AIAA JOURNAL
Vol. 40, No. 2, February 2002

Large-Scale Structural Analysis by Parallel Multifrontal Solver Through Internet-Based Personal Computers

Seung Jo Kim* and Chang Sung Lee†
Seoul National University, Seoul 151-742, Republic of Korea
and
Jeong Ho Kim‡
Korean Institute of Science and Technology Information, Taejon 305-333, Republic of Korea

Internet supercomputing methodology is introduced, and the concept is realized for large-scale finite element analysis. This is enabled by an efficient out-of-core parallel solver, which is based on domainwise or elementwise multifrontal approach. The primary resources of Internet supercomputing are numerous idling personal computers connected by the Internet irrespective of their locations. The computing ability of hundreds or thousands personal computers connected by the Internet can be as powerful as that of supercomputer such as the IBM SP system if these personal computers are utilized for solving a problem simultaneously through an efficient parallel computing algorithm. Under the described concept, a virtual supercomputing system InterSup I is constructed and tested. To establish the InterSup I system, 64 personal computer nodes, which are located in several places and connected by the Internet, are conscripted. When the established InterSup I system is used, linear static analyses of a finite element model having around 2×10^6 degrees of freedom are done through the developed parallel multifrontal solver. Some numerical tests have been carried out to investigate the affordability and effectiveness of Internet supercomputing. The possibility of Internet supercomputing through Internet personal computers as an alternative supercomputing power for high-performance computing is provided in this research.

Introduction

FINITE element analyses are often required in the processes of structural design and certification of large and complex structures such as aerospace vehicles. In this case, most of the computation time is spent on solving the system of algebraic equations associated with the finite element model. Kim and Kim[1] have developed an efficient multifrontal solver that can reduce fill-ins by using multiple fronts to solve the systems of linear equations. The solver was combined with graph-partitioning algorithms to obtain partitioned fronts, and all of the operations were performed in an elementwise manner. In this study, a new out-of-core parallel solver based on this multifrontal technique is developed for large-scale structural analysis.

For large-scale structural analysis and high-precision simulation of structures, supercomputing power as well as efficient parallel algorithms must be implemented to obtain reliable results within a reasonable time. Whereas various types of parallel hardware architectures have been developed, Internet supercomputing, which uses general-purpose personal computers simply connected by the Internet as a large parallel computing resource, can be a cost-effective way of implementing high-performance computing considering the startling progress in microprocessor technology in recent years. These computers in laboratories and companies are, for the most part, interlinked by the Internet and are idling after the closing-hour time. However, these computers can be utilized as enormous computing resources if they are combined through the network. This is a basic concept of Internet supercomputing. To realize this concept, an efficient parallel computing algorithm should be developed, and interface tools for handling computers simultaneously are needed.

In this study, the parallel performance and efficiency of the developed parallel multifrontal solver are tested in distributed parallel environments. Then, the effectiveness and possibility of Internet supercomputing proposed in this study was tested by numerical examples of large-scale structural analysis through the developed parallel multifrontal solver with Internet-based personal computers.

Multifrontal Solver

The concept of the multifrontal method was first introduced by Duff and Reid[2] as a generalization of the frontal method of Irons.[3] The elimination process of the frontal method is shown in Fig. 1, and that of the multifrontal method applied to finite element procedure is shown in Fig. 2.

The difference between the elimination process of the single frontal method and that of the multifrontal method can be understood by comparing Fig. 1 with Fig. 2. In the single frontal method, only one front is used, and it spreads out the whole domain, as degrees of freedom (DOF) fully assembled are eliminated from the front. By the multifrontal method, a given domain is divided into two subdomains recursively, and each subdomain has its own fronts, shown in Fig. 2. After the recursive bisection of the given domain is completed, the new fronts are constructed by the remaining interface DOF of each domain and are merged with each other recursively. At each merging state, DOF fully assembled are eliminated immediately.

Coefficients related to the eliminated DOF and the inactive front matrices, which are not involved in current elimination process, can be stored to secondary storage without loss of efficiency, as shown by Kim and Kim.[1] This feature can considerably reduce the required memory size and, consequently, make it possible to handle a much larger number of DOF. Kim and Kim also investigated the influence of the mesh-mapping schemes and the partitioning quality on the performance of the multifrontal solver.

Parallel Multifrontal Solver

The parallelization of multifrontal solver has been tried by several researchers including Lucas et al.,[4] Ingel and Mountziaris,[5] Geng et al.,[6] and Gupta et al.[7] The multifrontal solver for general sparse systems, which were proposed by Duff and Reid,[2] was parallelized

Presented as Paper 2000-1417 at the 41st Structures, Structural Dynamics, and Materials Conference, 3–6 April 2000; received 6 February 2001; revision received 31 July 2001; accepted for publication 5 August 2001.

*Professor, Department of Aerospace Engineering, Kwanak-gu. Member AIAA.
†Graduate Research Assistant, Department of Aerospace Engineering, Kwanak-gu.
‡Senior Researcher, Supercomputing Application Department, 52 Eoeun-dong Yusong-Gu.

by Lucas et al.[4] This multifrontal solver was used to solve the sparse systems arising from semiconductor design. It can be applied to the sparse linear system whose adjacency structure can be represented by a planar graph, for example, two-dimensional finite element problems. Geng et al.[6] solved two-dimensional finite element problems using four processors with one front in each processor. Ingle and Mountziaris[5] implemented the multifrontal solver to solve the large systems of equations arising in two-dimensional finite element analyses of chemical transport and reaction processes in a network-based parallel computing environment. They exploited only the inherent parallelism of the multifrontal solver. In other words, computation to eliminate fully assembled DOF on the subdomain boundary was performed on only one processor. It was not a serious problem for them because they solved two-dimensional problems and used just eight processors at most. However, in the case of three-dimensional analyses, which have a large number of DOF in the subdomain boundary, it is very difficult to obtain a good parallel performance. Using a large number of processors will only make the problem worse. The most recent and efficient implementation of the parallel multifrontal solver is the one developed by Gupta et al.[7] They implemented a general parallel sparse solver based on a multifrontal technique. Although it is well organized and efficient, it is not adequate for finite element analysis because it is an on-core solver, which requires a much greater amount of memory than its out-of-core counterpart, and it also requires a separate construction process of global stiffness matrices, which are not going to build. Therefore, in this study, a new out-of-core implementation of multifrontal technique that intermixes the matrix construction process and the solution process by a fully domain or elementwise approach and does not require any separate matrix construction process is proposed to obtain solutions for large-scale structural analyses. Detail of the procedures of the parallel multifrontal solver proposed in this study are discussed in this section.

The basic parallel implementation of the multifrontal solver is shown in Fig. 3. Mesh partitioning must be carried out to perform parallel computing. Mesh-partitioning algorithms based on graph-partitioning algorithms was used in this study. To use this graph-partitioning algorithm, finite element meshes to be partitioned must be converted into graph structures. A graph-mapping scheme, weighted-edge mapping (WEM), proposed by Kim and Kim[1] was used for converting finite element meshes to graph structures as shown in Fig. 4. Each element of the finite element mesh is mapped to a vertex. To reflect the connectivity information of finite element meshes, an edge is constructed between two elements that have at least one common node, and the weight of an edge between two neighboring elements is set to the number of common nodes. To obtain a partitioned graph, graph-partitioning routines provided by Metis[8] software were used to divide the graph into two parts recursively. The graph partitioning using Metis was conducted to assign the same vertexes to processors and to minimize the number of edge cuts. Therefore, the number of elements assigned to each processor is the same, and the communication resulting from the placement of adjacent elements to different processors is minimized.

After the mesh-partitioning stage, partitioned domains are distributed to each processor. Each domain assigned to each processor is partitioned by two parts recursively into smaller fronts until appropriate number of elements are given to each front. According to numerical tests performed by Kim and Kim,[1] the proposed multifrontal solver performs best for three-dimensional analysis when about eight elements in average are given to each initial front. After this front-partitioning stage, each processor eliminates fully assembled DOF in each partitioned domain. In the results, if totally N fronts and P processors are used for parallel computing, N/P fronts are distributed to each processor, and each processor can perform the elimination process of N/P fronts independently, as shown in Fig. 3. Therefore, the elimination process can be simply parallelized due to its inherent parallelism of the multifrontal solver.

After each processor successfully performs the elimination procedure, fronts of adjacent processors are merged as shown in Fig. 3. In Fig. 5, the front-merging processes between two processors is shown:

$$\boldsymbol{i}^{(1)} = [38 \quad 30 \quad 22 \quad 14 \quad 13 \quad 12 \quad 11]$$

$$\boldsymbol{i}^{(2)} = [38 \quad 30 \quad 22 \quad 14 \quad 15 \quad 16 \quad 17]$$

$$\boldsymbol{K}^{(1)} = \begin{bmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} & k_{15}^{(1)} & k_{16}^{(1)} & k_{17}^{(1)} \\ & k_{22}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} & k_{25}^{(1)} & k_{26}^{(1)} & k_{27}^{(1)} \\ & & k_{33}^{(1)} & k_{34}^{(1)} & k_{35}^{(1)} & k_{36}^{(1)} & k_{37}^{(1)} \\ & & & k_{44}^{(1)} & k_{45}^{(1)} & k_{46}^{(1)} & k_{47}^{(1)} \\ & \text{Sym} & & & k_{55}^{(1)} & k_{56}^{(1)} & k_{57}^{(1)} \\ & & & & & k_{66}^{(1)} & k_{67}^{(1)} \\ & & & & & & k_{77}^{(1)} \end{bmatrix}$$

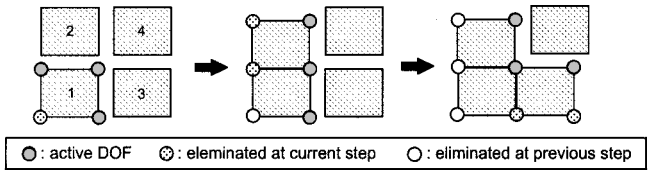

Fig. 1 Elimination process of the frontal method.

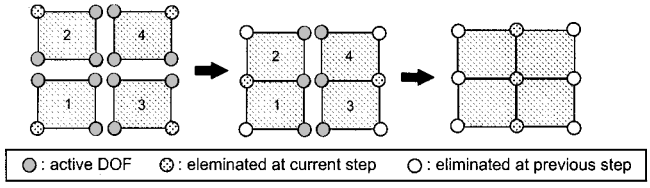

Fig. 2 Elimination process of the multifrontal method.

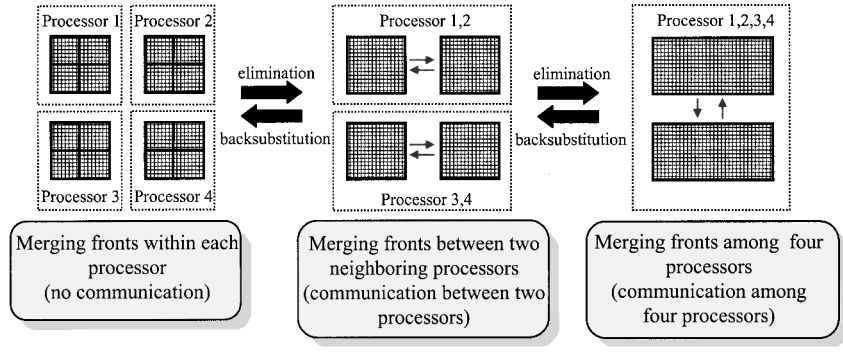

Fig. 3 Computing procedure of the parallel multifrontal solver.

$$\boldsymbol{K}^{(2)} = \begin{bmatrix} k_{11}^{(2)} & k_{12}^{(2)} & k_{13}^{(2)} & k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} & k_{17}^{(2)} \\ & k_{22}^{(2)} & k_{23}^{(2)} & k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} & k_{27}^{(2)} \\ & & k_{33}^{(2)} & k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} & k_{37}^{(2)} \\ & & & k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} & k_{47}^{(2)} \\ & \text{Sym} & & & k_{55}^{(2)} & k_{56}^{(2)} & k_{57}^{(2)} \\ & & & & & k_{66}^{(2)} & k_{67}^{(2)} \\ & & & & & & k_{77}^{(2)} \end{bmatrix}, \qquad \boldsymbol{i}^{(1+2)} = [38 \quad 30 \quad 22 \quad 14 \quad 13 \quad 12 \quad 11 \quad 15 \quad 16 \quad 17]$$

$$\boldsymbol{K}^{(1+2)} = \begin{bmatrix} k_{11}^{(1)}+k_{11}^{(2)} & k_{12}^{(1)}+k_{12}^{(2)} & k_{13}^{(1)}+k_{13}^{(2)} & k_{14}^{(1)}+k_{14}^{(2)} & k_{15}^{(1)} & k_{16}^{(1)} & k_{17}^{(1)} & k_{15}^{(2)} & k_{16}^{(2)} & k_{17}^{(2)} \\ & K_{22}^{(1)}+k_{22}^{(2)} & K_{23}^{(1)}+k_{23}^{(2)} & K_{24}^{(1)}+k_{24}^{(2)} & k_{25}^{(1)} & k_{26}^{(1)} & k_{27}^{(1)} & k_{25}^{(2)} & k_{26}^{(2)} & k_{27}^{(2)} \\ & & k_{33}^{(1)}+k_{33}^{(2)} & k_{34}^{(1)}+k_{34}^{(2)} & k_{35}^{(1)} & k_{36}^{(1)} & k_{37}^{(1)} & k_{35}^{(2)} & k_{36}^{(2)} & k_{37}^{(2)} \\ & & & k_{44}^{(1)}+k_{44}^{(2)} & k_{45}^{(1)} & k_{46}^{(1)} & k_{47}^{(1)} & k_{45}^{(2)} & k_{46}^{(2)} & k_{47}^{(2)} \\ & & & & k_{55}^{(1)} & k_{56}^{(1)} & k_{57}^{(1)} & 0 & 0 & 0 \\ & & \text{Sym} & & & k_{66}^{(1)} & k_{67}^{(1)} & 0 & 0 & 0 \\ & & & & & & k_{77}^{(1)} & 0 & 0 & 0 \\ & & & & & & & k_{55}^{(2)} & k_{56}^{(2)} & k_{57}^{(2)} \\ & & & & & & & & k_{66}^{(2)} & k_{67}^{(2)} \\ & & & & & & & & & k_{77}^{(2)} \end{bmatrix}$$

This merging process is very difficult to parallelize effectively. If, as in Ref. 5, all computations related with this front-merging process are carried out in one processor, the merged front matrix ($K^{(1+2)}$) must be constructed and saved by one processor. Therefore, load balancing is not achieved well, and almost the same quantity of main memory required in serial computing may be needed. For this reason, parallel performance is deteriorated in three-dimensional problems that have many DOF in the subdomain interface. In this study, these front-merging processes are conducted at the same time by the adjacent processors having common elements through a parallel Gauss-elimination algorithm (see Ref. 9). In the elimination process using ith row as pivot, as shown in Fig. 6, we must divide $A(j, i)$ by $A(i, i)$ and subtract $A'(j, i)(= A(j, i)/A(i, i))$ from $A(j, i)$. This operation can easily be parallelized if the ith row is distributed to all of the processors, as shown in Fig. 7. As in Fig. 7, $A(1, 1{:}100)$ can be distributed to four processors through broadcasting operations, and computing burdens can be distributed according to the remnants of row number divided by processor number. Thus, computation loads are balanced, and the elimination process can be done simultaneously using four processors. As mentioned before, the front-merging processes are parallelized thorough the parallel Gauss-elimination algorithms in this study. In Fig. 5, processors 1 and 2 carried out the front-elimination process in each domain, respectively. The $K^{(1)}$ front matrix is constructed and saved in the main memory by processor 1 and the $K^{(2)}$ front matrix is constructed and saved by processor 2. There are a total four DOF (38, 30, 22, and 14) in the interface region between the two processors. If processor 2 sends the $K^{(2)}$ matrix to processor 1, processor 1 can construct the merged front matrix $K^{(1+2)}$. However, in our algorithms, $K^{(1+2)}$ is not constructed explicitly in any processors. The values in $K^{(2)}$ and $K^{(1)}$ related with common DOF (38, 30, 22, and 14) are exchanged and updated between processors 1 and 2. Thus, two processors have

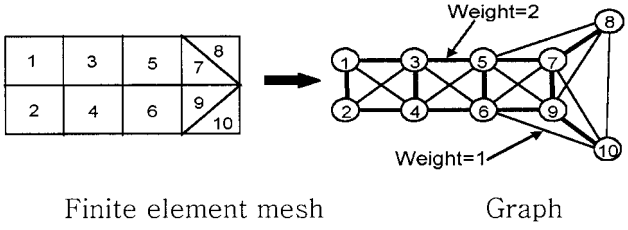

Fig. 4 WEM scheme.

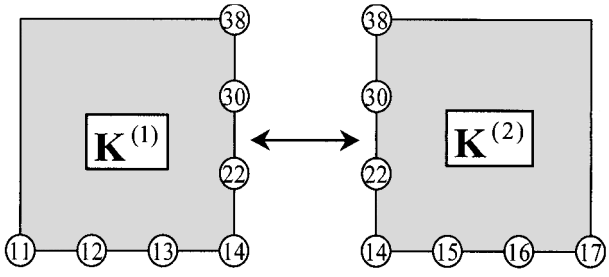

Fig. 5 Front-merging procedure.

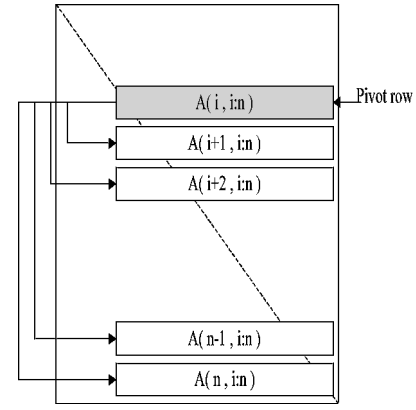

Fig. 6 Gauss-elimination using ith row as pivot.

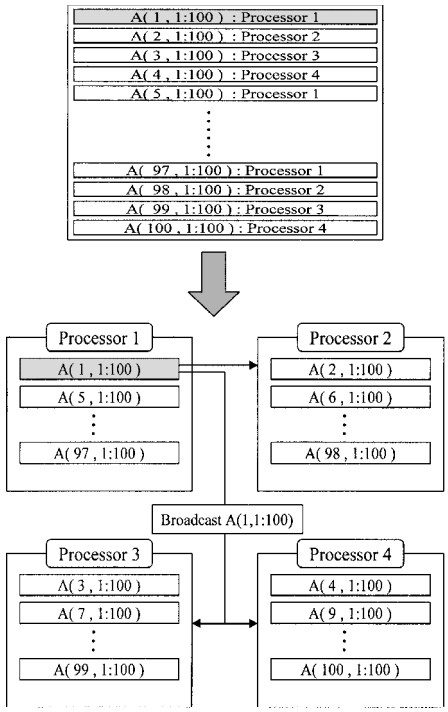

Fig. 7 Distribution of matrix for parallel Gauss elimination.

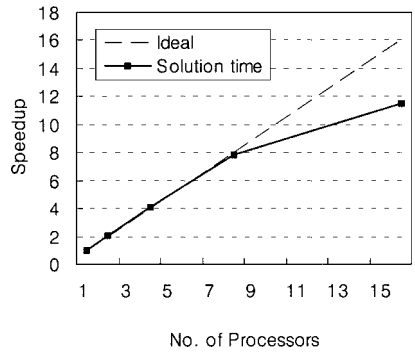

Fig. 8 Speedup for 16 × 16 × 16 solid elements.

the same value in front of matrix related with common DOF. When DOF 38 is eliminated, the elimination operations related with rows 1, 3, and 5–7 of $K^{(1+2)}$ can be carried out in processor 1, and the elimination operations related with rows 2, 4, and 8–10 of $K^{(1+2)}$ can be carried out simultaneously in processor 2. These operations are basically based on parallel Gauss-elimination algorithms, and the rows of the front matrix engaged in the current elimination process are distributed to the processors. Therefore, the front-merging process between two processes can be parallelized without constructing the $K^{(1+2)}$ matrix.

To obtain final solutions, backsubstitutions must be carried out in each processor in the same way as the multifrontal solver, and the displacements obtained by the parallel multifrontal solver are sent to one processor and stored in file.

Performance of Developed Parallel Multifrontal Solver

The parallel performance of developed parallel multifrontal solver is checked in a massively parallel processors (MPP) system (Cray T3E). A message-passing interface (MPI) library is utilized for the development of the parallel multifrontal solver. The parallel performances are measured in the elapsed time required in the solving process excluding the mesh-partitioning process and front-partitioning process, and the speedup is defined as follows:

$$S(n) = \frac{\text{elapsed time using one processor}}{\text{elapsed time using } n \text{ processors}}$$

Figure 8 is the results of finite element analysis with 16 × 16 × 16 three-dimensional solid elements, and the measured speedup is tabulated in Table 1. Detailed information of the communications of domain 1 is given in Table 2. Figure 8 shows a good parallel performance up to 8 processors, but the rate of increase on speedup is decreased over 16 processors. This is because the problem size is so small that the ratio of communication operations to elimination operations is large. The results of 32 × 32 × 16 three-dimensional solid elements are shown in Fig. 9, and this problem has 55,539 DOF. In Table 3, the measured speedups are also tabulated, and the communication information is given in Table 4. Excellent parallel performance is obtained up to 32 processors. As described in Table 4, the communications are increased about two times, but the number of elements allocated to each processor is increased four times. Therefore, the ratio of communication operations to elimination operations is smaller than the first example. Moreover, the deterioration of performance due to the usage of hard disk memory can be avoided in parallel computation. In general, the utilization of the slower hard disk memory must be needed in serial computing due to the large number of DOF, but sufficient memories are guaranteed if the number of processes is increased through the parallel computing. From these results, it can be concluded that the developed parallel multifrontal solver has shown good parallel performance in MPP systems.

To investigate the parallel performance of developed parallel multifrontal solver in general distributed parallel environments, a fast-networked personal computer system (eight CPUs) is constructed. Every node (personal computer) has the same system

Table 1 Speedups for 16 × 16 × 16 solid elements

No. of processors	Elapsed time	Speedup	No. of elements
1	171.6	1.00	4096
2	84.4	2.03	2048
4	41.8	4.11	1024
8	21.9	7.84	512
16	15.0	11.46	256

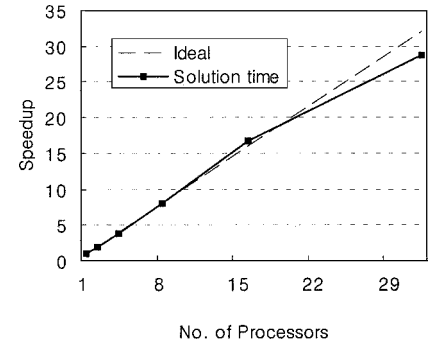

Fig. 9 Speedup for 32 × 32 × 16 solid elements.

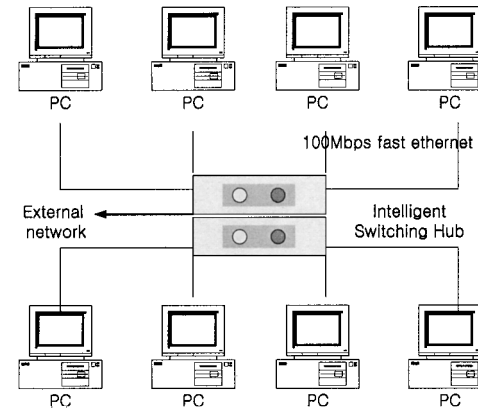

Fig. 10 Fast-networked personal computer system: CPU: Pentium III 550 MHz, RAM: 1 GB, HDD: 20 GB (E-IDE), OS: LINUX, and HUB: Intelligent switching hub (24 port, 2.1-Gbit bandwidth).

Table 2 Communication profile in domain 1 for 16 × 16 × 16 solid elements

No. of processors	Integer (4 B)		Double (8 B)		Total	
	Communication count	Communication volume, MB	Communication count	Communication volume, MB	Communication count	Communication volume, MB
2	993	4.77	2454	14.14	3447	18.71
4	1002	4.80	3286	17.93	4288	22.73
8	773	3.71	3606	18.97	4397	22.68
16	803	3.85	4028	21.13	4831	24.98

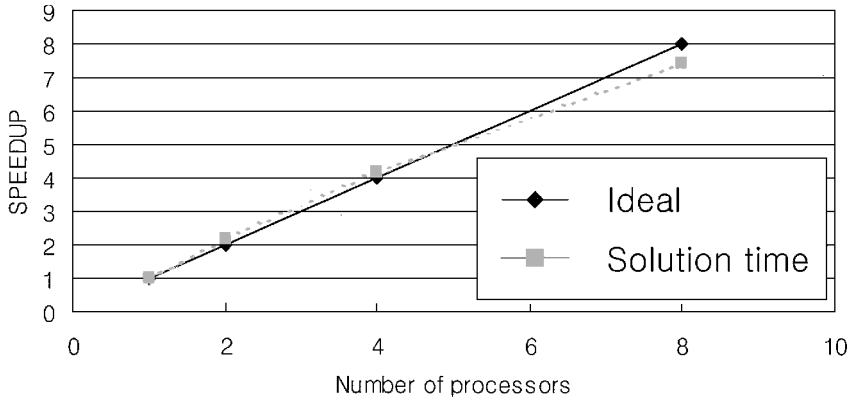

Fig. 11 Speedup results.

configuration (Fig. 10). A local area multicomputer (LAM)[10] is adopted for the message passing library. The Linux system is used for the operating system. The test problem has 116,400 DOF, and the parallel performance measured by numerical experiments is shown in Fig. 11. The results are almost the same up to eight processors with ideal speedup, that is, 7.5 speedup is obtained in the eight processors. Therefore, the parallel performance of developed parallel multifrontal solver is good in general distributed parallel environments, as well as in dedicated MPP systems.

Paradigm of Internet Supercomputing

The Internet supercomputing paradigm that we introduce in this study relies on two major fundamental changes in computing and networking environments. One is the unbelievable progress in performance of personal computers during the past decade, and the other is the permeation of the Internet into daily life. Two decades ago, the performance of personal computers was too low to compare to that of dedicated supercomputing machine. Similarly, the performance of a workstation was much more powerful compared to that of a personal computer around 10 years ago. However, the CPU performance of the personal computer has made startling progress in the past 10 years and has already reached or exceeded that of a dedicated workstation level these days. As a result, nowadays the boundary between the CPUs in a supercomputer and a personal computer has become vague. As is widely known, several MPP machines, such as the IBM SP2, are also equipped with the CPU that is commonly used in personal computers such as Macintosh. On the other hand, along with the remarkable improvement in the performance of the personal computer, the propagation of Internet changes our whole daily life patterns and business patterns substantially. This trend enforces most of the computers to be connected by the Internet with no regard to their locations (laboratory, office, home, etc.). Furthermore, the speed of the network has become faster due to the explosively increasing demand for networking.

The key idea of Internet supercomputing is to actively use the numerous general-purposed personal computers in the Internet environment. As mentioned before, most of the personal computers are connected by Internet. During office hours, these general-purposed personal computers are used for several purposes such as CAD, graphics, word processing, database application, etc. However, after office hours these computers connected by Internet start to idle without jobs or are turned off. These tremendous computing resources can be fully utilized, if a parallel algorithm appropriate to the Internet environment is developed to concentrate the scattered powers of personal computers onto one large-scale job. In addition, the resources are free and unlimited. These idling personal computers can be metamorphosed into a virtual supercomputer system after office hours and may show powerful computing capabilities. In this strategy, the supercomputing ability could be obtained for almost free. The cluster of personal computers could also be a similar way for affordable computing, but the clustered personal computer system is a kind of dedicated computing machine. Therefore, we have to pay for acquisition and maintenance of the system. Furthermore, the full utilization of the system is important so as not to waste the invested money because it is a dedicated computing machine. For these reasons, Internet supercomputing can be considered to be one of the cheapest ways and one of the most powerful ways to high-performance computing.

Basically a similar idea, the search for extraterrestrial intelligence (SETI)[11] project, has tried to utilize personal computers connected by the Internet. However, the main idea of SETI is a kind of sharing the job. All parts of the job are fully independent of each other. This is a kind of job distribution/collection through the Internet. However, the nature of solution procedure of physical problems is totally different from the problem characteristics of SETI. In solution procedure of physical problems, especially in finite element analysis, all of the DOF are coupled with each other, and a lot of communication is done to solve the problem in a parallel environment. This is not a collection of a bunch of independent jobs, but one huge-sized job. Thus, our idea of Internet supercomputing is totally different from the previous ones such as SETI project.

InterSup I: Realization of Internet Supercomputing

In this study, the virtual supercomputing system InterSup I is constructed to materialize the concept of Internet supercomputing. In the virtual supercomputing system, general-purposed personal computers connected by Internet are utilized. Even though the concept of Internet supercomputing is simple and fascinating, there are

a few obstacles to be overcome. We should first search for the available computers among the candidate resources because the status of available systems is changed everyday. Furthermore, because the resources are different in CPUs and operating systems (OSs) from each other, heterogeneous systems should be handled. Because our InterSup I system is quite different from the dedicated supercomputing machines, we should perform the computation within the limited access privilege because most of the resources in Internet supercomputing can be accessed not by superuser access level but by ordinary user access level. In this work, we conscript heterogeneous personal computers based on Linux/Unix operating system because Linux/Unix have several networking advantages over the Windows. LAM based on the Interoperable MPI (IMPI[10]) is adopted to handle the communication between heterogeneous systems. A secure shell is utilized to obtain the minimum access privilege to carry out the Internet supercomputing. We also make it possible to awaken the MPI daemon only when it is required. Unnecessary MPI daemon service gives rise to loss of computer resources when the personal computer is used for ordinary general purposes such as graphics, CAD, word processing, etc. Figure 12 shows the scattered locations of personal computers utilized in InterSup I. Each personal computer has an Ethernet network device, which was bought off the shelf. All communications among the personal computers are conducted through transmission control protocol/Internet protocol (TCP/IP).

To control the parallel computing jobs run through InterSup I, several utilities are also developed. Node monitoring tools can check

Table 3 Speedups for 32 × 32 × 16 solid elements

No. of processors	Elapsed time	Speedup	No. of elements
1	2,987	1.00	16,384
2	1,589	1.88	8,192
4	774.5	3.86	4,096
8	375.0	7.97	2,048
16	178.0	16.8	1,024
32	103.9	28.8	512

Table 4 Communication profile in domain 1 for 32 × 32 × 16 solid elements

	Integer (4 B)		Double (8 B)		Total	
No. of processors	Communication count	Communication volume, MB	Communication count	Communication volume, MB	Communication count	Communication volume, MB
2	1695	8.14	4208	24.20	5903	32.34
4	1722	8.27	5843	32.12	7565	40.39
8	2160	10.34	7487	40.11	9647	50.45
16	1346	6.46	8121	42.30	9467	48.76
32	931	4.47	8148	41.50	9079	45.97

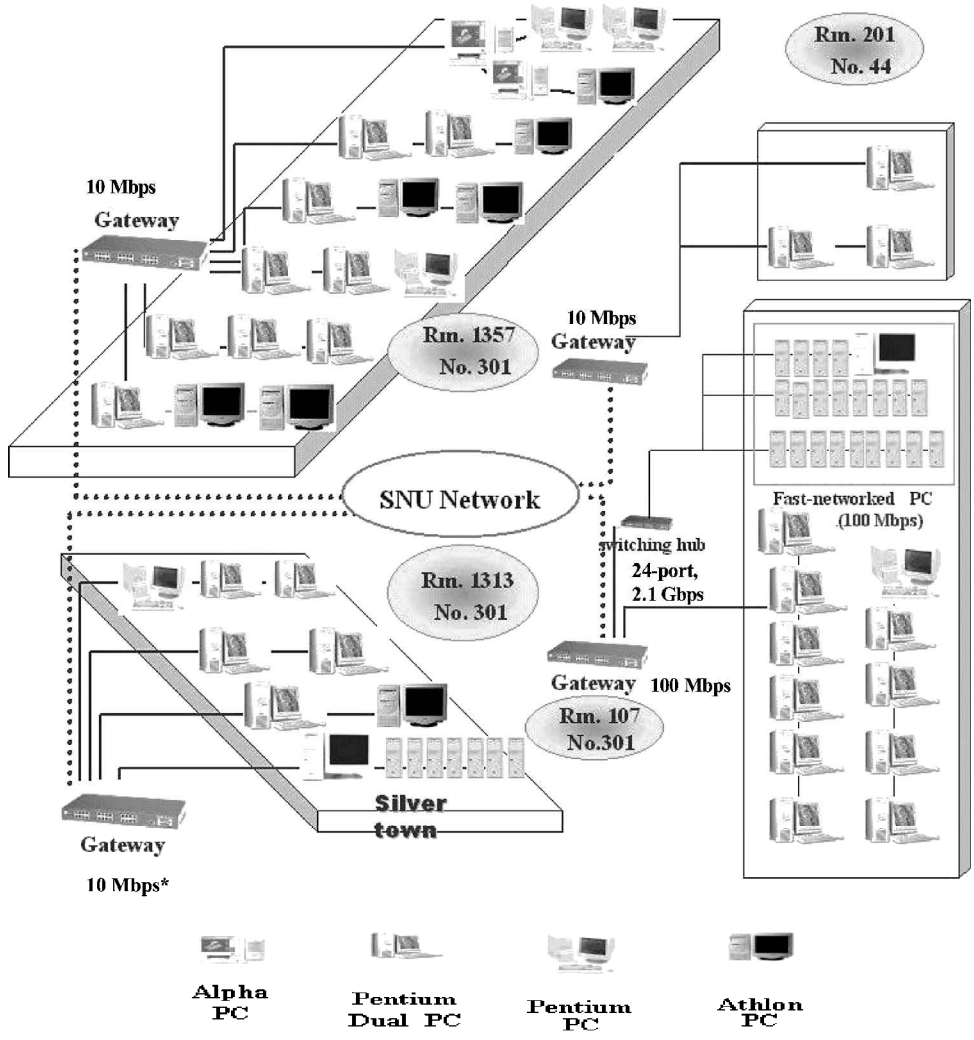

Fig. 12 Locations of personal computers utilized in Internet supercomputing.

Table 5 System and compiler configuration for benchmark problem

CPU	Peak performance (theoretical) Gflops	No. of CPUs used for parallel computing	System and compiler configuration
		IBM SP(Nightawk II)	
Power3 375 MHz	1.5	32	OS: AIX 4.3.3 Compiler: xlc, xlC SP switches and Fast Ethernet
		InterSup I	
Athlon 1 GHz	1	1	OS: Linux (kernel 2.4.0) Compiler: Athlon gcc-2.95.3 Athlon optimization flag[a] Fast Ethernet (100 Mbps)
P-III 866 MHz	0.866	2	OS: Linux (kernel 2.2.18) Compiler: Pentium gcc-2.95.3 Pentium Pro optimization flag[b] Ethernet (10 Mbps)
P-III 800 MHz	0.800	1	
P-III 650 MHz	0.650	4	OS: Linux (kernel 2.2.16) Compiler: Pentium gcc-2.95.3 Pentium Pro optimization flag[b] Ethernet (10 Mbps)
Athlon 550 MHz	0.550	13	OS: Linux (kernel 2.4.0) Compiler: Athlon gcc-2.95.3 Athlon optimization flag[a] Fast Ethernet (100 Mbps)
P-III 550 MHz	0.550	10	OS: Linux (kernel 2.2.16) Compiler: Pentium gcc-2.95.3 Pentium Pro optimization flag[b] Fast Ethernet (100 Mbps)
P-III 450 MHz	0.450	1	

[a]Athlon optimization flag= -O3 -fomit-frame-pointer -Wall -mathlon -march=athlon -mcpu=athlon -malign-functions=4 -funroll-loops -fexpensive-optimizations -malign-double -fschedule-insns2.
[b]Pentiumpro optimization flag= -O3 -fomit-frame-pointer -Wall -mpentiumpro -march=pentiumpro -malign-functions=4 -funroll-loops -fexpensive-optimizations -malign-double -fschedule-insns2.

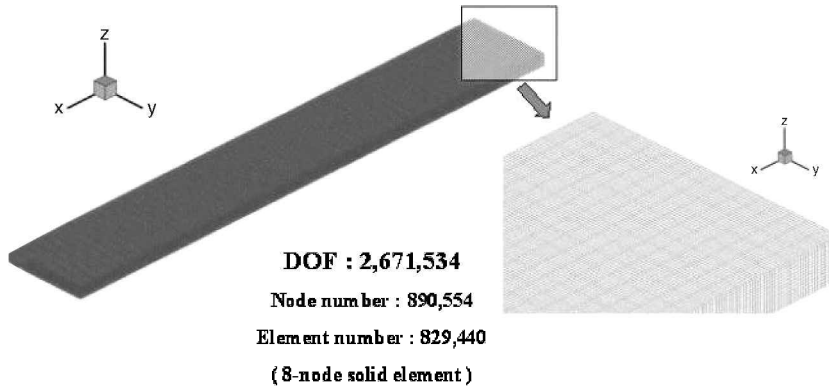

Fig. 13 Finite element model of test problem 1.

the status of each personal computer and acquire the system information including CPU clock, main memory capacity, hard disk usage, and CPU charge rate through simple network management protocol.[12] Job submitting and control can be done by graphic-user interface software based on Gimp Tool Kit (GTK) library and Practical Extraction and Report Language (PERL).

Numerical Examples

Benchmarking tests with a huge number of DOF are performed using the currently proposed Internet supercomputing methodology.

Test Problem 1: Composite Specimen (2.7×10^6 DOF)

The benchmark problem with finite element mesh of composite specimen is performed in dedicated MPP system and also InterSup I system. The IBM SP system [Nighthawk II model, Power3 375 MHz, 144 CPUs, Symmetric Multi-Processors (SMP), High Node: 9 nodes] in Seoul National University is utilized. This IBM SP system was ranked as the 116th in the 16 TOP500 list.[13] The composite specimen is discretized by 0.8×10^6 8-node three-dimensional solid elements as shown in Fig. 13. The total DOF are 2,671,534, and total number of nodes is 890,554. The numerical tensile test in the x direction is carried out by displacement control. The discretized finite element model is shown in Fig. 13, and the partitioned domains allocated to each personal computer are shown in Fig. 14. The benchmark problem is solved by using 32 CPUs through the developed finite element code based on the parallel multifrontal solver. In the two systems, the developed parallel multifrontal solver is compiled as the best optimization. The detail system tuning status, network speed, compiler, and compile optimization flags are given in Table 5. The elapsed time is 26,460 s in the IBM SP system and 31,212 s in the InterSup I. When the difference of network configuration between two systems is considered, this result is very promising and interesting. Therefore, it is evident that the computing powers of personal computers simply connected by the Internet can be treated as an alternative supercomputing power if parallel algorithms are well designed for Internet supercomputing environments and implemented efficiently.

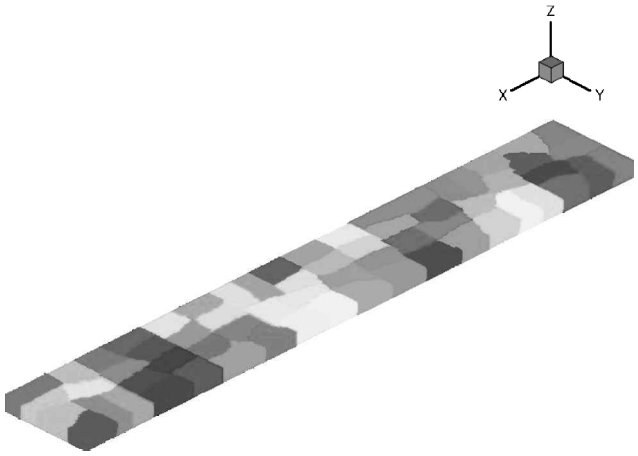

Fig. 14 Partitioned domains allocated to each personal computer.

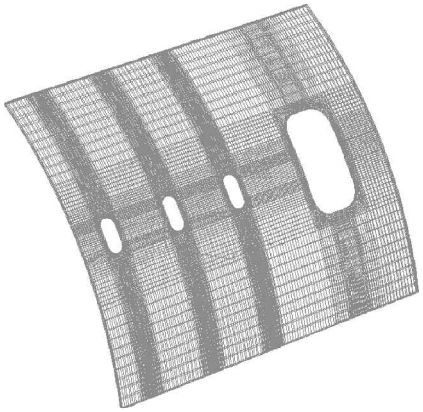

Fig. 15 Finite element method mesh of three-dimensional composite shell structures with cutouts (boron/aluminum composite, four layers).

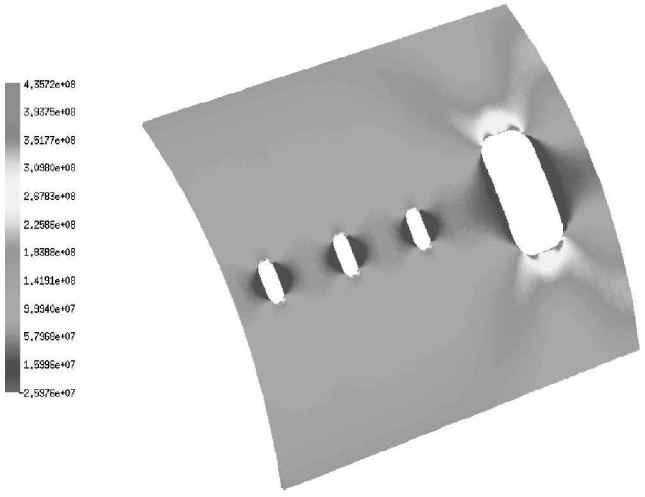

Fig. 16 Normal stress σ_{zz} distributions of three-dimensional composite shell structures with cutouts (boron/aluminum composite, four layers).

Test Problem 2: Three-Dimensional Composite Shell Structures with Cutouts (2.2×10^6 DOF)

Three-dimensional composite shell structures with cutouts are modeled by 151,456 three-dimensional solid 20-noded elements, as shown in Fig. 15, and the stress analysis is performed through the developed virtual supercomputing system InterSup I. Tensile forces are applied along the z direction. The problem size and resources used in computation are as follows: total DOF = 2,224,524; number of elements = 151,456; number of computers = 16; main memory usage = 2.9 GB; and elapsed time = 26,434 s (7 h).

This kind of stress analysis is commonly carried out for the preliminary design and system analysis. However, in most cases, the whole domain of the problem cannot be considered at one time because of the limitation of computing resources. Therefore in usual cases, coarse analysis is performed for the whole model first, and then based on the coarse results, detail analyses are carried out part by part. However, this approach may have a critical limitation in terms of accuracy. In some cases, we have to assume the local stress or strain condition when adopting this conventional analysis approach. Thus, we attempt to solve this example directly by the developed virtual supercomputing system InterSup I. In this case, a total of 16 computers among the candidate resources in Fig. 12 are utilized for the computation. The result of stress analysis of this problem is presented in Fig. 16.

Conclusions

In this study, large-scale structural analyses using a parallel multifrontal solver through Internet personal computers are introduced and realized. A new out-of-core parallel solver based on domainwise or elementwise multifrontal technique is proposed and implemented. When parallel Gauss-elimination algorithms are used, a good parallel performance can be obtained in three-dimensional analysis. In numerical tests, 28.8 speedup is measured in 32 processors of the Cray T3E. Therefore, it could be concluded that the developed parallel multifrontal solver may be widely used in large-scale structural analysis due to its robust stability and its good scalability.

To use the computing power of personal computers simply connected by Internet, a new paradigm, Internet supercomputing technology, is introduced. A virtual supercomputing system, InterSup I, is constructed to materialize the concept of Internet supercomputing. In the construction of InterSup I, LAM based on IMPI is chosen and tuned to handle the heterogeneous system in the Internet. A secure shell is used to obtain the minimum access privilege to carry out the Internet supercomputing. In numerical experiments, the first benchmark problem, which has 2,671,534 DOF, is solved by using 32 CPUs and the elapsed time is 31,212 s. The elapsed time required to solve the same problem in IBM SP system in Seoul National University is 26,460 s. Moreover, the detail stress analysis of three-dimensional composite shell structures with more than 2×10^6 DOF is successfully performed by using 16 CPUs in InterSup I, and the elapsed time is 26,434 s. From these results, it is confirmed that the proposed Internet supercomputing methodology is very promising and has great potential as the next-generation high-performance computing technology. When the rapid advancement of technology in computer hardware and networks is taken into account, Internet supercomputing can be considered as a worthy alternative for supercomputing.

Acknowledgment

This study has been supported by the Ministry of Science and Technology through National Research Laboratory Programs under Contract 00-N-NL-01-C-026.

References

[1] Kim, J. H., and Kim, S. J., "Multifrontal Solver Combined with Graph Partitioners," *AIAA Journal*, Vol. 37, No. 8, 1999, pp. 964–970.

[2] Duff, S., and Reid, J. K., "The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations," *ACM Transactions on Mathematical Software*, Vol. 9, 1973, pp. 302–325.

[3] Irons, B. M., "A Frontal Solution Program for Finite Element Analysis," *International Journal for Numerical Methods in Engineering*, Vol. 2, 1970, pp. 5–32.

[4] Lucas, R. F., Blank, T., and Tiemann, J. J., "A Parallel Solution Method for Large Sparse Systems of Equations," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-6, No. 6, 1987, pp. 981–991.

[5] Ingle, N. K., and Mountziaris, T. J., "A Multifrontal Algorithm for the Solution of Large Systems of Equations Using Network-Based Parallel Computing," *Computers and Chemical Engineering*, Vol. 19, 1995, pp. 671–681.

[6] Geng, P., Oden, J. T., and van de Geijn, R. A., "A Parallel Multifrontal Algorithm and Its Implementation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 149, No. 1–4, 1997, pp. 289–302.

[7]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.

[8]Karypis, G., and Kumar, V., "Metis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Ordering of Sparse Matrices," Dept. of Computer Science, TR, Univ. of Minnesota, Minneapolis, MN, Sept. 1998.

[9]Quinn, M. J., *Parallel Computing Theory and Practice*, 2nd ed., McGraw–Hill, New York, 1994, pp. 229–237.

[10]LAM Team, "LAM/MPI Parallel Computing," Univ. of Notre Dame, Notre Dame, IN, URL: http://www.lam.mpi.org [cited 13 March 2001].

[11]SETI Team, "History of SETI," SETI Inst., Mountain View, CA, URL: http://www.seti-inst.edu/general/history.html [cited 2 Feb. 2000].

[12]Taenbaum, A. S., *Computer Networks*, 3rd ed., Prentice–Hall, Upper Saddle River, NJ, 1996, pp. 630–643.

[13]Meuer, H., Strohmaier, E., Dongarra, J., and Simmon, H. D., "16th TOP500 List," Univ. of Tennessee, Knoxville, TN, URL: http://www.top500.org/list/2000/11 [cited 13 Nov. 2000].

Sunil Saigal
Associate Editor