DAO Office Note 97-13

Office Note Series on Global Modeling and Data Assimilation

Richard B. Rood, Head
Data Assimilation Office
Goddard Space Flight Center
Greenbelt, Maryland

Design of the Goddard Earth Observing System (GEOS) Parallel General Circulation Model (GCM)

Peter M. Lyster, William Sawyer, Lawrence L. Takacs

Data Assimilation Office, Goddard Laboratory for Atmospheres

Goddard Space Flight Center
Greenbelt, Maryland 20771
Aug 27 1997

Abstract

This document contains the requirements and design of the parallel Goddard Earth Observing System General Circulation Model (GEOS-3 GCM). This derives from a week-long internal workshop that were conducted at the NASA/Goddard Space Flight Center Data Assimilation Office (DAO) from January 13 to 17 1997, and subsequent discussions within the DAO Modeling group. The purpose of this workshop was to review the requirements for the developmental parallel GCM, and to lay the groundwork for the design and prototyping of the operational GCM that will be a part of the Goddard Earth Observing System data assimilation system GEOS-3. The Modeling group discussions determined the detailed design.

The subsystem will use the *Message Passing Interface* (MPI) library, with considerable heritage FORTRAN 77 code, and with an overarching Fortran 90 modular design. A parallel computer architecture is being targeted for running production code.

Note that this is a working document and has not yet been baselined by the Configuration Control Board (CCB) for the GOES-3 Project (May 1, 1997)

Contents

1 Introduction

The Goddard Earth Observing System General Circulation Model (GEOS GCM) is a significant part of the GEOS atmospheric data assimilation system (DAS) that is used by the Data Assimilation Office (DAO). Apart from the considerable technology surrounding transmission of data to and from data facilities, storage, Input/Output, and data visualization, the core components of GEOS are: an atmospheric general circulation model (GCM); the data quality control (QC); and a data assimilation analysis algorithm Physical-space Statistical Analysis System (PSAS). These algorithms are, because of the nature of their underlying physical models, internally highly coupled and compute-intensive. The coupling between the core algorithms is less complicated, and is achieved through defined interfaces (Staff, 1997a, Sawyer 1997).

The DAO is preparing to move its data assimilation system to advanced computing platforms. The system is being designated GEOS-3. This will be part of its regular operation, although an important role is expected for the Mission to Planet Earth (MTPE) system in the coming years. The end-to-end data system will comprise considerable data handling algorithms that will run on workstation environments. However the core compute algorithms require in excess of gigaflop/s of performance and will have to be implemented in high-performance computers. A parallel computer architecture is being targeted for production, and the Message Passing Interface (MPI) parallel library is being adopted as the defacto standard for communication software (Farrell *et al.*, 1996). A number of workshops were held in 1996 and 1997 focusing on the parallel aspects of online QC and PSAS. The requirements, design, and implementation of these algorithms is proceeding (Lyster *et al.* 1996) (note that online algorithms are implemented on the high-performance computers, whereas off-line algorithms are implemented on workstations.)

The GCM is a grid-point based algorithm (Takacs *et al.* 1994) that comprises: a dynamical core; a number of physics packages (moisture, turbulence, land-surface processes, shortwave and longwave radiation); I/O; and a driver that can be configured to run the algorithms in either forecast mode or in assimilation mode. A number of these components have been parallelized and the performance assessed by the group led by Max Suarez at GSFC.

The workshop that led to this document was intended to first summarize the requirements (Stobie 1996) and to initiate the prototyping and design for the parallel GEOS-3 GCM. Key issues for the design was the extent to which existing parallel algorithms could be used, the difficulties of converting the GCM to both Fortran 90 and MPI, and the identification of potential "show stoppers" for performance.

Since this development is a substantial shift for the DAO there is a considerable risk to achieving satisfactory performance. Fortran 90 can be used to impose a more modular (object oriented) software approach, but compilers are not yet optimized across all the potential platforms. Also, the use of parallel hardware involves a shift from optimized vector code to the more arcane world of optimizing cache on RISC processors for single-processor performance. Finally, message-passing itself often requires the use of customized code that minimizes the time-cost of transmitting data to remote memory (in this write-up local and remote memory refer to physical random access memory on distributed-memory platforms), and minimize the load imbalance that may occur for distributed processing.

The difficult task of configuration management which will (among other things) ad-

dress the process of synchronizing the concurrent development of the core GCM, QC, and PSAS so that they can run on distributed memory computers at the same time will be addressed in the GEOS-3 Software Design Document.

2 The Scientific Algorithm

The GEOS-3 GCM is an incremental development of the GEOS-2 GCM (Staff, 1996). For the GCM, this involves some changes to the physics algorithms and considerable changes to the software implementation – this document is mainly concerned with the latter.

The GCM is comprised of the following modular components (Takacs *et al.* 1994):

- **Dynamical core** which solves the primitive equations of atmospheric dynamics. This algorithm uses a finite-difference latitude-longitude grid, and there are a number of schemes for implementing the temporal evolution in finite-difference form. These schemes produce tendencies which are gridded increments to the primary prognostic fields (surface pressure, horizontal velocity, and flux forms for potential temperature, moisture content, *etc.*). The dynamical core uses a high-latitude Fourier filter to suppress the CFL instability.

The remaining items are generally categorized as physical processes.

- **Moist Processes** which handles, among others, convection and precipitation processes. At present moist processes produces tendencies for temperature and moisture. Moist processes uses a physical grid that may be different from the dynamical core. As in all the physical process modules, this algorithm does not involve couplings between different horizontal grid-points.
- **Radiation** processes are handled by two algorithms that treat longwave and shortwave parts of the spectrum. The present algorithms produce tendencies for temperature. There is some dependency on fields (such as cloud fractions) that are produced by moist processes.
- **Turbulence** handles subgridscale nonlinear processes. Turbulence produces tendencies for horizontal velocity, temperature, and moisture.
- **Land Surface** handles the surface boundary. A complex algorithm (Koster and Suarez, 1992) is being prototyped, and is expected to be implemented in GEOS-3.

There are other important functions that are not included in the above list. An eighth-order Shapiro filter is used for the wind, potential temperature, and specific humidity tendencies. This is a grid-point based filter (*i.e.*, it uses a finite difference template). It iterates a second-order template four times. A local transformation of grid-points from an C grid (where dynamics calculations are performed) to a A grid (physics modules), and a rotation of the coordinate system to suppress an finite-difference cancellation instability (Suarez and Takacs 1995) are performed. At present

the dynamics are performed in a rotated coordinate system (the computational pole is on the equator of the physical grid), and the physics modules are performed in the physical grid.

Apart from the above-mentioned dynamics and physics modules, tendencies are also generated by a gravity-wave drag algorithm. Tendencies are also input from the PSAS analysis module. A single function in the driver (subroutine step) is responsible for assembling tendencies into the full fields during each time-step of the algorithm.

At present the dynamics generates tendencies on the shortest time-scales. Other modules of GCM produce tendencies consistent with the inherent physical processes, and to a lesser extent as often as the computational cost of each allows. The time-steps for the key modules are:

- Dynamics: 3 minutes.
- Moist Processes: 9 minutes.
- Turbulence and gravity-wave drag: 30 minutes.
- longwave and shortwave radiation: 3 hours.

3 Requirements

The scientific and software requirements are set out in *GEOS-3 System Requirements* (Stobie 1996). Of relevance are the following:

- **3.1** The model will use a horizontal resolution of 2^o (latitude), 2.5^o (longitude), and 70 vertical levels.
- **3.2** In reanalysis operation, the end-to-end system should be capable of approximately 30 days of assimilated datasets in one day of wall-clock time.
- **3.3** The software in distributed memory computing will use the *Message Passing Interface* (MPI). This is a derived requirement, that was assessed in review (Farrell *et al.* 1996). If necessary, mixed language third-party software may be used provided portable Fortran bindings are available.

We outline the following derived requirements that pertain specifically to the parallelization effort:

- **3.4** The parallel design must generate scientific software that will have a long life cycle. The incorporation of Fortran 90 affords the opportunity to use a modular approach that allows for expandability, clarifies data dependencies through the use of derived types, enables the use of memory management through `allocate/deallocate` statements, decreases the likelihood of bugs, and makes it easier for a larger group of scientists to use and modify the same code.

- **3.5** It is commonly acknowledged that parallel computing, and message-passing in particular, are sufficiently complex that some effort has to be made to hide the communication modules from a substantial population of the regular programmers.

- **3.6** The parallel code should scale to meet the primary performance of 30 assimilated days per day. In addition there are gigaflop/s performance that were negotiated as part of the HPCC Earth and Space Sciences ESS project (Lyster *et al.* 1995).

- **3.7** Memory will be managed by Fortran 90 `allocate/deallocate` procedures. The general rule is as follows: within the modular structure of the code memory should be allocated and deallocated where possible so as to facilitate the safe and effective heterogeneous application for GEOS DAS (viz, in assimilation mode it must be possible for GCM, QC, PSAS to run together.) Where possible allocations/deallocations should occur at the same (preferably high) level system-wide in the calling tree. The pitfalls are: memory leaks; allocating/deallocating trivial memory that incurs excessive time overhead; and allocating/deallocating significantly large memory so as to cause fragmentation. Software should be designed with this in mind so that remedial steps can be taken if problems (*e.g.*, runtime fragmentation) occur.

- **3.8** The proposed parallel GCM is expected to produce bit-wise identical results on any number of processors since there are no global reduction operations in the algorithm. That is, all floating point operations are performed in the same order as they would be on a serial implementation. This should be verified, and any exceptions explained, by prototyping.

Note that a number of vendors support (or will soon) Fortran 90 and MPI on their hardware. Issues of hardware will not be further discussed, except to note that message-passing is a safe approach for the design of large-scale tightly-coupled algorithms. This is because a strong coupling between the user-generated data domain decomposition and the physical layout of memory affords the ability to optimize and scale against communications (latency and bandwidth) overhead, as described in the next section.

4 Algorithmic and Performance Issues Associated with Parallel Computing

This section is a summary of the key algorithmic and performance issues associated with the transfer from vector-based to distributed-memory, RISC-based computing. Apart from the scientific modifications that are being made in GEOS-3, the parallel issues represent the most significant change that is being adopted by the core DAS modules.

4.1 General Discussion

The advantage of converting to message-passing software using MPI is that this portable library enables the use of parallel processing in distributed or shared memory hardware. The wall-clock time to solution may be reduced by the overlapping (parallelizing) of floating-point operations. Distributed memory also offers the promise of scaling problems to larger memory (*i.e.*, more grid-points, the assimilation of more observations, *etc.*) MPI also holds the promise of greater efficiency in distributed or shared memory configurations because it effectively manages the coarse-grained arrangement of memory (such as groups of grid-points in a GCM). Software written using MPI will be portable to all high-performance computers in the medium-term future.

The disadvantage is that MPI requires greater effort to implement than standard serial vector techniques. There is considerable arcana associated with the need to concatenate messages (to overcome, *i.e.* amortize, the latency of messages that access remote memory) to use asynchronous communications, etc. Software needs to be modularized to hide this arcana from the regular scientific programmers (in particular, key scientific functions have to be made to "look" serial by hiding message-passing functions). A lot of the heritage of multitasking on shared-memory multiprocessors is lost.

4.2 Current State of RISC-based Parallel Computing

Apart from the potential performance differences between distributed and shared memory computers, the algorithms will have to be tuned to key parameters of the hardware – notably the single-processor performance and the network performance.

- **Single-Processor Performance (Table 1):** RISC-based processing is popular because it is cheaper than vector-based. For scientific computing, the main problem is that there is a great disparity between the time for the arithmetic units to access local cache versus slower on-processor memory. This fact, combined with the small ratio of cache size to total memory means that data that are in cache must be marshaled and reused as much as possible. There are no constructs in standard languages such as Fortran or C to do this. Memory reuse is achieved by sorting data (to create locality) and shortening `do/for` loops (to achieve reuse). The reason why the PSAS analysis module has been able to achieve high performance is because of the extensive sorting of data that precedes the use of efficient BLAS linear algebra calls. The need for locality and cache reuse means that it is extremely unwise to use indirection (especially if it is done for reasons of software style rather than algorithmic necessity). For example, Cray T3D on-processor memory fetches take one hundred times longer than the time to fetch from cache, that is, unstructured memory access has the capacity to make a processor one hundred times slower! Unfortunately, the long loops that were favored for vector processors often need to be replaced by shorter loops, or at least loops that guarantee data reuse. As the size of cache increases with newer processors that problem may diminish, but it will only be meaningful if the ratio of cache to main memory increases.

Table 1.	Cray T3E	SGI Origin
Sustained single-processor megaflop rates(*)	50 - 100	50 - 100
Memory size per processor	128 Mbytes	200 Mbytes
Primary Cache size per processor	8 kbytes	1 Mbytes
Secondary Cache size per processor	96 kbytes	1 Mbytes

(*) Sustained, means for reasonably optimized code

- **Network Performance (Table 2):** The well-known Amdahl's law states that for an implementation with a finite component of unparallelized code there always exists a number of processors beyond which very little improvement in wall-clock time can be achieved. The communication overhead associated with fetching memory from remote processors also acts as a serial bottleneck. It is almost impossible to predict the scaling (*e.g.*, the maximum number of useful processors for a particular job), so prototyping is essential. This has been done for both PSAS (Lyster *et al.* 1996) and the GCM. We will discuss the GCM in the next section. In general, analogous rules apply to network message passing as to on-processor memory access. Reuse is encouraged; sometimes it takes less time to recalculate a quantity, or have every processor calculates the same number, than to communicate quantities between processors. It is better to "cache" messages in large buffers so that the cost of latency is amortized; in Table 2 the "best message size" gives a lower bound above which the sustained interprocessor bandwidth is achieved (*i.e.*, not reduced by latency).

Table 2.	T3E	SGI Origin
Interprocessor Bandwidth (sustained)	100 Mbyte/s	10 Mbyte/s
Latency	2 μs	30 μs
Optimal message size	1 Kbyte	?

In general the total flop/s rate (read, "inverse time to solution") is given by: the (hopefully optimized) on-processor flop/s rate $\times$ the number of processors $\times$ the parallel efficiency. The parallel efficiency is usually found empirically by prototyping; it depends on the amount of unparallelized algorithm, the network parameters, the skill of the programmers in minimizing messages and amortizing latency, and possibly through the use of asynchronous communication. The results of prototyping to date will be summarized in the next section.

4.3 Input/Output

I/O is a notorious problem for parallel computing because of the need to coordinate and assemble data that are resident in remote processors through a limited number of I/O channels to disk. The strategies for handling this for GEOS-3 DAS will be discussed in the forthcoming *Requirements and Preliminary Design of the GEOS-3 Parallel I/O (GPIOS) Subsystem* (Lucchesi 1997).

5 Present State of Parallel GCM

As described in section 2, the model time-step is 3 minutes (for $2^o \times 2.5^o \times 70$ levels resolution). In order to achieve 30 days of assimilated data (120 analyses) per day of wall-clock time (section 3), this means that on average a model time-step should take about 1.9 seconds (assuming the model takes 40% of the analysis). Table 3 shows the approximate percentage breakdown of CPU-time (without running diagnostics) used by modules of the GCM on a Cray J916. Further numbers are presented in Takacs 1997, and some timings are discussed in the next section. One of the main results is that on a sixteen processor J916 the model alone, (*i.e.*, without diagnostics or analysis) achieves 25 days per day performance, which is significantly below the requirements. The GCM performed at about 240 megaflop/s for this experiment.

Table 3.	Percentage of CPU Time
Dynamics and Fourier filter	26
Radiation	39
Moist Processes	22
Turbulence	13
Shapiro filter	?

At present the parallel decomposition is dominated by the decomposition of the dynamics. This is partly because the dynamics solves the atmospheric primitive equations and this involves significant couplings between horizontal grid-points. The present algorithm is fourth-order accurate, meaning that the horizontal finite-difference template is 2 grid-points in each direction.

A two-dimensional horizontal data decomposition is used (sometimes called "checker board"). Domains on each processor are compact rectangular regions of grid-points that are stored on $N_p = N_{px} \times N_{py}$ processors, where N_{px} (N_{py}) is the number of processors in the longitude (latitude) direction of the decomposition. Message passing is used to fill boundaries that surround the region of grid-points in each processor. In this way the finite difference operators act without using repeated small messages to gather the grid-points from nearby domains. The present version of DYCORE (the plug-compatible subroutine) repeatedly refreshes the boundaries during the course of one time-step; the messages were not rationalized into a single buffers because the intention of the original design was not to alter the serial code in a significant manner. This strategy may have to be revisited – it may be more efficient (and modular) to do all the communications immediately on entering DYCORE (this is discussed in more detail in section 6.4).

There are no plans to apply a decomposition over vertical levels. One reason for this is because all the physical processes have vertical dependencies, and none have horizontal dependencies. The two-dimensional decomposition of the dynamics is satisfactory for the physics modules, unless significant load imbalance occurs (*e.g.*, for shortwave physics half the domains have no work, and for land-surface processes there is obviously significant horizontal inhomogeneity in the floating-point work). In general, load imbalance may be minimized by forming a separate decomposition for the time-consuming modules. In that case, there is a trade off between the communication cost of generating a new decomposition, and the inefficiency of load imbalance. The current status and plans for the main modules of the GCM are summarized be-

low (performance on the T3D are current as of 1996 only, and should be updated as appropriate):

- **Dynamics and Fourier filter:** The parallel dynamical core (DYCORE) of Suarez was designed to be interchangeable with the DAO GCM version. The finite-difference template uses local communications to fill domain boundaries as described above. The Fourier filter is a zonal filter that is applied at high latitude. This uses global communications and there is some load balancing in the current algorithm. The parallel DYCORE subroutine scales with at least 50% efficiency up to 800 processors of a T3D (extrapolated from 512 processors). In particular the ratings are 2.0 gigaflop/s (256 pe's), 3.7 gigaflop/s (512 pe's), and 6.0 gigaflop/s (1024 pe's).

- **Shortwave Radiation:** Parallel shortwave radiation scales with at least 50% efficiency beyond 1024 processors of the T3D. This is not surprising since there is no horizontal coupling, and hence no message passing. The ratings are 9.5 gigaflop/s (512 pe's) and 17.5 gigaflop/s (1024 pe's), representing the single-processor performance in a load-unbalanced implementation (the degree of load imbalance is undocumented, but if the default two-dimensional decomposition was used the load imbalance is probably 50%.) A compressed-shortwave algorithm was rated at 8.0 gigaflop/s (512 pe's) and 14.5 gigaflop/s (1024 pe's) with a worse load imbalance. Jim Abeles reports unimpressive results for attempts to load balance the parallel shortwave radiation module.

- **longwave Radiation:** There is presently no parallel implementation of this. It should be relatively well load balanced, since longwave processes are evaluated approximately uniformly on the globe.

- **Moist Processes:** There is presently no parallel implementation of this. It should be relatively well load balanced, since moist processes are evaluated approximately uniformly on the globe.

- **Turbulence:** There is no parallel implementation of this. It should be relatively well load balanced, since turbulent processes are evaluated approximately uniformly on the globe.

- **Land Surface:** A tiled-land surface model scales with at least 50% efficiency up to 100 processors, and was rated at 0.6 gigaflop/s (512 pe's) and 0.7 gigaflop/s (scaled) (1024 pe's). The problem is probably due to a load imbalance. A balanced tiled land surface model scales with at least 50% efficiency up to 512 processors and was rated at 2.5 gigaflop/s (512 pe's) and 5.0 gigaflop/s (1024 pe's). The proposed land surface model (based on Koster and Suarez 1992) uses a maximum of 10 tiled land-surface types per grid-point of the latitude-longitude grid (the ocean is one type). There is some possibility of serious load imbalance due to surface heterogeneity across the parallel decomposition, however the large number of tiles (up to 64000) may ameliorate this.

Important functions that act on or transform field arrays are:

- **Shapiro Filter:** Although the percentage of time spent in the Shapiro filter is not given above, it is known to be about 23% of the time for DYCORE. Hence

this represents a significant time cost. The present eighth-order filter is implemented as four iterations of a five-point finite-difference filter template. Each iteration involves message passing. The current implementation scales with at least 50% efficiency up to 32 processors, and doesn't get above 0.31 gigaflop/s. This poor performance is due to a combination of poor on-processor optimization, poor load balance, and excessive communications. The last of these three may be improved by evaluating the full template for the four iterations and performing the message-passing only once per call to the filter to amortize the latency cost. Other aspects of the poor performance are being studied by Dan Schaffer.

- **A to C grid transformation:** The model development group is modifying the present spectral algorithm to use local interpolation. If this is satisfactory then parallelization should not be a problem since single-vector ghost cells will be used to implement the parallel transformation. There will be some message-passing communication cost.
- **Pole rotation:** This involves bicubic (16 point) interpolation from one latitude-longitude grid to second latitude-longitude grid whose pole is on the equator of the first (section 2). It is clear that, the simplest parallel transformation that provides the best load-balanced algorithm transforms a two-dimensional decomposition of the first grid to morphologically identical two-dimensional decomposition of the second grid. This is potentially a "show stopper" because a brute-force implementation of the transformation takes on order [8 megawords per three-dimensional gridded field / 10 megawords per second network speed] = 1 second. the present GCM performs the rotation about 7 times per time-step, which we noted above should take about 1.9 seconds in order to achieve the primary requirement of 30 days of assimilation per day. Actually, an initial analysis performed at the model workshop in January reveals that there will be a message bottleneck at the pole of the destination grid for the transformation. Therefore, messages of size [total grid size / $(N_p N_p^{1/2})$] must be sent using $N_p^{1/2}$ messages from the originating decomposition. The size and number of messages may be reduced by using a logarithmic reduction algorithm (Max Suarez personal communication). Also, the size of the messages may be reduced by performing the interpolation in the source or destination decomposition, depending on where the latitude-longitude grid is finer or coarser.

Note that the proposed parallel GCM is expected to produce bit-wise identical results on any number of processors since there are no global reduction operations in the algorithm. That is, all floating point operations are performed in the same order as they would be on a serial implementation. This should be verified, and any exceptions explained, by prototyping.

6 Design and Development

6.1 Performance Issues for GEOS-3

The following discussion applies to all three components of the core GEOS DAS system: GCM, QC, PSAS (excluding GPIOS). A more detailed discussion on the

GCM timings, including the impact of model resolution can be found in Takacs 1997.

Already, prototype PSAS has achieved 11.0 gigaflop/s for realistic problems on 512 processors of the NASA Goddard Cray T3D. Parallel GCM has achieved 3.7 gigaflop/s (dynamical core) and 8.0 gigaflop/s (compressed short wave radiation package) on 512 processors of the T3D; other model modules (turbulence, long wave, and land surface) are expected to be lie somewhere in this range of performance. The increased performance that will be required for Shapiro filter and land surface module are exceptions. The filter achieves only 0.31 gigaflop/s at present and it occupies a significant amount of CPU time. The tiled land surface model achieves only 2.5 gigaflop/s on 512 processors of the T3E, and the version which is expected to be implemented in GEOS-3 will use a significant amount of CPU time. Increased performance to reach the GEOS-3 Requirements will be gained through: improved single-processor performance (cache optimization *etc.*); higher processor speed of the SGI Origin, and the large number of processors (approximately 1000 pe's T3E) that will be made available for the HPCC ESS project; improved communication strategies; and improved load-balancing strategies if possible. Communications may be optimized by both reducing the net volume where possible and concatenating messages in order to amortize latency. Single-processor performance is particularly important, and may involve significant changes to code since the previous versions of GEOS GCM were optimized for vector computers.

The relationship between flop rates for the codes and the wall-clock-time (which is actually more important for production) is obtained by weighting the flop rates according to the proportion of time taken be each module. The generic requirement is that 30 days of earth data be assimilated in one wall-clock day. Since there are four analysis cycles (6 hour forecast and analysis) per day, this means that we must perform an analysis cycle every 12 minutes. At 18.3 gigaflop/s (Paragon) prototype PSAS takes 156.5 seconds. The QC usually takes about 10 percent of the analysis, so this would add an additional 15.6 seconds. The model time-step is 3 minutes, while the physics modules are advanced on longer time-scales (9 minutes for moist processes, 30 minutes for turbulence, and 3 hours for longwave and shortwave calculations). At present, GEOS-2 GCM takes about 40 percent of the time of the analysis (GCM, PSAS, and QC) so, in order to achieve at least 30 days per day turnaround, each time-step must take no more than 1.9 seconds (on average). Timings for the present vectorized, multitasked version of the GCM indicate that if we can achieve a weighted average 10 gigaflop/s then the 6 hour forecast will take about 106 seconds. This time plus the 172 seconds (at 18.3 gigaflop/s) for the analysis (PSAS and QC) fall well within the requirement of 12 minutes per analysis cycle. It is clear that the production performance requirements of GEOS-3 (30 days assimilated per day) are less stringent than that of HPCC ESS (50 gigaflop/s in 1997 and 100 gigaflop/s in 1998 on modified systems); this is appropriate since, for MTPE, the DAO will be using less-expensive medium-range parallel computers such as the SGI Origin.

6.2 Software Approach

Following extensive discussions with the group led by Suarez, a framework has been adopted for the modularization of GEOS-3 using Fortran 90 (Staff 1997a). This is called the Goddard Earth Modeling System (GEMS). Within this framework, the GCM, QC, and PSAS are high-level modules that interface via a small number of well-defined data types. These become arguments for subroutine calls in the driver

application. These include input and output couplers. The states of the models are also included in the calling arguments, because components of the states may be needed by other models, and in order to facilitate an approach where multiple simultaneous states can be maintained in a single run. Couplers are modified by a HERMES library. Functions in HERMES modules are used for the cases where grid transformations are required between the high-level modules. Is is expected that a hierarchy of HERMES libraries will be built to accommodate the coupler/state interface approach of GEMS at a number of levels of the DAS. However, this is expected to be done incrementally. In particular, the first implementation of the parallel GCM will have its internal components (dynamical core, moist processes, radiation, land-surface) closed. This means that the GCM interfaces with the DAS in the strict GEMS framework, but the internal interfaces will not be significantly changed from GEOS-2. Note that considerable modularity already exists in theses internal interfaces, and that will be maintained and incrementally enhanced in an Fortran 90 framework.

6.3 Design and Prototyping

The following components will go into the GEOS-3 GCM. The following are steps to be taking in approximate chronological order: we anticipate adding points incrementally into each subsequent release (Jun. 1, Sep. 1, Dec. 1). The components are intended to go into the first release, unless explicitly stated otherwise.

- The present GCM will be converted to a minimal Fortran 90 configuration, that will use user-defined types to enable a codification of data and dataflow between modules. There will be no common blocks. `allocate/deallocate` functions will be used to manage memory.

  It should be noted that the use of Fortran 90 pointers may lead to reduced single-processor performance. For example, pointers will have to be dereferenced before time consuming `do` loops (or even outside of subroutines) to indicate to the compiler that there is no indirection. This and other aspects of Fortran 90 optimization are highly compiler dependent, and will have to be prototyped carefully.

- The parallel dynamical core of Schaffer/Suarez will be prototyped for use by the DAO. At first, a partially optimized second order DYCORE that was obtained from Jim Abeles will be used. As a second step, forth order DYCORE will be integrated into the Fortran 90 code described above.

  Because of the critical performance needs of the DAO, DYCORE may have to be opened up for later optimization (i.e., rather than just black boxed). In the 1995 HPCC ESS review of the parallel dynamical core, it was noted that there appeared to be a lot of unnecessary memory copies in the algorithm. This can be a serious problem, especially on RISC processors, and the source or explanation of the problem should be sought (see next section).

- The physics modules from GEOS-2 GCM will be prototyped for single-node performance (especially for cache optimization) on at least the Origin and Cray T3E machines. There is some experience among Max Suarez, Dan Schaffer, and Jim Abeles. This will be leveraged, but all physics modules will have

to be modified. In general this is not difficult since there are no horizontal dependencies.

The physics modules will be rewritten and any explicit or implicit references to absolute geographical coordinates in the physics domain will be removed. That is, the physics modules will be rewritten to work on a rectangular lat-long sub-domain of the entire earth. Clearly, this sub-domain will correspond to one processor's portion of the decomposed problem.

The physics modules of GEOS-2 GCM are multitasked for Cray C90/J90 operation using `strip/paste` subroutines that extract contiguous sets of horizontal grid-points and pass them to a process. This facility could be maintained for the parallel GCM because strips may be a way of generating horizontal data locality (this is similar to "blocking" in standard cache-optimization parlance). The cost of this is the extra memory copies involved in stripping and pasting. The only other way to generate data locality would be to reverse the current order of indexing of three-dimensional loops from `array(i,j,lev)` to `array(lev,i,j)`.

- A version of the parallel I/O (GPIOS) of Lucchesi (1997) will be used.

- Shapiro filter needs to be optimized in later releases. At present it is not understood why the performance is so bad. This needs to be cleared up. It may be useful to write out the complete template for the four-times iterated filter and implement it with one, optimized, message-passing call.

- The problem of load imbalance in the moisture, radiation, turbulence, and land surface models needs to be addressed in later releases

- In later releases, the following algorithms (that mostly belong to the HERMES class) need to be implemented and optimized because they represent serious parallel bottlenecks: Shapiro filter; rotate (forward and back); and to a lesser extent a-to-c and c-to-a. Also, an algorithm for transforming parallel decomposition of grids between GCM and PSAS needs to be developed and optimized.

- For the first release of the code, all calculations will be performed in a non-rotated frame. In later releases, the rotation will make use of the HERMES library currently under development. The transformation operator (or matrix) will be defined once, in an efficient form, as soon as the two grids have been determined. The application of the transformation is optimized to maximize data locality, and should be relatively efficient. Due to its inherent communication, however, it will not scale perfectly (see section 5).

  Because the rotate (forward and back) algorithms are so important, and because they are invoked several times every timestep, the possibility of running both dynamics and physics modules in the rotated frame should be considered in later releases.

- Memory allocation and deallocation procedures need to be systematized to conserve memory in later releases. The current GEOS-2 GCM has a high water memory mark of 130 megabytes. Parallel PSAS will need in excess of 10 gigabytes. Later modifications may be needed if fragmentation is found to be a problem.

- In the long term, we will migrate to a GEMS structure, although the existence and integration of existing software constrains its full realization. Therefore, only a GEMS-like approach will be adopted in the first releases (Staff 1997a).

- Current parallel algorithms will be documented.
- A user's manual will be written, including test data and installation instructions.

6.4 Optimization of DYCORE

Because DYCORE is so important for the June test phase, the following section discusses issues of its optimization in greater detail. This does not address the optimization of other important modules such as: Shapiro filter, load balance and single processor optimization of the physics modules, polar rotation, a-to-c and c-to-a transformations, and the high-latitude filter in DYCORE.

One easily available version of parallel DYCORE may be obtained from the HPCC ESS benchmark suite. Max Suarez submitted his code to this suite, and the version may be obtained from anonymous ftp at `farside.gsfc.nasa.gov` in the directory `/pub/HPCC/ESS/testcases`

Shown at the end of this section is the subroutine FRM_WST that was obtained from the directory dycore/src from Suarez's benchmark code. This shows that the finite-difference equations are implemented by using message-passing to emulate a vector CSHIFT. The input array is AI, which is actually a rectangular group of gridpoints corresponding to a checker-board domain of the parallel decomposition. The output array is AO, which is the equivalent to the input array shifted one gridpoint to the right; hence the name "FRM_WST" (i.e., from the west). There are similar subroutines for the other 3 horizontal directions. There is no parallel decomposition in the vertical. By looking at the subroutine one can see that the interior gridpoints of AI are simply shifted into AO. The boundary values are message-passed (see subroutine PUTTMN) from the corresponding logical processor that owns the domain to the west. This is equivalent to the traditional "ghosting" in that only necessary boundary strips are passed between processors.

The following items summarize some of the performance issues. The main principal is that the cost of message-latency can be amortized by concatenating small messages. Once messages are large enough (usually more than 1 kilobyte) the interprocessor bandwidth is achieved. After that there is only modest utility in concatenating messages. Also, some effort should be made to avoid unnecessarily sending redundant information (if that is being done.)

- **1** The subroutines sends messages (PUTTMN) separately for each level. In this case, vertical "walls" of gridpoints can be concatenated (buffered) and sent as one larger message. This should not be difficult to implement. This may have already been done in Jim Abeles' partially optimized DYCORE, and should be checked.
- **2** The finite-difference algorithm will be fourth order so the actually two "walls" of gridpoints should be sent when necessary. This will also help because it generates larger messages (i.e., as opposed to calling FRM_WST twice). This also shouldn't be too difficult to implement.
- **3** It may be more efficient to do pure ghosting. In this case the domains are simply overdimensioned and messages are sent which fill the appropriate ghost

cells. This may eliminate unnecessary memory and memory copies. Finite differencing would be done by standard do loops with indexing into single arrays (i.e., not using data-parallel shifts). It is unclear how this affects cache efficiency (i.e., single processor performance).

- **4** The last improvement is difficult to implement because it requires careful knowledge of the timestepping sequence in order to avoid damaging the algorithm. At present we cannot name an example, but can only describe the problem in general. The prognostic variables are wind, potential temperature, surface pressure and moisture (sometimes fluxes are used and a scalar velocity field). The point is that by careful investigation of the update sequence of each variable within one timestep it may be found that different variables can be concatenated into single messages; in some instances earlier than necessary. This generates larger messages, with the efficiency that has been described above, but it may obfuscate the simple sequence of events that is performed for a "regular" serial code. The person who does this needs to proceed carefully in order not to break the algorithm. break the algorithm; it should be done in a clean modular manner.

The following subroutine was obtained from the HPCC ESS benchmark code of Max Suarez (utils.f).

```
      SUBROUTINE FRM_WST(IM,JM,AO,AI)
      IMPLICIT NONE
      INTEGER IM,JM,SND_N_RCV
      REAL    AO(IM,JM),AI(IM,JM)
      INTEGER I,J

c/*  CSHIFT is a FORTRAN 90 intrinsic   */
c      AO = CSHIFT(AI,-1,1)

      INTEGER NX,NX0,NY,NY0,NYNTH,NYSTH,NXEST,NXWST,NPE,NGRP
      COMMON/COMMVARS/ NX,NX0,NY,NY0,NYNTH,NYSTH,NXEST,NXWST,NPE,NGRP

      DO J=1,JM
       DO I=2,IM
        AO(I,J) = AI(I-1,J)
       ENDDO
      ENDDO

      CALL PUTTMN(AO(1,1),IM,1,JM, AI(IM,1),IM,1,JM, NXEST,NY0)

      RETURN
      END
```

7 Detailed Design for June Release

As mentioned in previous sections, the June release will consist of a minimal system without analysis (i.e., analysis increments = 0), with all calculations performed in a non-rotated frame.

The driver will only call the GCM and GPIOS functionality, but will be designed to include the Analysis capability later on.

The following modules will be written for this release:

`dynamics_state_module`	Create/Destroy dyn. state
`physics_state_module`	Create/Destroy phys. state
`land_state_module`	Create/Destroy land state
`tendency_module`	Create/Destroy tendencies
`coupling_module`	Create/Destroy couplings
`GridTypes`	Grid and decomposition types
`GridModule`	Initialize/Destroy Grids
`ParUtilitiesModule`	Fundamental comm. operations
`ParRestartIOModule`	Parall Restart I/O routines
`GpiosModule`	Parallel I/O (GPIOS)
`DycoreModule`	Parallel DYCORE with cover
`PhysicsModule`	Revised Physics legacy codes
`DriverModule`	Driver, GCMInit, etc.

The detailed design of the individual modules, and the implementation as it becomes available, can be found in individual directories (with the same name) in the DAO Intranet under:

`http://dao.gsfc.nasa.gov/Intranet/GEOS3/Software/Core/GCM/`

Within each module the types and the interfaces of all constituent routines are specified in sufficient detail to make their implementation straightforward.

8 Portability, Reusability, and Third-Party Software

There may be some portability problems where Fortran 90 types are used as arguments for MPI message-passing functions (Hennecke 1996). This should be prototyped, even to the extent of generating a set of diagnostic functions to run on target parallel platforms libraries and compilers.

Acknowledgments

Staff would like to acknowledge useful discussions with the following, who attended January's workshop: Jim Abeles, Jay Larson, Sharon Nebuda, Carlos Pabon-Ortiz, Dan Schaffer, Max Suarez, Spencer Swift, Ricardo Todling, and Jose Zero.

References

Farrell, W. E., A. J. Busalacchi, A. Davis, W. P. Dannevik, G-R. Hoffmann, M. Kafatos, R. W. Moore, J. Sloan, T. Sterling, 1996: *Report of the Data Assimilation Office Computer Advisory Panel to the Laboratory for Atmospheres.*

Hennecke, M., 1996: A Fortran 90 interface to MPI version 1.1. RZ Universität Karlsruhe, Internal Report 63/96.
`http://ww.uni-karlsruhe.de/~Michael.Hennecke/`

Koster, R. D., M. J. Suarez, 1992: Modeling the Land Surface Boundary in Climate Models as a Composite of Independent Vegetation Stands. *J. Geophys. Res.*, **97**, D3, 2697-2715.

Lamich, D., and A. da Silva, 1996: Data and Architectural Design for the GEOS-2.1 Data Assimilation System Document Version 1. *DAO Office Note 96-XXX.* Data Assimilation Office, Goddard Space Flight Center, Greenbelt, MD 20771.

Lucchesi, R., 1997: Requirements and Preliminary Design of the GEOS-3 Parallel Input/Output (GPIOS) Subsystem.

Lyster, P. M., and Co-I's, 1995: Four Dimensional Data Assimilation of the Atmosphere. A proposal to NASA Cooperative Agreement for High Performance Computing and Communications Program Earth and Space Sciences (HPCC ESS) project.

:Lyster, P. M., J. W. Larson, C. H. Q. Ding, J. Guo, W. Sawyer, A. da Silva, I. Štajner, 1996: Requirements and Preliminary Design of the GEOS-3 Physical Space Statistical Analysis System (PSAS) Subsystem.

Pfaendtner, J., S. Bloom, D. Lamich, M. Seablom, M. Sienkiewicz, J Stobie, A. da Silva, 1995: Documentation of the Goddard Earth Observing System (GEOS) Data Assimilation System – Version 1. *NASA Tech. Memo. No. 104606*, Vol. 4, Goddard Space Flight Center, Greenbelt, MD 20771.
`ftp://dao.gsfc.nasa.gov/pub/tech_memos/volume_4.ps.Z`

William, W., A. da Silva, R. Lucchesi, P. Lyster, L. Takacs, 1997: GEOS-3 Data Assimilation System: Core GOES-3/DAS Data and Interface Design Document.

da Silva, A., C. Redder, 1995: Documentation of the GEOS/DAS Observation Data Stream (ODS) Version 1.01, *DAO Office Note 96-01.* Data Assimilation Office, Goddard Space Flight Center, Greenbelt, MD 20771.

da Silva, A., J. Guo, 1996: Documentation of the Physical-space Statistical Analysis System (PSAS). Part I: The Conjugate Gradient Solver, Version PSAS-1.00. *DAO Office Note 96-02.* Data Assimilation Office, Goddard Space Flight Center, Greenbelt, MD 20771.

da Silva, A., J. Pfaendtner, J. Guo, M. Sienkiewicz, and S. Cohn, 1995: Assessing the Effects of Data Selection with DAO's Physical-space Statistical Analysis System. *Proceedings of the second international symposium on the assimilation of observations in meteorology and oceanography*, Tokyo Japan, World Meteorological Organization.

Staff 1996: Data Assimilation Office Algorithm Theoretical Basis Document.

Staff, 1997a: GEOS-3 Data Assimilation System, System Design.

Stobie, J. 1996: GEOS-3 System Requirements.

Takacs, L. L., A. Molod, and T. Wang, 1994: Documentation of the Goddard Earth Observing System (GEOS) General Circulation Model – Version 1. *NASA Technical Memorandum 104606,* Volume **1**, NASA Goddard Space Flight Center, Greenbelt, MD 20771.
`ftp://dao.gsfc.nasa.gov/subpages/tech-reports.html`.

Takacs, L. L., 1997: Impact of Resolution on GEOS GCM Model Development.

Zero, J., R. Lucchesi, R. Rood, 1996: Data Assimilation Office (DAO) Strategy Statement: Evolution Towards the 1998 Computing Environment.