Design and performance analysis of a massively parallel atmospheric general circulation model

Daniel S. Schaffer* and Max J. Suárez
NASA Seasonal to Interannual Prediction Project, NASA Goddard Space Flight Center, Code 971, Greenbelt, MD 20771, USA
Tel.: +1 303 497 7252; Fax: +1 303 497 6301; E-mail: schaffer@fsl.noaa.gov

In the 1990's, computer manufacturers are increasingly turning to the development of parallel processor machines to meet the high performance needs of their customers. Simultaneously, atmospheric scientists studying weather and climate phenomena ranging from hurricanes to El Niño to global warming require increasingly fine resolution models. Here, implementation of a parallel atmospheric general circulation model (GCM) which exploits the power of massively parallel machines is described. Using the horizontal data domain decomposition methodology, this FORTRAN 90 model is able to integrate a 0.6° longitude by 0.5° latitude problem at a rate of 19 Gigaflops on 512 processors of a Cray T3E 600; corresponding to 280 seconds of wall-clock time per simulated model day. At this resolution, the model has 64 times as many degrees of freedom and performs 400 times as many floating point operations per simulated day as the model it replaces.

1. Introduction

The general circulation modeling community constantly demands more computing power to meet its needs. Short to medium range weather forecasters have used increasingly faster machines to run higher resolution models. The improved solutions obtained from higher resolution in numerical weather prediction is well known; Simmons et al. [21], among others, document this. Higher resolution is also important to seasonal and interannual variability studies (e.g. [7,11]). For studies of longer time scale phenomena, completing model runs at any reasonable resolution becomes the challenge. Coupled atmospheric/ocean simulations of El Niño require enormous computational power. Recently, some modelers have turned to ensembles of runs to produce better predictions; a strategy that magnifies resource demands. For the time scales of global climate change, coupled model runs can last hundreds of simulated years (e.g. [15]); for studies of the thermohaline circulation, those numbers stretch into the thousands.

To meet these needs, supercomputer manufacturers have offered a variety of solutions. Since the 1980's, parallel vector processors have been the most widely used by the GCM community. However, in the 1990's cache-based massively parallel processor (MPP) machines have become increasingly prominent. These machines challenge model designers to write code that runs efficiently within a single processor yet scales well for hundreds of processors. In addition, these models must be easily adaptable to rapidly changing machine architectures and communication software so as to avoid time-consuming code rewrites.

A snapshot of the progress of (mostly atmospheric) GCM designers toward meeting these challenges was presented in a special issue of Parallel Computing in 1995. Drake et al. [8] wrote a message passing implementation of the National Center for Atmospheric Research (NCAR) Community Climate Model (CCM2) for the IBM SP2 and Intel Paragon machines. Most notable was the poor single processor performance they attributed to inefficient cache use (a result noted repeatedly in the literature). Jones et al. [9] implemented a parallel version of the Geophysical Fluid Dynamical Laboratory (GFDL) Atmospheric General Circulation Model (AGCM) running on the CM-5 and SGI/Cray C90. Single processing element (PE) performance and scaling were quite good on the C90 but hampered on the CM5 by over-use of memory they attributed to poor algorithmic design. Lou and Farrara [13] optimized a parallel version of the UCLA AGCM for the Paragon

*Also at CIRA, Fort Collins, CO, USA.

Scientific Programming 8 (2000) 49–57
ISSN 1058-9244 / $8.00

and SGI/Cray T3D/E. The model scales fairly well but their preliminary attempts at cache-based optimizations have yielded modest improvements.

More recent GCM implementations on cache-based MPP's have yielded little improvement in single processor performance. The Canadian Mesoscale Compressible Community (MC2) model [24] and the Norwegian high resolution limited area model [22] both report speeds on the order of 10% of the T3E theoretical peak. Levesque [12] offers some hope as he was able to more than double performance of the Los Alamos Parallel Ocean Program (POP) on the SGI Origin 2000 by meticulously mapping cache misses and then re-writing the code to minimize them. However, this kind of restructuring is time consuming and the resulting code may not perform well on other architectures (particularly vector processors). Other modelers have instead chosen to re-direct efforts away from machine specific optimizations and toward enhancing code adaptability. The MC2 code [24] includes an intermediate communications layer between the model functional routines and communication packages such as MPI. The isolation of machine and communications package dependent code made the model quite portable. Michalakes [16] went even further by developing a special compiler to nearly automatically parallelize the serial version of the Penn State/NCAR Mesoscale Model (MM5). The results on the SGI Origin were as good as a hand parallelized version. Development of this compiler was facilitated by the fact that MM5 has only local data dependencies.

Here we describe the parallel design and performance of an AGCM designed for climate studies on MPPs. The primary objectives are efficient single PE performance, scalability, and portability. Section 2 describes the scientific basis of the model. Section 3 explains the high-level model design, the parallelization methodology, and gives highlights of the detailed design. Section 4 analyzes the model performance. Section 5 discusses how the model is currently being used and describes on-going optimization efforts.

2. Model description

The dynamical portion of the GCM is based on a finite-differenced, primitive equations dynamical core (Dycore) [23] that allows arbitrary horizontal and vertical resolution. It is the dynamical core used by Goddard's Data Assimilation Office in the Goddard Earth Observing System (GEOS) GCM and by NASA's Seasonal to Interannual Prediction Project (NSIPP). Its prognostic variables are the two horizontal wind components, the potential temperature, the surface pressure, the water vapor mixing ratio, and an arbitrary number of passive tracers. In the vertical, the discretization scheme closely follows that proposed by Arakawa and Suarez [1], but applied to a generalized vertical coordinate ($\sigma - p$). In the horizontal, the equations are finite-differenced on a staggered latitude-longitude grid (the C-grid). To avoid linear computational instability due to convergence of meridians near the poles, a Fourier filter is applied to all tendencies pole-ward of 45 degrees latitude. The model also uses a scale-selective filter [20] to damp grid-scale variance that can lead to non-linear computational instability. The model is integrated in time using a leapfrog scheme modified as proposed by Brown and Campana [3] and by applying a weak time filter [2].

The solar parameterization [5] models absorption due to O_3, CO_2, water vapor, O_2, clouds, and aerosols, as well as gaseous, cloud, and aerosol scattering. The infrared parameterization [6] includes absorption by water vapor, CO_2, O_3, methane, N_2O, CFC-11, CFC-12 and CFC-22 within eight spectral bands. Other parameterizations include the Louis et al. [14] turbulence and Zhou et al. [25] gravity wave drag schemes. Penetrative convection originating in the boundary layer is modeled using the Relaxed Arakawa-Schubert (RAS) scheme [17]. The Mosaic land surface model (LSM) [10] computes area-averaged energy and water fluxes from the land surface in response to meteorological forcing. A grid square is sub-divided into relatively homogeneous sub-regions ("tiles" of the mosaic), each containing a single vegetation or bare soil type.

3. Computational design

We begin by describing the high level structure of the GCM so as to provide context for the results in Section 4. The model is divided into self-contained components, each operating on its own space (grid) and time scales. Presently, the major components for this atmospheric GCM are:

1. Dynamics – Dycore, the Shapiro filter and the model stepping functionality.
2. Slow Physics – The longwave and shortwave radiation calculations.
3. Fast Physics – The remainder of the atmospheric GCM; convection, turbulence, land processes, etc.

"Coupling" software ties together the model components. The couplers serve two main purposes. One is to do the necessary time averaging of model component outputs. For example, the Dynamics runs more frequently (every few minutes) than the Slow Physics (typically every 3 hours). The coupler sums the Dynamics outputs during each 3 hour interval and then sends the time average to Slow Physics just before it runs. The second purpose of the coupling software is to convert data from one model grid to another in parallel. In general, this is accomplished by computing the area of intersection of overlapping grid boxes and then doing the appropriate weighted averages. When flux quantities are exchanged between model components, the global means are conserved. It is important to note that the results in Section 4 are for a version of the atmospheric model in which all components operate on the same grid. As a result, the couplers simply do time averages and data copies.

The couplers and model components operate synchronously within a single executable program. For example, the GCM driver might call the Dynamics, then the Dynamics to Slow Physics coupler, then the Slow Physics, etc. This model structure is sufficiently general to allow easy application to other models. For example, the atmospheric model has a component called "Ocean" which just reads SST from a file. A coupled atmosphere-ocean model was coded simply by replacing the "dummy" ocean component with a true ocean model and adding the appropriate atmosphere to ocean couplers (see Section 5).

The coupling software is similar to the NCAR Climate System Model (CSM) flux coupler [4] in that it does the appropriate space and time averaging in parallel. However, in the CSM, the flux coupler and model components operate asynchronously as separate executable programs. This will, of course, yield different time truncations from the approach here. It should be noted that the CSM approach has the advantage that model components can be run on separate computers.

The atmosphere model components are parallelized using a horizontal data domain decomposition. Put simply, each processor operates on a slab of data extending from the surface to the top of the atmosphere (Fig. 1). The primary advantage of this decomposition is that the number of horizontal grid points available to divide among the processors is large, allowing utilization of hundreds of PE's. In addition, physics calculations such as longwave, shortwave, etc. require no communication. Finally, at a practical level, using this scheme means that the original plug compatible physics subroutines can be retained, unmodified, in the parallel implementation.

The processors are laid out in a rectangular array so that each PE has exactly one neighbor on each of four sides. The number of PE's in the X and Y direction (NX and NY) as well as the number of grid points within each PE (IM and JM) are selectable at run-time. In particular, IM can be different for each of the NX columns of PEs and JM different for each of the NY rows. Ghost (shadow) regions are defined to facilitate local addressing and nearest neighbor communication. When code such as horizontal advection needs to access an array element outside the local processor, a communications call is made to fill in the ghost region. Once the data are in place, the code runs as if it were written for a serial computer. The communication is bundled over all levels to reduce the impact of latency.

Since the primary objective is implementation on a distributed memory MPP, a message-passing scheme is used for the communication. Generic synchronous point to point send/receive routines provide the backbone for this scheme. Several rules have been established to simplify the programming model:

1. Communication is "one to one". At every point where communication occurs, every processor sends data to exactly one other processor (although a processor can also send to itself).
2. Communication is one-sided. Every processor "puts" data to another processor. It knows to which other processor it is sending data but does not know from which other processor it is receiving.
3. Synchronization is automatically handled by the communication routines.

Currently the communications routines are implemented using calls to either native Cray shared memory software (SHMEM), message passing interface (MPI) routines, or to a single processor package which "communicates" via simple data copies. This backbone is packaged into a single "communication primitives" module. Since this is the only code that varies between implementations, porting the model is quite simple. The code currently runs on the DEC Alpha workstation, the SGI Origin 2000, the Cray T3E, and the Cray J90/C90.

While most of the communication in the model is nearest neighbor, the polar filter is a significant exception. It is implemented by first transposing the data from an (X,Y) to a (Y,Z) decomposition, then executing local FFTs, then transposing back. This implies that the

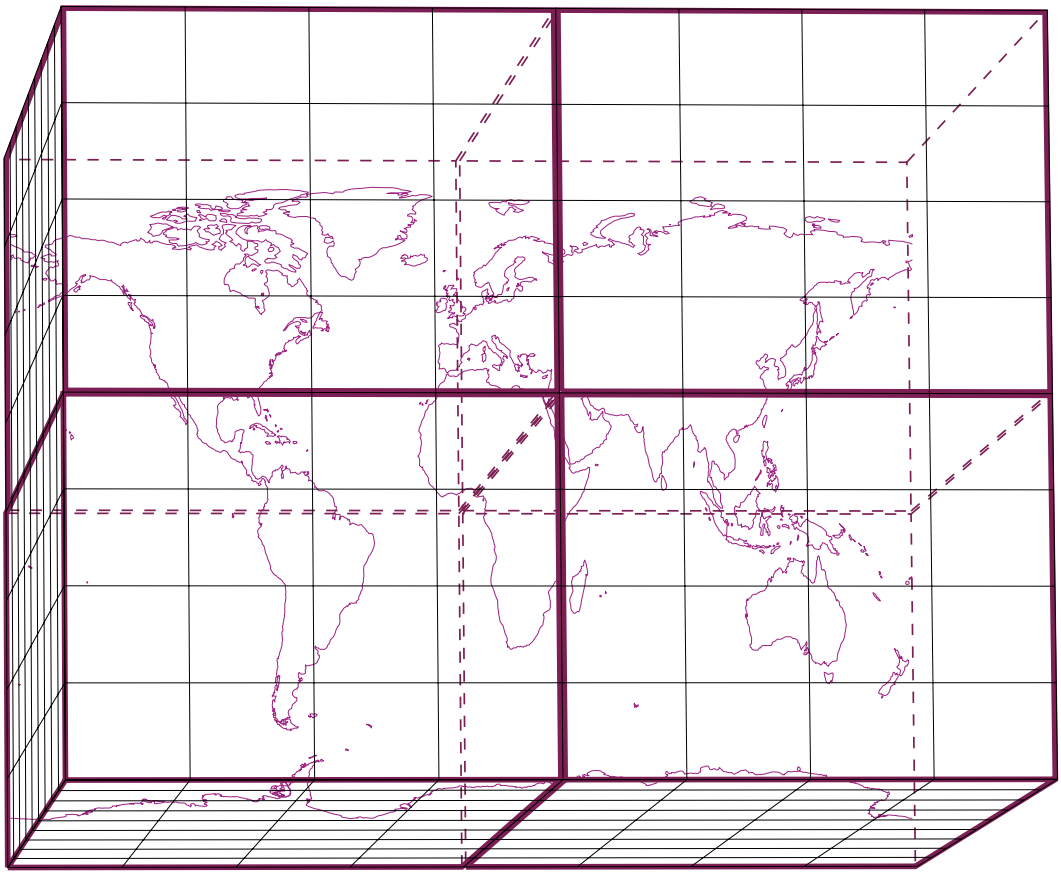

Fig. 1. Representation of a horizontal data domain decomposition. The thin lines demarcate model grid boxes. The thick lines indicate processor boundaries. In this case, the model data are divided among 4 processors.

greater the decomposition in X, the poorer the performance of the polar filter. Conversely, nearest neighbor communication scales as $\frac{1}{\sqrt{PEs}}$ only if the processor layout is close to symmetrical. These conflicting performance considerations guide optimal processor layout choice and represent the most obvious disadvantage of this decomposition strategy.

Currently, no load balancing is implemented. The sources of imbalance are:

1. The shortwave code; radiative transfer calculations need only be performed for sunlit soundings.
2. The land surface code; no computations are needed for ocean points and the uneven distribution of tiles further un-balances the problem.
3. Cumulus convection; fewer computations are needed where convection does not occur.
4. The polar filter; it only operates pole-ward of 45 degrees latitude. The utility of implementing load balancing schemes will be discussed in Section 5.

The Dynamics, all upper-level Physics routines and control and communication routines are written in FORTRAN 90. Most of the low-level, plug-compatible, computational routines in the Physics have been left in FORTRAN 77. Array syntax, user-defined types, subroutine overloading, modules, and dynamic memory allocation are used extensively. Use of these features has helped to create reasonably well-structured code and greatly facilitated debugging. Since all memory is dynamically allocated, the model runs at any resolution using any processor layout without recompilation. The use of FORTRAN 90 already has yielded negative impacts. On the Cray J90/C90 machines, code which uses FORTRAN 90 pointers does not vectorize. Consequently, the code had to be structured so as to convert pointer references to array references. This was accomplished by, for example, passing pointer variables to subroutines which have array dummy arguments. Also, arithmetic using assumed shape arrays was less efficient than that for automatic

or allocated arrays. Finally, dynamic memory use may hamper future cache-based optimizations. On the positive side, the use of array syntax does not appear to degrade performance on the Cray T3E as compared to F77 style loop structures. Figure 2 shows samples of F90 array syntax and F77 loop versions of the fourth order horizontal advection scheme inside Dycore. The two codes were repeatedly executed as part of a one day run of a 2.5° by 2° by 22 layer resolution atmospheric GCM on 64 processors. The codes executed in the same amount of time. In this case, communication (calls to GHOST) represented 20% of the run-time of the routine.

4. Results and performance

The model is currently being run on the DEC Alpha workstation, Cray T3E, and Cray J90. To validate the code, results were compared to those from the serial, FORTRAN 77, production version for the same initial and boundary conditions at a resolution of $72 \times 45 \times 22$. At this resolution, Dynamics and Fast Physics are run at 9 minute intervals and Slow Physics every 3 hours. After 3 hours, checksums of state variables, budgets and other diagnostic quantities for the old and new code differ at the round-off level; for one or multiple processors.

To assess performance, the floating point operations (FLOPs) are counted for a one processor run on a CRAY J90 using the PERF utility. These numbers are generally more conservative (up to 25%) than the operation counts produced by T3E-native counters. Initialization and finalization times are not counted. No model output is done during the "run" phase for purposes of these benchmarks. Performance is then computed by dividing the FLOP count by the run-time measured by wall-clock timers. The $72 \times 45 \times 22$ resolution problem was run on the Cray T3E-600 using 32 bit words for up to 64 PEs. The peak performance is 1.35 Gflop/s, corresponding to 20 seconds run-time per simulated model day. A 64 bit version runs longer (28 seconds per day) largely due to the fact that the code is memory-access bound. In comparison, the original production version running multi-tasked on the Cray J90 (64 bit) simulates one model day in 50 seconds.

With the successful implementation of the model on the T3E, it is now possible to change the resolution used in long-term simulations from 4° by 5° to 2.5° by 2°. A time step of 225 seconds is sufficient to satisfy the Courant-Friedrich-Levy (CFL) condition for linear numerical stability at this resolution. The Dynamics and Fast Physics run at this interval; the Slow Physics at 3 hour intervals. For the typical case of 128 processors using 32 bit words, the model simulates one day in 80 seconds. Of this, 32 seconds is spent in Dynamics, 34 seconds in Fast Physics and 9 seconds in Slow Physics. Only 3 seconds is spent doing I/O.

To truly exploit the power of the T3E machine, we turn to a high resolution problem; $576 \times 360 \times 22$ (0.625° by 0.5° by 22 levels). Preliminary tests show a Dynamics time step of one minute is required to satisfy the (CFL) condition at this resolution. The Fast Physics is run every 10 minutes and Slow Physics at 3 hour intervals. For a 3 hour run, the floating point operations total 686 billion. The 32 bit version requires approximately 1 billion words of memory; translating to a minimum of 64 Cray T3E-600 PEs. The GCM was tested for processor configurations totaling up to 512 PEs. Experimentation showed that for 512 PEs, a processor layout of 16 PEs in longitude, 32 in latitude is optimal. For that case, the performance is 19.6 Gflop/s. This corresponds to 280 seconds of wall-clock time per simulated model day.

The details of the T3E performance are shown in the speedup plots in Fig. 3. The solid lines in the figure are curve fits of the data to Amdahl's speedup law:

$$S = \frac{1}{Fs + \frac{Fp}{Np}}$$

where S is the speedup, Fs is the serial fraction, Fp is the parallel fraction and Np is the number of processors. For a perfectly load balanced code, the effective single processor performance is an estimate of how fast it would run on 1 PE if that were possible. Notice that, in Dynamics, this number is higher than the per-processor performance because it does not include the degradation due to communication as the problem is scaled to 512 PEs. The floating point operation counts show that Dynamics is responsible for the great majority of the work. This is largely due to its relatively short time steps. Although the Slow Physics does virtually no communication, Fig. 3 shows that it does not scale perfectly. We speculate that performance for large numbers of processors is hampered by excessive loop overhead in the longwave and shortwave routines. As the number of processors increases, the loop extents shrink and the effects of loop overhead become significant.

Table 1 shows a breakdown of performance of the major GCM components. The dynamical core consumes the most run-time and will need the greatest attention during future optimizations. The poor scaling

```
F77 Loop Version

      do L=1,LM
        do J = J1,JN
          do I = I1,IN
            TND(I,J,L) = TND(I,J,L) + (1./24.) &
              * ( (VT1(I, J, L) - VT1(I-1, J, L))  &
                + (VT2(I, J+1, L) - VT2(I, J, L)) )
          end do
        end do
      end do


F90 Array Syntax Version

      do L=1,LM
        TND(I1:IN, J1:JN, L) = TND(I1:IN, J1:JN, L) + (1./24.) &
          * ( (VT1(I1:IN, J1:JN, L) - VT1(I1-1:IN-1, J1:JN, L)) &
            + (VT2(I1:IN, J1+1:JN+1, L) - VT2(I1:IN, J1:JN, L)) )
      end do
```

Fig. 2. Code samples of F77 loop and F90 array versions of the fourth order horizontal advection code inside Dycore. At the resolution of 2.5 by 2° with 8 processors in the X and Y direction, the local loop sizes are 18 in X and either 12 or 11 in Y. The arrays shown in the code were allocated using F90 automatic array syntax.

of the Shapiro filter is expected; it does relatively few floating point operations per communication. That the Step function does not scale perfectly is merely an artifact of the code design. It fills the ghost regions of the state variables; work that could just have easily been done in Dycore.

The LSM and RAS codes are "super-scaling". This commonly observed result occurs because as the number of processors increases, the amount of memory needed per pe decreases and, consequently, the data fit better in cache.

The rated performance of the Cray T3E 600 is 600 Mflop/s. While, in practice, few codes reach 200 Mflop/s per PE, it is clear from table 1 that our per-PE performance is much lower. One reason is poor cache re-use. As a first cut, this code was written to mimic the original serial code which was designed to run efficiently on vector machines. As of yet, no serious cache-based optimizations have been attempted. A second reason is communication costs. Measurements indicate that 22% of the Dycore run-time is communication. Latency is significant. Even with bundled Ghost calls, preliminary measurements indicate that 35% of the nearest neighbor communication time is latency. When the Ghost calls are unbundled, Dycore performance degrades by 20%. A third cause of the poor single-pe performance is load-imbalance as described earlier. Strategies to address these inefficiencies are discussed in the next section.

Table 1
Floating point operations (in billions), run-time, total performance, and per pe performance for a 3 hour run of the $576 \times 360 \times 22$ resolution problem at 512 PEs.

Code	GFLOP	Time (s)	Gf	Mf/PE
Dycore	427.3	17.77	24.1	47.0
Shapiro	74.6	5.56	13.4	26.2
Step	33.6	1.57	21.4	41.8
Longwave	34.3	0.97	35.4	69.1
Shortwave	48.0	2.77	17.3	33.8
Lsm	6.8	0.96	7.1	13.8
Ras	4.6	1.21	3.8	7.4

A Cray J90 SHMEM version of the code for the same resolution performed at 90 Mflop/s on one processor. Since the rated performance of the J90 is 200 Mflop/s, the model is clearly vectorizing quite well. Although a multiple processor J90 version has not been run for this resolution, past experience suggests that it should perform at about 1 Gflop/s for 16 PEs. An MPI implementation on the J90 was found to significantly degrade the code's performance; presumably due to the high level of overhead in the MPI software. A T3E MPI version has not been tested. A SHMEM version on the SGI Origin 2000 runs on multiple processors but the results have not yet been analyzed.

5. Discussion

As currently written, the code performs well enough to enable production runs at high resolution (0.6° lon-

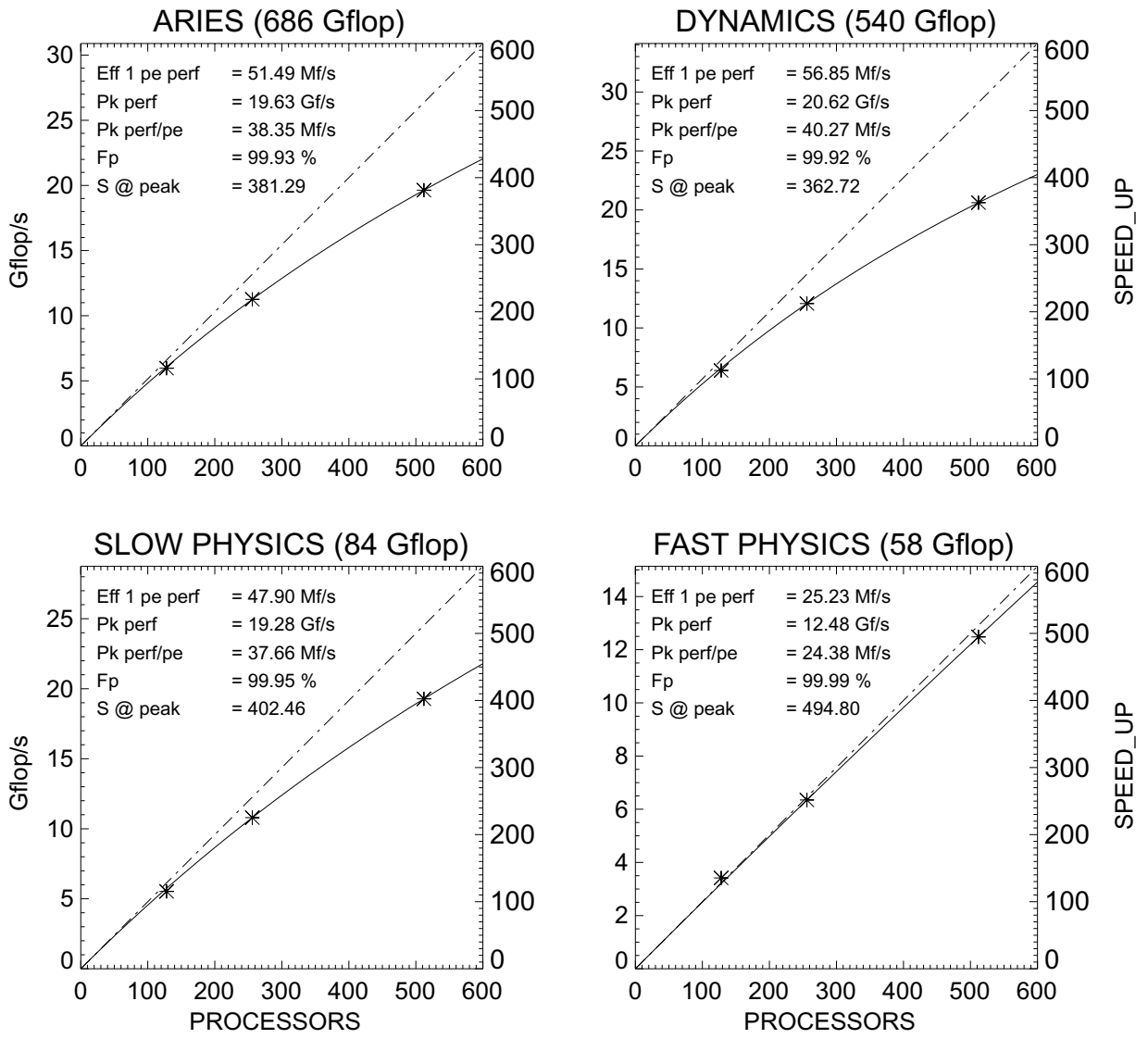

Fig. 3. Speedup plots for 3 hour runs of the full GCM and its three major components. The floating point operations in billions are given at the top of each graph. The asterisks represent the speed in Gflop/s for 128, 256 and 512 PEs. The dot-dashed line represents a perfectly linear speedup. The solid curve was obtained by fitting the run-times to Amdahl's speedup law (see text). Fp and S are as given in Amdahl's law. The effective single PE performance is the curve value for $Np = 1$.

gitude by 0.5° latitude by 22 levels) using 512 processors. In fact, a one year run at this resolution has already been completed. The model can also run efficiently at lower resolutions. For example, a 1.25° longitude by 1° latitude problem running on 128 processors would actually out-perform the high resolution case. The Dynamics, Fast Physics and Slow Physics would run with the same efficiency as in the high resolution case since the amount of work and number of processors have both decreased by a factor of 4. However, for the lower resolution, a Dynamics time step of 2 minutes could be taken, significantly improving the model throughput. For lower resolutions, an ensemble of runs most effectively utilizes the 512 PE machine. Such ensemble runs have already been completed at a resolution of 1.25° by 1°. The 2.5° by 2° atmospheric model has also been coupled to a parallel version of the Poseidon ocean model [19]. A 100 year run using 128 processors is currently underway.

Five major avenues of optimization are under investigation; faster time differencing schemes, single PE optimization, reduction of software latency in

the communication code, load balancing, and parallel/asynchronous I/O. As the results indicate, for the high resolution case, Dynamics is the bottleneck due to the small time step. One way to increase the time step is to handle terms in the differential equations responsible for gravity waves with an implicit integration step. This is known as a semi-implicit scheme. Another approach is to integrate the gravity wave terms at a smaller time step than the other terms (the split-explicit approach). Stability problems posed by advection could be relaxed with a semi-lagrangian time-integration scheme. It is expected that some combination of these and other schemes will double Dynamics throughput.

Single PE optimization will largely be achieved by better cache re-use. Preliminary analysis shows that the local storage for one sounding in the longwave code for the high resolution case could fit entirely in cache. Obtaining such a fit should enhance performance. A similar strategy could be applied to the shortwave and Fast Physics codes. Further single PE optimization may require more severe measures such as re-organizing data structures and writing key components in assembly language. Of course, such modifications would degrade vector performance on parallel vector machines as well as the clarity of the code itself.

As mentioned, communications latency is significant. Much of this latency appears to be due to unnecessary overhead in the "communication primitives" software. Efforts are underway to eliminate this overhead by eliminating communication buffers and superfluous memory access. Elimination of this excess latency should, for example, enable the aforementioned coupled run to scale well beyond 32 processors.

Off-line experimentation suggests that load balancing will improve the performance of the shortwave and LSM calculations. The re-distribution of data is determined ahead of time so the only cost is the actual communication. Some benefit could also be gained from a load-balanced polar filter since at 512 PEs, 60% of the polar filter time is spent doing the actual FFT. For RAS, it is possible no improvement at all will be achieved since a great deal of the run-time would have to spent determining how the data should be re-distributed.

For the long runs currently in progress, relatively little diagnostic output is needed so the cost of I/O is insignificant. It is estimated that even a planned 5-fold increase in model output will not present any great difficulty. Should this turn out not to be the case or if even more extensive output is needed then parallel/asynchronous I/O may be required. Development of parallel I/O software is discussed in Sawyer et al. [18].

In conclusion, a parallel atmospheric general circulation model that successfully exploits the power of MPP's such as the Cray T3E has been developed. The model is being used for high resolution runs as well as ensembles of low resolution cases. On-going efforts to improve single processor performance, reduce communication overhead, and mitigate load imbalances will enable even more effective use of these powerful machines.

Acknowldegements

This project is supported by the Global Modeling and Analysis Program in NASA's Office of Mission To Planet Earth under RTOP No. 622-24-47. Access to the Cray T3E-600 was provided by the Earth and Space Sciences (ESS) component of the NASA High Performance Computing and Communications (HPCC) Program. We would like to acknowledge Jim Abeles of SGI/Cray for the help he has provided over the years in design and optimization. Thanks also go to Tom Head of Carnegie Mellon for early work on the project and to Paul Schopf of George Mason University for his ideas on the GCM design. Further information on the parallel code may be obtained by contacting the author at the email address listed on the first page.

References

[1] A. Arakawa and M.J. Suárez, Vertical differencing of the primitive equations in sigma coordinates, *Mon. Wea. Rev.* **111** (1983), 34–45.

[2] R. Asselin, Frequency filter for time integrations, *Mon. Wea. Rev.* **100** (1972), 487–490.

[3] J.A. Brown and K. Campana, An economical time-differencing system for numerical weather prediction, *Mon. Wea. Rev.* **106** (1978), 1125–1136.

[4] F.O. Bryan, B.G. Kauffman, W.G. Large and P.R. Gent, The NCAR CSM flux coupler, *NCAR Technical Note*, NCAR/TN-424+STR, May 1996.

[5] M.D. Chou, A solar radiation model for use in climate studies, *J. Atmos. Sci.* **49** (1992), 762–772.

[6] M.D. Chou and M.J. Suárez, An efficient thermal infrared radiation parameterization for use in general circulation models, *NASA Technical Memorandum* **3** (1994), 104606, 84pp.

[7] M. Déqué and J. Piedelievre, High resolution climate simulation over Europe, *Climate Dynamics* **11** (1995), 321–339.

[8] J. Drake, I. Foster, J. Michalakes, B. Toonen and P. Worley, Design and performance of a scalable parallel community climate model, *Parallel Computing* **21** (1995), 1571–1591.

[9] P.W. Jones, C.L. Kerr and R.S. Hemler, Practical considerations in development of a parallel SKYHI general circulation model, *Parallel Computing* **21** (1995), 1677–1694.

[10] R.D. Koster and M.J. Suárez, Modeling the land surface boundary in climate models as a composite of independent vegetation stands, *J. Geophy. Res.* **97** (1992), 2697–2715.

[11] M. Lal, U. Cubasch, J. Perlwitz and J. Waszkewitz, Simulation of the indian monsoon climatology in ECHAM3 climate model: sensitivity to horizontal resolution, *Intl. J. Climat.* **17** (1997), 847–858.

[12] J. Levesque, Optimizing POP for a cache based architecture, in: *Proceedings Second International Workshop on Software Engineering and Code Design in Parallel Meteorological and Oceanographic Applications '98*, Greenbelt, MD, USA, 1998.

[13] J. Lou and J. Farrara, Performance analysis and optimization on the UCLA parallel atmospheric general circulation model code, in: *Proceedings Supercomputing '96*, Pittsburgh, PA, USA, ACM-IEEE, 1996.

[14] J. Louis, M. Tiedke and J. Geleyn, A short history of the PBL parameterization at ECMWF, in: *ECMWF workshop on Planetary Boundary Layer Parameterization*, Reading, 1982, pp. 59–80.

[15] S. Manabe and R.J. Stouffer, Multiple-Century response of a coupled ocean-atmosphere model to increase of atmospheric carbon dioxide, *J. Climate* **7** (1994), 5–23.

[16] J. Michalakes, The same source parallel MM5, in: *Proceedings Second International Workshop on Software Engineering and Code Design in Parallel Meteorological and Oceanographic Applications '98*, Greenbelt, MD, USA, 1998.

[17] S. Moorthi and M.J. Suárez, Relaxed Arakawa-Schubert: A parameterization of moist convection for general circulation models, *Mon. Wea. Rev.* **120** (1992), 978–1002.

[18] W. Sawyer, R. Lucchesi, P.M. Lyster, L.L. Takacs, J. Larson, A. Molod, S. Nebuda and C. Pabon-Ortiz, Parallelization aspects of an atmospheric general circulation model for data assimilation, in: *Proceedings High Performance Computing '98*, Boston, MA, USA, 1998.

[19] P. Schopf and A. Loughe, A reduced-gravity isopycnic ocean model – hindcasts of El Niño, *Mon. Wea. Rev.* **123** (1995), 2839–2863.

[20] R. Shapiro, Smoothing, filtering and boundary effects, *Rev. Geophys. Space Phys.* **8** (1970), 359–387.

[21] A.J. Simmons, D.M. Burridge, M. Jarraud, C. Girard and W. Wergen, The ECMWF medium-range prediction models development of the numerical formulations and the impact of increased resolution, *Meteorol. Atmos. Phys.* **40** (1989), 28–60.

[22] R. Skalin and D. Bjorge, Implementation and performance of a parallel version of the HIRLAM limited area atmospheric model, *Parallel Computing* **23** (1997), 2161–2172.

[23] M.J. Suárez and L.L. Takacs, Documentation of the Aries/GEOS dynamical core Version 2, *NASA Technical Memorandum* **5** (1995), 104606, 58pp.

[24] S.J. Thomas, A.V. Malevsky, M. Desgagná, R. Benoit, P. Pellerin and M. Valin, Massively parallel implementation of the Mesoscale Compressible Community model. *Parallel Computing* **23** (1997), 2143–2160.

[25] J. Zhou, Y.C. Sud and K.-M. Lau, Impact of orographically induced gravity-wave drag in the GLA GCM, *Quart. J. Roy. Meteor. Soc.* **122** (1996), 903–927.