PARALLEL OPTIMIZATION OF AN EARTH SYSTEM MODEL (100 GIGAFLOPS AND BEYOND?)

L. A. Drummond, J. D. Farrara, C. R. Mechoso, J. A. Spahr
Department of Atmospheric Sciences
University of California, Los Angeles
Los Angeles, California 90095-1565

Y. Chao, D. S. Katz, J. Z. Lou, P. Wang
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

KEYWORDS: Climate Modeling, Earth Sciences, Distributed Processors and Performance Analysis.

ABSTRACT

We are developing an Earth System Model (ESM) to be used in research aimed to better understand the interactions between the components of the Earth System and to eventually predict their variations. Currently, our ESM includes models of the atmosphere, oceans and the important chemical tracers therein.

This paper reports our efforts to optimize the parallel implementations of two of the four major ESM components: the atmospheric and oceanic models. So far, the work has focused on improving the load balancing and single node performance of the code in the CRAY T3D. As a result, the atmosphere and ocean model components running side-by-side, can achieve a performance level of slightly more than 10 GFLOPS on 512 processors of that machine. We also briefly outline our plans for realizing performance levels of 100 Gflops by the end of the century.

1. INTRODUCTION

Numerical models of the general circulation of the atmosphere and oceans (AGCMs and OGCMs, respectively) are the most powerful tools currently used to study complex environmental problems. The archetypal example of those problems is the impact on climate of increasing concentrations of greenhouse gases. The Earth system comprises many other components, however, such as those associated with chemical and biological processes. The possibility of using such complex models has been greatly enhanced by the development of massively parallel computer architectures and other similar technological developments.

We are addressing the challenge of building a three-dimensional model of the Earth System to be used in research aimed to better understand and eventually predict changes that will occur due to natural or anthropogenic reasons. Currently, the ESM has three major model components; an AGCM, an OGCM, and an atmospheric chemistry model (ACM). We plan to incorporate into the ESM an ocean chemistry model (OCM) by 1998. The model code is being designed for high-performance computing environments in conjunction with a Data Base Management System (DBMS) to retrieve and store output from different ESM components. This work is an integral part of the NASA High-Performance Computing and Communications (HPCC) Earth System Science (ESS) Program.

One of the science questions to be addressed with the AGCM-OGCM component of the ESM will be the decadal modulation of the El Niño/Southern Oscillation (ENSO) phenomenon. For this investigation, we need to perform simulations of at least several decades or even a century long. In view of the large computational resources required for the task, there is always a trade-off between the integration time of the simulations and the resolutions of the grids used in the ESM components. Up to now, the resources we have had available have allowed for a relatively coarse model resolution. With the new optimized versions of the codes, we plan to run century-long coupled AGCM/OGCM simulations with the model resolutions required to resolve the process crucial to the phenomenon according to current theory (Philander and Gu, personal communication). Later on, we will analyze the impact of ENSO on the exchange of chemical tracers between atmosphere and oceans.

2. PARALLEL OPTIMIZATION OF AGCM AND OGCM COMPONENTS OF THE ESM

The AGCM component of our ESM is a parallel version of that developed at UCLA over many years. This is a "grid point model", which uses a three dimensional staggered grid to compute flow variables around the globe. In the horizontal, the model uses a staggered latitude-longitude "C" grid. In the vertical, the model is based on a coordinate system in which the planetary boundary layer (PBL) top and isobaric surfaces above a prescribed pressure level in the lower stratosphere are coordinate surfaces. In the horizontal, the parallel AGCM implementation is based on a two dimensional domain decomposition (Mechoso et al. 1993, Wehner et al. 1995). This choice is based on the fact that many of the physical processes included in the model are strongly coupled in the vertical, which makes parallelization along this axis less efficient

The OGCM component of our ESM is the LANL version of the Parallel Ocean Program (POP, Smith et al. 1992). This is a parallel version of the three-dimensional, primitive-equation ocean model developed at NOAA/Princeton University Geophysical Fluid Dynamics Laboratory (Bryan, 1969). The model solves the three-dimensional Navier-Stokes equations in spherical coordinates using finitedifference techniques. POP has been used to study the circulation in the Atlantic Ocean, and is among the first to produce realistic simulations of the Gulf Stream separation from the coast of North America (Chao et al. 1996). The model has been integrated for 30 years with prescribed atmospheric forcings on the 256-processor CRAY T3D the NASA/Caltech Jet Propulsion Laboratory (JPL) using a resolution of $1/6^{o}$ longitude x $1/6^{o}$ latitude x 37 vertical levels. This resolution is one of the highest being used with contemporary OGCMs. We plan to increase the OGCM resolution to $1/10^{o}$ longitude x $1/10^{o}$ latitude x 100 vertical levels by the end of the century.

2.1 AGCM Optimization

The two major components of the AGCM are AGCM/Dynamics, which computes the evolution of the fluid flow governed by the appropriate equations and AGCM/Physics, which computes the effect of processes not resolved by the model's grid (such as cumulus convection) on processes that are resolved by the model's grid. The AGCM/Physics and AGCM/Dynamics execute sequentially, with the AGCM/Physics driving the atmospheric motion simulated by the AGCM/Dynamics. The AGCM/Dynamics itself consists of two main components: 1) a spectral filtering along latitude circles, and 2) the actual finitedifference calculations. With the grid partitioned in the horizontal as described above, there are two types of interprocessor communications required: 1) message exchanges among neighboring processors in the finite difference component, and 2) non-nearest neighbor exchanges in the spectral filtering component. In addition, the spectral filtering operation is performed only in polar regions, and thus introduces a load imbalance. This load imbalance generally increases with the number of processors, since the number of idle processors during the filtering steps increases. The filters are implemented as Fast Fourier Transforms (FFTs). From previous parallel implementations of this filtering step (see Wehner et al. 1993), we have found that it is best to assign the task of filtering a given latitudinal band to a single processor because it reduces the amount of data to be exchanged and allows the use of vendor-supplied, optimized FFT routines.

We have implemented a static load redistribution algorithm to load balance the filtering step (Lou and Farrara, 1996). In the parallel AGCM, the horizontal plane is divided into N sub-domains with $N = N_i \times N_j$, where N_i and N_j are the numbers of processors working in the longitudinal and latitudinal directions, respectively. Let K be the total number of latitude bands multiplied by the number of variables to be filtered by the N processors. In our load balancing algorithm, K is divided by N_j to obtain the number of latitude bands to be filtered by a row of processors, and then these bands are evenly distributed among processors.

With the horizontal grid partition, the AGCM/Physics component consists of a large amount of local computation with no interprocessor communication required. We have measured a parallel efficiency of only about 50% for this part of the code on 256 nodes of a CRAY T3D. This is a relatively poor efficiency for a code component that does not require interprocessor communication. The AGCM/Physics computational load varies in both time and space as elements such as cloud distribution, solar zenith angle and conditional stability of the atmosphere change. Thus, addressing load imbalances in the AGCM/Physics requires a more dynamic and elaborate procedure than that of the static scheme implemented in the AGCM/Dynamics for the spectral filtering.

We have designed such a dynamic scheme to load balance the AGCM/Physics. As a starting point, this algorithm computes estimates of the computational load in each processor during the AGCM/Physics. Then, these computational loads are sorted and as a result a numeric rank, between 1 and N, is assigned to each processor. The process of sorting computational loads involves global communication that can increase the overhead in the parallel AGCM code. For this reason, computational loads are sorted as infrequently as possible. To estimate the load we periodically measure the computational load in each processor for one complete 'unbalanced' AGCM/Physics step and assume that these loads will remain unchanged for the next M AGCM/Physics timesteps. Based on timing tests performed so far, we have selected $M = 10$.

Once the computational loads are sorted and the processors are ranked, the load is balanced by pairing processors and exchanging work between pairs. Processors are paired on the basis of their ranks. A pairwise data exchange between processors with rank i and rank $N - i + 1$ then takes place. The amount of data exchanged is determined by the difference in loads between the two processors. Thus, processors with a high workload will share part of their work with their *partner* processor whose workload is low. Pairs of processors share work only when the difference in their loads is greater than an empirically determined value (currently 10%). Whether the resulting load distribution is satisfactory or not clearly depends on the initial load distribution. If it is not satisfactory, then the sort and pairwise data exchange can be repeated. The parallel efficiency of the AGCM/Physics code component improved to 75% on 256 nodes of the CRAY T3D after implementing this load balancing scheme.

Lastly, we have performed a number of single node optimizations to increase the per-node performance on the

T3D (Drummond 1996). On the T3D, the overhead is mainly caused by the retrieval of data from main memory into cache and use of expensive operations that involve repeated application of one or more basic operations. To address these issues we have exchanged the order of some operations inside three-dimensional loops to reduce by one dimension (from 3-D to 2-D) the temporary arrays. As a result, we save memory and allow more data from different arrays to fit into the cache at once. We have replaced several calculations in the code by calls to the Basic Linear Algebra Subroutines (BLAS) (Lawson et al. 1979). BLAS has already been tuned and optimized for several available computers. Thus, the use of these routines does not adversely affect code portability.

In addition, there is a set of optimized special functions (such and exponential and square root) for the CRAY T3D and several compiler flags that aid in the single node optimization. The optimized special functions are in the form of a library of fast mathematical routines written in assembly language. These routines require substantially fewer clock cycles the code generated by their standard FORTRAN 77 counterparts (approximately, 50-90% reduction). We have used calls to these routines in our CRAY T3D code wherever possible.

We have also compiled the T3D code using the automatic unrolling compiler flag, the no IEEE standard division flag, and the read ahead flag. We have found that it is more advantageous to use compiler flags for compatibility with other parallel computing environments.

Number of Processors	Seconds per Simulated Day	GFLOPS
16	2540	0.68
32	1540	1.12
64	780	2.21
128	488	3.52
256	267	6.44
512	182	9.45

TABLE 1. Performance of the UCLA-AGCM on CRAY T3D. Using a model resolution of $2.5 \times 2 \times 29$.

The computations in most of the AGCM/Physics part of the code are independent from results of a previous integration step. Thus, it is not necessary to carry out the computations to the highest degree of numerical accuracy because the relatively small errors in the accuracy of the results do not accumulate. We had previously used a 64-bit arithmetic throughout the parallel UCLA-AGCM code. We have begun by changing the longwave radiation computation, which is the most expensive component computationally in the AGCM/Physics, to 32-bit arithmetic. Upon this change, the routine runs 20-25% faster.

The AGCM now boasts an overall parallel efficiency of 50% on 256 nodes of the CRAY T3D. This efficiency is very high for an atmospheric model code because the surface-to-volume ratio for these models is rather high and there are inherent load imbalances present. Table 1 summarizes the UCLA-AGCM performance after parallel code optimization.

2.2 OGCM Optimization

In the POP code, the horizontal dimension is distributed over parallel processors, while the vertical dimension is in a processor. The scalable performance of POP for large-scale problems has been investigated by increasing the global grid size (to maintain a fixed local grid size) and adding processors (Stagg and Chao, 1995). As shown in Table 2, the parallel efficiency of the OGCM is very high. The scaled efficiency is computed by dividing the time on one processor by the time on N processors. On the Cray T3D, the scaled efficiency is 87%, 88%, and 86% on 256, 512, and 1024 processors, respectively. Furthermore, on 1024 processors, a rate of 12.8 Gflops is obtained.

Number of Processors	Normalized Efficiency	GFLOPS
1	1	0.02
64	0.93	0.87
128	0.91	1.69
256	0.87	3.24
512	0.88	6.55
1024	0.86	12.80

TABLE 2. Performance of POP on CRAY T3D.

To improve the single node performance of the POP code, we have converted some temporary two-dimensional and three-dimensional arrays into one dimensional arrays to save storage and fit more data into cache. We have also changed the loop structures from implicit FORTRAN 90 style to explicit FORTRAN 77. In addition, there were many logical **if/where** statements in the code. If it is over ocean, do the computation, else over land, continue. Within these **if/where** statements, the computation may be quite complex. The compiler usually cannot efficiently optimize these loops (especially loop unrolling). We have replaced these **if/where** statements with land/ocean masking arrays, in which the values are 1 for ocean points and 0 for land

points. For example, the routine which computes the equation of state for ocean water was optimized by eliminating the **if/where** statements, replacing the FORTRAN 90 array syntax with explicit loop structures, converting nested double-loops and triple-loops to single-loops, and performing explicit loop unrolling. On a single node of the CRAY T3D, the execution rate of this routine increased from approximately 10 MFLOPS to 50 MFLOPS.

In addition, we have replaced some operations involving do-loops statements by calls to BLAS routines. For example,

```
DO I=1,IMAX
X(I)=ALPHA*(Y(I)*Z(I))
ENDDO
```

was replaced with a call to the extended BLAS routine named SHAD,

```
CALL SHAD(IMAX,ALPHA,Y,1,Z,1,0.0,X,1)
```

The original do-loop code required 7.7 million clock periods and now with the call to SHAD the code requires only 1.3 million clock periods. Similarly, the FORTRAN expression,

```
DO I=1,IMAX
X(I)=ALPHA*(X(I)*Y(I)*Z(I))
ENDDO
```

was replaced with:

```
CALL SHAD (IMAX,ALPHA,X,1,Y,1,0.0,X,1)
CALL SHAD (IMAX,1.0,X,1,Z,1,0.0,X,1)
```

The original do-loop code required 13 million clock periods while the calls to SHAD require only 2.8 million clock periods.

3. CONCLUSIONS AND FUTURE WORK

Figure 1 shows the predicted performance of the AGCM/OGCM running side-by-side on 512 processors of the CRAY T3D at NASA's Goddard Space Flight Center (GSFC). The numbers of nodes in the upper and lower panels represent the number of nodes assigned to the AGCM and OGCM, respectively. We can see that the coupled AGCM/OGCM can currently achieve a performance of slightly more than 10 GFLOPS on this machine.

The T3D belongs to the previous generation of CRAY MPPs and we will soon be running on the current generation T3E and other machines. On the T3E, we expect increases in performance from the faster processors and larger interprocessor bandwidths that will allow us to achieve 30-40 GFLOPS for the coupled AGCM/OGCM on comparably sized machines. Further increases in performance will require both the additional computations without much additional interprocessor communication that the ACM provides, and further optimization work on each of the codes. Concerning the former, incorporating the ACM will require the presence of an efficient mechanism to transfer data between ACM and AGCM. We are developing such a mechanism in the form of what we call a 'Data Broker'. Concerning the latter, the more sophisticated cache structure on the T3E will allow us to experiment with more elaborate techniques to improve data locality and realize a greater percentage of the peak processor speed.

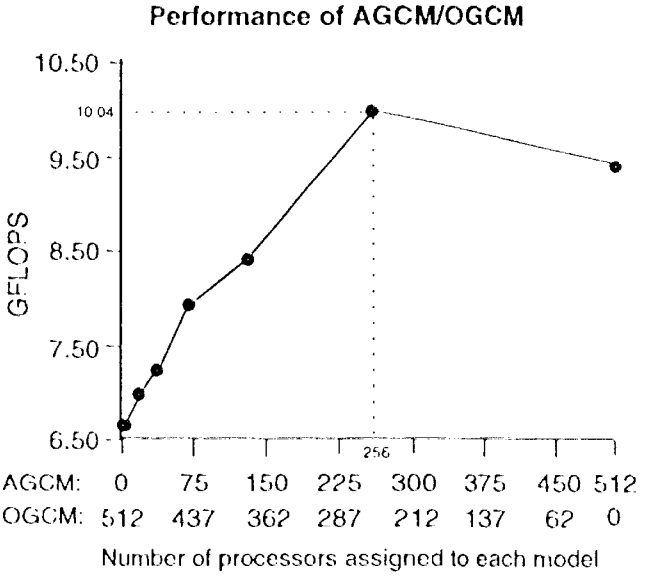

FIGURE 1. Parallel Performance of the AGCM and OGCM running on a CRAY T3D with 512 nodes.

With success in both of these areas, we expect to achieve a performance level of 50 GFLOPS using 512 processors of a CRAY T3E. To reach a level of performance approaching 100 GFLOPS will require a T3E twice as big as that currently available to us. Naturally, our quest will be facilitated by the advent of a next generation of computers with several improvements over the current, particularly in memory cache and communications bandwidths.

ACKNOWLEDGMENTS

This work has been supported by the NASA High Performance Computing and Communications for Earth Sciences Project under Grant NAG 5-2224 and Cooperative Agreement Number NCCS5-149. We thank Dr. Hong Ding for his help in the OGCM optimization effort.

REFERENCES

Bryan, K. 1969: A numerical method for the study of the circulation of the world ocean. *J. Comp. Phys.*, **4**, 1687-1712.

Chao, Y., A. Gangopadhyay, P. O. Bryan and W. R. Holland, 1996: Modeling the Gulf Stream System: How far from reality? *Geophys. Res. Lett.*, **23**, 3155-3158.

Drummond, L. A., J. D. Farrara, C. R. Mechoso and J. Z. Lou, 1997: Performance optimization of an atmospheric model in massively parallel computers. To appear in *European High Performance Computing and Networking 97*, April 1997.

Lawson, C., R. Hanson, D. Kincaid and F. Krogh, 1979: Basic Linear Algebra Subprograms for FORTRAN usage. *ACM Transactions on Mathematical Software*, 5.

Lou, J. Z., and J. D. Farrara, 1996: Performance analysis and optimization on the UCLA parallel atmospheric model code. *Supercomputing 96*, in press.

Mechoso, C. R., C.-C. Ma, J. D. Farrara and J. A. Spahr, 1993: Parallelization and distribution of a coupled atmosphere-ocean general circulation model. *Mon. Wea. Rev.*, **121**, 2062-2076.

Smith, R. D., J. K. Dukowicz and R. C. Malone, 1992: Parallel ocean general circulation modeling. *Physica D*, **60**, 38-61.

Stagg, A. and Y. Chao, 1995: Parallel ocean program on the 1024-processors CRAY T3D. Technical Report PATP, Cray Research.

Wehner, M. F., J. Ambrosiano, J. C. Brown, W. P. Dannevik, P. G. Eltgroth, A. A. Mirin, J. D. Farrara, C.-C. Ma, C. R. Mechoso, J. A. Spahr, 1993: Towards a high-performance, distributed memory climate model. *Proceedings, Second International Symposium on High Performance Computing (HPDC-2)*, Spokane, WA IEEE Computer Society, 102-113.

Wehner, M. F., A. A. Mirin, P. G. Eltgroth, W. P. Dannevik, C. R. Mechoso, J. D. Farrara and J. A. Spahr, 1995: Performance of a distributed memory, finite difference atmospheric general circulation model. *Parallel Computing*, **21**, 1655-1675.

BIOGRAPHY.

L. A. Drummond is a postdoctoral researcher in the Department of Atmospheric Sciences at the University of California, Los Angeles. His interests include parallel computing for scientific applications, numerical methods, and performance modeling of parallel codes. He received a PhD in Computer Sciences from Institut National Politechnique de Toulouse, France.

J. D. Farrara is an assistant researcher in the Department of Atmospheric Sciences at the University of California, Los Angeles. His interests include parallel and distributed computing for scientific applications, the dynamics of the stratospheric circulation, and global general circulation modeling of the atmosphere. He received a PhD in Atmospheric sciences from UCLA in 1989.

C. R. Mechoso is a professor in the Department of Atmospheric Sciences at the University of California, Los Angeles. He is involved in the Department of Energy Computer Hardware Advanced Mathematics and Model Physics Program, the CNRS/NASA Strateole Project, and the NASA HPCC Earth and Space Sciences Project. His research interests include numerical weather prediction, meteorology of the Southern Hemisphere, dynamics of atmospheric fronts, and ocean-atmosphere interactions. He received his PhD in Geophysical Fluid Dynamics from Princeton University in 1978.

J. A. Spahr is a senior computer programmer in the Department of Atmospheric Sciences at the University of California, Los Angeles. His interests include large-scale computing, particularly parallel and distributed processing for scientific applications, and atmospheric modeling. He received his BS in Atmospheric sciences from Cornell University in 1974.

Y. Chao is a researcher scientist in the Earth and Space Division at the Jet Propulsion Laboratory and California Institute of Technology. He is involved in the NASA HPCC Earth and Space Sciences Grand Challenge project, NASA Mission to Planet Earth TOPEX/POSEIDON and NSCAT Science Working Teams. His interest include ocean modeling, dynamics of ocean circulation, satellite oceanography and air-sea interaction. He received his PhD in Atmosphere and Ocean Sciences from Princeton University 1990.

D. S. Katz is a member of the technical staff at Jet Propulsion Laboratory, in the High Performance Computing Systems & Applications group. His interests include numerical methods applied to parallel computing, and computational methods in electromagnetic wave propagation. He received his PhD in Electrical Engineering from Northwestern University in 1994.

J. Z. Lou is a member of the technical staff in the Jet Propulsion Laboratory and California Institute of Technology. His interests include parallel and distributed scientific computing, parallel computational fluid dynamics and parallel software engineering. He received his PhD in applied mathematics/computational fluid dynamics from the University of California at Berkeley in 1991.