UCRL-JC-133451
PREPRINT

Design Considerations and Performance of a Scalable Version of a Nonhydrostatic Atmosphere Model

A.A. Mirin
G. Sugiyama
R.M. Hodur
J.M. Schmidt
S. Chen

This paper was prepared for submittal to the
DoD High Performance Moderization Program Users' Group Conference
Monterey, CA
June 7-10, 1999

April 27, 1999

Design Considerations and Performance of a Scalable Version of a Nonhydrostatic Atmospheric Model

A.A. Mirin and G. Sugiyama
Atmospheric Science Division
Lawrence Livermore National Laboratory
Livermore, CA 94550

R.M. Hodur, J.M. Schmidt and S. Chen
Marine Meteorology Division
Naval Research Laboratory
Monterey, CA 93943

Abstract

The Naval Research Laboratory's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS) is being developed into a parallel, scalable model in a joint collaborative effort with Lawrence Livermore National Laboratory. The initial focus is on the atmospheric forecast model, which solves a coupled, three-dimensional set of dynamical equations using finite differences. A distributed/shared memory parallel programming paradigm is used. Distributed memory parallelism is achieved through a two-dimensional domain decomposition technique, with internodal communication accomplished using Message Passing Interface (MPI), and OpenMP is used to provide parallelism within a node. Initial performance results on both the IBM-SP and Cray-T3E are presented.

Introduction

The Naval Research Laboratory (NRL) and Lawrence Livermore National Laboratory (LLNL) are collaborating in the massive parallelization of NRL's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS). The main purpose of this enterprise is to take advantage of emerging scalable technology, in order to treat the finer spatial and temporal resolutions needed in complex topographical or atmospheric conditions, as well as to allow the utilization of improved but computationally expensive physics packages. The parallel model will facilitate the ability to provide real-time, high-resolution, multi-day numerical weather predictions for forecaster guidance, input to atmospheric dispersion simulations, and forecast ensembles.

COAMPS consists of an atmospheric data assimilation system, a nonhydrostatic atmospheric forecast model, and an ocean model. The initial focus of this project is on the atmospheric forecast model [1], which solves the nonhydrostatic, compressible form of the dynamical equations and includes relevant physical processes such as explicit moist physics, and parameterizations for cumulus convection, radiation, and subgrid-scale mixing. A variety of map projection coordinate systems are supported and a vertical sigma coordinate is used to treat flow over an irregular surface. The equations are discretized using finite differences on an Arakawa C grid [2]. An arbitrary number of fixed nests are allowed, under the constraint of a 3:1 reduction of grid spacing between successive grids. The time-differencing scheme is fundamentally explicit in the horizontal direction, with subcycling to evolve the faster moving sound and gravity waves. An implicit treatment is made for the vertical terms responsible for sound waves and the Brunt-Vaisala frequency. Most horizontal derivatives are represented to second-order accuracy, with options to use fourth-order methods for the diffusion and advection.

Parallelization Methodology

Domain Decomposition

A domain decomposition technique is used to partition the full grid among an arbitrary, runtime-specified number of nodes of a parallel machine, with each grid potentially having its own decomposition. Because of the tight vertical coupling, the decomposition is limited to the two horizontal directions. Each subdomain contains an arbitrary number of extra halo rows (typically one or two) to facilitate finite differencing and data communication with nearest neighbors. Two halo rows are warranted when running with fourth order advection or diffusion, since those operator stencils extend two points in each direction. However, the use of a single halo row is supported, even when performing fourth-order differencing. This results in a slight savings of memory and less redundant border computation, but at the expense of additional communication calls.

Interprocess Communication

Communication between subdomain processes is achieved via Message Passing Interface (MPI). MPI derived data types are used for automatic gathering and scattering of communicated data. The use of contiguous buffers remains a code option and aids in debugging and performance comparison.

There are several patterns of communication, the most prevalent of which is nearest neighbor border communication, including support for periodic boundary conditions. Communication in each of the four directions is carried out using synchronous MPI *send and receive* calls. The inclusion of border cells in the transverse direction obviates the need for separate diagonal messages to handle the corner points. A small number of global reductions are required as well, primarily in the radiation (which is called every several time steps) and moisture physics; some global reductions also occur in the diagnostics, timing and initialization.

The other prevalent form of communication is between nested grids. Coarse grid points covering several outer zones of the finer grid must be communicated to the finer grid, so that the coarse mesh data can be properly blended with the fine mesh data. Because the finer grid represents an arbitrary geographical subset of the coarser grid, and because each grid is allowed its own domain decomposition, there is no obvious correspondence between the computational processes of the coarse and fine grids. Intergrid communication is therefore addressed by examining all of the possible connections between coarse mesh subdomains and fine mesh subdomains, and building communication tables during the initialization phase of the program. The actual communication consists of blocking sends and nonblocking receives. Communication from a fine mesh to a coarser mesh, which entails smoothing of the fine mesh data, is accomplished using a similar procedure.

Parallelization within a Node

Additional parallelization is carried out within a subdomain, invoking the de-facto standard OpenMP [3]. OpenMP is available at different levels of maturity on most high-performance computing platforms. At this stage of the implementation our use of OpenMP consists largely of loop-level parallelism. Most of the parallel loops cover a single physical dimension, one exception being the treatment of the long wave radiation. The OpenMP standard includes nested parallelization; however, this construct is not yet commonly supported.

Parallelization in a single dimension is adequate for architectures having a small number of processors per node, but it is not scalable. Even on the next generation IBM machine (which increases the number of processors per node from 4 to 16), dimensions would have to be chosen carefully to maintain a reasonable parallel efficiency. Multidimensional parallelism can be achieved through nested constructs or dimensionally combined loops, and represents a scalable methodology.

Parallel Vector Processors

COAMPS has historically supported vector processors by compressing two-dimensional loops into a single dimension. This practice is included as an option in the current parallel version. It should be noted, however, that compressed loops typically require redundant computations. For example, calculations that involve only the interior of the horizontal domain must include the "east-west" borders as well to avoid indirect indexing. This is another rationale for allowing a single border row even when invoking fourth order differencing.

Fortran-90

The code, originally written in Fortran-77, now utilizes a number of Fortran-90 constructs. Memory management is carried out using pointers and allocatable arrays, with physical quantities on multiple grids represented through derived data types. Unfortunately, Fortran-90 does not support the use of allocatable arrays as components of derived data types; hence one must use pointers instead. This leads to compiler inefficiencies for subroutines having large numbers of pointer array arguments, which are not assumed to be contiguous. This problem is circumvented through use of subroutine interfaces and assumed size arrays.

Input and Output

Global binary data files produced by the analysis code provide physical input to the forecast model. Data is read in by a single MPI process, which may be either one of the existing computational processes or a separate dedicated I/O process. The use of a separate I/O process (corresponding to a separate computational node) allows separation of large global arrays and smaller local arrays, which can be useful when the memory per node is limited. Once read in, the data is communicated to the various computational processes.

We have added the capability for each computational process to write its own component output file, with a separate postprocessor combining and analyzing the code data. Output can also be handled analogously to the input, with the logical requirement that the output file be independent of the number of processes used in the computation.

Load Balance

Attaining high parallel efficiency is dependent on balancing the computational load among the processors that are participating in the calculation, so that the amount of work per processor is comparable between barrier synchronization points. The dynamics is inherently load balanced, as the same equations are solved throughout the computational domain. However, the radiation and moisture physics are dependent on factors such as the available sunlight and water vapor concentration, which can lead to spatially and temporally varying computational times.

The degree of load imbalance typically increases as the domain decomposition becomes finer and finer. That leads to an interesting tradeoff between distributed and shared memory parallelism, especially on non-uniform memory access (NUMA) architectures such as the SGI Origin. One can choose to have a coarse decomposition (a small number of MPI processes) and a high degree of shared memory parallelism, or a fine decomposition (a large number of MPI processes) and smaller parallel loops. For a constant number of processors, as the decomposition coarsens, the distributed memory parallelization efficiency should improve due to the smaller surface area to volume ratio, but the shared memory parallelization efficiency should worsen due to memory conflicts and data nonlocality. As the subdomains get larger, the degree of load balance should improve. Load balance could determine the optimal strategy.

Performance

The parallelized code has been tested on a number of architectures, including the ASCI [4] Blue Pacific IBM-SP and the DEC-Alpha machines at Lawrence Livermore National Laboratory, and the Cray-T3E and SGI Origin 2000 machines located at the Naval Oceanographic Office (NAVO) Major Shared Resource Center (MSRC) in Stennis Space Center, MS. Tables 1 and 2 present performance results on the SP and T3E for an analytical, single-nested case that includes radiation (but not moisture). The grid contains 323 points in each of the horizontal directions and 30 points in the vertical direction, and square domain decompositions ranging from 16 to 256 subdomains are used. For the finest decomposition, the subdomains have roughly 20 points in each horizontal direction. On the IBM we assign MPI processes to each computational processor; that corresponds to four processes per node. The T3E by design supports only one process per node.

The relative parallel efficiency for the overall algorithm in going from a 4 x 4 decomposition to a 16 x 16 decomposition is 80% on the IBM and 61% on the Cray. Interestingly, the computer time on both machines is almost identical at 8 x 8 and 16 x 16 decompositions. The IBM shows a huge speedup when going from 16 to 64 subdomains, probably due to superior cache utilization at the finer decomposition.

The parallel efficiency of the radiation package is close to 100% on each machine (the 166 s. figure on the T3E seems anomalously large). That is because the radiation algorithm (which takes a larger fraction of the compute time on the T3E) is compute-intensive and involves few communications. This result is somewhat optimistic in that the test problem takes place wholly during the daytime, thereby avoiding day/night load imbalances.

The results suggest that the use of fewer than 400 points per subdomain horizontally is not recommended. However, neither machine is being utilized to its fullest. The IBM is expected to attain superior scaling using the mixed programming model (OpenMP rather than pure MPI on-node). The T3E is known to attain better communication performance under SHMEM. The use of MPI-2 (the follow-on to MPI) might reap some rewards, as it supports one-sided communication.

Conclusions and Future Directions

The initial parallelized version of the COAMPS forecast model is nearly complete. The dynamics and radiation have been fully parallelized (in distributed memory) for a single grid and analytical input data. Parallelization of the moisture, multi-nesting capability, real data, and I/O are in progress. Implementation of shared memory parallelism is also in progress.

The parallel performance of the model will undergo further evaluation and improvement as appropriate. In particular, the use of one-sided (asynchronous) communications under MPI-2 (when available) and of MPI-IO for code input and output will be investigated. As noted above, it will be important to implement the shared memory parallelization constructs in a scalable manner and to carefully examine the tradeoff between distributed and shared memory parallelism. Every effort will be made to adhere to standards in order to maintain portability, so as to minimize recoding and accommodate a variety of architectures.

Acknowledgments

This is LLNL Report UCRL-JC-133451. Work performed under the auspices of the U.S.D.O.E. by Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48, and by the Naval Research Laboratory with support from the Office of Naval

Research through Program PE-0602435N and from the Department of Defense through Program PE-0603755D.

References

[1] R.M. Hodur, The Naval Research Laboratory's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS), *Monthly Weather Review*, Vol. 125, 1997, pp. 1414-1430.

[2] A. Arakawa and V. Lamb, " Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model," *Methods in Computational Physics,* Vol. 17, 1977, pp. 173-265.

[3] "OpenMP," http://www.openmp.org.

[4] "Accelerated Strategic Computing Initiative," http://www.llnl.gov/asci, Lawrence Livermore National Laboratory Report UCRL-MI-125923.

Table 1. Parallel Performance on the IBM-SP. Times (in seconds) are shown for both the overall algorithm and the radiation only. Efficiencies are relative to the coarsest decomposition case.

Decomp.	Time (total)	Eff. (total)	Time (rad.)	Eff. (rad.)
4 x 4	1317	---	430	---
8 x 8	310	106%	103	104%
16 x 16	104	80%	26	103%

Table 2. Parallel Performance on the Cray-T3E. Times (in seconds) are shown for both the overall algorithm and the radiation only. Efficiencies are relative to the coarsest decomposition case.

Decomp.	Time (total)	Eff. (total)	Time (rad.)	Eff. (rad.)
4 x 4	1030	---	544	---
8 x 8	314	82%	166	82%
16 x 16	105	61%	35	97%