MASSIVELY PARALLEL IMPLEMENTATION OF THE PENN STATE/NCAR MESOSCALE MODEL*

Ian Foster
John Michalakes

Argonne National Laboratory
Chicago, Illinois

1. INTRODUCTION

Parallel computing promises significant improvements in both the raw speed and cost performance of mesoscale atmospheric models. On distributed-memory massively parallel computers available today, the performance of a mesoscale model will exceed that of conventional supercomputers; on the teraflops machines expected within the next five years, performance will increase by several orders of magnitude. As a result, scientists will be able to consider larger problems, more complex model processes, and finer resolutions.

In this paper, we report on a project at Argonne National Laboratory that will allow scientists to take advantage of parallel computing technology. This Massively Parallel Mesoscale Model (MPMM) will be functionally equivalent to the Penn State/NCAR Mesoscale Model (MM). In a prototype study, we produced a parallel version of MM4 using a static (compile-time) coarse-grained "patch" decomposition. This code achieves one-third the performance of a one-processor CRAY Y-MP on twelve Intel i860 microprocessors. The current version of MPMM is based on MM5 and uses a more fine-grained approach, decomposing the grid as finely as the mesh itself allows so that each horizontal grid cell is a parallel process. This will allow the code to utilize many hundreds of processors. A high-level language for expressing parallel programs is used to implement communication streams between the processes in a way that permits dynamic remapping to the physical processors of a particular parallel computer. This facilitates load balancing, grid nesting, and coupling with graphical systems and other models.

In the rest of this paper, we first introduce the Penn State/NCAR model and discuss issues that must be addressed in a massively parallel implementation. We then discuss the prototype parallel MM4 and the subsequent fine-grained parallelization of MM5 to produce MPMM.

*This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and was performed in part using the Intel Touchstone Delta System operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by Argonne National Laboratory.

1.1 Penn State/NCAR Model

The Penn State/NCAR Mesoscale Model is a limited-area primitive equation model, designed to simulate meso-alpha scale (200–2000 km) and meso-beta scale (20–200 km) atmospheric circulation systems (Anthes, 1987; Grell et al., 1992). It has been developed over a period of twenty years, first at the Pennsylvania State University and more recently also at the National Center for Atmospheric Research. The time integration uses a fourth-order finite difference scheme (Brown and Campana, 1978) with optional split-explicit time stepping. Model physics is highly configurable across a variety of convection, clouds, planetary boundary layer, radiative transfer, and other standard and optional parameterizations. The model may be run nonhydrostatically for high-resolution simulations (Dudhia, 1992). The model provides a nested-grid capability. A four-dimensional data assimilation module allows observed weather data dynamically incorporated into a simulation.

The MM is used for a range of computationally demanding applications, including real-time forecasting, storms research, and climate research. MM real-time forecasts provide frequent weather updates to aviation users, the general public, and specialty users interested in detailed forecasts in localized areas. However, vertical resolution is currently limited by available processing power. The MM is being used to investigate the development of damaging storms in the Midwest and cyclones in the tropics. Storm studies require the ability to switch to higher-resolution nested grids when advanced convective modeling processes engage for a developing system. Researchers working to improve the reliability of long-range climate change projections for a region couple MM runs with data from general circulation models. Climate change research demands the ability to model extremely long periods — on the order of decades — within reasonable limits of run time and cost. The flexibility and range of applications for MM reflect the diversity of its user community, which is drawn from universities, government agencies, laboratories, and other institutions.

The need for performance has up to now been met by designing and optimizing the model for vector supercomputers. For example, MM4 executes at a rate of 50,000 grid point steps per CPU second on a single CRAY Y-MP processor. Hence, a computation involving 50,000 grid points requires one second per time step, and just under 5 hours for a 30-day simulation with a 150-second time step. This performance is adequate for many current problems. However, while multitasking can increase performance by perhaps an order of magnitude, technological and physical constraints limit the absolute performance that can be attained by conventional supercomputer architectures. In addition, the reliance on custom components makes this approach very expensive. Hence, the MM is costly to run and is near its upper limit of performance on current problems. Clearly, new approaches are required to deal with the larger and more complex problems that will be of interest in the future.

1.2 MPP Approach

Massively Parallel Processing (MPP) achieves high performance by using, instead of one or a small number of very expensive vector processors, hundreds or thousands of inexpensive microprocessors. This approach is already competitive with conventional supercomputers and is far from reaching its limits; architectures capable of scaling to teraflops peak performance have already been announced by Intel and Thinking Machines Corporation, and teraflops computers should be available within five years. These successes build on recent developments in interconnect technology and microprocessor design. High-speed interconnects and cut-through routing allow inexpensive communications even to remote processors, while modern RISC-based microprocessors already match the scalar performance of high-end supercomputers and are likely to improve further with on-chip vectorization and pipelining.

Finite difference codes such as the MM have proven particularly well suited to parallel implementation, because of their regular nearest-neighbor communication pattern. However, one potential source of difficulty in MPP codes is parallel inefficiency resulting from uneven distribution of work to processors (load imbalance). The recoding and run-time costs to restore balanced performance by redistributing work can be quite complex if data domains are decomposed statically. This can be done efficiently and with a minimum of recoding by using a fine-grained decomposition and software tools that abstract out mechanisms for redistributing work between processors from the parallel model code. One such tool is Program Composition Notation (PCN), a high-level language for expressing parallel programs (Foster et al., 1992). Parallel processes and communication streams between parallel processes are easily and abstractly represented in this language.

Another benefit of MPP software tools such as PCN is that they provide portability over a wide range of parallel computers. Standardization of simple message-passing and parallel process paradigms makes it feasible to develop a single program that will execute with reasonable efficiency on an Intel Paragon, a Thinking Machines CM-5, a CRAY C-90, a network of RISC workstations, and most other high-performance computing platforms.

2. PROTOTYPE PARALLEL MM4

A prototype MPP implementation of version 4 of the MM code has been developed at Argonne National Laboratory. This is based on a one-dimensional west/east decomposition of the model grid and executes at about one-third the speed of a 1-processor CRAY Y-MP on 12 i860 microprocessors. The model grid was divided into p equally sized partitions, two of which represented the eastern and western boundaries of the grid, with the remainder representing the interior. Each partition was mapped to a physical processor on a parallel computer. Off-processor data was stored in array extensions and kept up to date by message passing (Figure 1). The widths of these extensions were determined by MM4's finite differencing stencil.

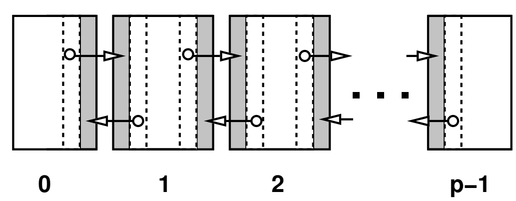

Figure 1: Static decomposition of MM4 grid. Data communication was through extended array "pads" that replicate off-processor memory.

The parallel code reads initial data on a single processor and then distributes it to the other processors. This is a potential bottleneck on a large machine but avoids the need to decompose the initial data set. Model output follows a reverse path. Although its principal purpose was as a prototype for exploring parallel processing issues, the parallel version of MM4 has been used to produce 8 months of climate data for the Pacific Northwest region of the United States. The parallel MM4 was also demonstrated with real-time graphics at the ACM SIGGRAPH Showcase '92 conference in Chicago. In this demonstration, data generated in in California on the Intel Touchstone Delta* computer was transmitted over a network to a Silicon

*The Intel Touchstone Delta computer connects 528 processing nodes, each with 16 Mbytes of local memory, over a high-speed mesh communication network. The Delta

Graphics high-performance visualization workstation at the SIGGRAPH conference. The workstation then rendered and displayed each new frame of model output as it was generated.

2.1 Performance

At NCAR, the sequential-vectorized version of MM4 runs approximately 110 times faster than simulation time on one processor of a CRAY Y-MP. In other words, a 24-hour simulation in the tested configuration† requires 780 seconds (13 minutes) on the CRAY.

Initial work with MM4 was conducted on a Sequent Symmetry. Since early work focused on producing a working reproducible model, performance was a secondary consideration. A 12-processor run of the 1-D parallel version on the Sequent Symmetry required 30 seconds to perform each 160 second time step. For a 24-hour simulation the model would require 15,750 seconds or 4.5 hours. This corresponds to a 6:1 simulation time to compute time ratio (the CRAY executes MM4 at approximately 110:1).

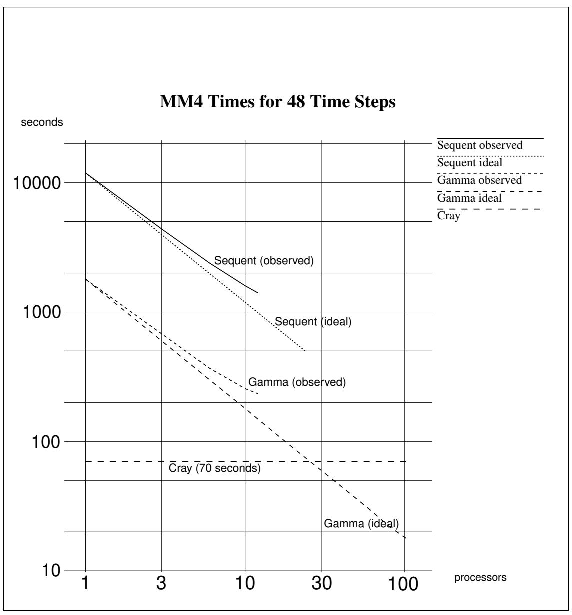

Figure 2: MM4 execution times for 48 time step simulation.

On an i860-based computer, performance was much better: a 12-processor run performed 1 time step in 4.8 seconds, yielding a 33:1 ratio of simulation time to CPU time, almost one-third the speed of the CRAY. Figure 2 shows the performance data for the Sequent, the Intel Gamma, and the CRAY. The figure shows that ideally, the Gamma would exceed CRAY performance at just under 30 processors; however, overhead is currently deflecting the observed performance in a curve that, if extrapolated, will never reach the CRAY at 70 seconds. Improved communication, reduced load imbalance, and other improvements to the parallel model are critical to reducing this deflection.

2.2 2-D Decomposition

To utilize a larger number of processors, the MM grid must be decomposed in a second horizontal dimension. This could be achieved by following the static decomposition strategy employed in the 1-dimensional prototype, adding array extensions in the second dimension and providing additional structure and communication to account for diagonal data dependencies between grid cells. The resulting code would be able to exploit 250 processors on a single 45×61 problem, giving performance equivalent to 2 CRAY Y-MP processors if parallel efficiency could be maintained through load balancing and communication tuning. However, relocating work to effect load balancing would be difficult, since the decomposition requires regular rectangular patches. The decision therefore was made to adopt a dynamic rather than a static decomposition when moving from the prototype to a production version of MM.

3. MASSIVELY PARALLEL MM5

Version 5 of the Penn State/NCAR Mesoscale Model, released in the fall of 1992, incorporates and standardizes a number of features that either were new or that had been added to MM4 for specific applications. Features include a nonhydrostatic option, four-dimensional data assimilation, movable nested grids, and improved physical parameterizations. A pre-release version of the MM5 code was made available for MPMM development in the spring of 1992, and work is continuing. In MPMM, the static decomposition strategy was abandoned in favor of an approach that would support dynamic load balancing and modular implementations of 4DDA, nested grids, and model coupling.

3.1 Fine-Grained Implementation

MPMM utilizes a fine-grained horizontal decomposition of the Mesoscale Model domain in which each multicomputer node is allocated some small but *not* statically determined number of columns in the grid. The shape of the processors' allocated region tends toward rectangular (where there are no load imbalances), but columns are able to migrate away from more heavily loaded processors when necessary. The technique that allows for this nonstatic decomposition of the grid is to make a distinction between the logical decom-

is situated at the California Institute of Technology and is owned by the Concurrent Supercomputing Consortium, of which Argonne is a member.

†Standard parameterizations on an 80 kilometer grid, with 46 latitudes, 61 longitudes, and 15 vertical levels.

position from the physical decomposition. The grid is first mapped into a virtual topology of logical processes; the virtual topology is then mapped dynamically into the physical processors of a particular parallel computer (for example, a mesh of processors as in the Intel Touchstone Delta computer). In the virtual topology, each grid column of the model is mapped onto a separate logical process in the parallel model. These *column processes* are connected by streams over which they communicate needed data to effect horizontal interpolation and finite differencing within the grid.

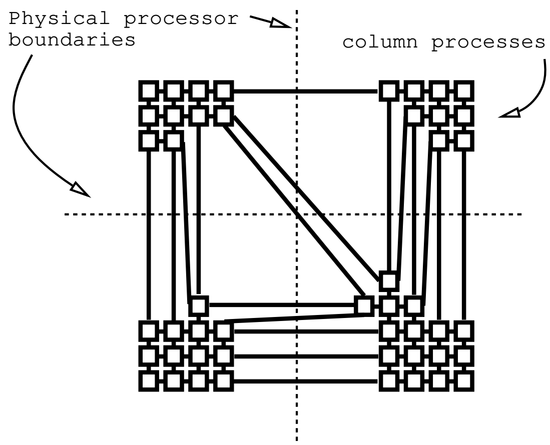

Figure 3: During an MPMM run, column processes in the virtual topology may be migrated away from more heavily loaded physical processors. Communication streams automatically follow under the run-time PCN system implementing the virtual topology.

Messages over streams between collocated processes are handled as memory references. Communication over streams that are cut by physical processor boundaries are handled by using interprocessor communication (message passing) between the processors. Further, moving a process to a different physical processor during model execution does not alter the virtual topology itself, so communication streams "follow" the process to its new physical location (Figure 3). The PCN parallel programming system handles the underlying mechanisms for constructing virtual topologies of processes, mapping them to physical processors, and implementing communication streams automatically.

In addition to the processes representing the model grid, we define a number of global or quasi-global *monitor processes* which implement such global functionality as managing input and output, coordinating load balancing, interfacing

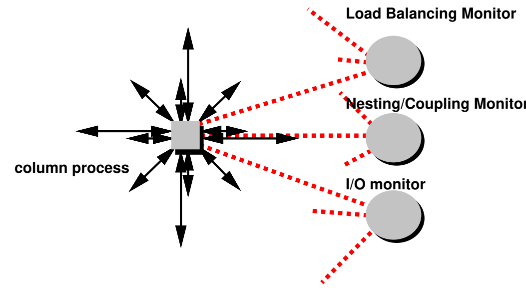

Figure 4: Schematic view of a column process representing a single grid column in the mesoscale model. Diagram shows communication streams to neighbors and to monitor processes, which are implemented and moved transparently in PCN.

with coupled model systems such as a general circulation model, and interfacing with interactive graphical systems. Figure 4 shows the process structure of MPMM. Only one grid process is represented with its communication streams to monitor processes and to neighbor column processes in a 12-point stencil. The monitor processes may be mapped to a single physical processor or may themselves be implemented as parallel programs executing on a separate virtual topology of logical nodes.

3.2 3-D Decomposition

We do not envision decomposing the model grid in the vertical dimension, for three reasons. First, we expect to obtain sufficient parallelism by a fine-grained horizontal decomposition. Second, we are concerned that a vertical decomposition would require too many changes to the MM code, in particular to physics routines. Third, we expect future parallel computers to provide either multitasking or vectorization facilities within each node, allowing vertical parallelism to be exploited by compilers.

3.3 Load Balancing

Atmospheric models are subject to load imbalances resulting from varying amounts of work over the model grid (Michalakes, 1991) when decomposed over a set of distributed memory processors in a multicomputer. MPMM will use dynamic load balancing to maintain parallel efficiency when the amount of work required per grid point is not constant; for example, because of the use of a more elaborate convection scheme in columns containing storm systems or because of dynamically created subgrids. The workload on each processor will be continuously monitored; periodically, imbalances will be corrected by moving vertical grid columns from heavily loaded to lightly loaded processors. This activity is coordinated by a load balancing monitor process which periodically collects and analyzes load data from each of the processors in the physical topology and instructs column processes in the virtual topology to relocate as needed. As stated previously, the undelying mechanisms supporting process and stream movement are provided in the PCN run-time system. Thus, alternative load-balancing algorithms can be substituted without changing other components of the parallel code, allowing a high degree of customization for load characteristics of a particular modeling application.

3.4 Nesting and Coupling

We intend that MPMM be usable by a broad community of scientists. Critical to this usability will be mechanisms to simplify the implementation of nesting and coupling to other models. We will implement both these capabilities using common mechanisms for transferring data between domains with different resolutions. In essence, a nested grid will be treated as a coupled run of the model at a finer resolution. Each grid will typically be distributed over the entire parallel computer, and appropriate interpolation/averaging routines will be used to transfer data between grids. In the case of coupled models, data transfers may also involve files or potentially parallel versions of other models running on the same computer. We anticipate supporting coupling with BATS and CCM2 initially; other models such as RADM will be considered if required.

3.5 Other Interfaces

The modularity of the design permits the installation of special-purpose monitor processes into the model. Work is currently under way at Argonne to develop a PCN/AVS parallel graphical interface that will allow real-time interactive 2- and 3-dimensional visualization of the model as it executes on a parallel computer. Such an interface could be easily encapsulated within a monitor process, and would permit scientists to interactively "explore" the data within their models. Additional modules will support the data movement necessary to implement 4-dimensional data assimilation in a parallel mesoscale model.

4. CONCLUSIONS

We have described a research and development project intended to develop a massively parallel mesoscale model (MPMM), capable of exploiting both current and future generations of parallel computers. Projected teraflops computers will allow MPMM to achieve performance superior by several orders of magnitude to that currently achievable on conventional supercomputers. In addition, MPMM opens the possibility of using more cost-effective platforms (e.g., networks of multiprocess workstations) for applications that do not require teraflops performance.

MPMM will provide the meteorological community with a cost-effective, high-performance mesoscale model. This in turn will broaden the range and size of problems that can be studied, permitting scientists to consider larger problem domains, longer simulations, finer-resolution grids, and more complex model processes, than have previously been possible. In addition, the parallel algorithms and code developed for MPMM will be directly applicable to projects developing parallel implementations of other, similar models.

REFERENCES

Anthes, R., E. Hsie, and Y. Kuo, 1987: Description of the Penn State/NCAR Mesoscale Model Version 4 (MM4). NCAR Technical Note, NCAR/TN-282+STR, 66 pp.

Brown, J., and K. Campana, 1992: An economical time-differencing system for numerical weather prediction. *Mon. Wea. Rev.*, 106, 1125–1136.

Dudhia, J., 1992: A nonhydrostatic version of the Penn State/NCAR mesoscale model: Validation tests and simulation of an Atlantic cyclone and cold front. Preprint, NCAR.

Foster, I., R. Olson, and S. Tuecke, 1992: Productive parallel programming: The PCN approach. *Scientific Programming*, 1(1), 51-66.

Grell, G., J. Dudhia, and D. Stauffer, 1992: MM5: A Description of the Fifth Generation PSU/NCAR Mesoscale Model. Draft NCAR Technical Note.

Michalakes, J., 1991: Analysis of Workload and Load Balancing Issues in the NCAR Community Climate Model. Argonne National Laboratory Technical Memo, ANL/MCS-TM-144, 20 pp.