A substructure based parallel dynamic solution of large systems on homogeneous PC clusters

Semih Özmen, Tunç Bahçecioğlu, Özgür Kurç *

Department of Civil Engineering, Middle East Technical University, 06531 Ankara, Turkey

ABSTRACT

This study focuses on developing a parallel solution framework for the linear dynamic analysis of large structural models on homogeneous PC clusters. The framework consists of two separate stages where the former is preparing data for the parallel solution that involves partitioning. The latter is a fully parallel finite element analysis that utilizes substructure based solution approach with direct solvers to perform implicit integration. The linear dynamic analysis of a large scale model was performed on a homogeneous PC cluster and the number of computers was varied in order to demonstrate the performance and the efficiency of the overall solution framework. The performance of the implemented framework was also compared with the widely acknowledged parallel direct solver, MUMPS.

ARTICLE INFO

Article history:
Received 13 May 2015
Accepted 8 July 2015

Keywords:
Dynamic analysis
Parallel solution
Substructure
Workload balancing
PC clusters

1. Introduction

Parallel computing techniques have been implemented in many finite element codes in consequence of parallel computers becoming more available and affordable. Over the last thirty years, extensive research on parallel solution algorithms has been performed. For the time being, there are numerous parallel solution methods based on different strategies with iterative or direct solvers (Sotelino, 1993) but their performance may be limited depending on the type of the analysis, the parallel environment, and the structural properties of the system.

In many structural engineering design offices, the most readily available computer system for parallel computing is the network of PC's (PC cluster). Thus, the civil engineering industry will benefit significantly from a parallel solution framework that utilizes the existing computer system at these offices. This way, not only the time spent during the analysis will decrease but also the existing computer system will be utilized more efficiently without the need of purchasing any additional hardware.

This study focuses on developing an efficient parallel solution framework for the linear dynamic solution of large structural models on homogeneous PC clusters. Homogeneous PC clusters are composed of identical computers having the same computational characteristics. The parallel solution is performed by a substructure based solution method (Kurç, 2008) where the substructures are condensed by a multifrontal solver and the interface equations are solved with a block-cyclic parallel dense solver (Blackford et al., 1997).

2. Method

2.1. Overview of the framework

The solution framework is composed of two main steps: data preparation and parallel solution. The aim of the data preparation step is to equalize the nodes at each substructure to balance the distribution of data among computers for the improvement of the performance of the parallel solution. Two main tasks at this stage is preparing the graph representation of the structure and then partitioning it into substructures. After the preparation of the substructures, the parallel solution is performed by a fully parallel finite element program which is capable of performing element stiffness, mass and damping matrix computations, assembly, solution, and element force computations in parallel. When the program completes the solution, it prepares the output for post-processing. All programs were developed with C++ programming language and utilized MPICH2, message passing library (MPICH2, 2010) for parallelization.

* *Corresponding author.* Tel.: +90-312-2102447 ; E-mail address: kurc@metu.edu.tr (Ö. Kurç)

ISSN: 2149-8024 / DOI: http://dx.doi.org/10.20528/cjsmec.2015.07.021

2.2. Data preparation

In order to divide the structure into substructures, graph partitioning algorithms are utilized. The objective of many existing partitioning algorithms (Hendrickson and Kolda, 2000) is to minimize the communication volume while keeping the number of nodes balanced in each substructure. The imbalances in local assembly and condensation times (local solution) can significantly decrease the efficiency of the parallel solution because the interface solution can not initiate until all local solutions are finalized. Thus, the time spent during the local solution step is governed by the substructure with the slowest assembly and condensation time. By partitioning the substructures with equal nodes in each substructure, it was expected that the imbalance among the local solution times would decrease and as a result the time spent during the parallel solution decreased.

The data preparation algorithm first partitions the structure into '*p*' substructures where '*p*' is equal to the number of available computers after preparing the nodal graph representation of the structural model. Partitioning is performed by recursive partitioning algorithm of METIS (Karypis and Kumar, 1998), a multilevel graph partitioning library. The data is prepared for the parallel solution. The node and element definitions of the substructures are created using the structural and the selected partitioning information. During that process, the interface elements, whose nodes are on two or more substructures, are assigned to one of their adjacent substructure.

Then, each computer orders the equations of their substructures utilizing MSMD ordering algorithm (Liu, 1989) to optimize the solution. MSMD algorithm numbers the vertices by stages, in other words, the vertices belonging to stage *i* are numbered before the vertices belonging stage *i*+1. Thus, during numbering, the internal and interface vertices are assigned to stages 0 and 1, respectively.

2.3. Parallel solution

2.3.1. Overview

Implicit Newmark method (Newmark, 1959) is a numerical integration method presented by Newmark for the solution of dynamic structural problems. The equation of motion that represents a dynamic structural system can be written as

$$[M]\{\ddot{U}\}_n + [C]\{\dot{U}\}_n + \{R^{int}\}_n = \{R^{ext}\}_n\,, \tag{1}$$

where for linear systems,

$$\{R^{int}\}_n = [K]\{U\}_n\,. \tag{2}$$

Implicit Newmark method is implicit as the solution of $\{U\}_{n+1}$ depends on variables both at time *n*+1 and *n* whereas in explicit methods solution of $\{U\}_{n+1}$ depends on only variables at time *t*. The general formulation of Implicit Newmark method (Wilson, 1962) can be written as

$$[\overline{K}]\{U\}_{n+1} = \{R^{ext}\}_{n+1} + [M]\left\{\frac{1}{\beta\Delta t^2}\{U\}_n + \frac{1}{\beta\Delta t}\{\dot{U}\}_n + \left(\frac{1}{2\beta} - 1\right)\{\ddot{U}\}_n\right\} + [C]\left\{\frac{\gamma}{\beta\Delta t}\{U\}_n + \left(\frac{\gamma}{\beta} - 1\right)\{\dot{U}\}_n + \left(\frac{\gamma}{2\beta} - 1\right)\{\ddot{U}\}_n\right\}, \tag{3}$$

where

$$[\overline{K}] = \frac{1}{\beta\Delta t^2}[M] + \frac{\gamma}{\beta\Delta t}[C] + [K]\,. \tag{4}$$

As $[\overline{K}]$ involves the stiffness matrix it cannot be a diagonal matrix. Thus full factorization of $[\overline{K}]$ is needed to solve Eq. (3). For linear systems $[\overline{K}]$ matrix can be factorized once and then repeatedly solved for each time step. It can be shown (Hughes, 1983) that Implicit Newmark method is unconditionally stable when $2\beta \geq \gamma \geq 1/2$. In this study γ=1/2 and β=1/4 values are used.

2.3.2. Implementation

The parallel solution initiates by creating separate data structures at each computer from the input file prepared by the data preparation program. Then, each computer assigns degrees of freedom to its nodes. The nodes of each substructure were written into the input file according to their optimized order. Hence, during the assignment process, each node is visited one by one and the nodes' active degrees of freedom are numbered consecutively. After that, stiffness and force vectors are assembled.

Besides the global stiffness matrix and load vector, mass and damping matrices shall be assembled prior to the initialization of repetitive solution by the implicit Newmark integration algorithm. These matrices are assembled at the element level to minimize the in-core memory consumption. Stiffness and mass matrices of every finite element of the structural model are computed and assembled to global matrices. Elemental mass matrices can be either lumped or consistent according to the model needs. The damping matrix is computed as a linear combination of the elemental stiffness and elemental mass matrices according to the Rayleigh damping method (Rayleigh, 1894).

Due to the nature of linear dynamic solution, the $[\overline{K}]$ matrix is factorized once and then this system is solved repeatedly at each time step. Factorization of $[\overline{K}]$ matrix consists of two steps as; condensation and interface system factorization. The first step of the factorization is static condensation of each substructure which means that contributions from internal nodes are reflected to the interface nodes. The condensations are performed by using a parallel direct solver, MUMPS (Amestoy et al., 2000) which performs LDL^T factorization of positive definite symmetric matrices. Up to this point, neither communication nor synchronization among computers is required.

As a second step; the interface stiffness matrix is assembled, where each computer sends and receives some portion of the interface stiffness matrix. In order to utilize the parallel dense matrix solver of ScaLAPACK library (Blackford et al., 1997), the interface matrix is distributed as 2D rectangular blocks in a cyclic manner. First each computer prepares a data distribution scheme and data buffers that involves the parts of the matrix that will be sent to a particular computer. Then, the data transfer initiates in such a way that none of the computers stays idle. As the distribution of the interface stiffness matrix is finalized, it is factorized by utilizing parallel LDL^T factorization method.

Similarly, right hand side of the Eq. (3) is computed in substructure level, condensed to interface nodes and then it is re-assembled to form interface system load vector. At each time step interface system load vector is computed and then it is solved with the factorized interface system matrix and the displacements at interface level are obtained. By recovering the interface displacements back to substructure level, right hand side vector can be computed for the next time step. In this manner, algorithm continues until the last time step reached. In addition to the nodal displacements, at each time step, the forces and the stresses for each element in a substructure can be computed.

3. Results and Discussions

The efficiency of the presented framework was tested on the homogeneous PC cluster composed of eight computers with identical hardware. The cluster is composed of eight Intel Core2Quad Q9300@2.5 GHz processors and 3.23 GB RAMs. Intel Core2Quad family processors involve four processors that are working at 2.5 GHz and are able to share the memory. However, this feature of this cluster is not utilized during these tests. All computers were running Windows XP and were connected with an ordinary 1 GBit network switch.

2D square mesh is a mathematical model having 40000 quadrilateral shell elements (Fig. 1) with 40401 nodes and ~130000 equations. Dynamic loading is applied to the system at certain time steps and behaviour is monitored for 20 seconds with 0.02 seconds intervals.

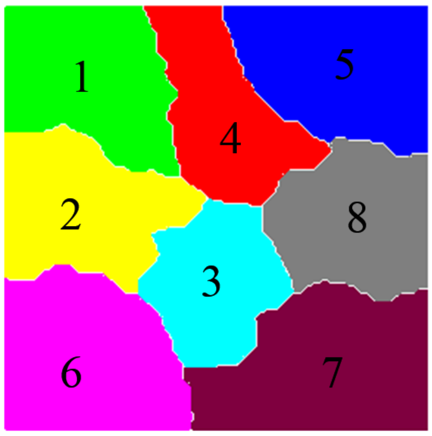

Fig. 1. 2D square mesh model with a sample partitioning among computers.

Fig. 2 presents the substructure assembly, dynamic system matrix factorization and back substitution timings for the dynamic solution of the test model by utilizing 1, 2, 4, 6 and 8 computers for FEMLib solver and MUMPS solver, respectively.

For both solution methods, substructure assembly timings are decreasing almost in the same order of the increase in number of computers utilized. For example, while assembly timing for a single computer was 94.6 seconds, this timing dropped to 46.1 seconds and 24.3 seconds for the solution with FEMLib by two and four computers, respectively. This is certainly expected due to the fact that the number of elements for each substructure is decreasing almost in the same order.

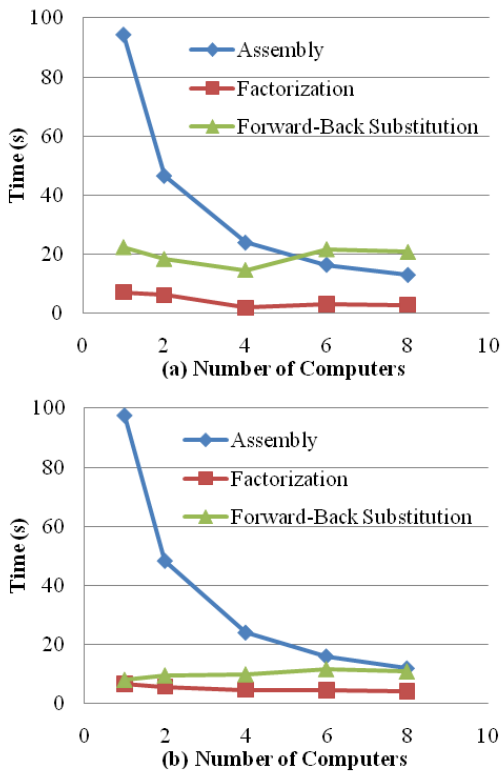

Fig. 2. Timings of main stages for FEMLib (a) and MUMPS (b) solvers.

Factorization stage, although it was one of the most time consuming stage for a static solution, was not a governing factor in dynamic solution, because it was computed only once compared to the forward and back substitutions for a thousand time-steps. Fig. 2, verifies this fact and also reveals a mild decrease in factorization stage with the increasing number of computers. When the timings of FEMLib and MUMPS solvers are compared, obviously the forward and back substitution stages were faster in the latter one. For solution by eight computers, back substitution for thousand iterations was computed by FEMLib in 20.1 seconds. However, the same computation was done by MUMPS in 10.8 seconds.

Fig. 3 illustrates the timings of different steps of dynamic system matrix factorization and those of dynamic system solution with FEMLib solver, respectively, for the various numbers of computers. For both, obviously condensation is decreasing as the number of computers utilized increase. On the contrary, interface assembly, interface factorization, load assembly and interface solution timings are gradually increasing. This behaviour is expected because the number of interface nodes is increasing with the increase in the number of substructures.

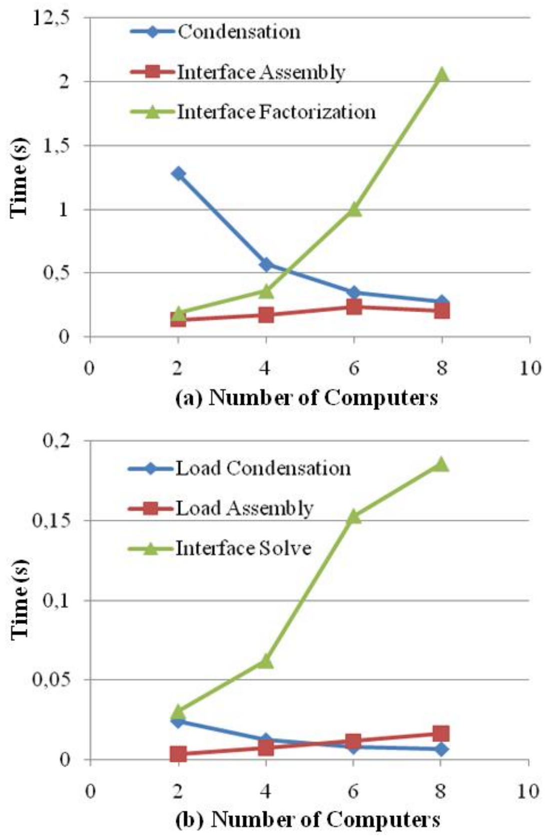

Fig. 3. Timings of factorization (a) and forward - back substitution (b) for FEMLib.

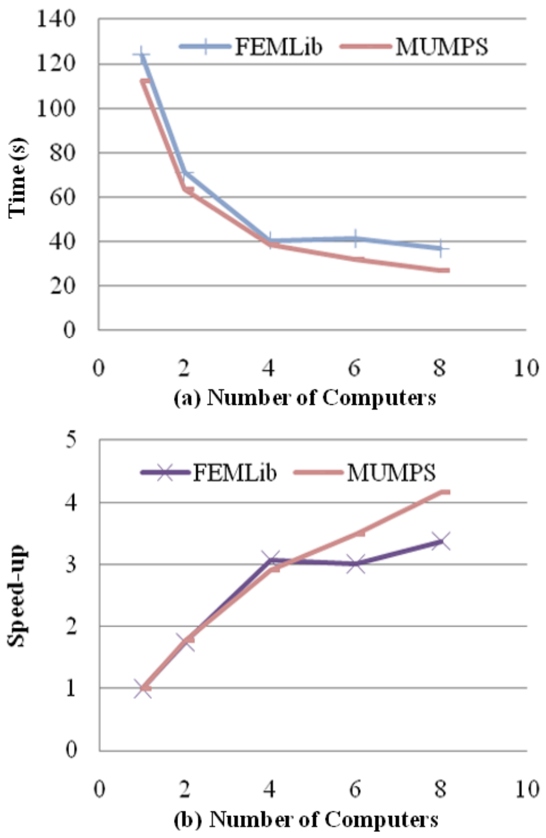

Fig. 4. Total Timings (a) and Speed-ups (b) for FEMLib and MUMPS solvers.

Assembly of the interface matrix and load vector increase as the number of computers increase because more communication is required among the computers to assemble a bigger system. These detailed timings are conclude the fact that by increasing the substructure size one can gain from condensation timings, but should lose from assembly and factorization because of the increasing interface system size.

Fig. 4 presents the total dynamic solution timings of the test model (a) and the speed-up values obtained (b) by utilizing 2, 4, 6 and 8 computers for FEMLib solver and MUMPS solver, respectively.

The dynamic solution by utilizing single computer completed in 123.1 seconds and 112.8 seconds for FEMLib and MUMPS solvers, respectively. From the previous figures it was revealed that this difference is mainly because the difference during the back substitution stage. By utilizing two, four and eight computers in parallel, the dynamic solution timings reduced to 71.0 seconds, 43.6 seconds and 35.9 seconds, respectively. Speed-up value which is the fraction of parallel solution timings to single computer solution timings are also given in Fig. 4(b). For the solution with two, four, six and eight computers, speed-up values 1.74, 3.06, 2.90 and 3.48 were obtained. Thus, although the speed-up values for two and four computers are close to theoretical values of 2.0 and 4.0, speed-up values for the six and eight computers are half of the theoretical values of 6.0 and 8.0. Because when the number of substructures is increased, interface system size is increasing.

4. Conclusions

This study presented a parallel solution framework for the linear dynamic analysis of large structures on homogeneous PC clusters. For the example problem considered, the presented method achieved almost comparative results to widely acknowledged parallel direct solver, MUMPS. Furthermore, the total solution time decreased as the number of computers increased. Thus, this framework is very efficient and can be utilized to solve large problems on cheap and ordinary PC clusters of homogeneous type.

Acknowledgements

Funding provided by TÜBİTAK under grant number 108M586 is gratefully acknowledged.

References

Amestoy PR, Du IS, Koster J (2000). MUMPS: A general purpose distributed memory sparse solver. *In Proceedings of PARA2000, 5th International Workshop on Applied Parallel Computing*, 122-131.

Blackford LS, Choi J, Cleary A, D'Azeuedo E, Demmel J, Dhillon I, Hammarling S, Henry G, Petitet A, Stanley K, Walker D, Whaley RC (1997). ScaLAPACK User's Guide, Society for Industrial and Applied Mathematics, USA.

Hendrickson B, Kolda TG (2000). Graph partitioning models for parallel computing. *Parallel Computing*, 26, 1519-1534.

Hughes TJR (1983). Analysis of transient algorithms with particular reference to stability behaviour. *Computational Methods for Transient Analysis*, T. Belytschko and T.J.R. Hughes, eds., pp. 67-155.

Karypis G, Kumar V (1998). METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0.

Kurc O (2008). Parallel Computing in Structural Engineering. VDM Verlag, Germany.

Liu JWH (1989). On the minimum degree ordering with constraints. *SIAM Journal on Scientific Computing*, 10, 1136-1145.

MPICH2 Library (2010). Message Passing Interface Standard v2.0 retrieved from *http://www-unix.mcs.anl.gov/mpi*.

Newmark NM (1959). A method of computation for structural dynamics. *ASCE Journal of the Engineering Mechanics Division*, 85, No. EM3.

Rayleigh JWS (1894). The theory of sound. 2nd ed. (reprinted by Dover Publications, New York, 1945), vol. I, pp. 102, vol. II, pp. 312.

Sotelino ED (2003). Parallel processing techniques in structural engineering applications. *ASCE Journal of Structural Engineering*, 29(12), 1698-1706.

Wilson E (1962). Dynamic response by step-by-step matrix analysis. *Proceedings, Symposium on the Use of Computers in Civil Engineering*, Labortotio Nacional de Engenharia Civil, Lisbon, Portugal.