APPLICATIONS OF HIGH PERFORMANCE COMPUTING NETWORK TECHNIQUES TO SONAR DESIGN

P.C.Macey (1), N.K.Allsopp (2) & A.S.Gill (2)

(1) SER Systems Ltd, 39 Nottingham Road, Stapleford, Nottingham, NG9 8AD
(2) Parallel Applications Centre, 2 Venture Road, Chilworth, Southampton, SO16 7NP

1. INTRODUCTION

Vibroacoustic analysis using finite element (FE) and boundary element (BE) methods were originally used only by large organisations because appropriate software was not readily available and the computational requirement was very great. However with the continuing trends of increasing computer power, decreasing computer cost, refinement of numerical techniques and evolution of commercially available software using these methods, vibroacoustic analysis is now more generally available to all types of organisation. These techniques can be used in many situations including, individual transducer design, transducer array design, scattering/radiation by underwater vehicles, loudspeaker design, analysis of rooms/vehicle interiors for part of the frequency range and many other application areas. However it is in underwater acoustics that the justification for its use is most easily made, because alternative design techniques, based on much testing, can be very expensive. Despite the beneficial changes mentioned above there is still a need to analyse problems which require greater computation than that which is available at easily affordable prices. Using the BE method the surface mesh required is generally related to the acoustic wavelength. Thus if the frequency is doubled then the surface mesh requires 4 X as many patches. The storage in memory/disk and the time needed to set up the equations will increase by a factor of 16 and the time needed to solve the equations, using Gaussian elimination, will increase by a factor of 64. Similar considerations apply to FE meshes. Parallel computing techniques, considered by this paper, enable processors to "add" their computational speed, memory and disk to address higher frequencies/larger problems. However there is a communication overhead in splitting the computation between the processors.

2. PARALLEL COMPUTING

A high performance computing network (HPCN) is a collection of processors linked by a network. They can be viewed as a single computer with the computational task split between the processors. The early supercomputers such as the Cray X/MP in 1986, had this structure. Lower cost HPCN computers based on transputer technology subsequently became available. However writing programs for these multiprocessor systems was an order of magnitude harder than writing code for conventional serial processors. Sometimes the serial nature of an algorithm makes it quite unsuitable for parallel processing. Furthermore communication

between processors was non-standard, and software vendors were reluctant to invest effort creating a parallel version on one particular platform if it would not port onto other systems.

The use of workstation clusters to run parallel applications is becoming a popular alternative to specialised, typically expensive, parallel computing platforms such as the Cray T3E or the IBM SP2. Traditionally a workstation referred to some sort of UNIX platform and PCs were mainly used for administrative tasks, such as word processing. However over the last three years there has been a rapid convergence in high-end workstations and PC based machines. At the same time the Microsoft Windows based operating systems have become more dominant for both home and commercial use. These factors, together with the comparatively low cost of PCs, has led to the utilisation of PC based systems as some form of computational resource for parallel computing.

The Message Passing Interface (MPI) [1], [2] is a portable message-passing standard that facilitates the development of parallel applications and libraries. MPI makes it feasible to transfer code written in high level languages between different types of parallel computer ranging from tightly coupled, massively parallel machines, through to networks of workstations. MPI has a range of features including point-to-point synchronous and asynchronous communication modes; collection and communication (i.e. barrier, broadcast, reduction operations). One of the attractions of MPI is its availability for various platforms. MPICH is a version of MPI built on top of Chameleon [3], which was developed by Argonne National Laboratory and Mississippi State University. This is probably the most popular implementation of MPI being used on UNIX platforms. There are currently three implementations of MPI for Windows NT. WMPI [4] is an implementation of the MPI standard with support for C and FORTRAN for x86-based systems. This has become commercially available under the name PaTENT MPI, marketed by Genias. MPI/Pro is a commercial product from MPI Software Technology Incorporated, a spin-off company from Mississippi State University. It is derived from the WinMPICH project, is implemented over fast Ethernet and VIA hardware, and is specifically designed for x86 and Alpha PCs [5]. The High Performance Virtual Machines (HPVM) project offers MPI on top of Illinois Fast Messages [6]. This is primarily designed to run over high performance Myrinet communications hardware, and is currently used in production compute clusters by NPACI.

3. PARALLEL VIBROACOUSTIC SOLUTION

The algorithm described in this paper and the results included were computed using a parallelization of the PAFEC VibroAcoustics code, created by PACAN-D, EU funded project number 24871. Traditionally a parallel system consists of a master processor and a large number of slave processors. The master shares out the work to the slaves and collects the results after the computational phase of the analysis. Since we were dealing with a small cluster of NT workstations, the parallel algorithm was constructed such that the master acted as a slave during the numerically intensive sections of code. Usually a system is analysed for

many frequencies, and the computations for different frequencies are independent. Hence one strategy would have been to share out the frequencies between the processors. This would have been fairly easy to implement since there would be virtually no communication between processes. However this result can be achieved simply by concatenating the output files from some serial analyses run on sections of the frequency range on unconnected processors. A more ambitious approach was chosen, parallelizing at a lower level, so that all processors are used in the computation of each frequency. Hence the combined resources of cache/memory/disk are available to tackle each frequency, which should be beneficial for very large problems, exceeding the capacity of a single processor, but there is a greater need for communication between processors, which may delay the computational procedure. Profiling revealed, as was originally suspected, that the numerically intensive routines formed a small part of the total number of lines of code. Hence the technique used was to let the master follow the original serial path through the program, but when a computationally demanding section is reached all of the processors are given a similar amount of work. Hence the system is load balanced within the parallel sections of code, whilst the main I/O is dealt with by the master processor only. This is illustrated in figure 1, which shows the underlying methodology of the parallelism of the analysis routines.

The PAFEC VibroAcoustics system comprises a suite of programs called phases. There are 10 phases in total. Phases 1-6 perform pre-processing tasks. Phase 7 performs the main equation solution. Phases 8-10 perform postprocessing tasks. In phase 7 the master processor proceeds through the program just as in the serial version, see figure 1. When it is about to enter a numerically intensive routine a message is sent to the slave processors. All processors then perform an equal part of the required calculation. After the routine the master processor continues through the serial code whilst the slave processors wait for the next numerically intensive section to be reached by the master processor. The efficiency of this technique depends on the time spent communicating information to share out the computation being small compared with the computation time. In many cases it is possible for the data required by the slave processors to be already distributed from the end of the previous calculations. As the size of the analysis increases, both computation and communication increase. However the computation increases more rapidly, and thus it is expected that parallel vibroacoustic analysis will be more efficient for larger models.

The equations for a fully coupled vibroacoustic solution using an acoustic BE mesh coupled to a structural FE mesh are [7]

$$\begin{bmatrix} [S]+i\omega[C]-\omega^2[M] & [T]^T \\ -\omega^2\rho[G][E]^T & [H] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{p\} \end{Bmatrix} = \begin{Bmatrix} \{F\} \\ \{p_I\} \end{Bmatrix} \qquad (1)$$

where {u} is a vector of displacements on the structural mesh, {p} is a vector of pressures on the BE. [S], [C] and [M] are structural stiffness, damping and mass matrices and are large and sparse. [H] and [G] are small dense matrices derived from the BE formulation. [T] and [E] are

coupling matrices. Sometimes the structural representation is simplified by using a smaller, modal model of the structure. The current work was based on a full solution of equation (1) using the 4 stages below.

Stage 1 – FE merging/reduction

The dynamic stiffness matrix and coupling matrix [T] are formed by merging contribution from individual finite elements. The equations are simultaneously solved. Degrees of freedom are eliminated as early as possible. The matrices are shared between processors and the elimination is done in parallel.

Stage 2 – forming the BE matrices

For each collocation point on the BE surface it is necessary to integrate over the surface to form a row in the BE matrices. Parallelization is achieved by sharing these collocation points between the processors. Distributed BE matrices are formed.

Stage 3 – reducing the BE matrices

The matrix $-\omega^2 \rho[G][E]^T$ is formed and reduced using resolution with the structural elimination equations from stage 1. As above the matrices are distributed between the processors.

Stage 4 – Gaussian elimination of final equations

The resulting compact dense set of equations is solved using a parallelized form of Gaussian elimination on the distributed matrix.

4. RESULTS

A piston transducer embedded in a cylindrical baffle was taken as a test problem. The structural model had 10923 degrees of freedom and a front size of 813. The acoustic boundary element had 1664 acoustic degrees of freedom. Both FE and BE models are shown in figure 2. The FE model of the structure is shown in figure 3.

The results were computed on a network of up to four 450Mhz Pentium II PCs each with 128 Mbytes of RAM and linked by a 100Mbit Ethernet. MPI Pro Version 1.2.3 was used. Table 1 gives CPU times in seconds for the 4 stages of a single frequency.

APPLICATIONS OF HPCN TECHNIQUES TO SONAR DESIGN

	1 processor	2 processors	4 processors
Stage 1	2724	1602	1620
Stage 2	242	151	67
Stage 3	4069	2434	1634
Stage 4	209	152	166

Table 1: CPU times for coupled FE/BE analysis of piston transducer in baffle

It should be noted that the timings in stage 2 can be adjusted by altering the integration order. High order integration was used for the run producing table 1. For an analysis over many frequencies stage 2 could also be reduced by interpolating the BE matrices. This also could be done in parallel. The alternative modelling technique, using a modal representation of the structure would require a procedure similar to stage 1 to be performed once and a much reduced form of stage 3 for each frequency. The routines for doing this type of analysis have not been parallelized at the time of writing this paper.

Stage 2, which requires virtually no communication between processors, is efficiently parallelized. Stage 3 requires small amount of communication and is less efficiently parallelized. Stages 1 and 4, which require relatively large amounts of communication actually run slower on 4 processors.

Benchmarking has also been performed for some audio examples. One of these was analysing a point source in a room. This used only BE. The model, composed of 912 quadratic patches with 2738 nodes, is shown in figure 4. Without FE there is no stage 1 or 3. Solution times for a single frequency are shown in table 2. A comparison between timings for MPIPro 1.2.3 and WMPI 4.09 is also made.

	1 Processor		2 Processors		3 Processors		4 Processors	
	Pro	WMPI	Pro	WMPI	Pro	WMPI	Pro	WMPI
Stage 2	661	661	341	341	218	222	173	174
Stage 4	917	917	596	552	449	452	491	384

Table 2 : BE analysis time for point source in room. Pro is MPIPro 1.2.3 and WMPI is PaTENT 4.09

Comparing tables 1 and 2 it is seen that both stages 2 and 4 are more efficiently parallelized on larger problems. Table 2 shows that WMPI performs better on stage 4. Other results for analysis of a loudspeaker [8] show that MPI/Pro performing better on stage 1. This is understood by considering the latency and bandwidth of low level communications, shown in table 3. Stage 1 requires more smaller messages compared with stage 4, which has fewer longer messages. PaTENT takes longer than MPI/Pro to set up a message, but the data transfer rate is greater.

Interconnect (100Mbit Ethernet)	Latency μsec	Asymptotic bandwidth Mbytes/sec
MPIPro 1.2.3	209	4.4
PaTENT 4.09	259	7.1

Table 3: communications performance for unidirectional MPI Ping-Pong benchmark

It should be borne in mind that the performance of MPI software will probably improve with further releases.

5. CONCLUSIONS

A network of fast PCs has been shown to be a useful tool for solving big vibroacoustic analysis problems. The combined power and speed should enable large problems, such as interaction problems for large arrays or analysis at high frequency with a fine mesh to be performed using cost effective hardware.

6. REFERENCES

[1] M.Snir, S.Otto, S.Huss-Lederman, D.Walker & J.Dongarra, "MPI The complete reference", MIT Press 1996

[2] "Message Passing Interface Forum, MPI: A Message-Passing Interface Standard", 5th May 1994, University of Tennesse, Knoxville, Report No. CS-94-230

[3] W.Gropp & B.Smith, "Chameleon parallel programming tools users manual", Technical Report ANL-93/23 Argonne National Laboratory, March 1993.

[4] M.Marinho & J.G.Siva, "WMPI Message Passing Interface for Win32 Clusters", Instituto Superior de Engenharia de Coimbra, Portugal and Departmento de Engenharia Informatica, Universidada de Coimbra, Portugal

[5] S.Pakin, V.Karamcheti & A.A.Chien, "Fast Messages (FM): Efficient Portable Communications for Workstation Clusters and Massively-Parallel Processors", IEEE Concurrency, Vol 5 No 2, pp 60-73, 1997

[6] A.Chien, S.Scott, M.Lautia, M.Buchanan, K.Hane, L.Giannini & J.Prusakova, "High Performance Virtual Machines (HPVM): Clusters with Supercomputing APIs and Performance", Eighth SIAM Conference on Parallel Processing for Scientific Computing (PP97) 1997

[7] P.C.Macey,"Finite element/boundary element modelling techniques applied to ring transducers with viscoelastic coating", Ferroelectrics, Vol 187 1996 pp 201-211

[8] P.C.Macey,J.R.Wright & N.K.Allsopp,"High performance computing network techniques for vibroacoustic analysis: Audio applications", 106th Convention Audio Engineering Society, Munich, May 1999

APPLICATIONS OF HPCN TECHNIQUES TO SONAR DESIGN

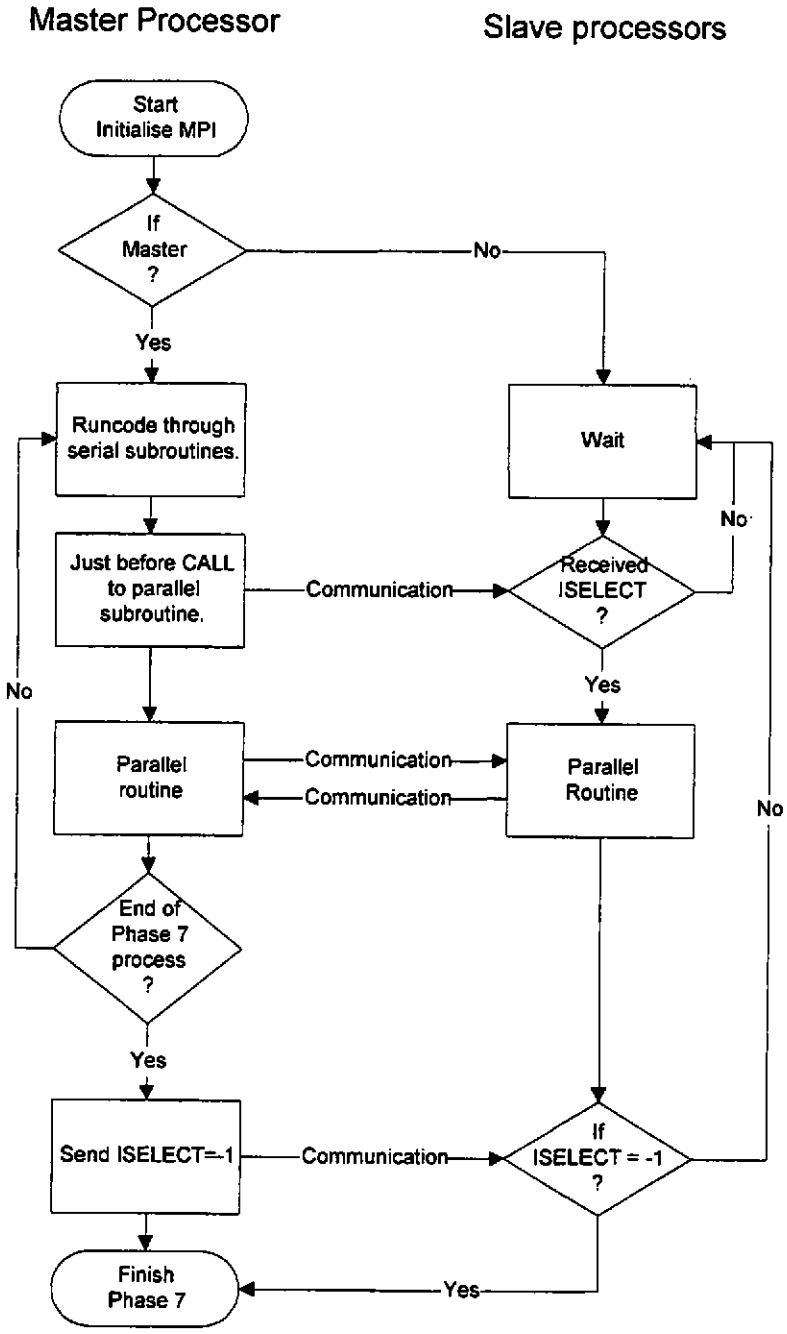

Figure 1 : Flow diagram of overall methodology of serial code parallelism

APPLICATIONS OF HPCN TECHNIQUES TO SONAR DESIGN

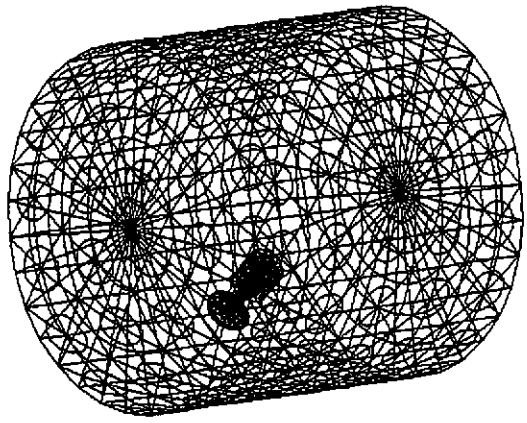

Figure 2: Finite/Boundary element model of piston transducer in cylindrical baffle

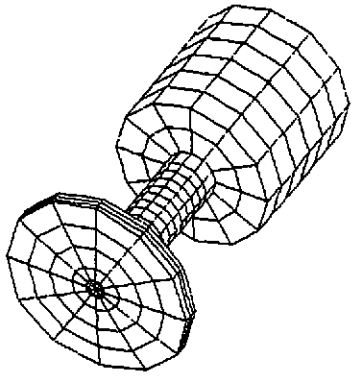

Figure 3: Finite element model of piston transducer

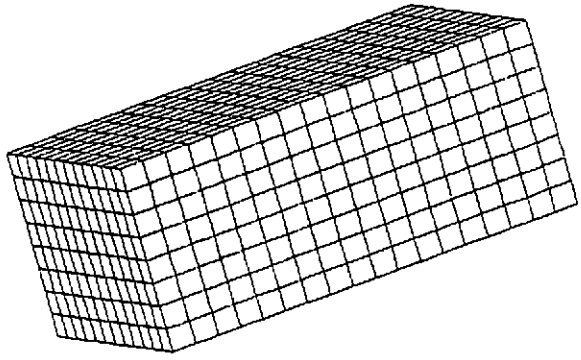

Figure 4: Boundary element model of room