Grid4Build: A High Performance Grid Framework for the 3D Analysis and Visualisation of Building Structures

José M. Alonso, Vicente Hernández, Roberto López, and Germán 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

Abstract. This paper describes Grid4Build, a high performance Grid approach that enables to perform 3D linear analysis and visualisation of building structures. The system involves three independent components: a GUI Client, a Structural Analysis Grid Service and an HPC-based structural simulator. The GUI Client is in charge to assist the user in the pre-processing and post-processing stages. The HPC-based application is a simulator that performs the 3D linear static and dynamic analysis of building structures. Finally, the Grid Service is an on-line middleware application that exposes the functionality of the Structural Simulator through Internet. The user's structural simulations are distributed through the available computational resources of a Grid Computing-based infrastructure thanks to GMarte metascheduler.

1 Introduction

Structural analysis of buildings is the process to determine the response of a structure to different prescribed applied loads. This response is usually measured by establishing the forces and displacements at any point of the structural elements.

When designing a building, security must be assured for persons and goods. Thus, accuracy in structural simulations is a key factor, where 3D realistic structural models and accurate and numerically efficient methods of analysis must be applied. In order to find the most appropriate structural design, according to distinct criteria of safety, economic limitations or construction constraints, a large amount of different configurations will have to be simulated, following a trial-error process. Each of these alternatives is defined by the structural engineer varying the size of the structural elements, the material that composes them (concrete, steel, etc.), or the external loads applied, together with any combination of them. Then, these structural alternatives will be analysed and the results will be interpreted, maybe giving place to a new iteration in this trial-error scheme.

Therefore, structural analysis can demand a huge computational power and become one of the most time consuming phases in the design cycle of a building.

High Performance Computing (HPC) techniques provide powerful numerical and programming tools in order to develop applications able to simulate, efficiently and in a realistic way, large dimension structures, in very reasonable response times. However, studios for architecture and engineering rarely own parallel platforms to execute an HPC-based application.

Fortunately, Grid Computing technology, or supercomputing on demand, emerged in the mid-90s as a solution for the computational requirements of organisations, enabling the collaborative usage of non-owned remote resources to satisfy the execution of computationally expensive tasks [1]. The aim of the Grid is to provide a coherent view of geographically distributed heterogeneous computational resources so that they act as a single, huge, powerful and self-managed computer, in a transparent way for the user. Moreover, Grid technology allows to share not only computing power, but also another kind of resources such as storage space, specialised devices, data and even software packages. In this way, users neither need to acquire nor maintain these resources in property but only to establish usage agreements.

This paper describes Grid4Build, a novel approach for combining HPC techniques, Grid Computing technology and a realistic graphical visualisation in order to provide the structural engineers with a high performance Grid framework for the 3D structural analysis of large dimension buildings. The remaining of the paper is structured as follows: First, section 2 presents the parallel structural simulator. Then, section 3 describes the Structural Analysis Grid Service. The graphical client application that interacts with the Grid Service is introduced in section 4. Finally section 5 summarises the paper.

2 The Parallel Structural Simulator

The component of Grid4Build framework in charge of carrying out the 3D structural linear analysis of buildings is a HPC-based application that takes into account all the nodes of the structure and six degrees of freedom per joint. Node condensation techniques have not been assumed. The MPI library has been used to perform the communications among the processors that participate in the different phases of the analysis. Thanks to this library and the different parallel public domain numerical libraries employed, the application is highly portable, and it can be easily migrated to a wide variety of parallel platforms.

Depending of the changing nature of the external load applied to the building, a static or dynamic structural analysis will be needed. The Stiffness Method has been parallelised to carry out the static analysis, in contrast dynamic analyses are performed by means of parallel implementations of Direct Time Integration methods and Modal Analysis techniques.

2.1 The Parallel Static Structural Analysis

In a static analysis, where the external loads (dead loads, snow load, etc.) do not change along the time, the Stiffness Method [2] employs the stiffness properties of the structural elements to form the static equilibrium equation, that represents the relationship between the nodal forces acting on the structure and its displacements:

$$KD = F. \tag{1}$$

In it, $K \in \Re^{6Nx6N}$ is the stiffness matrix derived from the assembly of the individual member stiffness matrices of all the structural elements, being N the total number of nodes in the structure, $F \in \Re^{6NxL}$ represents the force matrix, and it is computed from all the different forces applied, where L stands for the total number of load hypothesis combination. Finally $D \in \Re^{6NxL}$ are the displacements at the joints of the structure for each load combination.

Each processor involved in the parallel static structural analysis implements the following phases:

1. Obtain its group of nodes and its set of structural members assigned.
2. Generate the stiffness matrix K.
3. Generate the external load matrix F.
4. Impose the initial conditions.
5. Solve the system of linear equations for joint displacements $KD = F$.
6. Calculate the member end forces.
7. Compute the stresses and deformations at any point of the structure.

In the step 1, the data of the problem are divided into the processors following two different distribution strategies. On one hand, a domain decomposition of the whole building is carried out and each processor is assigned a group of N/p consecutive nodes, where p stands for the number of processors (node-based approach). On the other hand, the B structural members are divided into p groups composed of B/p consecutive elements and each processor is assigned one of them (element-based approach). The node-based distribution is thus employed in the computation stages 2 to 5 and the element-based approach is used in the phases 6 and 7.

The step 5, where the matrix K is sparse, symmetric and positive definite, represents the most time-consuming phase in all the analysis process. In our case, parallel direct and iterative methods can be used to solve these systems of linear equations, respectively implemented in WSMP [3], MUMPS [4], and PETSc [5] numerical libraries.

2.2 The Parallel Dynamic Structural Analysis

The second order differential equations in time that governs the motion of structural dynamic problems [6], where the applied external loads (earthquake, wind, etc.) do change along the time, can be written as follows:

$$MA(t) + CV(t) + KD(t) = F(t), \tag{2}$$

where M, C and $K \in \Re^{6Nx6N}$ are the mass, damping and stiffness matrices respectively, $F(t) \in \Re^{6NxL}$ are the applied dynamic loads, and $D(t)$, $V(t)$ and

$A(t)$ represent the unknown displacement, velocity and acceleration matrices, for each external load combination, at the joints of the structure. The initial conditions at $t = 0$ are given by $D(0) = D_0$ and $V(0) = V_0$.

Direct time integration algorithms and modal analysis are the basic techniques usually applied for solving this computationally demanding equation. Both of them provide the response of the structure along the time.

Direct Time Integration Methods. In direct time integration techniques, the equation of motion (2) is integrated using a time step-by-step numerical procedure. In general, they involve a solution of the complete set of equilibrium equations at each time increment [7]. Although it can take a significant amount of time, they have been widely employed. In our case, the following eight well-known time integration methods have been parallelised: Newmark, Wilson-θ, Central Difference, Single-step Houbolt, HHT-α, WBZ-α, Generalized-α and SDIRK. Consistent-mass matrices have been considered, a more realistic alternative than lumped (diagonal) mass matrix. On the other hand, Rayleigh damping has been employed, what means that $C = \alpha M + \beta K$.

Except from the intrinsic differences among the different parallelised integration methods, each processor involved in the linear dynamic analysis follows this general procedure:

1. Obtain its group of nodes and its set of structural members assigned.
2. Generate the stiffness (K), mass (M) and damping (C) matrices.
3. Generate the effective stiffness matrix $\hat{K}$.
4. Impose the initial conditions.
5. For each time step t=Δt, $2\Delta t$, ..., $N\Delta t$:
   (a) Evaluate the effective dynamic load vector $F_{efect}(t)$.
   (b) Solve the system of linear equations for joint displacements $\hat{K}D(t) = F_{efect}(t)$.
   (c) Compute velocities and accelerations at joints.
   (d) Calculate the member end forces.
   (e) Compute the stresses and deformations at any point of the structure.
   (f) Go to step 5, with $t = t + \Delta t$.

In a similar way to the static analysis, the N nodes and the B beams are assigned to the processors following distribution approaches based on nodes and elements. The nodal distribution is applied in the steps 2 to 5.c, whereas the element distribution is used in the steps 5.d and 5.e. Movements at the joints 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 $\hat{K}$ coefficient matrix does not change during the simulation process. Parallel direct and iterative methods implemented in WSMP, MUMPS and PETSc libraries have been used again for solving these linear systems.

The Modal Analysis or Superposition Method. This method represents the most common and effective approach for analysis of linear structural systems. Actually, Modal Analysis method is an appropriate superposition of the modal deformational shapes of the structural model. Its starting point is the analysis of the free vibration problem, in which there is no damping or external forcing in Equation 2. For a linear analysis, free vibration will be simple harmonic, and the equation governing free undamped vibration of multiple degree of freedom systems can be mathematically described as:

$$K\Phi = w^2 M\Phi. \tag{3}$$

Φ represents the natural modes of vibration (a particular pattern of vibration) and w are the natural circular frequencies of vibration. This system of n algebraic equations, being n the number of degrees of freedom considered, can be interpreted as a generalized eigenvalue problem, where up to n eigenvalues w_i^2 and n eigenvectors ϕ_i must be determined to define the natural vibration characteristics and avoid the resonance. In practice, the interest is focused on the lowest modes and the solution of the problem can be obtained with a good accuracy just employing the smallest q eigenvalues, where $q \ll n$. However, even the computation of just a few modes may be a very intense computational problem in large dimension structures.

After that, this method reduces the large set of global equilibrium equations to a q uncoupled second order differential equations, where displacements, velocities and accelerations at joints can be computed by means of a superposition of the response of each individual mode. More in detail, each processor involved in the simulation carries out the following phases:

1. Obtain its group of nodes and structural members assigned.
2. Generate the stiffness (M) and mass (M) matrices.
3. Compute natural frequencies (w) and the natural mode shapes of vibration (Φ) by solving the eigenvalue problem.
4. For each time step t=Δt, $2\Delta t$, …, $K\Delta t$:
   (a) Evaluate the effective dynamic load vector.
   (b) Solve a set of independent equations of one degree of freedom.
   (c) Compute displacement, velocity and acceleration vectors at joints.
   (d) Calculate the member end forces.
   (e) Compute the stresses and deformations at any point of the structure.
   (f) Go to step 4, with $t = t + \Delta t$.

Regarding the step 1, each processor is assigned a group of consecutive n/p degrees of freedom, being n the total number of degrees of freedom without movement constraints, and p the number of processors. This nodal partitioning is employed in the phases 2 to 4.c. The SLEPc library [8] is employed to compute the eigenvalue problem. Besides, the systems of linear equations appearing in each step of the eigensolver process will be solved by means of MUMPS or PETSc. On the other hand, routines involving Duhamel integral [6] and the eight direct time integration methods previously mentioned have been developed for solving the uncoupled equations.

3 The Structural Analysis Grid Service

Nowadays, Grid Computing has been established as the more extended framework to share and employ non-owned remote resources. In this way, the inherent computational requirements of large-scale problems, such as a dynamic analysis of high-rise buildings, makes essential the use of the Grid Computing philosophy to tackle them. The Globus Toolkit (GT), one of the most employed Grid middleware developed by the Globus Alliance, offers a set of high level services that provides an homogeneous view of the different heterogeneous resources available in the Grid infrastructure. In order to satisfy the OGSA (Open Grid Services Architecture) standard [9], this high level services are in charge of task management, data management, resource monitoring and discovering, etc. The classical batch-oriented approach offered in the GT2 release [10], was enough to offer an interactive distributed application to explode the available Grid infrastructure. However, in its latest release (GT4) [11], a Service Oriented approach has been adopted, employing Web Services as the base for defining its architecture and interfaces. As a result, the so-called Grid Services have appeared, which are Web Services used in the Grid domain, defining a standard for discovering as well as accessing to the Grid resources distributed in the network.

In this work, we have developed and deployed under the GT4 middleware a Structural Analysis Grid Service (SAGS) for the 3D static and 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 Graphical User Interface (GUI) Client, the SAGS itself and the Globus-based computational infrastructure.

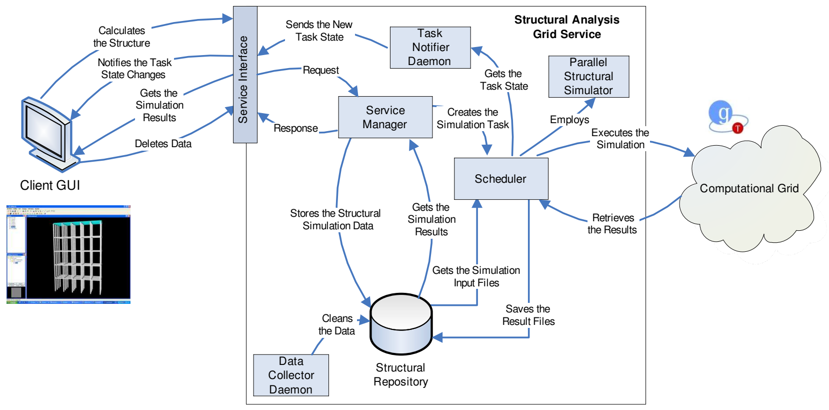

Fig. 1. The Structural Analysis Grid Service Architecture.

The *Service Manager* is the main component of the system. It receives and processes the client requests, interacting with the rest of the components and interconnecting them. The *Task Notifier Daemon* represents an interactive information system that notifies each client about the state of their simulations, thus enabling the users to permanently know the state of all of them. The *Parallel Structural Simulator* is the MPI-based application explained in the section 2 in charge of carrying out the static and dynamic analysis of large-scale buildings.

On the other hand, the structural simulation execution, in the available Grid infrastructure, is managed by the *Scheduler* module. Firstly, this component involves a resource discovery to obtain a list of candidate execution machines. Then, a resource selection takes place in order to select the best computational resource available for each structural simulation.

All the simulation data are temporarily stored in a Structural Repository, implementing a data persistence schema and also enabling the use of the system also as a Storage Service. However, the *Data Collector Daemon* component is responsible of inspecting the Structural Repository and cleaning all the output files that the user has not retrieved after a specified period of time.

3.1 The Structural Simulation Process in the SAGS

In order to explain the process carried out to perform the structural analysis of a building, a step by step sequence is detailed in the following paragraphs.

First, the client submits a simulation request, sending the corresponding files that define the structure, such as its structural and geometric properties and the different external load hypotheses to be evaluated, together with the needed parameters to specify the type of analysis. In our case, we have developed a GUI Client to define these structural properties.

The client requests are received by the *Service Manager*, which processes all the input data, storing them in the Structural Repository, and returns to the client a simulation identifier, that will be used in later invocations to identify it. The data are stored in the Structural Repository and formatted using the appropriate requirements imposed by the parallel simulator. After that, the *Service Manager* creates an execution task containing 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 increase 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 parameters used to decide the number of processors that carries out the execution.

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

In a static analysis, the output results are automatically saved into the Structural Repository by the *Scheduler* when the task execution has finished, and the user is informed about their availability thanks to the *Task Notifier*. However, the process followed in a dynamic analysis is very different, because it implies an iterative process that generates output data for each simulation time step. This feature makes possible a notably reduction of the waiting time, since the results are periodically sent from the remote resource to the SAGS machine and the Task Notifier informs the clients when there are enough results to be retrieved. 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 can be performed by the client by means of two different approaches. The first one is based on SOAP protocol [12] in which the Service Manager processes and analyses a set of output files, generating, as a result, an XML file that is sent to the clients in a SOAP message, once they have invoked the *Get Results* method. In order to reduce the message size overhead related to this XML-based format, an hexadecimal codification scheme for including binary data can also been employed, instead of inserting all of them in a text-based format. The second retrieval procedure approach is based on the GridFTP protocol [13] offered by GT4. In this alternative, the clients have direct access to their simulation storage space inside the Structural Repository and, in consequence, they can download directly the binary result files. For this purpose, a user friendly GridFTP client, that performs the transport process in a transparent way to the user, has been integrated in the GUI Client.

An erasing method that deletes all the simulation data is also available. Notwithstanding, this method invocation is optional, due to the SAGS is also offered as a Data Storage Service and a client could employ it as a particular structural repository on-line available. Nevertheless, due to the fact that there can be 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.

3.2 SAGS Fault Tolerance and Security

Several fault tolerance levels have been included in the system, involving the SAGS itself, the task scheduling and execution, and the GUI Client, guaranteeing that all the simulations successfully submitted will be attended. On one hand, the SAGS 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 SAGS failure, all the non-finished tasks would be launched later, and the identifiers of those ones 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.

A robust security system has been integrated in the SAGS, including user authorization and authentication, and privacy and integrity of results based on the Globus GSI. On one hand, the user authorization and authentication capabilities establish an control in the access 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 allowed to interact with the SAGS. 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 have 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.

3.3 Grid Task Execution

The SAGS executes the *Parallel Structural Simulator* over a computational Grid by using the functionality of the GMarte middleware [14]. GMarte is a software abstraction layer which offers 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.

The *Service Manager*, in Figure 1, uses the GMarte API to provide the description of the computational tasks, which are assigned to the *Scheduler* daemon that waits for new tasks to be allocated and 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 SAGS 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 launched. When the simulation has finished, all the generated output files are moved to the machine hosting the SAGS.

The metascheduling policy implemented in GMarte considers the application requirements specified by the SAGS, as well as the dynamic state of the computational resources to select the most appropriate remote machine. 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.

3.4 Monitoring and Discovering the SAGS

In a dynamic environment like a Grid infrastructure, it is essential to offer publication and query mechanisms to inform about the state of the Grid resources as well as their characteristics. This kind of information it is very important to perform a classical remote task execution, because it has a direct impact in the load distribution. However, in a Service Oriented environment, the services themselves could be important information sources, mainly when they could be part of a multi-service infrastructure. In this situation, the state of all the individual services, that compose the global service infrastructure, it is needed to appropriately distribute the specific Service simulations. These simulations would be directly submitted to the system by a client or, in an advanced infrastructure, by a high-level service scheduler also offered.

Therefore, two different mechanisms have been included into the SAGS in order to, on one hand, offer to the clients information about the SAGS state, i.e. the number of active simulations or clients, etc., and on the other hand, to enable the development of a high level global MSAGS (Multiple Structural Analysis Grid Services) infrastructure, where there could be multiple SAGS available. These mechanisms included are the Globus Information Service MDS (Monitoring and Discovery Services) [15] and the more advanced Information Service R-GMA (Relational Grid Monitoring Architecture) [16].

The SAGS MDS publication mechanism is based on a log file system that registers all the events produced in the Service, together with a MDS registered provider, also implemented by us and included in the SAGS. The provider is in charge of parsing these log files, generating an output text data containing the information according with the format defined in the MDS scheme. As a consequence, once the provider has been added and activated in MDS, the publication process is transparent to the SAGS, because it is indirectly registering the information in the log files and publishing it.

Regarding R-GMA, the publication mechanism is very different compared with the adopted in the MDS. In the MDS approach the information is published by a provider on query demand. As a consequence the information is not physically stored in any place. In contrast, the R-GMA approach works like a database and there are a producer which goal is to insert the information in a R-GMA server. This information is stored on server tables that would be directly accessible by means of SQL queries. Therefore, it forces to directly insert the information into de R-GMA server, changing considerably the philosophy of the MDS approach. Taking into account these features, it has been developed a Java producer agent over an available high level Java API. This way, the provider is strongly integrated in the SAGS, due to the SAGS has to notify about all it state changes in order to be published on the R-GMA server.

As a first approach, there are currently published four parameters, that are the number of active users, their names, the number of simulations active and the number of available nodes on the Grid infrastructure. Nowadays we are working on the introduction of a security layer on the information publication, because it is essential to difference the information that could be visible to a user in contraposition of the administration information.

4 The Graphical Client Application

An advanced GUI, that assists the user in the pre-processing and post-processing stages of the structural analysis, has been developed. On one hand, the pre-processing phase involves the definition of the different structural properties, such as the beam sections, the movement constraints, the external loads, and so on. On the other hand, the post-processing shows the structural analysis results, and its main goal is to check if the structure fulfils the construction standards.

Regarding the pre-processing stage, the GUI Client supports a high range of beam sections that are taken from a catalogue. Using the Java 3D graphical libraries, this application shows, in a user-friendly way, a 3D scene in which the user can interact with the structure, make rotations, translations, select elements, etc. The high level of interaction offered by the use of 3D graphics, makes easier the pre-processing phase, turning it into an intuitive and fast task.

The Client application analyses the structures by means of the SAGS, accessing it via its public interface. In this way, a SOAP request is sent with all the structural properties and the different analysis parameters. Next, the Client is subscribed to the simulation notifications and begins to receive information about the analysis process. Finally, when the Client is notified about the task finalisation, or when there are enough data to be collected, they will be retrieved in SOAP messages or via GridFTP, depending on the user requirements, and the data will be erased from the Structural Repository. The received results enable the post-processing stage, representing them on the graphical interface in order to ease its interpretation and evaluation.

The fault tolerance mechanism implemented guarantees that a failed or disconnected Client during the analysis process will imply neither the cancellation of the simulation nor the lost of the results. In fact, the results will be retrieved in a later Client execution.

5 Conclusions

This paper describes a high performance Grid Computing framework to perform a structural analysis of high-rise buildings that includes a GUI Client, a Structural Analysis Grid Service and a HPC-based structural simulator. The whole application is weakly connected and different components could be easily substituted or new ones could be incorporated.

The *GUI Client*, by means of the use of advanced 3D graphical libraries, assist the structural engineers, in a user-friendly way, in the pre-processing and post-processing stages. The *HPC-based Structural Simulator*, using the currently state-of-the-art public domain numerical libraries, carries out the static and dynamic analysis of large-scale buildings. Regarding the *Structural Analysis Grid Service*, it represents the middleware application that interconnects the GUI Client with the Structural Simulator. This way, this service-oriented application exposes its functionality over the network to the structural engineers community and by means of XML-based Service-Oriented architecture standard protocols. The simulations are distributed into a Grid-based infrastructure employing the GMarte metascheduler.

Finally, it is important to remark, the high fault tolerance level achieved in the Grid4Build framework, due to a robust multi-level scheme extensively employed and integrated in all its components. Also, the security layer included guarantees the required data privacy and integrity, with a special attention to the data transported through the network.

References

1. Foster, I., Kesselman, C.: The Grid: Blue print for a new computing infrastructure. Morgan Kaifmann (2004)
2. Livesley, R.K.: Matrix methods of structural analysis. Pergamon Press (1975)
3. Gupta, I.: WSMP: Watson Sparse Matrix Package part I - Direct solution of symmetric sparse system. Technical Report IBM Research Report RC 21886(98462), IBM (2000)
4. Amestoy, P., Duff, I., L'Excellent, J.Y., Koster, J.: MUltifrontal Massively Parallel Solver (MUMPS Version 4.3) users guide (2003)
5. Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M., Curfman-McInnes, L., Smith, B.F., Zhang, H.: PETSc users manual. Technical Report ANL-95/11 - Revision 2.3.2. Argonne National Laboratory (2006)
6. Clough R.W., Penzien, J. Dynamics of structures. Computers and Structures, Inc. (2004)
7. Fung, T.C.: Numerical dissipation in time-step integration algorithms for structural dynamic analysis. Progress in Structural Engineering and Materials **5** (2003) 167–180
8. Hernández V., Román, J.E., Tomas, A., Vidal, V.: SLEPc users manual. Scalable Library for Eigenvalue Problem Computations. Technical Report DSIC-II/24/02, Universidad Politécnica de Valencia (2006).
9. Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., Von Reich, J.: The Open Grid Services Architecture. OGSA-WG (2005)
10. Foster I., Kesselman C.: Globus: a metacomputing infrastructure toolkit. International Journal of Supercomputer Applications **11**(1997) 115–128
11. Foster,I.: Globus Toolkit Version 4. Software for service-oriented systems. IFIP International Conference on Network and Parallel Computing, LNCS **3779** (2005) 2–13
12. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web Services Architecture, W3C, Working Draft (2003)
13. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S.: Data management and transfer in High-Performance computational Grid environments. Parallel Computing Journal **28(5)** (2002) 749–771
14. Alonso, J.M., Hernández, V., Moltó, G.: GMarte: Grid Middleware to abstract remote task execution. Concurrency and Computation: Practice and Experience **18(15)** (2006) 2021–2036
15. Zhang, X., Schopf, J.: Performance Analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2. Proceedings of the International Workshop on Middleware Performance (MP 2004). 23rd International Performance Computing and Communications Workshop (IPCCC) (2004)
16. Coghlan, B., Cooke, et al.: R-GMA: A Grid Information and Monitoring System. WP3. UK e-Science All Hands Conference, Sheffield (2002)