IPSAP : A High-performance Parallel Finite Element Code for Large-scale Structural Analysis Based on Domain-wise Multifrontal Technique

Seung Jo Kim[†] and Chang Sung Lee[♣]
Department of Aerospace Engineering, Seoul National University, Korea

Jeong Ho Kim[♦], Minsu Joh[♦] and Sangsan Lee[*]
High Performance Computing and Networking Supercomputing Center, Korea

Abstract

Most of researches for large-scale parallel structural analysis have focused on iterative solution methods since direct solution methods generally have many difficulties and disadvantages for large-scale problems. However, due to the numerical robustness of direct methods that guarantees the solution to be obtained within estimated time, direct methods are much more desirable for general application of large-scale structural analysis, if the difficulties and disadvantages can be overcome. In this research, we propose the domain-wise multifrontal solver as an efficient direct solver that can overcome most of these difficulties and disadvantages. By using our own structural analysis code *IPSAP* which uses the proposed solver, we can solve the largest problem ever solved by direct solvers and can sustain 191 Gflop/s with 256 CPUs on our self-made cluster system, *Pegasus*. By implementing the block Lanczos algorithm using our solver, *IPSAP* can solve eigen problems with 7 millions of DOFs within one hour.

Introduction

In the process of implicit finite element analysis, small element stiffness matrices are assembled to build a global stiffness matrix that has sparse non-zero pattern, and the matrix equation is solved to get the solution of the problem. This procedure may be repeated depending on the characteristics of the given problem. Throughout the whole finite element analysis procedure, the equation solution part is the most time-consuming. In the case of large-scale finite element analysis, most of the computation time is spent on the equation solution and overall performance of finite element analysis code is determined by the performance of the equation solver used.

We can use just simple Gauss elimination algorithm to solve the matrix equations arising from finite element analysis, but it is too inefficient in terms of computation time and storage requirement because matrices assembled by finite element procedure are generally very sparse. Therefore, various types of equation solvers that can deal with sparse matrices for finite element analysis have been developed. They can be classified into two categories, the direct and the iterative solvers.

---

† *Professor, San 56-1, Shilliom-dong Kwanak-gue, Seoul, 151-742, Korea, E-mail: sjkim@snu.ac.kr*
♣ *Graduated Student,*
♦ *Senior Researcher, Korean Institute of Science and Technology Information, Taejon 305-333, Korea*
* *Director, Korean Institute of Science and Technology Information, Taejon 305-333, Korea*

SC'03, November 15-21, 2003, Phoenix, Arizona, USA Copyright 2003 ACM 1-58113-695-1/03/0011...$5.00

Direct solvers perform direct factorization of a global stiffness matrix with the non-zero pattern of the matrix considered and then solve the equation by substitution procedure, so the solution always can be found after required computational steps are completed unless the rank of the stiffness matrix is deficient. On the other hand, iterative solvers perform matrix-vector or vector-vector computations repeatedly until the solution converges, so iterative solvers may take too many iterations or even fail to find the solution depending on the numerical condition of the stiffness matrix. Iterative solvers may show much better performance than direct solvers for certain classes of problems, but we do not know in advance when or whether we can get the solution with iterative solvers. In the case of direct solvers, total operation counts to solve a given problem can be estimated quite exactly prior to the beginning of the computation, so we can expect when we can get the solution. The numerical robustness that guarantees the solution to be obtained within estimated time is very important, advantageous and essential property of direct solvers for general application of finite element analysis technique to wide range of problems from academic and industrial communities. For structural analysis or solid mechanics problems, the numerical robustness of direct solvers matters much more since these applications have numerically more stiff system than fluid dynamics problems, which leads to troubles in using iterative solvers. Thus, most of commercial finite element packages for structural analysis or solid mechanics problems use a direct solver as their main equation solver. Direct solvers are also much more efficient than iterative solvers for the problems with multiple right-hand side vectors such as structural analysis problems with multiple load cases or implicit time integration problems with constant stiffness matrices. The block Lanczos algorithm for eigen analysis of structures is also a good example that can benefit from direct solvers since it requires solutions for multiple right-hand side vectors.

However, in spite of all the strong points, direct solvers have many weak points for large-scale problems, especially for parallel processing of the finite element analysis procedure for very large-scale problems. Direct solvers generally have much larger storage requirements than iterative solvers and the operation counts of a direct solver increase faster with the growing problem size than that of an iterative solver. Furthermore, direct solvers are generally very difficult to parallelize compared with iterative solvers, and the parallelized versions of direct solvers also require much more communications between processors and have poor scalability. To make matters worse, direct solvers generally require an explicit construction of a global stiffness matrix to solve the problem, which results in another difficulty in the parallelization of the whole finite element analysis procedure. Because of the difficulties and the disadvantages of direct solvers as mentioned above, iterative solvers have been preferred for large-scale parallel finite element analysis and most of researches that have been performed so far focus on iterative methods [1-4]. One of the most successful research codes for large-scale parallel finite element analysis with iterative solvers may be Salinas [3] with FETI [1,2] style algorithm.

However, in spite of all the difficulties and disadvantages for large-scale problems, direct methods are still more desirable choice especially for structural analysis or solid mechanics problems. Thus, if one can develop a direct solver that can overcome most of the difficulties and the disadvantages for large-scale problems and can show comparable results in both solvable problems size and performance, it can be considered as very significant progress in large-scale parallel finite element analysis technique.

In this research, we proposed the domain-wise multifrontal solver as an efficient direct solver for large-scale parallel finite element analysis. And we showed the capability and performance of the proposed solver by using the *IPSAP* (Internet Parallel Structural Analysis Program) code based on the proposed domain-wise multifrontal solver. Most of performance tests were run on the Pegasus system, a self-made cluster supercomputer system with dual XEON processors, where *IPSAP* code sustained very high Gflops/s and could solve very large-scale finite element analysis problems.

As real applications for *IPSAP* code, the characteristics of smart structures, AFC(Active Fiber Composites) [5] was investigated through the DNS (Direct Numerical Simulation) approach [6] which requires a huge-size finite element model and tremendous computations. And the vibration analysis of an aerospace launch vehicle was also successfully carried out through *IPSAP* on the *Pegasus* cluster system.

Cluster supercomputer: *Pegasus*

In this work, the numerical experiments were executed on the *Pegasus* cluster system [7] which consists of 400 Intel Xeon 2.2/2.4/2.8 GHz processors. Each node in Pegasus runs dual Intel Xeon 2.2(or 2.4, 2.8) GHz processors on the E7500 chipset motherboard with 3GB DDR RAM and 80GB HDD. Gigabit network system is utilized for parallel computing. The Nortel Baystack 380-24T L2 switches are connected to 200 nodes and to Passport 8600 Routing switch at the core of Gigabit Ethernet network system. For cluster management and NFS(Network File System) service, the Fast Ethernet network system is also constructed. The netpipe test [8] was carried out to measure the network performance. The tuned-up network bandwidth of two nodes is about 920 Mbps and the latency time is about 21 μsec.

Domain-wise multifrontal solver

The concept of the multifrontal method was first introduced by Duff et al. [9] as a generalization of the frontal method of Irons [10]. The frontal method was originally proposed as a solution procedure for the finite element method and was designed to minimize core storage requirements. The main idea of the frontal solution method is to eliminate the variable while assembling the equations. As soon as the coefficients of an equation are completely assembled from the contributions of all relevant elements to a small dense matrix called ‘frontal matrix’, the corresponding variables can be eliminated. Therefore the complete global stiffness matrix is never formed as such, since after elimination the reduced equation is immediately transferred to secondary storage.

Duff et al. analyzed the process of the frontal method and extend the concept of ‘element’ so that it can be applied to general sparse matrix. Generalized frontal method for general sparse matrices was further improved to be able to deal with multiple frontal matrices, which results in significant reduction of total operation counts. This solution procedure is called the multifrontal method. There have been many significant progresses in the multifrontal method and it is now considered as one of the most efficient direct solvers for general sparse systems of linear equations. Therefore, it can be used efficiently for implicit finite element analysis. Some of finite element analysis packages such as ABAQUS use the multifrontal method as their main equation solver and shows very good performance compared with other packages that do not use the multifrontal algorithm. One of the most successful implementation of multifrontal algorithm is that of Gupta et al. [11]. It is also parallelized for a distributed memory system and has quite good parallel performance and scalability for a direct solver.

However, by the generalization process from the original frontal method, the multifrontal method has lost many advantages of the original frontal method for finite element analysis. The multifrontal method requires the construction of the complete global stiffness matrix like other direct solvers except the frontal method, which may be a significant demerit for large-scale parallel finite element analysis. In addition, most of available multifrontal solvers including Gupta’s do not use the out-of-core technique that is extensively used for the original frontal method to reduce the required core memory size, and thus, the required memory size of the multifrontal solvers may be too large to perform large-scale finite element analysis though it is quite small compared with other direct solvers.

The idea of the domain-wise multifrontal method proposed and established by Kim et al. [12], which extends the original frontal technique to use multiple fronts without the generalization process for general sparse matrices, is an excellent alternative to the multifrontal method for finite element analysis. Although both the domain-wise multifrontal method and the multifrontal method can be considered to be almost equivalent algebraically, the merits for finite element analysis that the original frontal method has can be kept for the domain-wise multifrontal method by the domain-wise approach. The domain-wise multifrontal method does not require the global assembly of the completed stiffness matrix and can use the out-of-core technique easily and seamlessly as the frontal method does. As far as large-scale parallel finite element analysis is concerned, the advantage of the domain-wise approach for the domain-wise multifrontal method becomes evident. Each frontal matrix of the domain-wise multifrontal method can be always mapped to a finite element domain, so we can accomplish domain-wise parallelism by decomposing the whole finite element domain into multiple subdomains, whereas the multifrontal method requires global redistribution of the global stiffness matrix after it is completely assembled from element stiffness matrices.

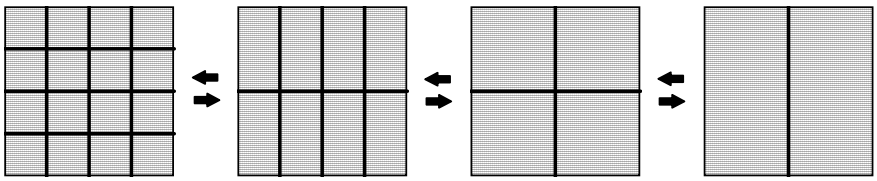


Fig. 1 Illustration of the recursive substructuring procedure

Let us explain the domain-wise multifrontal method in terms of structural analysis. By the domain-wise multifrontal method, the whole finite element domain is divided into two subdomains recursively until appropriate numbers of elements are included in each domain. Then, fully assembled degrees of freedom (DOFs) in each domain are eliminated by the procedure called static condensation, and neighboring domains are merged together in the reverse order to form a new domain. Fully assembled DOFs in each new domain are eliminated again by static condensation, and then neighboring new domains are merged together. This process is repeated recursively until all of the finite element domains are merged. Fig. 1 illustrates a simple example for a 2-D finite element domain. The domain-wise multifrontal method can be regarded as the ‘recursive substructuring’ technique since the elimination of internal or fully assembled DOFs of the given domain is called ‘substructuring’ in terms of structural analysis.

Serial implementation

Modern cache-based systems can perform dense matrix computations very efficiently, so we can get very high FLOPS(FLoating-point Operations Per Second) with dense matrix computations. The domain-wise multifrontal method can be organized to perform most of its computations in the form of dense matrix computations, so we can expect very high FLOPS as well as reduced operation counts. However, Kim et al. who established the domain-wise multirontal method focused on reduction of operation counts and did not focus on its actual performance. In this research, the performance of our domain-wise multifrontal solver is much enhanced by implementing dense matrix computations using high performance dense linear algebra libraries that are highly tuned to hardware architecture for optimized performance.

The recursive substructuring procedure, which is the core operation in the domain-wise multifrontal method, can be divided into two stages. The first stage is to construct a new frontal matrix by merging old frontal matrices from two neighboring domains, which is called the ‘extend-add’ operation by Gupta et al. [11]. The

second stage is to eliminate fully assembled DOFs from the newly formed frontal matrix equation. Most of the computing time is spent on the second stage. Let us label internal or fully assembled DOFs as $\mathbf{u}_1$ and external or surface DOFs as $\mathbf{u}_2$. Then the matrix equation of the new domain can be written as follows:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{Bmatrix} \tag{1}$$

After static condensation process, equation (1) is transformed as follows:

$$\begin{aligned} \overline{\mathbf{K}}_{22}\mathbf{u}_2 &= (\mathbf{K}_{22} - \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\mathbf{K}_{12})\mathbf{u}_2 \\ &= \mathbf{f}_2 - \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\mathbf{f}_1 = \bar{\mathbf{f}}_2 \end{aligned} \tag{2}$$

The core operation of the recursive substructuring process is to compute $\overline{\mathbf{K}}_{22}$ in equation (2). All the computations required to obtain $\overline{\mathbf{K}}_{22}$ can be implemented using level 3 BLAS[13] and LAPACK[14], which means that we can use fully optimized library routines to perform these operations so that we can take full advantage of the given hardware performance. Furthermore, since BLAS and LAPACK subroutine interface is being accepted as de facto standard, using BLAS and LAPACK interface guarantees high portability with high performance. We evaluated the performance of our domain-wise multifrontal solver by comparing the performance of *IPSAP* which uses our solver with those of most popular finite element structural analysis packages such as MSC/NASTRAN and ABAQUS.

As shown in Table 1 and Table 2, *IPSAP* shows the best performance on both EV67 system and POWER4 system. Mflops/s values in Table 2 were measured by using hpmcount [15]. On both systems, optimized BLAS and LAPACK routines developed by the hardware vendors are used. For Intel-based systems such as Pegasus, the most popular implementation of BLAS and LAPACK is ATLAS [16]. However, since GOTO [17] library shows better performance than ATLAS for wide range of matrix size on Intel Xeon 2.2GHz system as shown in Fig. 2, our domain-wise multifrontal solver uses GOTO library to obtain better performance. In Table 3, the performance of *IPSAP* is twice as fast as that of ABAQUS. *IPSAP* sustains 2.153 Gflop/s and shows 45% of CPU efficiency.

Table 1 Performance comparison on Alpha EV67 667MHz system (R_{peak}=1.3Gflops/s)

Mesh	No. of unknowns	CPU time (sec)		
		IPSAP	ABAQUS	NASTRAN
32×32×32	107,811	305	331	1,345
48×48×48	352,947	2,909	3,150	18,885

Table 2 Performance comparison on IBM POWER4 1.3GHz system (R_{peak}=5.2Gflops/s)

Mesh	No. of unknowns	CPU time(sec)		Performance (Mflops/s)	
		IPSAP	ABAQUS	IPSAP	ABAQUS
32×32×32	107,811	112	147	1,836	1,331
48×48×48	352,947	1,159	1,213	2,016	1,890

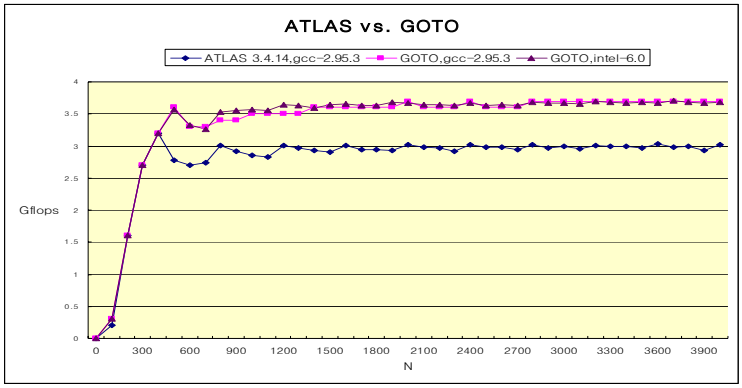


Fig. 2 Performance comparison for ATLAS and GOTO

Table 3 Performance comparison on Intel Xeon 2.4GHz system (R_{peak}=4.8 Gflops/s)

Mesh	No. of unknowns	CPU time(sec)		Performance (Mflops/s)	
		IPSAP	ABAQUS	IPSAP	ABAQUS
32×32×32	107,811	93	203	2,153	986

Parallel implementation

By using domain-wise approach, the domain-wise multifrontal solver can have the same level of domain-wise parallelism with iterative solvers based on domain decomposition method such as FETI. In other words, the domain-wise independent computations to construct an interface problem for iterative solvers based on domain decomposition method can be considered to be equivalent to the computations to build a frontal matrix equation for each domain assigned to each processor. After this step, iterative solvers perform parallel computations which requires global communications while the domain-wise multifrontal solver performs local parallel computations with neighboring processors. The localized communication pattern for the domain-wise multifrontal solver is more desirable in spite of total communication volume is larger than those of iterative solvers. Fig. 3 illustrates parallel implementation of the domain-wise multifrontal solver.

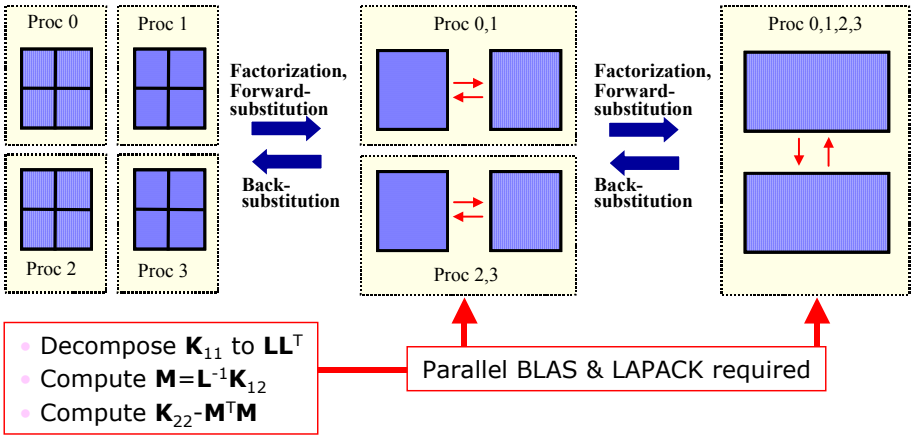


Fig. 3 Parallel implementation of the domain-wise multifrontal method

As shown in Fig. 3, after the independent computations within each processor are completed, domain-merging operation is performed across the processors. And then, dense matrix computations for substructuring are performed in parallel over related processors. To perform parallel dense matrix computations, parallel BLAS and LAPACK is required. PBLAS and ScaLAPACK [18] is known as most successful implementation of parallel BLAS and LAPACK. In order to use PBLAS and ScaLAPACK, matrices should be distributed in block-cyclic manner with constant block size. In this case, domain merging operation or parallel extend-add operation of frontal matrices requires almost all-to-all communications among related processors, so it is too inefficient to use and too complicated to implement. Thus, the parallel extend-add algorithm proposed by Gupta et al. [11] for their parallel multifrontal solver is applied to our domain-wise multifrontal solver since that algorithm is considered to be very efficient and simple to implement. The resulting communication patterns for parallel extend-add algorithm used are illustrated in Fig. 4. By this algorithm, only one-to-one communications are required in each step. However, since the block size of each distributed matrix is not constant, PBLAS and ScaLAPACK can't be used with this extend-add algorithm. Therefore, we implemented our own parallel dense linear algebra subroutines required for our domain-wise multifrontal solver. The computations of all of them are implemented using BLAS and LAPACK, and all the communications are implemented using BLACS [19] so that the performance of each developed subroutine can compete with that of the corresponding subroutine from PBLAS and ScaLAPACK.

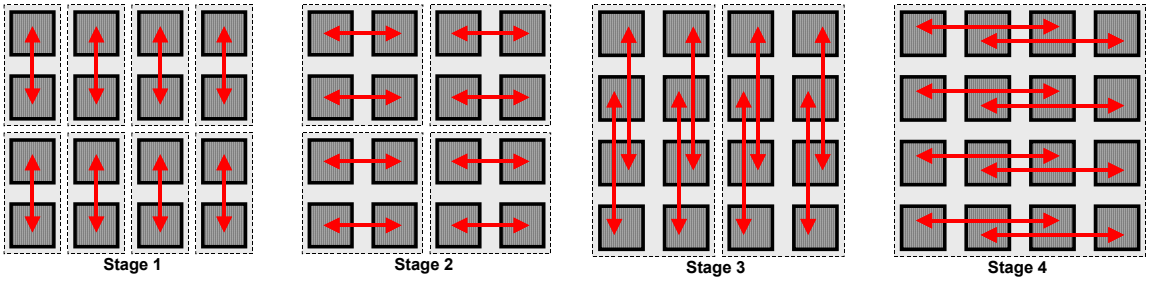
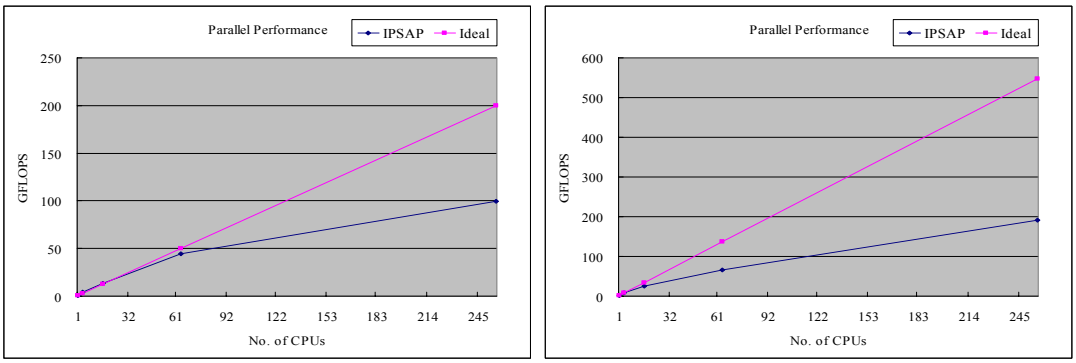


Fig. 4 Resulting communication pattern for parallel extend-add operation

In order to investigate the parallel characteristics of the parallelized domain-wise multifrontal solver, scalability tests were performed using *Pegasus* system. We performed scalability tests for both 2-dimensional mesh topology and 3-dimensional mesh topology. The size of the given problem was increased in proportion to the number of processors used. For 2-dimensional scalability test, the size of the mesh assigned to each processor was kept constant. A 128×128×1 finite element mesh was assigned to each processor. For 3-dimensional scalability test, the total operation count in each processor was kept nearly constant. Specification for 3-dimensional scalability test is shown in Table 4. The developed solver scales quite well for a direct solver as shown in Fig. 5, though the optimization of parallel dense matrix subroutines has not performed yet. According to the results on Fig. 5, 2-dimensional problem scales better while 3-dimensional problem shows better Gflops/s.



2-D mesh topology

3-D mesh topology

Fig. 5 Scalability test results

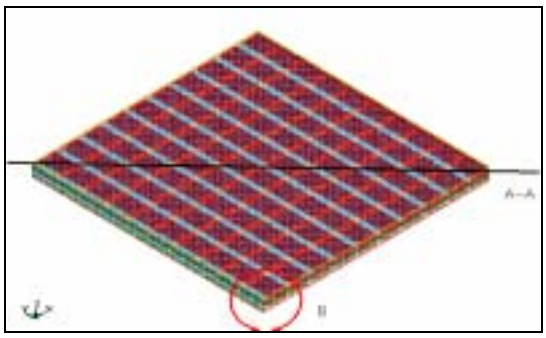
Table 4 Specification of data for 3-D scalability test and results

No. of CPUs	Mesh	Number of Unknowns	Operation counts	Performance (GFLOPS)	Scaled Speedup
1	40x40x40	201,720	8.11E+11	2.1	1.0
4	50x50x50	390,150	3.06E+12	8.2	3.8
16	64x64x64	811,200	1.33E+13	25.3	11.8
64	80x80x80	1,574,640	5.05E+13	65.5	30.6
256	100x100x100	3,060,300	1.92E+14	191.0	89.3

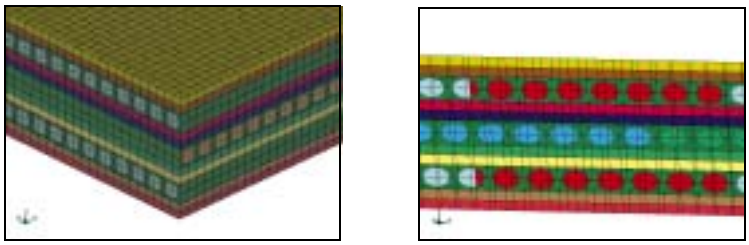
Practical Application : DNS of Active Fiber Composites (10 millions DOFs)

Active Fiber Composites (AFC) possesses desirable characteristics for smart structure applications. One major advantage of AFC is the ability to create anisotropic laminate layers useful in applications requiring off-axis or twisting motions.

AFC is naturally composed of two different constituents: piezoelectric fiber and matrix. Therefore, homogenization method, which is utilized in the analysis of laminated composite material, has been used to characterize the material properties. Using this approach, the global behaviors of the structures are predicted in an averaged sense. However, this approach has intrinsic limitations in describing the local behaviors in the level of the constituents. In particular, the failure analysis of AFC requires the knowledge of the local behaviors. Therefore, microscopic approach is necessary to predict the behaviors of AFC. In this work, a microscopic approach for the analysis of AFC was performed on the *Pegasus* system by using *IPSAP*. Piezoelectric fiber and matrix were modeled separately and a total of 3,369,600 8-node solid elements were used as shown in Fig. 6. The total number of DOFs of AFC model were 10,543,005. This approach is called Direct Numerical Simulation (DNS) of structure [5,6].

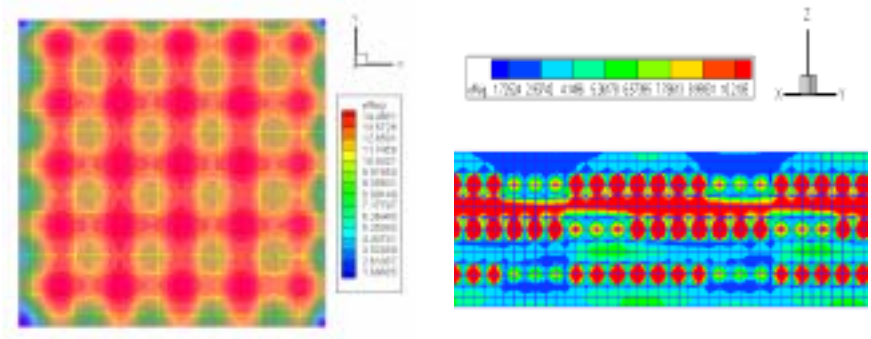


Finite element mode of AFC



Magnified view of region B Cross section of A-A

Fig. 6 The finite element model of AFC (10 M DOFs)



In-plane distribution Thickness distribution

Fig. 7 Von-Misses Effective Stress Distributions of AFC

The elapsed time was 2,139 sec. with 256 CPUs. The Von-Misses effective stress distributions are given in Fig. 7. With this approach, local stress distributions that could not be observed with homogenized models could be predicted. Moreover, inhomogeneous non-periodic effects that could not be obtained with unit cell models could be evaluated using the DNS approach.

Eigen solver based on Block Lanczos Algorithm

As mentioned before, *IPSAP* has eigen solution module based on the block Lanczos method. The Lanczos eigen solver may be considered as a powerful tool for extraction of the extreme eigenvalues and the corresponding eigenvectors of a sparse symmetric generalized eigen problem [20,21]. The block Lanczos algorithm has shown efficiency for large-scale eigen analysis [22]. In this work, BLZPACK [23] is utilized for the implementations of the block Lanczos algorithm.

In typical structural dynamic analysis, the formulated equation for vibration analysis is a generalized eigenvalue problem, with real, symmetric system matrices:

$$(\mathbf{K}-\lambda\mathbf{M})\, \mathbf{x} = \mathbf{0} \qquad (3)$$

where **K** and **M** are, respectively, the stiffness matrix and the mass matrix of the structure. The eigenvalues λ and the eigenvectors **x** correspond to the natural frequencies and their mode shapes of the structures. Shifting-and-Inverting technique is used to improve the solution through modifying the original equation to

$$\mathbf{M}(\mathbf{K}-\delta\mathbf{M})^{-1}\, \mathbf{M}\mathbf{x} = \mu\mathbf{M}\mathbf{x} \qquad (4)$$

with shifting value δ and shifted eigenvalue

Within Lanczos steps, invert process of **(K -** δ**M)**, and multiplication of **M** and vectors are required. During the actual computation, inverting process is replaced by factorization and substitution with multiple right-hand side vectors. Therefore an efficient linear equation solver can be regarded as the kernel of block Lanczos eigen solver. Especially, an efficient direct solver can maximize the advantages of block Lanczos eigen solver since multiple right-hand side vectors can be handled without unnecessary factorization. We will show the effectiveness of an efficient parallel direct solver for large-scale parallel eigen solution by applying our domain-wise multifrontal solver to block Lanczos algorithm.

For the verification and performance check of the developed eigen solution module, the vibration analysis of a plate model which has 1,638,400 8-node solid elements and 7,389,225 DOFs was conducted. The total of 10 eigenvalues and eigenvectors were extracted and the elapsed time was 3,402 sec with 256 CPUs on the *Pegasus* system. The mode shapes of 4^{th} and 7^{th} mode are given in Fig. 8

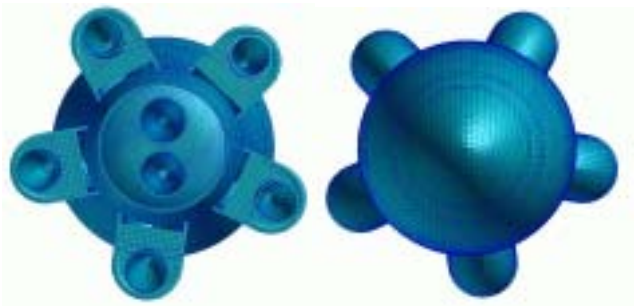
shape of the 4^{th} mode shape shape of the 7^{th} mode shape

Fig. 8 The mode shapes of plate model (7.0 M DOFs)

Practical Application : Vibration analysis of rocket model (1.2 M DOFs)

To illustrate the usefulness of *IPSAP* program for real practical applications, the full-scale vibration analysis of an aerospace launch vehicle was carried out on the *Pegasus* system. The ATLAS V500 launch vehicle was modeled by using 255,550 solid finite elements as shown in Fig. 9. The total number of DOFs was 1,201,511.

ATLAS V500 Launch Vehicle



Bottom view Top view

Fig. 9 The finite element mesh of *ATLAS V500* Launch Vehicle

The total of 20 eigenvalues and eigenvectors were extracted by the *IPSAP* program and 64 processors were utilized. The elapsed time was 584 sec. including pre/post-processing time for mesh data input and result data output, and mesh partitioning time. The elapsed time of Lanczos steps was 215 sec. including factorization, 20 substitutions and 21 multiplications of **M** and Ritz vectors. The elapsed time of factorization, substitutions and multiplications of **M** and *Ritz* vectors were 89.0 sec., 80.5 sec. and 45.5 sec., respectively. The broadcasting communication time of *Ritz* vectors was included int the elapsed time of multiplications of **M** and *Ritz* vectors. The mode shapes of 7^{th} and 10^{th} mode are shown in Fig. 10.

The Salinas program, which is based on FETI-DP [1,2] iterative solver, shows good parallel scalability. The program has shown good performance of less than 10 minutes to solve a dozen of eigen modes of a million-DOF structural model using 3,000 processors [3]. However, by applying the parallel direct solver we developed to the block Lanczos algorithm, we were able to achieve comparable or better performance on much smaller scale system than the Salinas program used. It would not be possible without the direct parallel solver, the domain-wise multifrontal solver.

In a practical point of view, more than thousands of processors may not be available to almost companies and academic research groups. Therefore, better performance on dozens or hundreds of processors of a cost-effective cluster system may be much more meaningful and useful for real applications than better scalability on thousands of processors.

shape of the 7th mode vibration mode shape of the 10th mode vibration mode

Fig. 10 The mode shapes of ATLAS V500 Launch Vehicle

Concluding Remarks and Future work

In this work, the capability and performance of *IPSAP* (Internet Parallel Structural Analysis Program) based on the domain-wise multifrontal solver were provided. The serial and parallel performance of *IPSAP* were investigated and compared with the available performance of the commercial FE packages such as ABAQUS and NASTRAN on several hardware platforms. On the Xeon 2.4 GHz system, *IPSAP* achieved 2.1 Gflop/s that was twice as fast as that of 0.986 Gflop/s of ABAQUS. *IPSAP* sustained 191 Gflop/s with 256 CPUs on our self-made *Pegasus* cluster system and computed a total of 10 eigenvalues and eigenvectors for the seven million DOF finite element model in less than one hour. The local stress distribution of smart structures, AFC was investigated through the DNS approach. And the vibration analysis of an aerospace launch vehicle was also successfully carried out through *IPSAP* on the *Pegasus* cluster system.

By the results mentioned above, we showed that we could make a direct solver work for large-scale parallel finite element analysis by proposing and implementing the domain-wise multifrontal solver which had the scalability in terms of storage as well as performance. This is considered to be very significant progress in large-scale parallel finite element analysis technique since it means that now we can benefit from the merits of direct solvers, including the numerical robustness which is an essential property for general application of finite element analysis technique to wide range of problems from academic and industrial communities, even when we run large-scale parallel finite element analysis.

References

[1] C. Farhat, M. Lesoinne and K. Pierson. A scalable dual-primal domain decomposition method. *Numer. Lin. Alg. Appl.* 7, 687-714 (2000).

[2] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson and D. Rixen. FETI-DP: a dual-primal unified FETI method - part I: a faster alternative to the two-level FETI method. *Internat. J. Numer. Meths. Engrg.* 50, 1523-1544 (2001).
[3] Manoj Bhardwaj, Kendall Pierson, Garth Reese, Tim Walsh, David Day, Ken Alvin and James Peery, Charbel Farhat and Michel Lesoinne, "Salinas: A Scalable Software for High-Performance Structural and Solid Mechanics Simulations," Proceedings of the IEEE/ACM SC2002 Conference, Baltimore, Maryland, USA, 16-22 November, 2002
[4] P. LeTalle, "Domain-decomposition methods in computational mechanics," *Computational Mechanics Advances* 1, 121-220 (1994)
[5] Seung Jo Kim, Jun Seok Hwang, Seung Hoon Paik, "Direct Numerical Simulation of Active Fiber Composites", SPIE's 10th Annual Symposium on Smart Structures and Material, 2-6 March 2003 Town and Country Resort & Convention Center, San Diego, CA USA
[6] Seung Jo Kim, Chang Sung Lee, Hea Jin Yeo, Jeong Ho Kim, and Jin Yeon Cho, "Direct Numerical Simulation of Composite Strucutres", *Journal of COMPOSITE MATERIALS*, Vol. 36, No. 24/2002t-2
[7] the 20th top500 list, http://www.top500.org/lists/2002/06
[8] http://www.scl.ameslab.gov/netpipe/
[9] S. Duff and J. K. Reid, 'The multifrontal solution of indefinite sparse symmetric linear equations,' *ACM Trans. Math. Software*, 9, 302-325 (1973)
[10] B. M. Irons, 'A frontal solution program for finite element analysis,' *International Journal for Numerical Methods in Engineering*, 2, 5-32 (1970)
[11] Gupta, A., Karypis, G., and Kumar, V., "Highly Scalable Parallel Algorithms for Sparse Cholesky Factorization on a Hypercube," *Parallel Computing*, Vol. 10, 1989, pp. 287–298.
[12] J.H. Kim and S.J. Kim, "A Multifrontal Solver Combined with Graph Partitioners", *AIAA Journal*,Vol 38, No.8, August 1999, pp.964-970.
[13] http://www.netlib.org/blas/index.html
[14] http://www.netlib.org/lapack/index.html
[15] http://www.alphaworks.ibm.com/tech/hpmtoolkit
[16] http://www.netlib.org/atlas/index.html
[17] http://www.cs.utexas.edu/users/flame/goto/#overview
[18] http://www.netlib.org/scalapack/index.html
[19] http://www.netlib.org/blacs/index.html
[20] Parlett, B. N., and B. Nour-Omid, "Toward a Black Box Lanczos Program," *Computer Physics Communications*, vol.53, pp. 169–179, 1989
[21] Simon, H. D., "The Lanczos Algorithm with Partial Reorthogonalization," *Mathematics of Computation*, vol.42, pp. 115–142, 1984.
[22] Grimes, R. G., J. G. Lewis, and H. D. Simon, "A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Generalized Eigenproblems," *SIAM Journal on Matrix Analysis and Applications*, vol.15, pp. 228–272, 1994.
[23] O. A. Marques, "BLZpack User's Guide", http://www.nersc.gov/~osni/#Software