Linear Static Structural and Vibration Analysis on High-Performance Computers

Majdi Baddourah and Olaf O. Storaasli and Susan Bostic
Computational Mechanics Branch
NASA Langley Research Center
Hampton, VA 23665-5225
(804)-864-2927

Abstract

Parallel computers offer the opportunity to significantly reduce the computation time necessary to analyze large-scale aerospace structures. This paper presents algorithms developed for and implemented on a massively-parallel computers hereafter referred to as Scalable High Performance Computers (SHPC) for the most computationally intensive tasks involved in structural analysis, namely, generation and assembly of system matrices, solution of systems of equations and calculation of the eigenvalues and eigenvectors. Results on SHPC are presented for large-scale structural problems (i.e. Models of high speed civil transport).

The goal of this research is to develop new efficient technique which extend structural analysis to SHPC and make large-scale structural analyses tractable.

1. Introduction

The finite element method is the most widely used algorithm to analyze large-scale aerospace, automotive, marine and building structures. By far, the bulk of calculations in structural analysis codes (exceeding 90% for large-scale analysis models) is associated with the generation and assembly of the global stiffness matrix, [K], and the solution for displacements, u, of [K] {u} = {p}, where p is the applied load. Initial attempts to implement finite element methods on SHPC, using the traditional element-based approach, have resulted in severe interprocessor communication/synchronization bottlenecks. Attempts to eliminate these bottlenecks such element-based codes to eliminate these bottlenecks have not yet been fully successful. A new nodal-based algorithm to generate and assemble finite elements[1] eliminates interprocessor communication on SHPC. This algorithm, which generates and assembles global stiffness (or mass) matrices simultaneously on multiple processors, is described in section 2 with examples given in section 5. The global stiffness matrix is generated and stored in a distributed manner on multiple processors so an equation solver designed for SHPC can be used. Sections 3 and 4 describe an accurate and efficient Gauss elimination equation solver for static and dynamic analysis on SHPC.

2. Generation and Assembly of Systems of Equations

Traditional methods to generate a system of equations are element-based. In this approach, each element is generated and then assembled into a global stiffness matrix. However, the algorithm presented in this paper operates on nodes as opposed to elements. In this "nodal approach", the contribution of each element at a node is generated and then assembled into the global stiffness matrix. Each processor is assigned a node, or many nodes, in a wrapped fashion, (i.e. node one is assigned to processor one, node two is assigned to processor two and so on). Each processor computes only the contribution of

each element at its assigned node, and assembles the results into the global stiffness matrix. Using this approach, no interprocessor communication is required. For simplicity, the same element stiffness matrices are computed repeatedly by different processors as they are needed. The element matrices may be stored in memory or on a disk and subsequently retrieved. On existing SHPC, the computation time to assemble element stiffness matrices is so negligible compared to the time required to communicate data between processors which makes the nodal method an excellent choice.

The amount of interprocessor communication required by the traditional element-based approach is qualitatively compared to that of the new nodal-based approach in Table 1. for the three tasks which dominate finite element analysis on SHPC.

Task	Interprocessor Communicaion	
	Element approach	Nodal approach
Element generation	None	None
Matrix assembly	Significant	None
Matrix distribution	Significant	None

Table 1: Interprocessor Communication Eliminated

Although both approaches require no interprocessor communication for element generation, the element-based approach requires additional memory to store element information associated with each neighboring element. Furthermore it requires significant communication and synchronization between the processors. No such communication or synchronization is required by the nodal-based approach. In addition, only partial element stiffness matrices are actually required for the nodal-based approach for matrix assembly and distribution. Achieving parallelism with no the communication overhead makes the nodal method ideally suited for SHPC.

3. Solution of system of equations

The solution of systems of equations that have been generated and assembled by the nodal-based algorithm is the most computationally intensive part of the finite element analysis. An equation solver based on Gauss elimination has been developed and compared with an iterative sovler; Preconditioned Conjugate Gradient (PCG) with diagonal preconditiong. Both solvers were implemented and performance compared on an Intel i860 Gamma supercomputer. The solution times are compared for both methods for the Mach 2.4 High Speed Civil Transport (HSCT) Model (see Fig. 1).

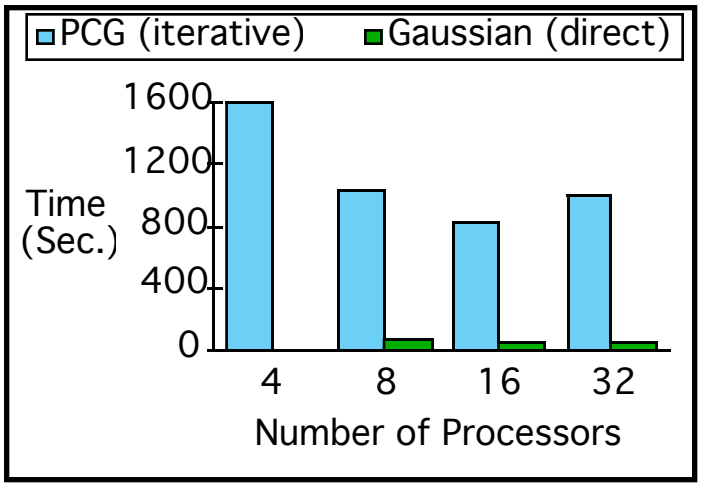


Fig. 1. Solution time for the HSCT Mach 2.4 using Iterative (Preconditioned Conjugate Gradient) and Direct (Gaussian Elimination) Methods

The equation solution time for Mach 2.4 HSCT problem using PCG was 850 seconds on 16 processors. However, the same problem took 54 second using Gauss elimination. For other structural models, the PCG method failed to converge, a drawback for iterative solution technique. Thus, after numerous comparisons, the speed and reliability of direct methods was found to be preferable for structures applications. The following paragraphs describe the variable band Gauss elimination equation solver developed.

The Gauss solver was developed to exploits SAXPY operations. Timing studies show that to achieve equal load balancing and maximum performance, blocks of six or more equations should be assigned to each processor in a wrapped manner. Since the generated system of equations is distributed on each processor, there is no need to redistribute the matrix.

The variable band storage scheme was selected over the skyline storage scheme to eliminate the matrix distribution time (see Table 1). The slight increase in storage required by the variable band scheme over the skyline storage scheme does not significantly increase the computation time.

Two communication methods were compared: binary tree broadcast and ring communication. In both methods, the solution procedure sends the first matrix row to all processors, which they use the to update their own rows. This is accomplished on the Intel i860 by using the *csend* command with a "-1" (broadcast) option. This option broadcasts coefficients of the first row to all processors in a binary tree fashion. Since the speed for this broadcast scheme was found to be slow, an alternative ring communication scheme was developed. Using ring communication, the coefficients of the first row are sent using the *csend* command by specifying the receiving processor number. In this case, the coefficients are sent to only one processor which is then free to perform computation. The coefficients are then sent to the next processor, and so on as in a "bucket brigade". Details of the comparison of the two broadcast methods is given in section 5.2.

4. Vibration Analysis

To determine the dynamic response of a structure, a free vibration analysis is carried out to find the lowest natural frequencies (or eigenvalues) and their associated mode shapes. An eigensolver, based on the Lanczos method, has been compared to other

widely-used methods and shown to be efficient and accurate when only a few eigenvalues of a large system are required. The most computationally intensive steps in the Lanczos method are factoring the global stiffness matrix and the forward/backward equation solution steps. The Gauss equation solver was implemented within the Lanczos method to improve the efficiency of these computational steps.. The Lanczos algorithm has been implemented on various computer architectures, including parallel computers with shared-memory, vector computers and parallel/vector computers. Results of the Lanczos eigensolver adapted for SHPC and run on the Touchstone DELTA system are presented in Section 5.3.

5. Results

This section contains results of three SHPC:

1. Nodal generation and assembly of stiffness matrices
2. Solution of matrix equations systems
3. Vibration analysis

To evaluate these parallel methods, results were obtained for a variety of structural models including a Control-Structures interaction (CSI) geostationary platform(Fig 2), Mach 2.4 (fig 3), and 3.0 (Fig 5) versions of a high-speed civil transport(HSCT). The important characteristics for the analysis are shown on the Figures, and more details may be found in reference 2.

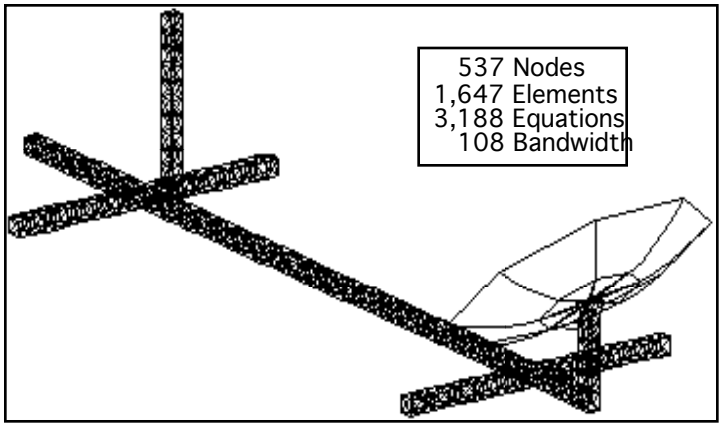


Fig. 2. CSI Structural Model of Geostationary Platform

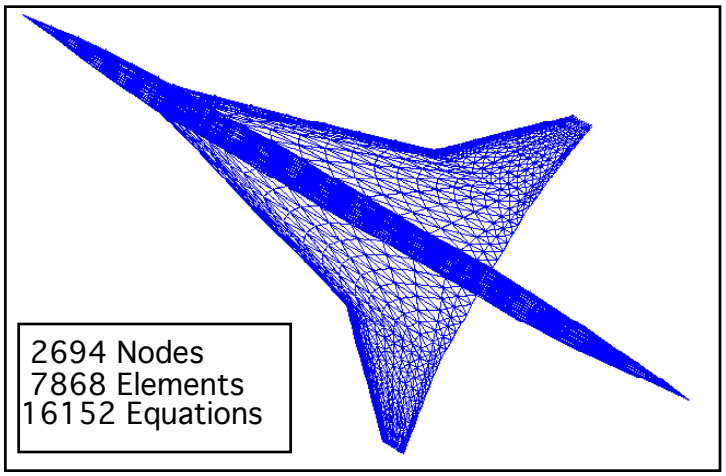


Fig. 3. Mach 2.4 High-Speed Civil Transport Finite Element Model

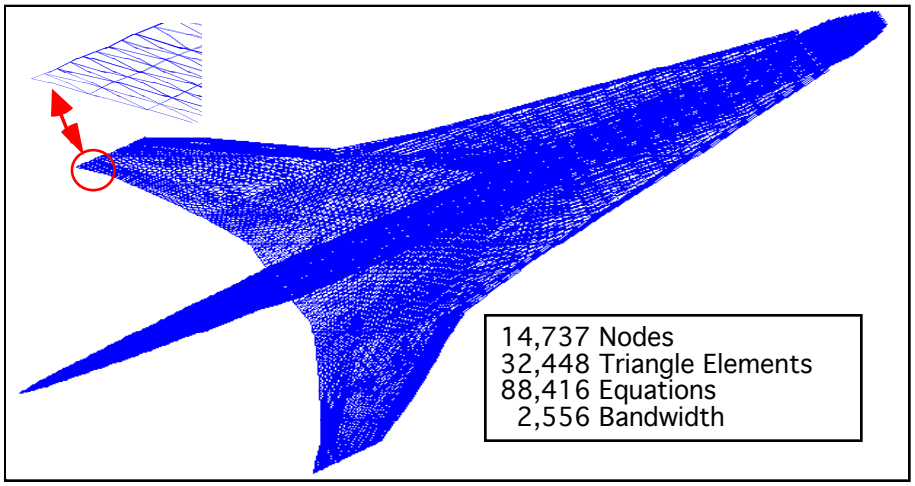


Fig. 4. High-Speed Civil Transport Structural Model

Detailed structural analyses were performed for the CSI and HSCT models on a Cray Y-MP and Intel i/860 Gamma and Delta supercomputers.

The 8 processors Cray Y-MP contained 256 megawords of memory and a 512 megaword solid state disk. The communication rate on the Cray Y-MP is fast (memory-access speed). However, care was taken to minimize the synchronization time between the processors as they simultaneously update the shared memory. Although communication time is minimal, synchronization time may be critical and was minimized to achieve an efficient equation solver. The Cray Y-MP is not considered a SHPC.

The Intel Gamma i/860 consists of 128 processors each with peak performance of 60 MFLOPS and 8 million bytes of memory (8MB). The Intel Delta SHPC contains 576 processors with 16 MB of memory arranged in a two-dimensional mesh with a peak communication rate between two adjacent processors of 12 megabytes/sec. The inter-processor communication on the Intel i/860 is performed via message passing. The inter-processor communications bandwidth and the communications software limit the communication speed that can be achieved. The communications bandwidth is recognized as critical to maximizing total computational performance and has been

undergoing significant improvement(i.e. 2.7 MB/sec on the Intel Gamma, 12.2 MB/sec on the Intel Delta, and 200 MB/sec on the Intel Paragon).

5.1 Nodal Generation and Assembly

The nodal algorithm to generate and assemble the stiffness matrix for the Mach 2.4 HSCT structural model was run on from 16 to 512 Delta processors is shown in Fig. 5. Since no communication is involved., a near perfect speedup is obtained as the number of processors increases. The solution time on 512 Delta processors is clearly faster than similar results obtained on the Cray Y-MP (left of Fig 5).

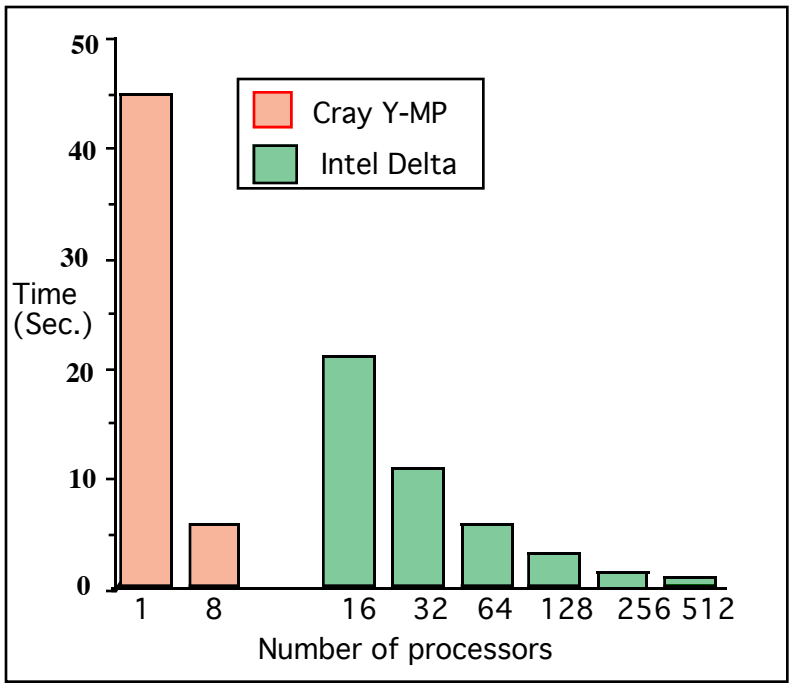


Fig. 5. Node-based Matrix assembly time for Mach 2.4 HSCT

The generation and assembly of the stiffness matrix for the Mach 3.0 HSCT resulted in a global stiffness matrix with 88,416 unknown displacements (equations) having a maximum bandwidth of 2,556. This bandwidth was reduced using a Reverse Cuthill McGee node-reordering technique from an initial bandwidth of 78,000. The HSCT application required 256 processors on the Intel Delta SHPC since the stiffness matrix and geometric data required 3.328 Gigabyte of memory (13 MB/processor). The computation total time to generate and assemble the global stiffness matrix for this HSCT model on both the Cray and Delta supercomputers is compared in Fig. 6.

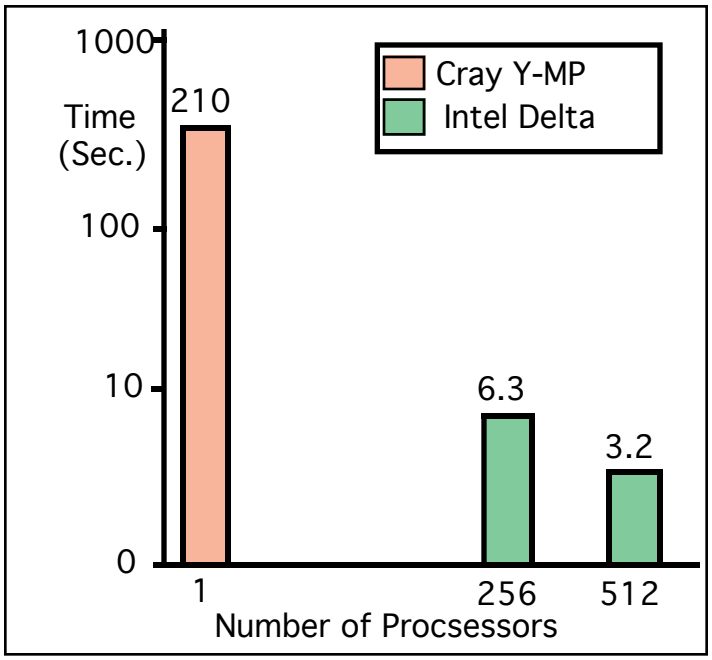


Fig. 6. Node-based Matrix assembly time for Mach 3.0 HSCT

The generation and assembly of the stiffness matrix for this example executed on one Cray Y-MP processor initially took 630 seconds using the new nodal method with compiler optimization. After the key routines were rewritten to include "loop unrolling", the time was reduced to 210 seconds. Loop unrolling reduces computation time since it permits simultaneous use of the add and multiply functional units. Moreover, the best time obtained for the element-based algorithm on the Cray Y-MP was 70 seconds with loop unrolling optimization. Nevertheless, as previously mentioned, the element-based approach does not parallize well, and is not suitable for SHPC.

The times to generate and assemble the stiffness matrix for the Mach 3.0 HSCT on the corresponding Delta supercomputer were 15.1 and 7.5 seconds on 256 and 512 processors, respectively, using compiler optimization. After rewriting the algorithm to include loop unrolling, the times were further reduced to 6.3 and 3.2 seconds on 256 and 512 processors, respectively. The algorithm involves matrix-vector and matrix-matrix multiplications on small element matrices (at most 18x18), which leads to short-vector operations. Thus, it would not be expected that this algorithm would perform very well on vector computers such as the Cray. Since the nodal algorithm is perfectly parallel a excellent performance is achieved on SHPC. For the Mach 3.0 HSCT problem, the measured performance was 1.6 and 3.05 GigaFLOPS on 256 and 512 processors, respectively.

5.2 Solution of system of Equations

Equation solution time dominates static and vibration analyses. The matrix factorization in Gauss method is most critical for static analysis, forward/backward substitution is most critical for vibration analysis. This section focuses on factorization, while the next section focuses on forward/backward substitution. A Gauss elimination algorithm using a variable band storage scheme was implemented in FORTRAN on the Intel i860. The first implementation of this algorithm, denoted RowSolver was not vectorized. The time to solve a 1000 full, symmetric equations on the Intel Gamma for three direct methods is shown in Fig. 7.

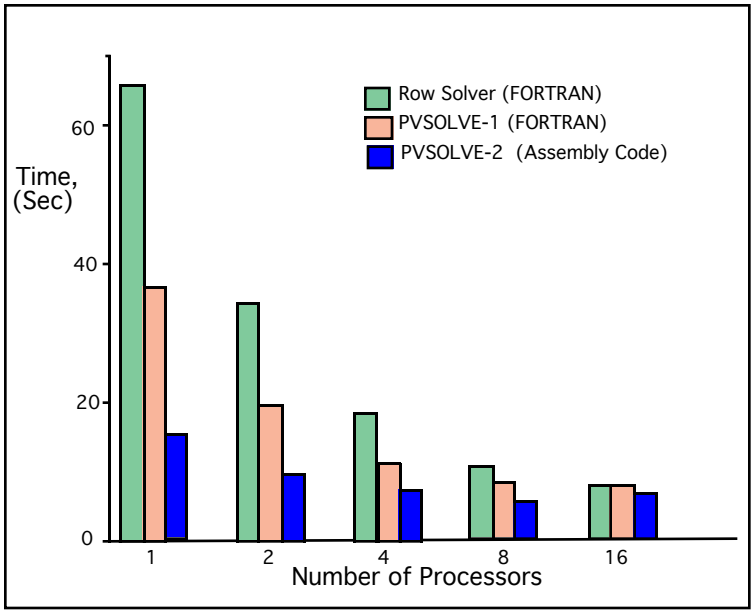


Fig. 7: Time comparison for three direct equation solvers on the Gamma for the 1000 x 1000 equations.

RowSolver is a Gauss elimination solver written in FORTRAN with no vector optimization. PVSOLVE-1 is a Choleski solver using loop unrolling (level six). PVSOLVE-2 is a Choleski solver using loop unrolling (level six) and a dot product routine written in assembly code. The RowSolver solution time is 65 seconds on one processor, while PVSOLVE-2 took 18 seconds. On 16 processors, RowSolver took 6.5 seconds and PVSOLVE-2 took 5.5 seconds. Regardless of how well the equation solvers were vectorized on one processor, the vector speed becomes less significant when many processors are used. Based on these results, loop unrolling is not used to obtain vector speed for subsequent results.

Solution time for a simplified Mach 3.0 HSCT model with 1646 equations and an average bandwidth of 321 were obtained using an early version of RowSolver to obtain the breakdown of the solution time on the Delta shown in Fig. 8.

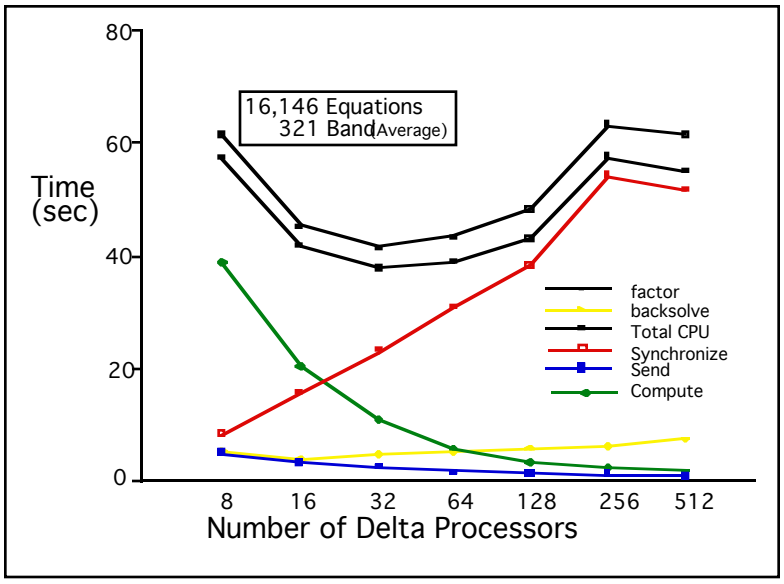
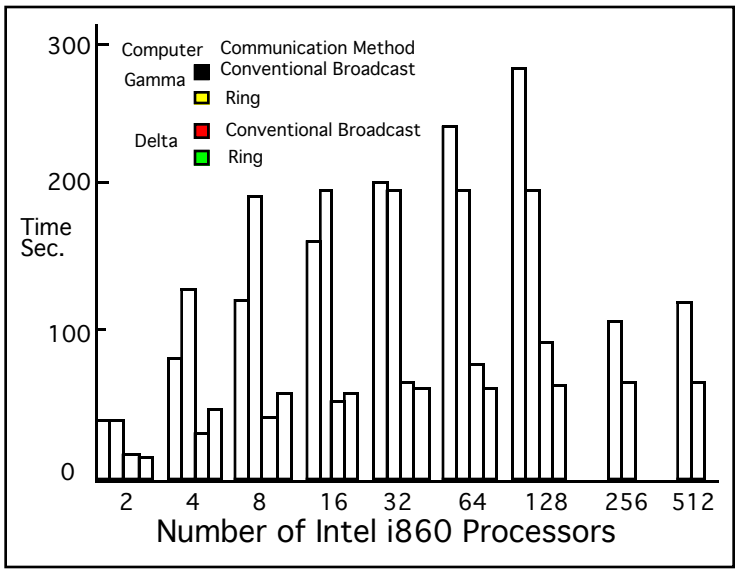


Fig. 8. Solution Time Breakdown for Simplified Mach 3.0 HSCT Model

Although the compute time decreases as the number of processors increases, the communication time increases, resulting in a net decrease in overall performance beyond 32 processors. This is a typical tradeoff between computation and communication time for a fixed problem size. When a certain number of processors is fixed, there may not be sufficient computations to perform on each processor to offset the communication required. Thus, reducing the communication time is even more critical than reducing the CPU time. Two communication schemes were evaluated. Of these, ring communication was found to perform best for equation solution.



.

Fig. 9. Communication Time when solving HSCT M2.4

When the broadcast communication method is used, the time to communicate between processors increases as the number of processors increases linearly on both the

Gamma and Delta SHPC. On the other hand, ring communication remains constant as the number of processors increases. Thus, ring communication is preferred, in particular as the number of processors is large.

The effect of changing the number of equations assigned to each processor(block size) for the Mach 2.4 HSCT is shown in Fig. 10.

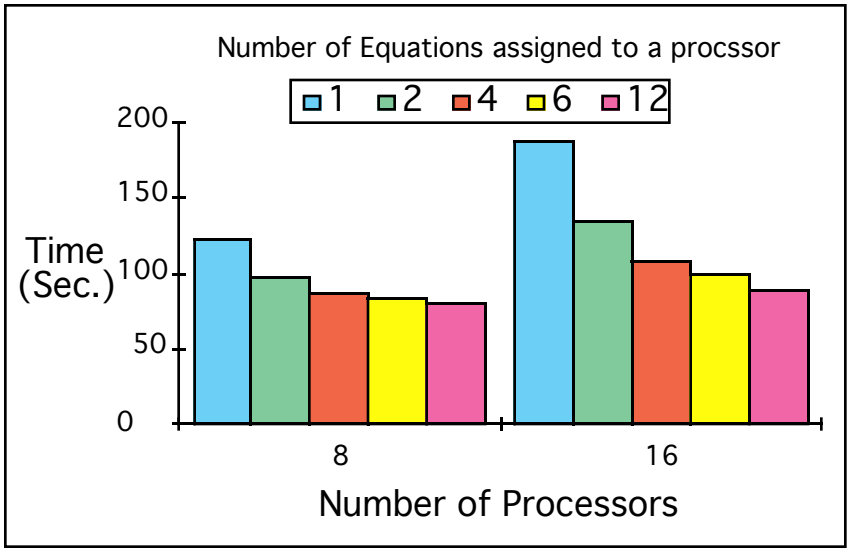


Fig. 10. Communication time vs. block size for Mach 2.4 HSCT on Intel Gamma.

As the number of equations per block increases, the communication time decreases. For example, when 16 processors are used, and one equation is assigned to each processor, the time to communicate is 180 seconds. This time reduces to 90 seconds when 6 equations are assigned to a processor. As the number of equations per processor increases beyond six equations per processor, the reduction in communication time becomes less dramatic and appears asymptotic. For structural applications, assigning each structural node (with 6 degrees of freedom) to a processor (i.e. six equation per processor), should result in good performance.

The RowSolver performance solution time reduces in direct proportion to the bandwidth of the structural matrix. In order to illustrate the point a symmetric banded matrix with a fixed bandwidth of 1000 and a varying number of equations was considered. The solution time for varying the number of equations from 5000 to 10000 is shown in Fig 11.

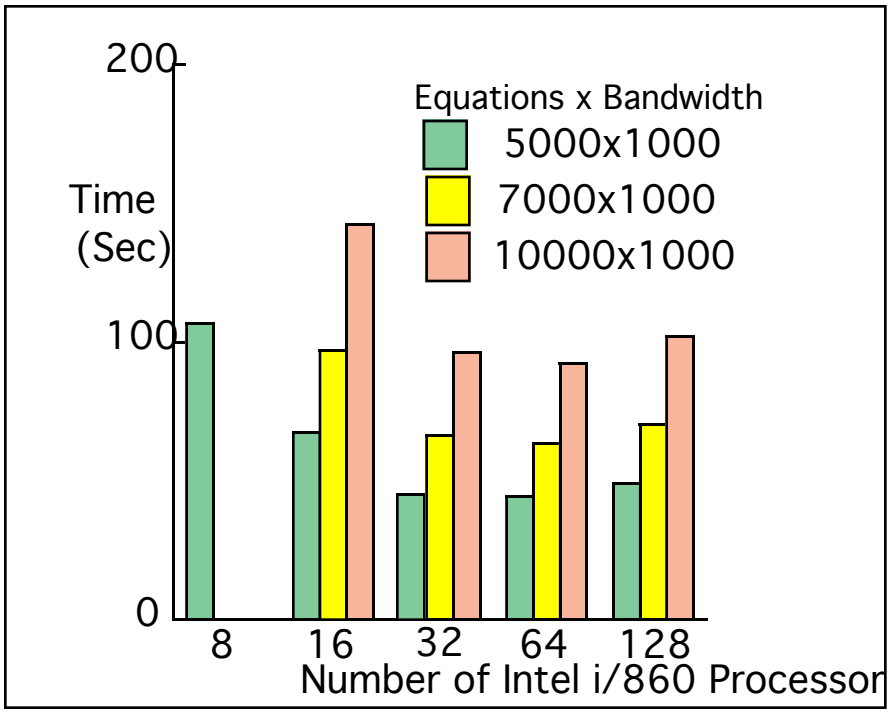


Fig. 11. Solution time vs. number of equations and processors for fixed bandwidth.

The time to solve 5000 equations with on 16 processors is 66.7 seconds, and 44.5 seconds on 32 processors in 44.5 seconds. The speed up between 8 and 16 processors is 1.5. On the other hand, a matrix with 7000 equations took 97 seconds to solve on 16 processors, 64.5 seconds on 32 processors. The speed up between 16 and 32 processors is 1.5. For a constant bandwidth of 1000, a reduction in computation time is shown in Fig. 11 for up to 32 processors, regardless of the number of equations solved. However, if more than 32 processors are used, there is no further time reduction because the slow communication rate offsets the faster computation rate.

A different case is a "computation bound" problem solved on multiple processors, when the number of equations is held constant (i.e. 5000 as shown in Fig. 12). Here the bandwidth is increased from 500 to 1500, so even though the number of computations increases, but the computation time decreases as the number of processors is increased.

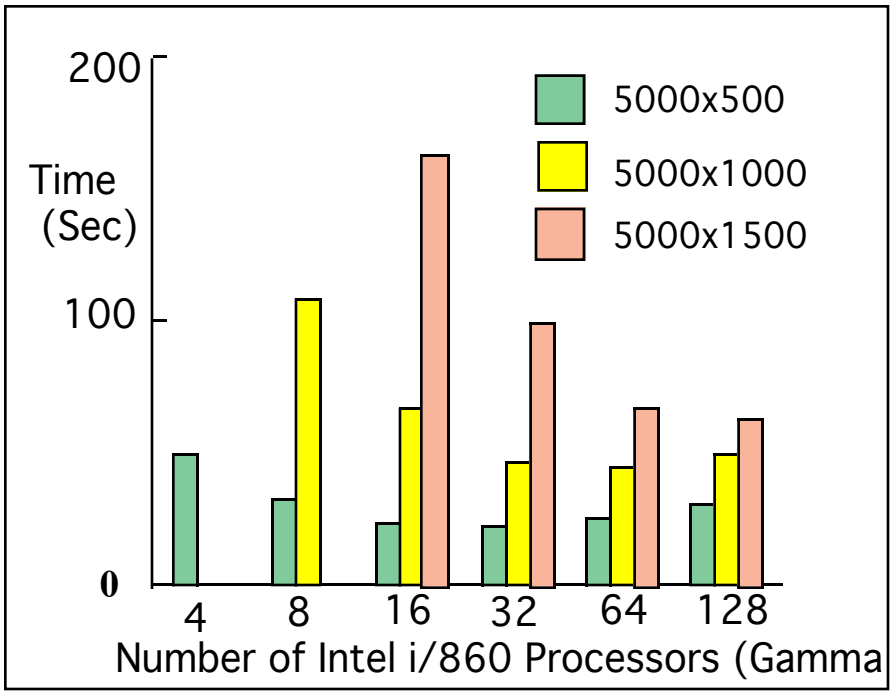


Fig. 12. Solution time for increased work (bandwidth) on the Gamma.

For a problem with a fixed number of equations and varying bandwidths, the solution time increases linearly as the bandwidth increases. However as Fig. 12 illustrates, for a small bandwidth of 500, the minimum computational time was obtained using 16 processors, while for a bandwidth of 1000, it was obtained using 32 processors and for a bandwidth of 1500, it was obtained using 64 processors. For a matrix with 5000 equations and 1500 bandwidth, the time used to solve the problem on 16 processors is 164.5 seconds, and 100 seconds on 32 processors. The speed up between 16 and 32 processors is 1.64, while the speed up to solve 5000 equations with 1000 bandwidth is 1.5 (from previous example). Thus, one can expect the computational efficiency to improve as the bandwidth increases since the number of computations performed on each processor is increased. This is not true if the communications rate were increased dramatically, but is typical of future well-balanced multiple processor computers having a high interprocessor communication rate is.

For the Mach 2.4 HSCT model, the minimum computation time obtained on one Cray Y-MP processor was 8.7 seconds (see Fig. 13), while the time was 32 seconds on 32 Intel Delta processors.

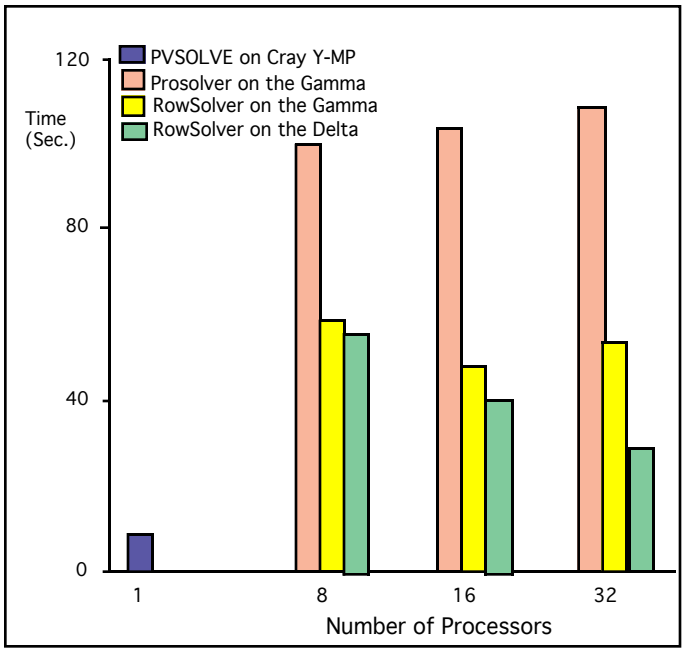


Fig. 13. Equation Solution Comparison (HSCT Mach 2.4)

A continual reduction in solution time is shown in Fig. 13 for RowSolver as the number of processors increases from 8 to 32. Unfortunately, for more than 32 processors, the computation time increases. This is attributed to primarily to the relatively slow communication rate (12.2 MB/sec) on the Delta. This rate expected to be 200 MB/sec for the next generation Intel Paragon.

The minimum time to solve the Mach 3.0 model on 512 processors on the Intel Delta was 532 seconds as shown in Fig. 14.

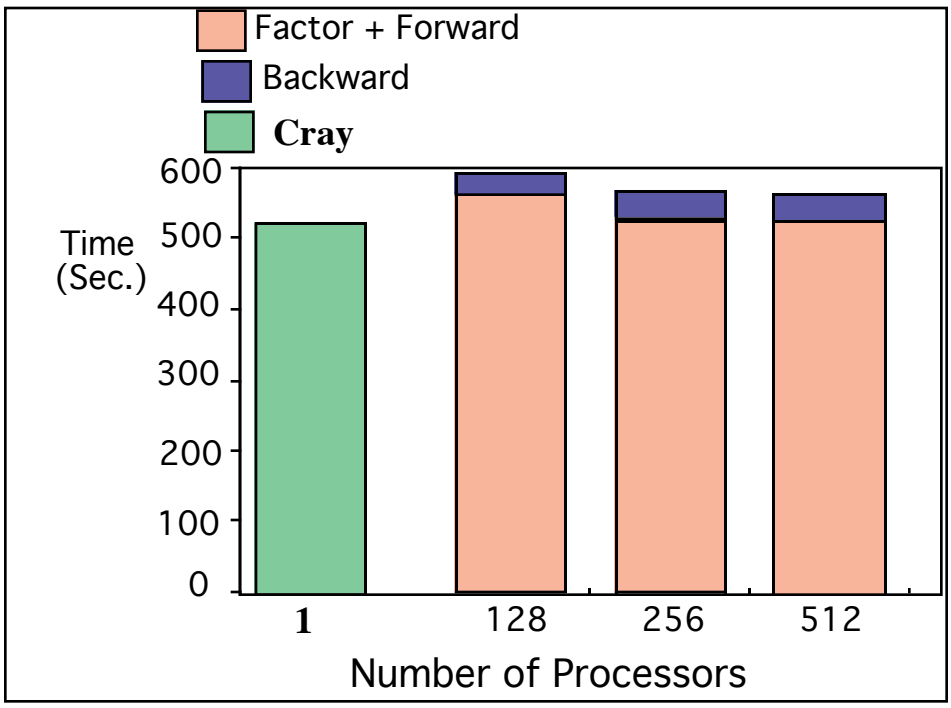


Fig. 14. Solution time for Mach 3.0 HSCT

The same problem took 530 seconds on one Cray Y-MP processor using an optimized banded solver, PVSOLVE[3]. Fig. 14 which shows a breakdown time for solving Mach 3.0 HSCT with factorization and forward substitution consuming the majority of time. For 256 processors, 500 seconds is spent in factorization and forward substitution, with only 30 seconds in backward substitution. The time breakdown for matrix factorization on the Intel Delta shown in Fig. 15.

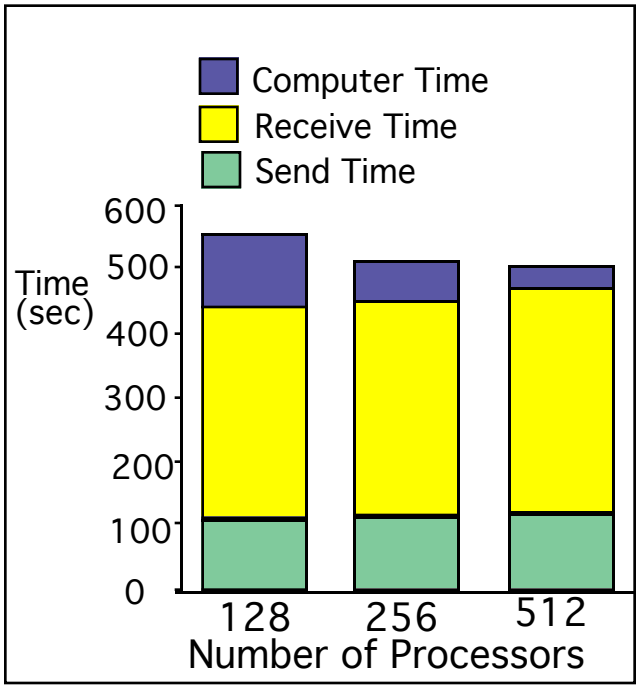


Fig. 15. Factorization time breakdown for Mach 3.0 HSCT

Although the computation time decreases as the number of processors increases, the communication time increases. For such large problem, the majority of time (over 450 seconds), is associated with communication while the computation time only 25 seconds on 512 processors.

For the Mach 2.4 HSCT analysis with the small bandwidth, the total Intel Gamma time was 4 times that than for one processor on a Cray. For the Mach 3.0 HSCT analysis with the large bandwidth, the Cray and the Intel solution times were approximately the same. This shows that SHPC become more efficient as the problem size increases.

5.3 Vibration Analysis

An analysis of the computation time for the Lanczos eigensolver shows that the equation solver, which is called at each iteration step, consumes nearly all of the time, as shown in Fig. 16.

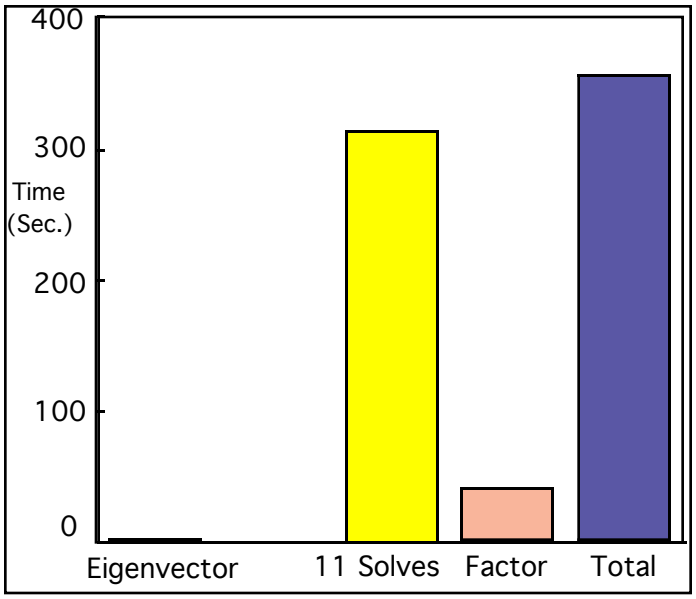


Fig. 16 Total solution time for HSCT Mach 2.4 HSCT

A new SHPC eigensolver based on the Lanczos method, has a modular design which enables the most up-to-date equation solver to be incorporated in it. An improved version of RowSolver (using ring communication) was implemented in the eigensolver. Fig. 17 shows the resulting decrease in computation time using this solver, solving the Mach 2.4 HSCT problem on 8 processors of the Delta computer.

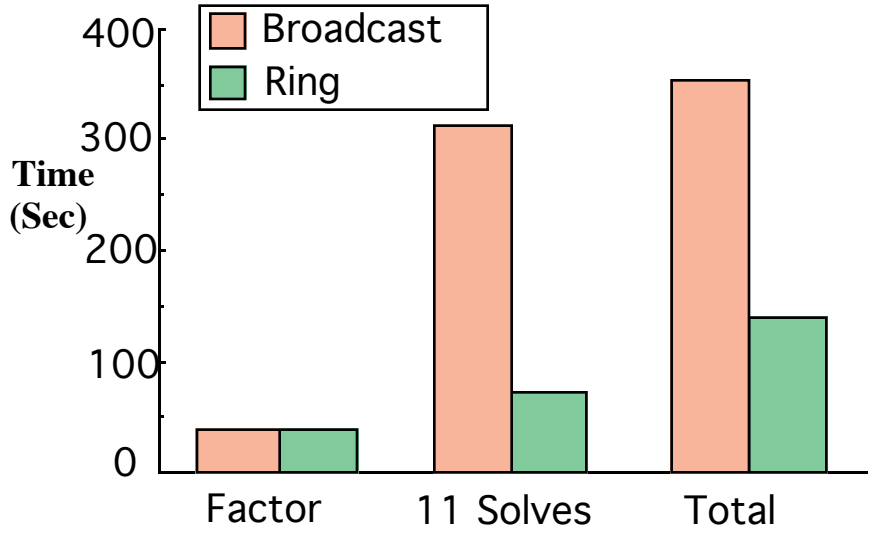


Fig. 17 Vibration analysis time study for Mach 2.4 HSCT comparing ring and broadcast communication

A comparison of the time to compute the eigenvaluse for the Mach 2.4 HSCT on 8 and 16 processors is shown in Fig. 18. The time to factor the matrix shows a small decrease in computation time. However, the time to solve remains constant due to the required serial operations in the forward/backward solution step. The eigenvector

computation parallize well, as do the many matrix-vector and matrix-matrix multiplications.

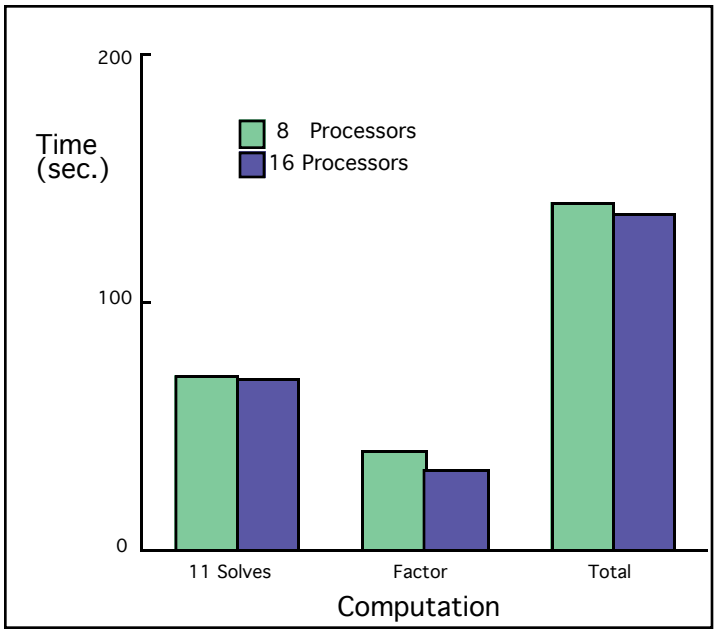


Fig. 18 Vibration analysis time for Mach 2.4 HSCT

Conclusion

The node-based algorithm for the parallel generation and assembly of stiffness matrices is perfectly parallel as no communication time is required. Near linear speedup is achieved. for all applications tested. For current SHPC vector speed in not important, communication time dominates when large number of processors are used. Ring communication is faster than the traditional broadcast for large number of processors. The speed of the parallel Gauss elimination equation solution algorithm performance depends directly on the square of the bandwidth of the application. For small bandwidth problems, insufficient computation is involved to realize gains possible by increasing the number of processors. However, as the bandwidth increases, the performance of the equation solver on an increasing number of processors improves. The communication was found to be critical and consume the largest time (even 10x CPU time) for large applications on the Intel i/860. This rate which is determined by both latency and the communication rate. It is expected that the equation solver will run without changes on the Paragon with a significant reduction in communication which in turn should dramatically reduce the overall time. Since the speed of the eigensolver is directly proportional to the speed of the equation solver, similar performance gains are expected for the eigensolver on the Paragon.

References

1. Baddourah, M. A., Storaasli, O. O., Carmona E. A. and Nguyen, D. T., "A Fast Parallel Algorithm for Generation and Assembly of Finite Element Stiffness and Mass Matrices", AIAA Paper No. 91-1006, *Proc. of the 32nd AIAA/ASME/ASCE/AHS/ ASC Structures, Structural Dynamics and Materials Conference,* Baltimore, MD April 8-10, 1991, pp. 1547-1553

2 Storaasli, O. Nguyen, D., Baddourah, M. and Qin, J.; "Computational Analysis Tool for Parallel-Vector Supercomputers", *AIAA/ASME/ASCE/AHS/ASC 34th Structures, Structural Dynamics and Materials Conference Proceedings,* Part 2, pp. 772-778, April 1993 (to appear in International Journal of Computing Systems in Engineering, 1993).

3 Agarwal, T., Storaasli, O., and Nguyen, D.,"A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers", AIAA Paper No. 90-1149, *Proc. of the AIAA/ASME/ ASCE/AHS 31st Structures, Structural Dynamics and Materials Conference,* Long Beach, CA, April 2-4, 1990, pp. 662-672.

Acknowledgment
This research was performed in part using the Intel Touchtone Delta System operated by Caltech on behalf of the Concurrent Supercomputing Consortium.