Parallelizing a Global Atmospheric Chemical Tracer Model

James Demmel*

Computer Science Division & Math. Dept.
University of California
Berkeley, CA. 94720
demmel@cs.berkeley.edu

Sharon Smith* *

Computer Science Division
University of California
Berkeley, CA. 94270
ssmith@cs.berkeley.edu

Abstract

We describe the design of a parallel global atmospheric chemical tracer model. This is part of a larger Earth Systems Model (ESM) for atmospheric dynamics and physics, and ocean physics. The ESM will be used for global warming and ozone depletion studies, among others. Our goal is to parallelize the chemistry model to run efficiently on current and future parallel architectures, including the Sequoia archival database. We describe our preliminary design, especially the detailed timing model, and some initial validations. We have found a timing modeling very valuable for designing for a large range of architectures.

1 Introduction

The Earth Systems Model we are building is an HPCC Grand Challenge project intended to give insight into the future of the earth's climate, including predictions of global warming, ozone depletion, and El Niño. To accomplish this, it must include coupled models of the atmosphere and oceans. The starting point will be the UCLA general circulation model of the atmosphere (AGCM) [1], the GFDL/Princeton University general circulation model of the ocean (OGCM) [2], and the AMES/UCLA chemical tracer model (GATOR) [7]. These models are being parallelized, coupled, and linked to the Sequoia database system [9]. GATOR, which stands for Gas, Aerosol, Transport and Radiation model, was developed by R. Turco, O. Toon, R. Lu and M. Jacobson. GATOR models atmospheric chemistry in the Los Angeles Basin, and has been used for detailed air pollution studies. GATOR includes gasses and aerosols, modeling gas-phase chemistry; aqueous-phase chemistry; radiation transfer; aerosol nucleation, coagulation, growth and evaporation; horizontal advection and vertical convection; dry and wet deposition; visibility and emissions. Our task is to scale up GATOR into a model for the whole earth, as well as parallelizing it and coupling it to the other system components.

Our design goals are as follows. First, since enormous amounts of computation are involved, we want our parallel implementation to be highly efficient on existing machines. Second, since the data sets are also enormous, we must carefully account for I/O time (between model components, to the Sequoia archiver, and to interactive visualization tools). Third, since the details in the model will continue to change over time, we want to establish a way for earth scientists' future prototypes to be easily parallelized. Fourth, since the lifetime of most of the model is likely to be much longer than any parallel machine, we want our design to port to future machines without great performance loss or significant rewriting effort.

We have decided that to accomplish these goals a detailed performance model is required. The body of this paper describes our timing model for GATOR. As a function of GATOR parameters (such as grid resolution and number of chemical species), machine parameters (such as number of processors, communication bandwidth and latency), data layout and algorithm, our model predicts the number of wall-clock seconds needed to simulate one day of climate. Similar timing models have been developed for benchmarking architectures using the shallow water equations [8], and in the development of parallel algorithms for various algorithms such as sorting and FFT using the LogP model [3]. This exercise has been very valuable in identitying bottlenecks, eliminating design alternatives, and giving feedback to earth scientists as to what simulations they can expect to run.

The rest of this paper is organized as follows. Section 2 describes the machine parameters we need to

*The authors acknowledge the support of UCLA/NASA subcontract NAG5-2224A.

predict performance, including those of the database. Section 3 describes GATOR in enough detail to justify our model of it, and talks about the major features affecting performance. Section 4 describes our various algorithms for horizontal advection, which is the internal part of GATOR requiring the most interprocessor communication, and describes the overall performance model. Section 5 presents performance predictions for various scenarios, and draws conclusions about the ultimate design and remaining bottlenecks.

2 Machine parameters

We examined the performance of our timing models on the architectures described in Table 1. The architectures are characterized by the average time to perform a logical or floating point operation, the communication startup cost, the communication bandwidth (or the per-byte cost of sending a large message) and the number of processors in the latitudinal (PLAT) and longitudinal (PLONG) directions. Those parameters marked with an '*' have been measured specifically for this application, as in the case of the CM-5 numbers which were measured using Fortran F77 and the CMMD message passing library, Those parameters marked with a '†' represent estimates of what is currently attainable performance on a particular architecture. For example, on the DEC Alpha gigaswitch network, we assume that the overhead and latencies arise from using a software message passing system such as PVM for communication [4]. The Mflop rate for the 'Alpha 1' group was measured using GATOR, although with adjustments to the code to improve cache performance, we expect to realize higher Mflop rates. On the C-90, the megaflop rates used are those measured in the existing LA Basin code and the communication is assumed to occur through shared memory. The first group of architectures represents configurations currently available and being used by the UCLA ESM group. The second group of architectures shows different configurations of workstation clusters with current and future architectural features. Workstation clusters are becoming increasingly popular as a computing medium, so our performance results include a discussion of these types of systems and their performance.

3 Structure of GATOR

The steps involved in the GATOR model are described by the following pseudo-code:

```
Initialization.
Repeat
   1. Read input every INPUT seconds.
   2. Transport chemical species in
      horizontal and vertical directions.
   3. Solve ODEs to predict new concentrations
      of species for the next time step.
   4. Write output data every OUTPUT seconds.
Until simulation is done.
```

The transport and ODE solver computations use a rectangular volume of grid cells that corresponds physically to the globe with axes latitude, longitude, and altitude. Depending on the type of chemical reactions and whether aerosols are included in the model, each grid cell may contain one hundred to a few thousand chemical species. The chemical reactions can be characterized by their degree of stiffness, which is influenced by the time of day, the season, and the concentration of chemical species and emissions in the atmosphere. These parameters form the basic characterization of the application for our performance models.

The input to GATOR is comprised of several types of information that are calculated by the meteorological model (the UCLA/AGCM), such as wind velocity, temperature, water vapor, pressure, humidity, convective coefficients and optical depth. Each of these quantities requires at least 1 value per grid cell and so the amount of data input to GATOR can be quite large (from 7 - 50 quantities per grid cell). It is also desired to periodically archive the species' concentrations to the Sequoia database (step 4). This also requires a large number of quantities for each grid cell, but the frequency of data archival is much less than that of the input. The first three steps occur every iteration, which may be every 5 - 30 simulated minutes, whereas the fourth step occurs every 12 simulated hours.

Communication arises in the atmospheric chemistry code because of the necessity to transport the concentrations of chemical species in the horizontal and vertical directions. The system also incorporates a radiation transfer model and meteorological processes that require the movement of data. As the bulk of data movement occurs in the vertical domain, we have chosen to partition the grid cells by vertical columns, so that vertical transport of data can occur within a processor. The greatest impact of the communication will thus be in the horizontal direction, so our models describe different methods for implementing the horizontal transport of chemical species. The chemistry code achieves this using a tridiagonal-like solver first from east to west and then from north to south.

Table 1: Parameterization of parallel architectures for GATOR model.

Architecture Characteristics	CM-5 (no VUs)	Paragon	C-90	Alpha 1 (gigaswitch)	Alpha 2 (gigaswitch)	RS/6000 1 Ethernet	RS/6000 2 ATM	RS/6000 3 ATM
# Procs	256	256	16	9	9	256	256	256
Mflops/node	2*	12†	300*	4*	40†	40†	40†	40†
per-node BW (MB/sec)	5*	25†	16,000†	12.5†	12.5†	0.0039†	19†	19†
Latency (mu-secs)	100* (OS)	300† (OS)	0.05† (Mem)	5000† (PVM)	100† (OS)	5000† (PVM)	5000† (PVM)	100† (OS)

The ODE solver, which is based on an implicit Gear code, is the most expensive routine in the system and can take up to 98% of the time on a single processor Cray C-90. This, together with the resolution of the physical domain and the stiffness of the ODEs in each grid cell dictates the amount of time required by the ODE solver routine for each cell. Because of the variation in the amount of computation from cell to cell, we also investigated various static mapping strategies of vertical columns to processors. In particular we considered block, cyclic, and block-cyclic static mapping strategies, as in HP Fortran [6].

4 Algorithms and Timing Models

The performance models for the parallel system include the time necessary for the ODE solver, all the routines that involve horizontal transport, and I/O. Each of the parameters associated with the models in terms of application parameters, system parameters, and layout parameters are shown in Table 2. In this discussion, we detail each part of the model and then give the "total cost" of the algorithm for the simplest case.

The ODE solver is highly parallelizable along the grid cell dimension and involves no communication between processors. We have assumed that this portion of the code will scale perfectly with the number of the processors, so the performance model estimates the cost of the ODE solver as being the sequential cost of the algorithm divided by the number of processors involved. The optimal parallel cost of the algorithm, assuming perfect load balance is given below.

$$\mathrm{T_{ode}} = \frac{\mathrm{NLAT \times NLONG \times NVS \times ode}}{\mathrm{PLAT \times PLONG \times m}} \qquad (1)$$

We extrapolated values for *ode* from runs of the atmospheric chemistry model on a computational grid for the LA Basin of resolution $10 \times 10 \times 5$ (latitude by longitude by vertical layer), with 92 species and stiff ODEs. This part of the model has been validated the with the measured performance for one processor of the C-90 on the LA Basin code using 10,000 grid cells (a $40 \times 50 \times 5$ resolution) and 92 chemical species. Our model was within 25 % of the experimental time. We do not expect that the ODEs for the globe will be as stiff as for the LA Basin, so a smaller value of *ode* is used to model this situation.

We have investigated several strategies for advecting the species concentrations between neighboring grid cells for this tridiagonal-like solver: (1) **minimum message**, a strategy which sends a minimum number of large messages containing the species data between processors in order to minimize the latency and overhead costs in communication startup; (2) **pipelining**, a strategy which pipelines species data between neighboring processors in the transport in order to overlap communication and computation as much as possible; and (3) **transpose**, a full transpose of the species information from a vertical column to a horizontal column layout in order to perform the forward and backward solves locally.

We developed performance models for all three of the above strategies for block, cyclic, and block-cyclic partitionings of the horizontal domain. The cost for each strategy is expressed in terms of latency and overhead, bandwidth, and computational costs. Equation 2 shows the overall cost of the horizontal transport.

$$T_{trans} = T_{EW} + T_{NS} \qquad (2)$$

The details of the east-west transport (T_{EW}) are described in Equation 3 for the minimum message strategy, and in Equation 4 for the pipelining strategy. Both strategies are expressed in terms of the system parameters described in Table 2. (The performance models indicated that the transpose was not competitive so we did not consider it further). In both the minimum message and the pipelining strategies, the equation for T_{NS} is symmetric to T_{EW}, with $NLONG$ terms interchanged with $NLAT$, $BLONG$

Table 2: Description of model parameters and abbreviations

Parameter	Description
$NLAT$, $NLONG$	Number of latitudinal and longitudinal divisions
$NVERT$, $NSPECIES$, NVS	Number of vertical layers, number of chemical species, and $NVERT \times NSPECIES$
$PLAT$, $PLONG$	Number of processors in latitudinal and longitudinal directions
$BLAT$, $BLONG$	Number of cells to a block in latitudinal and longitudinal directions
nr	Number of small messages in pipelining
N_INPUT	Number of variables input to GATOR
$NBYTES$	Number of bytes in a double precision word
i	Rate of input device (disk or network) (MB/sec)
o	Rate of output device (disk) (MB/sec)
lat	Communication latency and overhead (seconds)
bw	Communication bandwidth (MB/sec)
m	Mflop rate of processor (Mflops/sec)
$trans$	Number of floating point operations in transport inner loop
ode	Number of floating point operations per grid cell
$INPUT$	Frequency of input to GATOR (seconds)
$OUTPUT$	Frequency of output to GATOR (seconds)
S	Number of seconds in the simulation period

terms interchanged with $BLAT$, and $PLAT$ terms interchanged with $PLONG$.

$$\begin{aligned} T_{EW} &= \left\lceil \frac{2\ NLONG}{BLONG} \right\rceil \left(lat + \left\lceil \frac{2\ NLAT}{PLAT} \right\rceil NVS\ \frac{NBYTES}{bw} \right) \\ &\quad + \left\lceil \frac{2\ NLONG}{BLONG} \right\rceil \left(BLONG \left\lceil \frac{NLAT}{PLAT} \right\rceil NVS\ \frac{trans}{m} \right) \end{aligned} \quad (3)$$

$$\begin{aligned} T_{EW} &= 2\,N \left(lat + \frac{\lceil \frac{NLAT}{PLAT} \rceil NVS\ NBYTES}{nr\ \ bw} \right) \\ &\quad + \frac{2\ N\ \ BLONG \lceil \frac{NLAT}{PLAT} \rceil NVS}{nr}\ \frac{trans}{m} \end{aligned} \quad (4)$$

where $N = \min(PLONG, nr) - 1 + \max(PLONG, nr+1) \left\lceil \frac{NLONG}{BLONG\ PLONG} \right\rceil$ in Equation 4.

Input and output is modeled very simply in the GATOR timing models. There are N_INPUT variables read for every grid cell every $INPUT$ seconds and $NSPECIES$ species concentration values for every grid cell are written to disk every $OUTPUT$ seconds. The timing models for input and output are given in Equations 5 and 6.

$$\mathrm{T_{input}} = \frac{\mathrm{NLAT \times NLONG \times NVERT \times N\_INPUT}}{\mathrm{i}} \quad (5)$$

$$\mathrm{T_{output}} = \frac{\mathrm{NLAT \times NLONG \times NVS}}{\mathrm{o}} \quad (6)$$

For the total cost of the complete model, we assume the simplest situation in which there is no overlap between computation and communication or I/O. We also assume that the horizontal transport and the ODE solver steps are done as frequently as the input step. Thus, the total cost of the algorithm for a given simulation period (in seconds), S, is:

$$\mathrm{T_{GATOR}} = \mathrm{S} \left(\frac{\mathrm{T_{input}}}{\mathrm{INPUT}} + \frac{\mathrm{T_{trans}}}{\mathrm{INPUT}} + \frac{\mathrm{T_{ode}}}{\mathrm{INPUT}} + \frac{\mathrm{T_{output}}}{\mathrm{OUTPUT}} \right) \quad (7)$$

5 Performance Results

In this section, we present estimates for the execution time of GATOR for different architectures characterizations, data layouts, and the two horizontal transport algorithms. We also show timing results for the Earth Systems Model that illustrate some of the performance problems with communication between the GATOR model and the UCLA AGCM.

5.1 GATOR Performance

There are several factors that affect the standalone performance of GATOR. The horizontal transport phase is the communication intensive portion of the algorithm, so we wanted to choose an algorithm which minimizes this phase of GATOR. In addition, the possibility for load imbalance exists and we wanted to measure how badly it could affect performance, and whether the use of block-cyclic or cyclic layouts would help to reduce load balance without introducing excessive communication costs.

To study the effects of load imbalance, we used the timing models to examine how variation in computation due to seasons and night and day affect the overall performance of GATOR for various data layouts and architectures. The ODEs tend to take much longer to solve for daytime cells than nighttime cells. Seasons

Table 3: Predicted Performance for $2^o \times 2.5^o \times 20$, 92 species, without I/O

Machine	SPEEDUPS							
	Pipeline				Minimum Message			
	Block	Block-UB	B-Cyclic	B-Cyclic-UB	Block	Block-UB	B-Cyclic	B-Cyclic-UB
CM-5 (256 PEs)	248	99	245	189	108	65	95	86
Paragon (OS) (256 PEs)	233	97	214	170	105	64	91	82
Paragon (PVM) (256 PEs)	123	70	74	68	99	62	80	73
Alpha 1 (PVM) (9 PEs)	8.8	3.5	7.7	6	7.7	3.3	7.3	5.8
Alpha 2 (PVM) (9 PEs)	8	3.3	3.3	2.9	7.4	3.3	4.5	3.9
Alpha 2 (OS) (9 PEs)	8.8	3.5	7	5.7	7.4	3.2	4.6	4

ODE time in seconds/simulated day
(Transport time in seconds/simulated day)

CM-5 (256 PEs)	571 (66)	1522 (66)	571 (74)	761 (74)	571 (898)	1522 (898)	571 (1088)	761 (1088)
Paragon (OS) (256 PEs)	95 (18)	253 (18)	95 (28)	126 (28)	95 (155)	253 (155)	95 (194)	126 (194)
Paragon(PVM) (256 PEs)	95 (118)	253 (118)	95 (260)	126 (260)	95 (170)	253 (170)	95 (233)	126 (233)
Alpha 1 (PVM) (9 PEs)	8140 (813)	21700 (813)	8140 (2194)	10850 (2194)	8140 (2092)	21700 (2092)	8140 (2767)	10850 (2767)
Alpha 2 (OS) (9 PEs)	814 (89)	2171 (89)	814 (310)	1085 (310)	814 (258)	2171 (258)	814 (898)	1085 (898)
Alpha 2 (PVM) (9 PEs)	814 (183)	2171 (183)	814 (1566)	1085 (1566)	814 (260)	2171 (260)	814 (936)	1085 (936)

also effect load balance, with summer grid cells requiring more computation than winter grid cells. Because of these effects, a blocked layout of the computational grid to the globe will have the worst performance, since it is very likely that a single processor holding only daytime and summer grid cells will be much slower than a processor holding nighttime and winter cells. Using block-cyclic or cyclic data layouts, we expected that the load imbalance might be lessened, but perhaps at the expense of additional communication.

We examine 3 possible layouts for both pipelining and minimum message horizontal transport. The resolution of the computational grid is $96 \times 144 \times 20$ ($NLAT \times NLONG \times NVERT$), with 92 species, which is approximately a 2^o latitude $\times 2.5^o$ longitude $\times 20$ layer resolution of the globe. For this resolution, the maximum time for block-cyclic layout and cyclic layout will result if a processor has $\frac{1}{3}$ of its cells in summer daytime and $\frac{1}{3}$ of its cells in winter daytime. For block layout, a processor could have all of its cells in summer daytime. If we weight the cost of summer daytime cells to be twice as expensive as winter daytime cells, and assume that the remaining cells have negligible cost, then the ODE part of the computation for blocked layout can be $2\frac{2}{3}$ times more expensive than in the optimally load-balanced case shown in Equation 1. The block-cyclic and cyclic layouts will be $1\frac{1}{3}$ times more expensive than the optimally load balanced case.

Table 3 shows this scenario with the speedups and execution times for one simulated day that are observed in the ODE computation and the horizontal transport. The number of seconds are the number of wall clock seconds for that simulation period. The "Block" and "B-Cyclic" cases assumes perfect load balance, whereas the "Block-UB" and "B-Cyclic-UB" cases represent the unbalanced situations described above.

Table 3 illuminates several important aspects of GATOR performance. First, it shows that significant load imbalance can occur for the scenario we have considered and the choice of a mapping strategy will have a significant impact on GATOR's overall execution time. Note that the block-cyclic layout has the best "unbalanced" performance for all of the systems that have relatively low communication latency for the pipelining transport strategy. The penalty for extra communication is outweighed by the improvement in load balance, except when communication overhead costs are high, as with PVM. It is also evident from Table 3 that the pipelining strategy has consistently better performance than the minimum message strategy, except in the case of large numbers of messages and high communication latency for the block-cyclic layout.

The performance models for pipelining and minimum message have been validated using cyclic and block mappings on a 32 processor CM-5. The implementation of these strategies showed reasonable agreement with the predicted model timings, with a relative difference of at most 20 % between the predicted timings and the observed timings. Pipelining of messages on this machine was also observed to be the fastest strategy (up to 5 times faster than its main competitor minimum message).

5.2 Earth Systems Model Performance

In addition to investigating the stand-alone performance of GATOR, we have examined the role of I/O in GATOR's interaction with the Earth Systems Model and the Sequoia archival database. We have considered two scenarios. The first scenario is a"forced" model in which GATOR reads input from files stored on disk. The data may have been previously generated by the UCLA AGCM, or it might also come from meteorological data sets. For this situation we examine the performance of parallel vs. serial I/O. In the second scenario, GATOR is coupled with the UCLA AGCM and expects input from this model every $INPUT$ seconds. The performance issue of interest in this case is whether the models are physically distributed or placed on the same multiprocessor.

The performance of GATOR for the first scenario is shown in Table 4. The times for 12 simulated hours are shown for the ODE solver, the horizontal transport, and both serial and parallel I/O. We assume in each case that serial I/O comes through a host node via a 2 Mbyte/sec disk server and is the primary bottleneck in distributing the data to each of the processing elements. In the case of parallel I/O, we assume that the Cray has a 10 MB/s disk per node and that in the other systems every processor has its own local disk and inputs data from the disk at the rate of 2 MB/s per second. If some type of disk striping is used as in a RAID system [5], then the aggregate parallel I/O bandwidth would be $0.8 \times PLAT \times PLONG \times i$. (The 0.8 factor arises because of the necessity for data redundancy in RAID systems.) The particular test case shown is for a global resolution of 4^o latitude $\times 5^o$ longitude $\times 20$ vertical layers, with 92 species. The transport strategy used is pipelining with a block layout and we assume that the load is perfectly balanced. 50 variables of meteorological data (approximately 27 Mbytes) are input at a rate of once every 5 minutes, and all of the chemical species (approximately 51 Mbytes representing one double precision floating point number for each species for each grid cell) are output once every 12 hours.

Table 4 clearly shows that serial I/O presents a significant performance bottleneck, but that the use of parallel I/O greatly improves the situation and keeps the overall I/O cost in line with the other system costs.

The second scenario examines the situation in which GATOR and the UCLA AGCM run in a coupled mode, either on separate multiprocessors or on the same multiprocessor. In the former case, we assume that the models communicate over a T3 network, with a communication bandwidth of up to 6 Mybtes/sec. In the latter case, both models run on the same multiprocessor and communicate via the local interconnect network. We make the simplifying assumption in this case that the models have the same data layouts so that a processor i running GATOR on some partition of the globe communicates directly with a single processor j running that same global partition. Furthermore, we assume that the aggregate bandwidth of the interconnect network is $PLAT \times PLONG \times bw$. In practice, this number maybe smaller if all processors are communicating at once, however, this should give an indication of what type of performance might be observed.

Table 5 shows the input times for both of these scenarios. (We assume that all output goes to disk, which can be done with either serial or parallel I/O as indicated in Table 4.) The problem resolution, data layout, and transport strategy are the same as those used in Table 4. Considering only the input costs, it is evident that a distributed implementation of the models would be extremely costly. Even if the aggregate bandwidth of the interconnect network is 10 times slower, having both models on the same multiprocessor is a better approach.

5.3 Workstation Clusters

As observed earlier, workstation clusters have become increasingly popular as a communication medium. Table 6 presents a comparison of different workstation configurations, all based on clusters of RS 6000s. The purpose of this table is to show what system components are essential for the high performance of GATOR on workstation clusters. It also shows how each component of a "well" balanced system plays a crucial role in the performance of this application. The first table entry shows a workstation cluster with 256 processors connected via an ethernet and sharing a single 2 Mbyte/sec disk, with PVM as the communication medium. In the second table entry, the network has been upgraded to a 155 Mbit/sec ATM network. The increase in bandwidth improves the hori-

Table 4: Predicted Performance for $4^o \times 5^o \times 20$, 92 species, and parallel and serial I/O

	times for 12 hours of simulation					
			Parallel I/O		Serial I/O	
Machine	ODE	TRANSPORT	Input	Output	Input	Output
CM5 (256 procs)	70 s	71 s	9.7 s	0.12 s	1990 s	25 s
Paragon (256 procs)	12 s	23.5 s	9.7 s	0.12 s	1990 s	25 s
C90 (16 procs)	7.2 s	4 s	25 s	0.3 s	1990 s	25 s
Alpha 2 (OS) (9 procs)	102 s	83 s	276 s	3.5 s	1990 s	25 s

Table 5: Predicted Performance for $4^o \times 5^o \times 20$, 92 species, on physically distributed multiprocessors and the same multiprocessor.

	times for 12 hours of simulation			
			Distributed	Same Machine
Machine	ODE	TRANSPORT	INPUT	INPUT
CM5 (256 procs)	70 s	71 s	796 s	3 s
Paragon (256 procs)	12 s	23.5 s	663 s	0.6 s
C90 (16 procs)	7.2 s	4 s	663 s	0.01 s
Alpha 2 (OS) (9 procs)	102 s	83 s	663 s	35 s

zontal transport phase significantly, but the sequential file system is still a limiting factor. In the third table entry, the sequential file system is replaced by a RAID style parallel file system with a 2 Mbyte/sec disk at each node. This significantly diminishes the I/O cost. Finally, in the last table entry the overhead and latency costs of the network have been reduced to around 100 microseconds, using a low latency communication medium such as active messages [10] on the CM-5 parallel machine. This configuration makes the workstation cluster competitive with the MPP and vector machines shown in Table 4.

5.4 Summary

We have introduced performance models that have been useful in identifying the important aspects of the design of a parallel implementation of GATOR. First, the models (and their validation with experimental measurements) have demonstrated that the pipelining strategy has a significant performance advantage over the minimum message scheme in the horizontal transport of species, except for architectures in which the overhead costs associated with communication are very high. Second, our models have indicated that a static block-cyclic mapping strategy will provide reasonable performance in the face of load imbalance arising from seasonal and diurnal variations in the stiffness of ODEs. Third, the models revealed that I/O can be a limiting factor for performance on all the machines. The availability of parallel I/O will be essential to acceptable performance of forced simulations, and coupled model simulations will most likely need to be run on a single multiprocessor unless some other intermediate solution can be devised. Finally, the models have shown that in order to have balanced overall system performance, this application requires an architecture that has respectable computation speed, high communication bandwidth, low communication latency, and the availability of parallel I/O. Certainly any of these aspects would contribute to good performance in many parallel applications, however, for GATOR, all aspects must be present for the computation to achieve reasonable performance.

We have found the models to be extremely useful in our design of GATOR and in uncovering potentially serious bottlenecks in the Earth Systems Model. Given that our preliminary experimental results on the CM-5 agree well with the corresponding performance models, the models should be reliable indicators of the parallel system performance.

Table 6: Predicted Performance for $4^o \times 5^o \times 20$, 92 species, for various workstation configurations

	times for 12 hours of simulation			
Machine	ODE	TRANSPORT	Input	Output
RS 6000 (256) (Ethernet)	4 s	23340 s	3980 s	50 s
RS 6000 (256) + ATM	4 s	192 s	1990 s	25 s
RS 6000 (256) + parallel file sys	4 s	192 s	10 s	0.1 s
RS 6000 (256) + low latency	4 s	11 s	10 s	0.1 s

6 Future Work

The parallel implementation of GATOR is now underway. We are implementing the block-cyclic layout with pipelined horizontal transport for several of the architectures discussed here. We intend to continue to use and develop the models to aid in our current design and implementation of the GATOR code as well as in the coupling of Earth system components.

Our initial experience with running the ODE solver on the DEC Alpha indicates that it may be necessary to restructure the code for this kernel if reasonable cache performance is to be realized. Part of our effort will be focused on obtaining decent floating point performance for the ODE solver on the various target architectures.

Acknowledgements

The authors wish to thank Ken Stanley and Howard Robinson for their helpful comments during the development of the timing models.

References

[1] A. Arakawa and V. R. Lamb, *Computational design of the basic dynamical processes of the ucla general circulation model*, Methods in Comput. Phys., 17 (1977), pp. 173–265.

[2] M. D. Cox, *A primitive, 3-dimensional model of the ocean*, Tech. Rep. 1, GFDL Ocean Group, 1984.

[3] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, *Logp: Towards a realistic model of parallel computation*, in Proc. ACM SIGCOMM Conference, San Diego, 1993.

[4] J. Dongarra, A. Geist, R. Manchek, and V. Sunderam, *Integrated PVM framework supports heterogeneous network computing*, Computers in Physics, 7 (1993), pp. 166–175.

[5] A. L. Drapeau, K. Shirriff, J. H. Hartman, E. L. Miller, S. Seshan, R. H. Katz, K. Lutz, D. A. Patterson, E. K. Lee, P. M. Chen, and G. A. Gibson, *RAID-II: A high-bandwidth network file server*, in Proceedings of the 21st International Symposium on Computer Architecture, Chicago, IL, apr 1994. To appear.

[6] High Performance FORTRAN Forum, *High performance FORTRAN language specification*, Scientific American, 2 (1993).

[7] M. Z. Jacobson, R. P. Turco, R. Lu, and O. B. Toon, *Gator: A gas, aerosol, transport, and radiation modeling system for studying urban and regional air pollution*, (1994).

[8] R. Pozo, *Performance Modeling of Parallel Architectures for Scientific Computing*, PhD thesis, University of Colorado, Boulder, 1991.

[9] M. Stonebraker and J. Dozier, *Sequoia 2000: Large capacity object servers to support global change research*, Sequoia 2000 Technical Report 91/1, University of California, Berkeley, CA, July 1991.

[10] T.von Eicken, D. E. Culler, S. Goldstein, and K. Schauser, *Active messages: A mechanism for integrated communication and computation*, in Proceedings of the 19th Annual International Symposium on Computer Architecture, may 1992, pp. 455–472.