SMG on SMP Clusters: Performance Issues

Sue Goudy, Lorie Liebrock, and Steve Schaffer
New Mexico Institute of Mining and Technology
Socorro, New Mexico 87801
spgoudy@nmt.edu

February 3, 2003

Abstract

The symmetric multiprocessor (SMP) appears as a unit in systems from desktop computers to massively parallel systems. The focus of this paper is the performance of a particular algorithm, two-dimensional semicoarsening multigrid, on a cluster of SMPs. Complexity models for hybrid parallelization of a portion of this algorithm are derived. We examine system parameters, such as the capability for simultaneous message-passing in a symmetric multiprocessor, that can affect the performance of hybrid code. Complexity estimates are tested for a variety of decomposition strategies, line solve methods, and problem sizes. Results from the Intel Teraflops supercomputer and from a Beowulf cluster of dual-processor Xeons are presented.

Clusters of symmetric multiprocessors exist in many sizes and forms, from massively parallel supercomputers such as the Intel Teraflops computer to multiprocessor workstations linked on a network. In between are moderately parallel clusters such as the VPplant visualization cluster at Sandia National Laboratories. Performance modelling of algorithms for these computing platforms is a subject of current research.

Parallel computers can be classified on the basis of the tightness with which the processors are coupled in their ability to communicate with each other. In a symmetric multiprocessor system, the intercommunication occurs via a bus which is internal to the system. In a cluster the intercommunication occurs across a network: this can be a high speed interconnect as in Intel Teraflops or some relatively slow connection such as ethernet in a local area network. Some massively parallel computers, for example the “Q Machine” at Los Alamos National Laboratory, employ a hybrid architecture in which the machine is a cluster of symmetric multiprocessors.

Within a symmetric multiprocessor the software paradigm for interprocess communication can take a variety of practical forms. For simplicity, these forms are herein categorized as shared memory; there must be a mechanism for distinguishing shared data from local data. For a clustered system of computers all internode data sharing takes place via explicit interchange or message passing. A hybrid, or mixed-mode, parallel programming paradigm combines threads with utilization of message passing libraries.

This research focuses on the development of a performance model for hybrid parallelism that combines shared memory and message passing models. It is necessary to evaluate the prediction of performance given by the model to actual computation in order to evaluate the correctness of the model. Semicoarsening multigrid (SMG) is a robust numerical technique for solution of elliptic partial differential equations in two and three dimensions. After parallelization, the algorithm developed by Schaffer [5] contains both coarse grained and fine grained parallel execution paths. This character makes SMG a good basis for experimentation in a clustered SMP environment.

Semicoarsening multigrid is a V-cycle multigrid method for the solution of linear system arising from the discretization of a partial differential equation. These methods can be implemented to run on parallel computers; however, a significant sequential component exists in the V-cycle. The levels, or discretization scales, are processed sequentially with parallel computation occurring within a level.

A V-cycle in our implementation of Schaffer’s algorithm uses block Gauss-Seidel relaxation to reduce oscillatory error components. In this research we investigate the system characteristics that can enhance speedup of SMG. Since this algorithm has been shown to exhibit good convergence by other researchers

[1, 2], we present no convergence analysis. Restriction is defined to be the transpose of operator-induced interpolation for intergrid transfers. In our pseudcode for semicoarsening multigrid, the operators relaxation, prolongation, and restriction are denoted by G, P, and R. The coarse grid operator L^{2h} is defined as the matrix product RL^hP. In our implementation, direct solution of the coarse-grid equation is replaced by a call to $SMG2()$, resulting in a recursive V-cycle.

Experiments on a single processor have demonstrated that relaxation accounts for roughly ninety percent of computaton time for a V-cycle. In this sense we can regard the computational time of relaxation as representative of the entire algorithm. We confine our hybrid parallel study to the routine $RELAX2()$. Our implementation of block Gauss-Seidel relaxation proceeds in two phases: preparation of the right hand side and the subsequent line solve. We investigate four tridiagonal solvers: the Thomas algorithm, cyclic reduction, Wang's partition method, and a block multithreaded variant of Wang's method.

Conversion of Schaffer's algorithm to run on distributed memory computer systems proceeded in a series of steps. The Single Program Multiple Data (SPMD) model was chosen for implementation design. It was assumed that domain decomposition onto the parallel computer would be done prior to invocation of $SMG2$. The Message Passing Interface (MPI) library [3] was used to generate explicit transfer of information at subdomain boundaries. The first version of parallel relaxation was designed to work only for strip decompositions that did not partition a tridiagonal solve across processors; the Thomas method was suitable for the line solves. Next, parallel implementations of cyclic reduction and Wang's partition method were added. MPI optimization was limited to overlapping computation and communication. To simplify bookkeeping for the nearest neighbor communications as the grids become coarser, split communicators were employed. Thread capability was added using OpenMP directives [4].

Development of the model of hybrid parallelization and our computational results are presented in the full paper.

The Thomas algorithm and the strip decomposition of the problem domain shows the fastest execution time for our fixed size test cases. This combination appears to be suitable on clusters of modest size. The number of processors must be chosen to allow the strips to contain sufficient work to balance the cost of communication. The relative speeds of floating-point operations and communication network functions determine the efficiency of a domain to processor mapping.

In cases where it is infeasible to decompose the problem domain into strips, the block Wang algorithm can be used to minimize communication events during the line solves. For example, large numbers of processors cannot be applied effectively to a strip decomposition. Also, the decomposition in the user code may dictate the mapping strategy for semicoarsening multigrid. That is, the user may have selected a decomposition based on the physical arrangement of the domain. Remapping that decomposition for a call to our routine might be too expensive with respect to memory usage.

Efficient coding for hybrid parallel computation increases complexity. In some cases, to obtain the best efficiency in threaded portions of the code, it is necessary to rewrite algorithms. Our orignal implementation of $RELAX2$ set up the right hand side of a tridiagonal system and called the solver for each line on a processor. Threading of the line solver at the loop level was ineffective. Creation of a block solver for multiple simultaneous tridiagonal systems allocated more work to the threads. Hybrid parallelization of a small part of semicoarsening multigrid produced modest performance gains for some domain to processor mappings. Extension of the thread parallel section to include most of $SMG2$ is one of our current goals.

References

[1] C. Baldwin, P.N. Brown, R. Falgout, F. Graziani, and J. Jones. Iterative linear solvers in a 2D radiation-hydrodynamics code. *Journal of Computational Physics*, 154(1):1–40, September 1999.

[2] P.N. Brown, R.D. Falgout, and J.E. Jones. Semicoarsening multigrid on distributed memory machines. *SIAM Journal on Scientific Computing*, 21(5):1823–1834, 2000.

[3] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface.* MIT Press, 1994.

[4] OpenMP Architecture Review Board, http://www.openmp.org/specs/. *OpenMP Fortran Application Program Interface*, February 2003.

[5] S. Schaffer. A semicoarsening multigrid method for elliptic partial differential equations, with highly discontinuous and anisotropic coefficients. *SIAM Journal of Scientific Computing*, 20(1):228–242, 1998.