AIAA JOURNAL
Vol. 30, No. 12, December 1992

Programming Probabilistic Structural Analysis for Parallel Processing Computers

Robert H. Sues* and Heh-Chyun Chen†
Applied Research Associates, Inc., Raleigh, North Carolina 27165
and
Christos C. Chamis‡ and Papu L. N. Murthy§
NASA Lewis Research Center, Cleveland, Ohio 44135

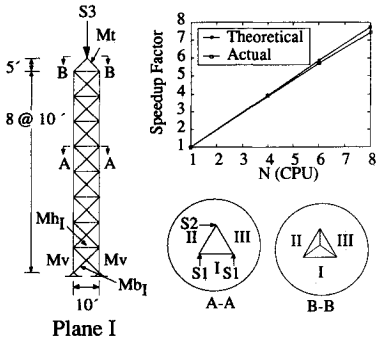
Abstract

SIGNIFICANT advances have taken place in probabilistic structural analysis (PSA) over the last two decades. Much of this research has focused on basic theory development and the development of approximate analytic solution methods. Practical application of PSA methods has been hampered, however, by their computationally intense nature. Solutions of PSA problems require repeated analyses of structures that are often large and exhibit nonlinear and/or dynamic response behavior. PSA methods, however, are all inherently parallel and ideally suited to implementation on parallel processing computers.

This synoptic[1] summarizes the results of the first phase of research to develop a parallel processing system that can significantly reduce the computational times for large-scale PSA problems. Parallel processing improves the speed with which a computational task is done by breaking it into subtasks and executing as many as possible of these subtasks simultaneously. A summary of the principal ideas in parallel processing and a survey of currently available architecture appears in Sues et al.[2] We identify here the multiple levels of parallelism in PSA, describe the development of a parallel stochastic finite element code, and present results of an example application. Although the example application achieved both excellent speedups and greater than 95% efficiency, we conclude that achieving massive parallelism will require overcoming limitations of current parallel architectures. New hardware/software strategies must be developed that keep large numbers of processors busy while minimizing memory requirements and interprocessor communication. Hence, we provide generic hardware and software recommendations for achieving large-scale parallel PSA implementation.

Contents

Presented as Paper 91-0920 at the AIAA 32nd Structures, Structural Dynamics, and Materials Conference, Baltimore, MD, April 8-10, 1991; received June 13, 1991; synoptic received Dec. 20, 1991; accepted for publication Dec. 28, 1991. Full paper available from AIAA library, 555 W. 57th Street, New York, NY 10019.



Notes: Eight independent material types, M1 - M8; Random Variables: Modulus (lognormal, $\delta = 0.15$); Area (lognormal, $\delta = 0.10$); Loading (lognormal, $\delta = 0.25$); Initial Strain (normal, zero mean, $\sigma = 10^{-4}$), δ = coefficient of variation, σ = standard deviation.

Fig. 1 Three-dimensional space truss example.

scoring (which requires a carry-around scalar) are not. Hence, special strategies are required to enable parallel implementation so that random-number sequences are not duplicated and so that erroneous scoring of results does not occur.

For random-number generation, a strategy developed and implemented in McPAP is to establish an independent stream of random numbers for each processor. Random seeds are generated for each processor using a different random-number generator. Since individual simulation histories can take different lengths of time to execute, and histories are dynamically allocated to the next available processor, a function call is first executed (on each invocation of a simulation trial) that returns the number of the processor that is allocated to the particular trial. The processor number is used to fetch the previous set of random numbers generated by this processor, which is then used to generate the next set of random numbers. A similar strategy is used to address the problem of scoring.

An alternative approach for parallel random-number generation is that presented in Fox et al.[3] Although it is more complex than the previous approach, it has the advantage that it results in generation of exactly the same stream of random numbers as would be generated on a serial computer, and it does not require a machine-dependent library function call. This is accomplished by employing the fundamental relation between the $n+k$th and nth random numbers for congruential random-number generators.

Example Application

Figure 1 shows a three-dimensional space truss composed of 99 members with 73 degrees of freedom and the speedups obtained in computing the cumulative distribution function (non-exceedance probability) for element stress. For this problem, 1000 Monte-Carlo histories were performed and the fraction of the computations that cannot be performed concurrently is 0.5%. The results show that the efficiency (ratio of actual speedup to theoretical speedup) is high, exceeding 96% for eight processors. The observed decrease in efficiency as the number of processors increases is because the overhead increases with the number of processors as expected.

Conclusions and Recommendations

Although the example implementations performed in this research were highly successful, new hardware and software strategies will be required to achieve massively parallel implementations for many practical applications. This is a result of two factors—concurrency overhead and storage requirements. First, concurrency overhead must be kept extremely small to achieve reasonable efficiency for massively parallel applications. Second, multiple levels of parallelism need to be exploited since storage requirements can easily exceed available memory if only one level of parallelism is implemented. When multiple concurrent solutions are performed in a parallel PSA code, so that each processor is assigned an independent performance function evaluation, the memory required is multiplied by the number of processors. Although secondary storage can be used (i.e., disk storage), its use will significantly degrade the parallel efficiency. Using multiple levels, multiple processors can work on a single performance function evaluation to reduce memory requirements.

The key conclusions and recommendations of this study are as follows:

1) A wide range of parallel architectures are currently available; however, none are ideally suited for massively parallel implementation of PSA.

2) There are several levels of parallelism in PSA problems that may need to be taken advantage of in order to fully exploit the potential of parallel processing computers.

3) Very high efficiency (greater than 96% for eight processors) can be achieved for parallel Monte-Carlo PSA codes.

4) To achieve large-scale parallelism, distributed memory machines will likely be preferable. Each distributed processor should be capable of performing the numeric floating-point operations required in structural mechanics problems with high speed and would ideally have vector pipeline capability.

5) For a distributed memory system, the interprocessor communication topology must be flexible enough (dynamic) to allow for direct communication (or memory sharing) among small clusters of the processors. This will allow more efficient handling of multiple levels of parallelism.

6) Specially adapted numerical techniques are required for efficient parallel implementation of many practical problems in order to reduce memory requirements and processor idling. An example of such an approach is the stochastic preconditioned conjugate gradient method (SPCG).[1,2] This method is obtained by using the mean stiffness matrix to precondition the equations that are solved in each Monte-Carlo history. Once preconditioned, the equations are solved using the SPCG method. The method is easily implemented in parallel, requires minimal storage, and converges rapidly.

7) Controlling software must be developed to optimally allocate the multiple levels of parallelism among the processors to minimize processor idling and achieve maximum speedup.

8) Availability of parallel computers with properly adapted software shows excellent potential for practical turnaround time on large-scale PSA problems.

Acknowledgment

This research was supported by the NASA Lewis Research Center under Contract NAS3-25824. This support is gratefully acknowledged.

References

[1]Sues, R. H., Chen, H-C., Twisdale, L. A., Chamis, C. C., and Murthy, P. L. N., "Programming Probabilistic Structural Analyses for Parallel Processing Computers," *Proceedings of the AIAA/ASME/ASCE/AHS/ASC 32nd SDM Conference*, AIAA, Washington, DC, 1991.

[2]Sues, R. H., Chen, H-C., and Twisdale, L. A., *Probabilistic Structural Mechanics Research for Parallel Processing Computers*, NASA-CR-187162, 1991.

[3]Fox, G. C., et al., *General Techniques and Regular Problems*, Vol. I, Solving Problems on Concurrent Processors, Prentice-Hall, Englewood Cliffs, NJ, 1988.