An Efficient Communication Strategy for Finite Element Methods on the Connection Machine CM-5 System

An Efficient Communication Strategy for Finite Element Methods on the Connection Machine CM-5 System

Zdeněk Johan
Kapil K. Mathur
S. Lennart Johnsson
Thomas J.R. Hughes

TR-06-93

March 1993

Parallel Computing Research Group

Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts

To appear in *Parallel CFD '93*, Paris, France, May 10–12, 1993.

An Efficient Communication Strategy for Finite Element Methods on the Connection Machine CM-5 System

Zdeněk Johan, Kapil K. Mathur, S. Lennart Johnsson
Thinking Machines Corporation
245 First Street, Cambridge, MA 02142–1264

Thomas J.R. Hughes
Division of Applied Mechanics, Stanford University
Stanford, CA 94305–4040

Performance of finite element solvers on parallel computers such as the Connection Machine CM-5 system is directly related to the efficiency of the communication strategy. The objective of this work is two-fold: First, we propose a data-parallel implementation of a partitioning algorithm used to decompose unstructured meshes. The mesh partitions are then mapped to the vector units of the CM-5. Second, we design gather and scatter operations taking advantage of data locality coming from the decomposition to reduce the communication time. This new communication strategy is available in the CMSSL [8]. An example illustrates the performance of the proposed strategy.

Partitioning algorithm

The recursive spectral bisection algorithm [3, 6] has been implemented on the CM-5 in a data-parallel fashion. This implementation is an enhanced version of the one developed for the CM-2 [1]. In this implementation, a two-level parallelization is applied both to the partitions generated at a given stage of the recursive process and to the elements in each partition. This prevents any loss of performance during the recursive process since the CM-5 always processes the same number of data, namely the number of elements in the whole mesh.

The dual mesh connectivity is used to define the Laplacian matrix of the graph. The smallest non-zero eigenvalue of the Laplacian matrix and its associated eigenvector are then computed using the Lanczos algorithm. A careful mapping of the arrays in the Lanczos inner loop restricts inter-processor communication to dot-products and matrix-vector multiplications. The latter can be reduced to scatter operations, and dot-products are expressed using a scatter followed by a gather. Both gather and scatter operations are performed using the utility routines `sparse_util_gather` and `sparse_util_scatter` provided by the CMSSL. Most of the algorithm has been implemented in CM Fortran and C. However, the computation of the smallest eigenvalue of the tridiagonal matrix generated by the Lanczos process has been written in CDPEAC (a macro-assembler) to achieve maximum performance for this critical part of the algorithm.

Gather/scatter operations

The subdomains generated by the partitioning algorithm can be viewed as meshes independent of one another. Each vector unit has its own mesh with local element and node numberings. The gather operation is then performed in two steps:

1. A global (off-processor) gather operation is executed between the global set of nodes (i.e., the nodes for the whole mesh) and the local sets of nodes (i.e., the nodes for each partition). The CMSSL routines used for the implementation of the partitioning algorithm are also used for this operation.

2. A local gather operation is then executed on each vector unit between the local sets of node and elements. The local gather requires *no* inter-processor communication. It has been implemented in C with calls to low-level library routines to take full advantage of indirect addressing available on the vector units.

The scatter operation is performed in a similar fashion by having a local scatter followed by a global scatter. It can be shown that, for tetrahedral meshes, this two-step procedure can reduce the number of data to be sent to the network by up to a factor of 20 over a strategy not taking advantage of data locality. Substantial speed-ups over the default communication scheme can therefore be expected.

Numerical example

The partitioning algorithm and associated gather/scatter routines were incorporated in a finite element program solving the 3-D compressible Euler and Navier-Stokes equations [2]. Convergence is achieved using a fully implicit matrix-free solver based on the preconditioned GMRES algorithm [1, 4, 5]. In this example, the 3-D inviscid shock-shock interaction on a swept cylindrical leading edge is computed [7]. The sweep angle is 15 degrees. Incoming flows at Mach 8.03 and Mach 5.25 are separated by an impinging shock. The unstructured mesh has 16,707 nodes and 86,701 tetrahedra (see Figure 1). One integration point per element is used and 100 time steps are performed, with an average of 10 residual evaluations per time step. This problem is solved on a 32-processing node CM-5 equipped with 128 vector units. Two communication strategies are used: Strategy 1 calls the CMSSL communication primitives `sparse_util_vec_gather` and `sparse_util_vec_scatter` which perform the gather/scatter operations by simply sending all data to the network. On the other hand, Strategy 2 partitions the mesh and calls the two-step CMSSL gather/scatter routines described in the previous section. The partitioning into 32 subdomains is shown in Figure 2 (128 partitions are actually needed for a 32-processing node CM-5). Timings for both strategies are presented in Table 1. The reduction in communication time by taking advantage of data locality is particularly striking. A factor of 2 speed-up for the overall code can be achieved, even when including the partitioning time. In the case of Strategy 2, the effective bandwidths per processing node for the gather and scatter are 7.2 Mbytes/s and 5.1 Mbytes/s, respectively. Note that these bandwidths are weighted averages between memory bandwidth and network bandwidth.

Conclusions

We have shown that substantial improvements in the overall performance of finite element solvers can be achieved by proper mapping of data to the vector units of the CM-5 and by taking advantage of that mapping in the design of the communication routines. It is also important to note that, given a mesh partitioner and a set of communication primitives, the finite element program can be architecture-independent. We expect that additional improvements in the implementation will further reduce the communication times in the near future. Finally, the scalability of this communication strategy is currently under evaluation.

References

1. Z. Johan, "Data parallel finite element techniques for large-scale computational fluid dynamics," *Ph.D. Thesis*, Stanford University, 1992.

2. Z. Johan, T.J.R. Hughes, K.K. Mathur and S.L. Johnsson, "A data parallel finite element method for computational fluid dynamics on the Connection Machine system," *Computer Methods in Applied Mechanics and Engineering*, **99** (1992) 113-134.

3. A. Pothen, H.D. Simon and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal of Matrix Analysis and Applications*, **11** (1990) 430–452.

4. Y. Saad and M.H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal of Scientific and Statistical Computing*, **7** (1986) 856–869.

5. F. Shakib, T.J.R. Hughes and Z. Johan, "A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis," *Computer Methods in Applied Mechanics and Engineering*, **75** (1989) 415-456.

6. H.D. Simon, "Partitioning of unstructured problems for parallel processing," *Computing Systems in Engineering*, **2** (1991) 135–148.

7. R.R. Thareja, K. Morgan, J. Peraire and J. Peiro, "A three-dimensional upwind finite element point implicit unstructured grid Euler solver," *AIAA 27th Aerospace Sciences Meeting*, paper AIAA-89-0658, 1989.

8. *CMSSL for CM Fortran, Version 3.1*, Thinking Machines Corporation, Cambridge, MA, 1993.

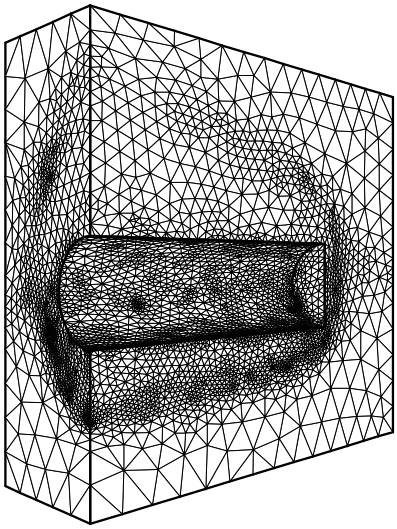

Figure 1. Cylindrical leading edge. Surface mesh.

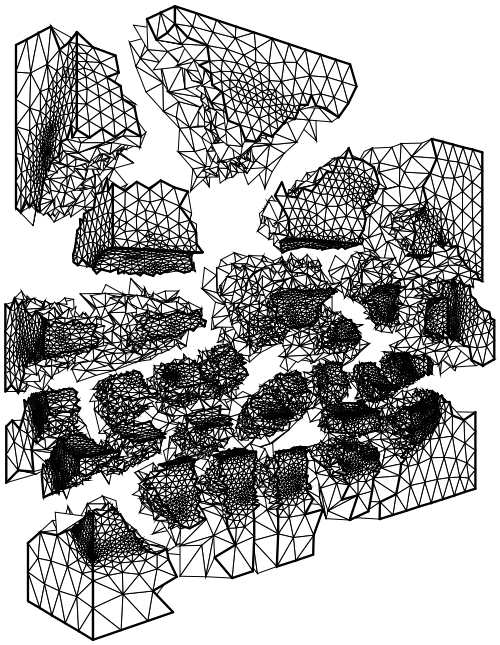

Figure 2. Decomposition into 32 partitions.

Table 1. Cylindrical leading edge. Timings on a 32-processing node CM-5 system.

	Strategy 1	Strategy 2
Partitioning	—	96 s
Gather	375 s	65 s
Scatter	509 s	125 s
Computation	363 s	353 s
Total	1247 s	639 s