Parallel and Grid Computing in 3D Analysis of Large Dimmension Structural Systems

José M. Alonso, Carlos de Alfonso, Gabriel García, and Vicente Hernández*

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia,
Camino de Vera s/n, 46022, Valencia, Spain
{jmalonso,calfonso,ggarcia,vhernand}@dsic.upv.es
http://www.grycap.upv.es

Abstract. In this article a parallel application for the 3D structural analysis of buildings is presented. Taking into account that solving the system of linear equations is the most time-consuming phase, several parallel public domain numerical libraries, that reflect the start-of-the-art, have been tested. A timing result comparison when analyzing two medium-sized buildings is included. Besides, a Grid-based demonstrator has been developed, integrating the parallel application and taking advantage of the different resources remotely distributed in the network. The Grid demonstrator enables the simulation of a larger number of different structural alternatives, with the purpose of finding the one which better accomplishes with all the economical limitations, safety requirements, aesthetical aspects and other criteria.

1 Introduction

Structural analysis of buildings is the process to determine the response of a structure to specified external loads or actions. This response is usually measured by obtaining tensions and displacements that take place at any point of each structural component.

This structural analysis plays a central role during the preliminary design stage of a building, when several alternatives must be considered. At this stage, construction standards obliges the designer to take into account a wide range of situations and load cases which, with a given probability, could occur during the lifetime of the structure. For each of these cases and model configurations, a structural analysis is required. A requirement of speed is imposed by the need of having comprehensive information in the preliminary design cycle. Usually, these initial designs have been based on simplified models to minimize the time and effort spent on this phase. Although, in some cases, all the designs are rejected,

* The authors wish to thank the financial support received from The Spanish Ministry of Science and Technology to develop the project GRID-IT (TIC2003-0131). This work has been partially supported by the Structural Funds of the European Regional Development Fund (ERDF).

M. Danelutto, D. Laforenza, M. Vanneschi (Eds.): Euro-Par 2004, LNCS 3149, pp. 487–496, 2004.

returning to the initial design stage, a selection among these alternatives has to be made before proceeding a detailed design phase. Now, a large number of different structural configurations have to be analysed quickly and in a realistic way to achieve the most efficient solution (the cheapest and the safest one). An iterative trial-error process is developed by the structural engineer where, varying sections or load conditions, the whole structure is analysed and the results are interpreted. At this final stage, calculations must be performed accurately, complying criteria of safety, cost limitations or construction constrains.

3D structural models deal with 6N degrees of freedom (dof), where N indicates the number of nodes considered in the structure. Current trend of erecting more complex and larger buildings is providing structural problems that can reach dimensions of about several hundreds of thousands of equations.

Therefore, structural analysis is one of the most time consuming stage in the design cycle of a building. Memory and computing power requirements to manage efficiently large systems implies the utilization of high performance computing techniques, even when Grid strategies are applicable to enable a decrease in the simulation time, a realistic analysis of larger buildings, and to test a greater number of alternatives in the preliminary and detailed design stage.

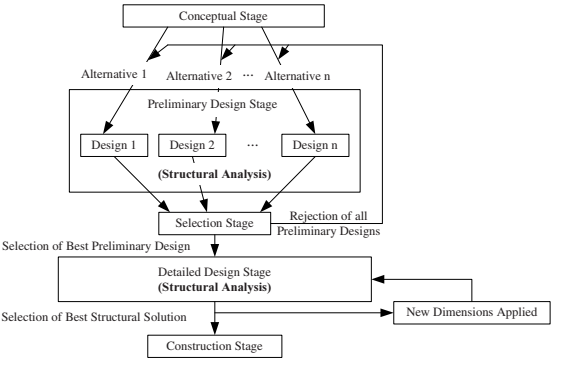

Fig. 1. Flow diagram describing the engineering design process.

The paper is structured as follows: Section 2 describes the parallelization of the structural analysis method applied. Advantages of using parallel numerical libraries with state-of-the-art capabilities, together with a brief summary of each of them, are detailed in Sect. 3. Performance of the parallel application developed is shown in Sect. 4. Section 5 presents a Grid Computing-based approach that allows performing multiple structural analyses. Finally, conclusions and further work are described in Sect. 6.

2 Parallelization of the Structural Analysis Method

The matrix methods [1] represent the most powerful design tool in structural engineering and they are appropriate to be implemented on computers. These

methods are based on the idea of replacing the actual continuous structure by a mathematical model made up from structural elements of finite size with known elastic and inertial properties that can be expressed in matrix form.

This paper introduces a MPI-based parallelization of the *stiffness method*. It employs the stiffness properties of the structural elements to form the equations that represent the relationship between nodal forces acting on the structure and its displacements. The unknown joint movements are computed by solving the equilibrium equations at the joints. At each joint, the loading conditions are specified by six force components, and the response is described by six displacement components. This method is based on the following five phases:

1. Generation of the force matrix $F \in \Re^{6N \times M}$, where M is the number of combinations of actions applied to the structure, and N represents the total number of joints in the building.
2. Generation of the stiffness matrix $K \in \Re^{6N \times 6N}$ of the structure, by assembling the stiffness matrices of the different elements that compose it. The K and F matrices are generated in parallel, where each processor will assemble a local part of them. As a result, these matrices will be partitioned among the processors, following a row-wise block-striped distribution.
3. Calculation of the joint movement. For each joint, six displacements must be calculated. The relationship between external forces F and displacements $D \in \Re^{6N \times M}$ is given by
$$KD = F. \tag{1}$$
Thus, a system of linear equations must be solved to compute the node movements. K matrix is sparse, symmetric and positive definite. Great precision is required at this point, as it determines the remaining structural phases.
4. Calculation of the member end forces. Once joint movements are known, the structural elements are divided into p groups. Each processor computes the 12 internal forces at both ends of its assigned structural elements, and the reactions at the points attached to the rigid foundation.
5. Computation of the deformations at any point of the structure. Finally, the bending moments and deformations at the predefined division points of the members are evaluated in parallel, to check that they do not exceed the established limits. Once the bending moments of an element have been worked out, its deformation is computed by means of the differential equation of an elastic curve, see (2). In this equation, $M(x)$ is the bending moment at the internal point x of a member, $y(x)$ represents its deflection function, E is its module of elasticity, and I the moment of inertia of the bending plane.
$$\frac{\partial^2 y(x)}{\partial x^2} = \frac{M(x)}{EI(x)}. \tag{2}$$

3 Parallel Modern Numerical Libraries to Solve the System of Linear Equations

Solving the large sparse symmetric system of linear equations is the crucial problem in the 3D static structural analysis, where direct or iterative methods can be

applied. Different efforts and initiatives have recently appeared encouraging the adoption and use of software tools that make it easier for programmers to write high-performance scientific applications and solving major scientific and technical problems [2]. The use of these mature tools provides important advantages, such as code portability, reusability, modularity and better performance.

In this way, multiple parallel software packages based on direct methods have emerged to solve large sparse symmetric systems: SPOOLES [3], MUMPS [4], PSPASES [5], WSMP [6], etc. WSMP and PSPASES employ the theoretically most scalable algorithms for Multifrontal Cholesky factorization and they also perform parallel ordering. Both the subtree-to-subcube mapping of the elimination tree among the processors and a two-dimensional distribution of frontal and update matrices among subgroups of processors are crucial to obtaining the highest scalability. SPOOLES employs subtree-to-subcube mapping but uses a one-dimensional distribution. MUMPS uses a two-dimensional distribution of data at only the topmost supernode of the elimination tree. However, PSPASES and WSMP apply both of them.

In addition, several parallel software packages based on iterative methods and preconditioners have been implemented: PETSc [7], Aztec [8], P-SPARSLIB [9], BlockSolve95 [10], HYPRE [11], etc. Even, some packages deal, exclusively, with parallel preconditioners for iterative methods: BPKIT [12], SPAI [13], etc. PETSc is nowadays the most popular, comprehensive and widely used.

3.1 Numerical Libraries Employed

In this work, WSMP, MUMPS, PETSc and BlockSolve numerical libraries have been applied to solve the linear system mentioned before.

Watson Sparse Matrix Package. WSMP is a high-performance, robust software package for solving large sparse symmetric and non-symmetric systems of linear equations. A node, composed of one or more processors of CPUs, communicates with other ones via message-passing (MPI). However, parallelism within multiprocessor nodes is exploited by threads or message-passing. The serial ordering heuristics used in WSMP is based on the Multilevel Nested Dissection (MND) algorithm implemented in METIS library [14]. Parallelization implemented in WSMP exploits the natural parallelism of MND ordering. The elimination tree is performed and adjusted in order to balance the numerical factorization work among the processors. The parallel symmetric numerical factorization is based on Cholesky Multifrontal algorithm. Both this phase and the parallel solution of triangular systems are guided by the supernodal elimination tree, following the same factor matrix distribution.

Portable, Extensible Toolkit for Scientific Computation. PETSc includes an expanding suite of parallel linear, nonlinear equation solvers and time integrators that may be used in application codes. It also provides many of the mechanisms needed in parallel application codes, such as parallel matrix and vector assembly routines. The combination of a Krylov subspace method (CG

and variations, GMRES, etc.) and a preconditioner (Jacobi, Block Jacobi, Incomplete Cholesky/ILU, Additive Schwarz, etc.) is the heart of the parallel iterative methods in PETSc. In addition, it allows uniform and efficient access to external direct solvers (MUMPS, SPOOLES, etc.) and preconditioners (BlockSolve95, HYPRE, ParPre, SPAI, etc.).

MUltifrontal Massively Parallel Solver. MUMPS is a MPI-based parallel package for solving linear systems of equations where the coefficient matrix is sparse and can be either unsymmetric, symmetric positive definite, or general symmetric. MUMPS uses a Multifrontal technique which is a direct method based on either the LU or the LDLT factorization of the matrix. A range of orderings to preserve sparsity is available: approximate minimum degree ordering (AMD), approximate minimum degree ordering with automatic quasi dense row detection (QAMD), an approximate minimum fill-in ordering (AMF) and the possibility of using an ordering provided by the user. PORD [15] and METIS packages are also possible choices.

BlockSolve95. It is a scalable parallel software library for solving large, sparse systems of linear equations. It contains implementations of several well-known iterative methods (CG, GMRES, SYMMLQ) and different preconditioners (Incomplete LU/Cholesky, SSOR or Block Jacobi). It uses an efficient implementation of a parallel coloring algorithm that allows for the efficient computation of matrix orderings and scalable performance of the linear solver. Just ILU and ICC preconditioners can be invoked from PETSc.

4 Experimental Results

Two medium-sized buildings, presented in Table 1, have been chosen to compare the results when using different numerical libraries.

Table 1. Features of the buildings used in test.

	Degrees of Freedom	Actions Applied
Building No. 1	Θ (60,000)	7
Building No. 2	Θ (100,000)	1

Table 2 shows the time, and efficiencies, spent on the whole structural analysis of these buildings. The linear system has been solved by means of the public domain numerical libraries WSMP, MUMPS, PETSc and BlockSolve95. WSMP has demonstrated to be the fastest solver, even more scalable than other direct solvers such as MUMPS, as expected. MUMPS library has been employed via PETSc, together with the MND ordering provided by METIS. Other orderings have also been tested in MUMPS, providing worse parallel performance.

Since coefficient matrices are very ill-conditioned, iterative methods are much slower than direct methods, although they present better efficiencies. Moreover,

the whole system need to be solved for every right hand sides (RHS) when using iterative methods, which is much less efficient than just computing the forward-backward substitution for each RHS as direct methods do. Regarding PETSc, the lowest solving times have been found for Conjugate Gradient (CG) solver, combined with Block Jacobi preconditioning, where Incomplete Cholesky (ICC) factorization has been applied as subblock preconditioner. The main disadvantage of Block Jacobi is that the number of iterations arises as the number of processors increases. Finally, the linear system has been solved applying the CG solver implemented in PETSc, and the ICC preconditioner of BlockSolve95, through the interface provided by PETSc. In order to achieve good performance, different coefficient matrix formats must be employed in PETSc, depending on the external numerical library used. Worse times have been obtained for any other combination of solver and preconditioner in PETSc.

Table 2. Structural analysis time (in seconds) and efficiencies (%) for the test battery buildings using up to 16 processors. Simulations have been run on a cluster of 20 Pentium Xeon@2GHz biprocessors with 1 Gbyte RAM, connected by a SCI network.

Building	Proc.	WSMP		MUMPS + METIS		PETSc (CG + Block Jacobi + ICC)		PETSc + Blocksolve95 (CG + ICC)	
No. 1	1	10.5	100.0 %	10.4	100.0 %	308.9	100.0 %	620.9	100.0 %
No. 1	2	5.6	93.0 %	6.8	76.8 %	164.4	94.0 %	311.1	99.8 %
No. 1	4	3.8	69.2 %	5.3	48.9 %	99.3	77.8 %	155.4	99.2 %
No. 1	8	2.9	45.0 %	4.8	27.3 %	64.7	59.7 %	88.1	88.1 %
No. 1	16	3.5	18.6 %	10.4	6.3 %	88.7	21.8 %	56.5	68.7 %
No. 2	1	10.7	100.0 %	16.5	100.0 %	60.8	100.0 %	115.0	100.0 %
No. 2	2	6.4	83.5 %	10.8	76.3 %	34.9	87.0 %	60.3	95.3 %
No. 2	4	4.7	57.5 %	8.7	47.2 %	19.3	78.7 %	32.4	88.9 %
No. 2	8	4.1	32.3 %	7.6	26.9 %	10.8	70.1 %	16.9	84.8 %
No. 2	16	3.9	16.9 %	7.1	14.6 %	6.8	56.2 %	11.1	64.4 %

5 Grid Computing-Based Structural Analysis

Grid Technologies main objective is to ease and coordinate resource sharing for collaborative problem solving in a dynamic multi institutional Virtual Organization [16] [17]. This technology aims at sharing further elements than information, by allowing direct, coordinated and secure access to different kind of computing resources, applications, communication, data, special devices, etc. In order to create the Grid infrastructure, some kind of middleware is needed. There are some well-known projects in Grid middleware, such as the open source Globus Toolkit (GT) [18] or Unicore [19]. Some private companies have developed their own middlewares, such as InnerGRID [20] or Avaki Data Grid [21]. In our case, a Grid Computing-based system, employing GT 2.4, have been implemented to analyze concurrently a group of different structural solutions at the preliminary and final design stages, making use of the parallel application developed.

5.1 Grid Structural Analyser Architecture

Figure 2 shows a conceptual view of the Grid Structural Analyser developed. Starting from a set of different structural alternatives and a group of machines distributed throughout internet, the Grid Structural Analyser performs the necessary work to analyse all of them by running the parallel application on the available resources.

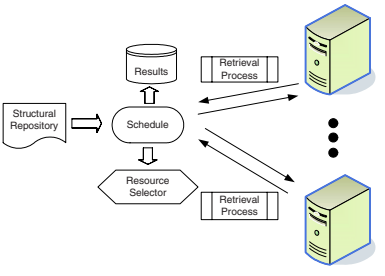

Fig. 2. Grid Structural Analyser architecture. The system consists in three main parts: the *scheduler*, which obtains designs from a structural repository; the *resource selector*, which selects an appropiate resource to run the parallel application; and the *retrieval processes*, which retrieve results files.

Scheduler. This module considers a pool of different structural configurations to be analysed and a warehouse where results will be stored. A structure is composed by a set of files which defines its geometry, properties and loads applied. All these input archives are needed by the executing host to be processed. Applying the well-known tar utility, all these files are compressed to lower the network usage time. The scheduler starts a Global Access to Secondary Storage (GASS) server which is used, by the executing hosts, to ask for the needed files.

Once the GASS server is launched, the scheduler gets a structure from the repository and asks the *resource selector* for a machine to run the parallel application. Then, the input files are staged in, via the GASS server, to the execution host. Next, the parallel application will be queued in the remote machine. Finally, the scheduler module starts a *retrieval process* which will recover results once the task has finished. If any part of the process fails, the scheduler will try to analyze it later. The scheduler finishes when all the structural alternatives have been analysed and the result files have been stored on the local host.

Resource Selector. This module uses the Monitoring and Discovery Service (MDS) utilities, provided by GT 2.4, to query the number of free processors of a computational resource in the Grid. This component handles with a list of ordered machines according to memory, computational power, workload, etc. These resources run the MDS daemon and they are enquired for their available processors. The system deals with a desirable minimum and a maximum number of processors to execute the parallel application. These numbers are thought both to increase productivity and to ensure executions with some minimum re-

quirements. The maximum number provided should make it possible parallel application to run optimally, according to structure size and library used. The minimum number will allow maintaining more resources working concurrently. The first host with enough number of free processors is chosen. The system will try to launch parallel simulations using the optimal number of processors.

With the purpose of enabling portability, all the platform-dependent optimizations should not be used. In addition, versions of numerical libraries, such as BLAS and LAPACK, tuned for a determined architecture must be avoided. Numerical software packages employed, such as PETSc, MUMPS, etc., have to be linked statically. Moreover, the parallel application should not depend on the MPI implementation on the executing host and the standard MPICH implementation should be included in a self-contained executable. As a result, two executable archives have been created, one for IA32 and the other one for IA64. Thus, the resource selector is also responsible to provide a suitable executable for each machine.

Retrieval Processes. Since the parallel application is not aware of Grid, it uses neither GT 2.4 utilities nor API to send the result files to the scheduler. A retrieval process is started for each Grid task. This process will be in charge of waiting for the Grid job to be finished and recovering the result files. Once the task has concluded, the results are compressed and sent back to the local host, where they will be saved on the appropriate directories created. All the temporary files on the executing host will be deleted. The waiting time for a job termination query must be tuned according to the resources in the Grid and their average workload. In case of the job finishes and it produces no results or any of the files expected is missed, the retrieval process will inform the scheduler and the structure will be analysed later.

5.2 Case Study

To check the performance of the Grid demonstrator, a case study of a hotel is presented. Hotels are singular buildings with a high cost of construction. The designer must be extremely careful in the design stage because in countries, like Spain, where the tourism is one of the biggest industry, an adequate design can be the differential factor for economical profitability. The design of hotel facilities requires taking into account a great number of relevant factors leading to a high number of structural solutions for the same problem.

Four different layouts were presented: two of these designs presented a reticular (bi-directional) slab solution, with large spanning of the beams, leading to geometric models with more than 600.000 dof, and two more designs with one-directional slab with approximately 330.000 dof. Three alternatives were considered for the construction material: steel, reinforced concrete and steel-concrete composite frame. In each case (layout design plus material type) eight combinations of different structural member dimensions were provided. Therefore, in the preliminary design stage, a total of 96 possible combinations should be analysed before selecting the most suitable option.

Table 3 summarizes the task distribution in the Grid. As maximum, parallel executions were limited to four proccesors, a polite policy with rest of the remote users that allows multiple concurrent simulations. Due to the memory requirements, the minimum number of processors was set to two. In the table, an entry like 9 (2 p.) indicates that nine simulations were performed with two processors each one. As table shows, the scheduler assignes dinamically the number of tasks to the resources according to their free processors and computational power. The available testbed was composed of 3 machines belonging to our research group: one cluster of 8 Pentium Xeon@2Ghz biprocessors (Kefren); another cluster of 11 Pentium III@866Mhz biprocessors (Ramses); and an Itanium II@900Mhz biprocessor workstation (Bastet).

Table 3. Distribution of the task in the testbed. The number in parentheses indicates the number of processors involved in the execution.

Machine	Simulations
Kefren	40 (4 p.)
Ramses	27 (4 p.), 20 (3 p.)
Bastet	9 (2 p.)

The execution of the whole structural study lasted for 108.3 minutes by using just one node of cluster Kefren (traditional sequential alternative). Following a high performance computing approach, 20.26 minutes were needed. Each simulation were launched with four processors in Kefren cluster, allowing two concurrent executions. Finally, the Grid Structural Analyser required 16.31 minutes for the whole case study. MUMPS library was used in all these executions.

6 Conclusions and Further Work

Firstly, a parallel application for the 3D structural analysis of buildings has been described in this paper. In order to compute the joint displacements, several parallel numerical libraries, with state-of-the-art capabilities, have been tested to solve a large sparse symmetric linear system. Direct methods, and more concretely WSMP library, have demonstrated to be the fastest solver. Besides, this parallel application has been integrated into a Globus-based Grid infrastructure. The Grid Structural Analyser developed gives the possibility to analyse in detail, concurrently, a high number of different alternatives in the preliminary and final design stages, saving time and effort, providing the designer with a powerful tool to select the best option based on quantitative measures. Small and medium-sized enterprises can now easily increase its productivity and business volume by subcontracting resource usage or employing their own office computers. Since Grid Computing enables efficient resource usage when a high coordinated computational power is demanded, the system developed can be very useful for very large and singular buildings, and moreover in a time-consuming 3D dynamic structural analysis where a building must be simulated under the influence of multiple earthquakes.

References

1. Livesley, R.: Matrix Methods of Structural Analysis. 2 edn. Pergamon Press, Oxford, UK (1975)
2. Drummond, L., Marques, O.: The Advanced Computational Testing and Simulation Toolkit (ACTS): What Can ACTS Do for You? Technical Report LBNL-50414, Lawrence Berkeley National Laboratory (2002)
3. Ashcraft, C., Pierce, D., Wah, D.K., Wu, J.: The Reference Manual for SPOOLES, Release 2.2: An Object Oriented Software Library for Solving Sparse Linear Systems of Equations. Boeing Shared Services Group. (1999)
4. Amestoy, P., Duff, I., L'Excellent, J.Y., Koster, J.: MUltifrontal Massive Parallel Solver (MUMPS Version 4.3) Users' Guide. (2003)
5. Joshi, M., Karypis, G., Kumar, V., Gupta, A., Gustavson, F.: PSPASES: Scalable Parallel Direct Solver Library for Sparse Symmetric Positive Definite Linear Systems. Users's Manual (version 1.0.3). University of Minnesota. (1999)
6. Gupta, A.: WSMP: Watson Sparse Matrix Package Part I - Direct Solution of Symmetric Sparse System. Technical Report IBM Research Report RC 21886(98462), IBM (2000)
7. Balay, S., Buschelman, K., Smith, B.F., et al.: PETSc Users Manual. Technical Report ANL-95/11 - Revision 2.1.6, Argonne National Laboratory (2003)
8. Tuminaro, R.S., Heroux, M., Hutchinson, S.A., Shadid, J.N.: Official Aztec Users's Guide. Version 2.1. Sandia National Laboratories, SAND99-8801J. (1999)
9. Saad, Y., Lo, G.C., Kuznetsov, S.: PSPARSLIB Users Manual: A Portable Library of Parallel Sparse Iterative Solvers. University of Minnesota, Morgan-Stanley (New-York), Institute of Mathematics (Novosibirsk, Russia). (1998)
10. Jones, M., Plassmann, P.: BlockSolve95 Users Manual: Scalable Library Software for the Parallel Solution of Sparse Linear Systems. ANL-95/48. (1995)
11. Falgout, R.D., et al.: HYPRE: High Performance Preconditioners. User's Manual. Version 1.6.0. Lawrence Livermore National Laboratory. (2001)
12. Chow, E., Heroux, M.A.: BPKIT: Block Preconditioning Toolkit. Reference Manual. University of Minnesota. SGI/cray Research, Inc. (1996)
13. Barnard, S.T., Grote, M.J.: A Block Version of the SPAI Preconditioner, 9th SIAM Conf. on Parall. Proc. for Sci. Comp. (1999)
14. Karypis, G., Kumar, V.: METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0. University of Minnesota. (1998)
15. Schulze, J.: Towards a Tighter Coupling of Bottom-up and Top-down Sparse Matrix Ordering Methods. BIT **41** (2001) 800–841
16. Berstis, V.: Fundamentals of Grid Computing. Technical Report IBM Redbooks Paper, IBM (2002)
17. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Intl. Journal Supercomputer Applications **15(3)** (2001)
18. Ferreira, L., Berstis, V.: Introduction to Grid Computing with Globus. Technical report, IBM (2002)
19. Unicore: Unicore Client User Guide. (2004)
20. GridSystems: Overview to InnerGrid. (2003)
21. Avaki: Avaki Data Grid 4.0 Software. (2003)