52	44.83%	6.45	51.78%	165.84	78.65%

Table 1. Simulation time (in minutes) and efficiencies (%) for Newmark method.

Proc.	WSMP		MUMPS (QAMD)		MUMPS (MND)		PETSc	
1	-	-	54.99	100.00%	-	-	2205.38	100.00%
2	32.85	100.00%	27.73	99.15%	27.17	100.00%	1182.03	93.29%
4	18.12	90.55%	16.31	84.29%	15.03	90.39%	618.89	89.09%
8	11.42	71.84%	10.76	63.88%	9.56	71.05%	324.98	84.83%
16	8.01	51.21%	7.63	45.04%	6.71	50.61%	173.17	79.60%

Table 2. Simulation time (in minutes) and efficiencies (%) for Generalized α-method.

Proc.	WSMP		MUMPS (QAMD)		MUMPS (MND)		PETSc	
1	-	-	-	-	-	-	7201.76	100.00%
2	-	-	49.90	100.00%	-	-	3903.79	92.24%
4	42.29	100.00%	33.56	74.34%	30.01	100.00%	2127.92	84.61%
8	28.3	74.19%	23.08	54.05%	20.53	73.31%	1158.86	77.68%
16	22.11	47.81%	18.43	33.84%	16.25	46.31%	594.80	75.69%

Table 3. Simulation time (in minutes) and efficiencies (%) for SDIRK method.

The shortest response times were achieved with MUMPS library, together with these ordering algorithms: MND (Multilevel Nested Dissection), implemented in METIS package [9], and QAMD (Approximate Minimum Degree Ordering with Automatic Quasi Dense Row Detection). WSMP achieved excellent results as well, with similar efficiencies than MUMPS with MND. WSMP ordering is also based on MND. Simulations with 1 processor overcame the RAM memory available in the approaches employing WSMP and MUMPS with MND. Clearly, the number of non-zero elements of the coefficient matrix, after numerical factorization, in MND is superior to QAMD. Therefore, the efficiency values appearing in Tables 1 and 2 for WSMP, and MUMPS with MND, are obtained with respect to 2 processors, or with respect to 4 processors at Table 3.

Regarding PETSc libraries, best results have been achieved by means of the combination of Conjugate Gradient as iterative method with block Jacobi preconditioning, where Incomplete Cholesky factorization is also applied as subblock preconditioner. Structural coefficient matrices are usually ill-conditioning, what explains that iterative methods have been much slower than direct methods. It should be noticed that the main drawback of Block Jacobi preconditioner is that the number of iterations can rise when the number of processors is increased, what obviously has influence on the simulation times and efficiencies obtained. Anyway, direct methods just need to carry out a forward-backward substitution for each time step, what is much more efficient than solving the whole linear system as the iterative methods do.

While Newmark and Generalized-α methods offer second-order accuracy on the results, stage parameter was set to four in the SDIRK method, trying to achieve third-order accuracy. As expected, it increased dramatically the simulation times, since two linear systems (the first one composed of four right-hand size vectors) must be solved. In spite of acceleration values were not calculated in this case, with the aim to avoid the factorization of the mass matrix, memory requirements of WSMP and MUMPS with MND ordering exceeded the available RAM even with two processors.

3 Service Oriented Dynamic Structural Analysis

Web services have emerged as the standard framework in distributed system development. They provide flexible and extensible mechanisms for describing, discovering, and invoking public network services by means of XML-based protocols. Globus Toolkit 4 (GT4)[10], the latest version of the current standard in Grid middleware systems, has performed a natural evolution to Web services technology, adopting them to define its architecture and interfaces. The result is the so-called Grid services, i.e. enhanced Web services that extend their conventional functionality into the Grid domain.

In this work, we have developed and deployed under GT4 middleware a Structural Dynamic Analysis Grid Service (SDAGS) for the 3D dynamic simulation of large-scale buildings. Figure 1 exposes the Grid service architecture proposed. The diagram shows some of the principal parts involved, such as the GUI client, the SDAGS itself and the Globus-based computational infrastructure.

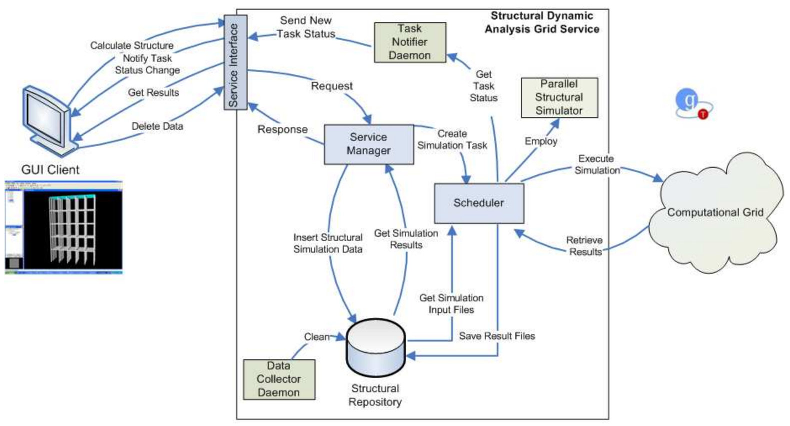

Fig. 1. Diagram of the implemented Grid service architecture

3.1 The GUI Client

The structural engineers can simulate the structures in the SDAGS thanks to an advanced graphical user interface (GUI) program. This software enables the user to perform the pre-processing phase, where the different properties are assigned to the structural members of the building (i.e. initial conditions, sections, external loads, etc.) in a user-friendly way. Using the Java 3D libraries, this highly portable application shows a 3D scene in which the user can interact with the building by means of different functionalities such as rotations, translations, zooming, selections, etc., employing the wired or solid modes of visualisation.

The GUI client interacts with the SDAGS, via its public interface, to analyse the structures. For this, the client sends, via a SOAP request, a XML file with the properties of the building to be simulated, together with different parameters related to the dynamic analysis. Then, the status of the simulation task is periodically received, and once the structure has been remotely analysed, the output data are retrieved in a SOAP message, and then deleted in the machine that runs the Grid service. Finally, the post-processing phase takes place and the results obtained are automatically mapped onto the graphical display and easily interpreted.

This GUI client incorporates a fault-tolerant procedure with the SDAGS for the data transfers. It should be clear that client and service are decoupled, what allows to recover later the analysis results in case of client, service or communication failure. In a dynamic analysis, all the result data successfully received by the client will not be sent again by the SDAGS.

3.2 Structural Dynamic Analysis Grid Service

The SDAGS is a flexible and extensible Grid service implementation that enables to remotely employ the previously mentioned HPC-based dynamic structural simulator. This service publishes a set of methods, by means of standard XML-based protocols, that are invoked via SOAP requests. On one hand, this enables to implement heterogeneous clients, developed in different programming languages and over a wide variety of platforms, to interact with the service. To include all the input and output binary simulation data in the XML messages, a hexadecimal encode schema has been employed. The SDAGS is composed of the following main components: the *Service Manager*, the *Scheduler*, the *Data Collector Daemon*, the *Parallel Structural Simulator* and the *Task Notifier Daemon.*

The *Service Manager* represents the core of the SDAGS and it is in charge of satisfying the requests from the clients. It acts as the front end, receiving the client requests as well as interconnecting all the system components. The *Task Notifier Daemon* is responsible of performing the notification process of the state changes of the tasks, thus enabling the users to permanently know the state of their simulations. The structures are analysed on the Grid resources by the *Parallel Structural Simulator*, which is able to perform efficiently and in a realistic way static and dynamic analyses.

The *Scheduler* agent executes the structural simulations in the available Grid infrastructure. Currently, we are employing several cost-effective cluster of PCs located at our research center. Firstly, the *Scheduler* involves the resource discovery to obtain a list of candidate execution machines. After that, a resource selection phase is carried out in order to select the best available computational machine for each structural simulation. Finally, the different phases related to achieve remote task execution, such as data stage and job monitor to detect failures, will be also performed by this component.

Input and output simulation data will be stored in a Structural Repository, implementing a data persistence schema and enabling the use of the system also as a Storage Service. Finally, the *Data Collector Daemon* component inspects periodically the Structural Repository and cleans the old simulation files.

3.3 The Structural Analysis Process

The implementation details of the Grid Service developed are exposed in the next paragraphs, by means of the sequence of steps to be followed to simulate a building. First, the client submits the request, sending the corresponding files that define the structure, such as its structural and geometric properties, the different external load hypotheses to be evaluated and the needed parameters to define the type of analysis.

This request is received in the SDAGS by the *Service Manager*, which processes all the input data, storing them in the Structural Repository, returns to the client a simulation identifier, which will be used in later invocations to identify the simulation, and generates the appropriate binary input file for the parallel simulator. Then, the *Service Manager* creates an execution task that contains all the required properties to execute it in the computational Grid. Next, this task is added to the *Scheduler* module, which, in a transparent way, performs the resource selection and the simulation execution management. A resource selection policy has been defined addressed to optimize the throughput and reduce the execution time of the each analysis. The simulation type, static or dynamic, the dimension of the structure and the user privileges will be values used to decide the number of processors involved in each execution.

For each simulation request, the SDAGS creates and publishes a notification item that is in charge of informing the client about its evolution. After subscribing to this item, the *Task Notifier Daemon* notifies the user any change that takes place in the analysis process. In this way, the client is perfectly aware of the status of the simulations: waiting, in execution, failed, finalized, etc. This approach dramatically reduces the overhead that would appear in the system if the clients periodically queried the service about the status of every simulation.

In a static analysis, and once the task execution has finished, the output results are automatically saved into the Structural Repository by the *Scheduler*, and the user is informed about their availability thanks to the *Task Notifier*. However, a dynamic analysis is performed by means of an iterative process that implies the generation of output data for each simulation time step. With the purpose of reducing the waiting time, the client is informed by the Task Notifer when there are enough results to be sent, thus submitting it different retrieval requests. In this way, the simulation and data retrieval phases are overlapped, what implies a clear benefit for the user who can begin to process the results before the analysis is completed.

The result retrieval procedure is performed by the *Service Manager*, which processes and analyses all the output files in order to generate a XML file that is sent to the client in a SOAP message. One of the main problems related to the use of this type of messages, being based on XML, is their size, thus introducing a communication overhead between the client and the Grid service. In our case, the solution adopted has lied in the use of a hexadecimal codification schema for including binary data, instead of inserting all of them in a text-based format. This approximation enables to reduce substantially the message dimension, which has a direct impact on the data transfer times.

An erase method that deletes all the simulation data is also available. Notwithstanding, the client is not required to invoke it, thus taking advantage of a Data Storage Service that can be employed during a certain period of time. Nevertheless, due to the fact that there are users with different privileges in the system, a component called *Data Collector Daemon* will be in charge of periodically erasing the simulation results of those lowest level clients.

Several fault tolerance levels have been implemented in the system, including the service itself, the task scheduling and execution, and the client, what guarantees that all the simulations submitted will be successfully attended. On one hand, the SDAGS implements a persistence schema that stores a description of all the tasks in course or waiting for execution, and those finalised simulations that still have results to be recovered by the client. Therefore, in case of service failure, all the non-finished tasks would be launched later, and the identifiers of those having pending results would be registered again. On the other hand, the fault tolerance level included in the *Scheduler* ensures that a failed execution will be transparently migrated to another Grid resource. Failures in the communication data between the service and the client, or the service and the computational resources, are also supported.

A robust security system has been integrated in the service, including user authorization and authentication, and privacy and integrity of results. On one hand, the user authorization and authentication capability establishes an access control to the published services, enabling to register all the actions performed by the clients. The authorization system employs a configuration file that contains all the users authorized to interact with the service. All the requests from users not registered will be directly rejected. The authentication process is implemented by means of a X.509 certificate that identifies the user. This certificate is sent to the service when the communication begins. The data privacy and integrity has been achieved using a private-public key approach. It employs the same certificate X.509 to perform the encryption and signature of all the data exchanged between the service and the client.

3.4 Interacting with the Computational Grid via GMarte

The SDAGS execute the *Parallel Structural Simulator* over a computational Grid by using the functionality of the GMarte middleware [11]. GMarte is a software abstraction layer, developed on top of the Java CoG Kit 1.2 [12], which provides an object-oriented API for the description of simulation tasks and computational resources. It provides all the required software infrastructure to perform the fault-tolerant allocation of tasks to machines based on the Globus Toolkit.

In order to achieve remote task execution, GMarte enables the user to focus on *what* should be executed, instead of messing around with all the implementation details of the underlying Grid middleware. For that, GMarte first introduces an abstraction layer over the information provided by the computational resources of a Grid infrastructure. This enables the user to access computational information of the resources, such as the number of available processors or RAM, in the same manner, regardless the underlying differences of the Grid middleware.

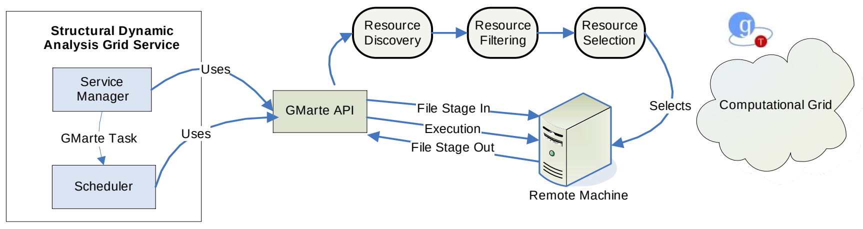

Fig. 2. Usage of GMarte within the Grid service

Figure 2 describes how GMarte fits in the service proposed. The *Service Manager*, in Figure 1, uses the GMarte API to provide the description of the computational tasks, which are assigned to a daemon *Scheduler* that waits for new tasks to be executed.

The implemented GMarte-based *Scheduler* is in charge of performing a sequence of steps in order to achieve successful execution of the tasks. This procedure involves, when the Grid service starts, the *Resource Discovery* and the *Resource Filtering* phases to obtain a list of currently available machines to host executions. Then, for each structural analysis request, the *Resource Selection* phase selects the current best computational resource to execute it. Later, all the needed input files are automatically transferred to the remote machine, before the remote parallel execution is started. When the simulation has finished, all the generated output files are moved to the machine hosting the SDAGS.

GMarte implements a multi-threaded metascheduler that enables to concurrently carry out the resource selection phase for the different simulations that have to be executed. This notably reduces the start-up time of the metascheduling procedure, when compared with other traditional single-threaded metaschedulers, what enables to notably increase the service productivity when it is concurrently used by multiple users. The metascheduling policy implemented in GMarte considers the application requirements specified by SDGAS, as well as the dynamic state of computational resources to select the most appropriate resource.

A multilevel fault-tolerance scheme is enforced to cope with the errors arising both during data transfers and remote execution. This ensures that executions will proceed as long as there are living resources in the Grid Computing infrastructure. The use of this proposed SDAGS, that uses a computational Grid for executions, enables to increase the productivity when the service is concurrently used by multiple users. In this way, preliminary results have shown that this approach is notably faster than simulations carried out in a sequential computer or in a parallel machine.

4 Multiuser Structural Case Study

In order to test the performance of the SDAGS in a multiuser environment, a structural case study composed of several simulations has been simulated on a Grid infrastructure.

The case study proposed, addressed to reproduce the Grid service availability with different clients, is composed of 30 user simulations, which must be concurrently managed. Each simulation represents the dynamic analysis of a building whose structural features (68,800 nodes and 137,390 beams) were described in section 2.

Different representative earthquakes, according to the geographical location of the building, have been applied. The accelerograms employed had an duration between 5 and 10 seconds and they include an equally-spaced ground acceleration every 0.01 seconds. Due to the accelerograms duration variability and in order to employ an homogeneous case study, the simulation time was fixed to 5 seconds using a time step of 0.01 seconds. The Newmark method was the chosen direct time integration procedure, and the Parallel Structural Simulator was configured to use the WSMP library. The output data contains information about the stresses and deformations at multiple predefined intermediate points of all the structural elements that compose the building. This was configured to be stored every 0.5 seconds. This resulted in an output data of 646 MBytes for simulation resulting in a total of 19 GBytes.

The execution of the case study was performed in a Grid infrastructure composed of computational resources which belong to our research group. It consists of 2 clusters of PCs, whose principal characteristics are detailed in Table 4. Both machines are interconnected via a local area network delivering 100 Mbits/sec. with the service host. The Globus Toolkit version 2.4 was previously installed on each machine of the Grid deployment. Due to the high volume of data to be transferred between the service and the Grid resources, we decided to rely only on our local resources, which offer bandwidth enough to cope with this problem.

Following the policy of selecting the number of processors according to the features of the structure, the service estimated a number of two processors involved in each parallel execution. This decision enabled to efficiently share the limited available computational resources, as many executions could be proceeded simultaneously.

Machine	Processors	Memory	Tasks Allocated
Kefren	20 dual Intel Pentium Xeon@2.0 Ghz	1 GByte	16
Odin	55 dual Intel Pentium Xeon@2.8 Ghz	2 GBytes	14

Table 4. Detailed machine characteristics of the Grid infrastructure.

4.1 Execution results

Table 4 shows that a similar number of simulations were allocated to each computational resource. In fact, the GMarte resource selection component implements a policy that distributes the workload on the different resources of a Grid, trying to minimise the impact in case of failure in a determined host. Clearly, resource selection is a fundamental key in the whole task allocation procedure. Fine-tuning this phase, by allocating more executions to Odin, could probably have obtained better results.

The execution of the structural case study on the proposed infrastructure required a total of 38 minutes, since the scheduling procedure started until the output data of the last simulation was retrieved to the Grid service machine. On one hand, executing all the simulations using a sequential platform, one execution after another and employing 1 PC of Odin, the faster cluster, required 566 minutes. On the other hand, using a High Performance Computing approach, assuming a typical cluster of 8 CPUs, and performing 2-processor executions on cluster Odin (4 simultaneous simulations) required a total of 105 minutes.

Therefore, the Grid Computing approach delivered an speedup of 14.89 with respect to the sequential execution and 2.76 compared to the HPC approach. Obviously, this improvement in speed depends on the amount of computational resources employed in the Grid deployment. Anyway, it is important to point out that the Grid approach introduces an overhead, both at the scheduling level (for the resource selection) and the data transfers involved in the stage in and the stage out phases.

5 Conclusions

In this paper, we have developed a Grid service oriented system, based on GT4, that enables to perform high performance and realistic 3D dynamic structural simulations of large dimension buildings on a Grid infrastructure. For that, an MPI-based structural application has been previously implemented, where 3 different direct time integration methods have been parallelised. Underlying linear systems of equations have been solved by means of WSMP, MUMPS and PETSc numerical libraries. The parallelisation strategy of the different stages that compose the parallel structural simulator has been discussed, as well as the parallel performance, in terms of speed-up and efficiency, in the dynamic analysis of a building, considering the time integration algorithms and the distinct numerical libraries employed.

Besides, the architecture of the Grid service has been described, emphasizing its design and implementation. GMarte framework has been presented as an appropriate metascheduler to carry out the remote task simulation in a Grid infrastructure. Finally, the behaviour of the Grid service has been tested when multiple clients try to analyse, at the same time, different structures, with the purpose of evaluating the needed high-throughput of the system. Simulation times corresponding to the analysis of all these buildings have been provided, comparing them with different computational approaches. From our point of view, the system presents an acceptable development level to begin to be tested by end-users.

References

1. Clough, R., Penzien, J.: Dynamics of Structures. Second edn. McGraw-Hill, Inc (1993)
2. Fung, T.: Numerical Dissipation in Time-Step Integration Algorithms for Structural Dynamic Analysis. Progress in Structural Engineering and Materials **5** (2003) 167–180
3. Wilson, E.L.: A Computer Program for the Dynamic Stress Analysis of Underground Structures. Technical Report SESM Report 68-1, Division of Structural Engineering and Structural Mechanics, University of California, Berkeley (1968)
4. Chung, J., Hulbert, G.: A Time Integration Algorithm for Structural Dynamics with Improved Numerical Dissipation: the Generalized α-Method. Journal of Applied Mechanics **60** (1993) 371–376
5. Owren, B., Simonsen, H.: Alternative Integration Methods for Problems in Structural Dynamics. Computer Methods in Applied Mechanics and Engineering **122**(1-2) (1995) 1–10
6. Gupta, A.: WSMP: Watson Sparse Matrix Package Part I - Direct Solution of Symmetric Sparse Systems. Technical Report Technical Report IBM Research Report RC 21886(98462), IBM (2000)
7. Amestoy, P., Duff, I., L'Excellent, J., Koster, J.: MUltifrontal Massively Parallel Solver (MUMPS Version 4.6.1) Users Guide. Technical report, IBM (2006)
8. Balay, S., Buschelman, K., Gropp, W., Kaushik, D., Knepley, M., Curfman-McInnes, L., Smith, B., Zhang, H.: PETSc Users Manual. Technical Report Technical Report ANL-95/11 - Revision 2.3.1, Argonne National Laboratory (2006)
9. Karypis, G., Kumar, V.: METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Technical Report Version 4.0, University of Minnesota, Department of Computer Science /Army HPC Research Center (1998)
10. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS. Volume 3779. (2005) 2–13
11. Alonso, J., Hernández, V., Moltó, G.: An Object-Oriented View of Grid Computing Technologies to Abstract Remote Task Execution. In: Proceedings of the Euromicro 2005 International Conference. (2005) 235–242
12. von Laszewski, G., Foster, I., Gawor, J., Lane, P.: A Java Commodity Grid Kit. Concurrency and Computation-Practice & Experience **13**(8-9) (2001) 645–662