DAO Office Note 1998-009

Office Note Series on
Global Modeling and Data Assimilation

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

GEOS-2.x (multitasked) to GEOS-3.x (MPI) GCM Migration Plan

William Sawyer, Lawrence Takacs, Robert Lucchesi, Ricardo Todling

Data Assimilation Office, Goddard Laboratory for Atmospheres

Goddard Space Flight Center
Greenbelt, Maryland 20771
April 1998

Abstract

This document suggest a step-by-step migration path to the GEOS 3.x (MPI) GCM from the for the GEOS 2.x (multitasked) GCM version vc5.9, once the latter has been frozen and baselined for the GEOS 3.x development. This migration plan address a wide range of issues from managerial to technical. Specific design issues will be dealt with in later documents.

Contents

1 Background

In Jan. 1997 a workshop was organized by Peter Lyster to discuss the message-passing parallelization strategy for the GEOS DAS General Circulation Model. The parallel Goddard Earth Modeling System (GEMS) being developed by Max Suarez and Dan Schaffer was presented. The DAO modeling group, represented by Lawrence Takacs, presented the status of GEOS-2.x GCM at that time. The design issues of the parallel version were discussed. The options open at the end of the workshop were the following:

- Adopt the Goddard Earth Modeling System (GEMS) framework and most or all of the model (*dycore* and, at a later time, physics modules) being developed by Suarez and Schaffer.
- The DAO modeling group would write their own MPI model, including a parallel *dycore*.
- The long-term path would be decided at a later time. In the short term, Suarez/Schaffer would provide a F77 "cover" for their latest, GEMS-compliant parallel *dycore*. An interim parallelization of the existing F77 GCM would be performed, calling the *dycore* cover. This "interim" parallel version would provide the HPCC benchmarks required in Dec. 1997.

The interim approach was actually agreed upon but then not pursued. Another path evolved: Lawrence Takacs upgraded the GEOS-2.x GCM (vc5.9) with F90 derived data structures (released July 1997). Adopting the concepts of that new work, Lyster and Sawyer prepared in May 1997 a document on the requirements and design of the parallel GEOS-3 GCM [1]. This document describes the results of the workshop and makes an attempt at the design of the MPI GCM. The use of GEMS framework was discussed at length but finally dropped, and Suarez/Schaffer provided an older (non-GEMS) F77 parallel *dycore* for integration into our model. Subsequent implementation work was performed on the basis of [1], resulting in a MPI GCM (prototype) completed in Dec. 1997. This prototype was used extensively within the framework of the HPCC project in order to obtain benchmarks.

The divergence of the MPI GCM prototype with the actual production code (which was not upgraded with F90 derived data structures and contains new scientific additions) resulted in renewed discussions in Nov.-Dec. 1997. A new decision was made: when the AM-1 system is frozen, the GCM would be MPI-parallelized again based on that system. With the AM-1 GCM code nearly stable in mid-March 1998, the authors discussed the migration path again and reach agreement on the way to proceed. The subsequent document discusses the current plan to obtain an fully operational, parallel, scientifically valid GCM written in F90 and using the Message-Passing Interface (MPI) for communication.

2 Requirements

The key to the success of this project is the validation of the MPI production GCM. The frozen AM-1 GEOS-2.x (multitasked) GCM running on the production platform (SGI Origins) in sequential mode will be considered as a reference version.

- The frozen AM-1 code forms the foundation for the MPI code and extensions or alterations will not be incrementally patched into the MPI development.

- MPI development will take place on the SGI Origins where portions of the code can be continually compared with the actually production code. On this platform stable F90 and MPI implementations are assumed, as well as uninhibited access to at least 16 development processors.
- Unit tests need to be made available by the modeling group which, at the very least, allow scientifically and computationally homogeneous sections of the sequential GCM (e.g., the long-wave radiation, the rotation, etc.) to be run and tested with realistic data. The MPI developers will then extend these unit tests to distribute the test problem to all PEs, calculate the problem in parallel, gather the data on 1 PE and compare it to the sequential results.
- The MPI GCM needs to give either zero differences or explicable round-off differences when compared with the sequential production code. Note that we anticipate round-off differences in many portions of the code. The rationale behind our expectations can be found in [2].
- The MPI GCM needs to have sufficient performance to ensure that the Core System can assimilate 30 days per day at $2^o \times 2.5^o \times 70$ resolution on a 64 PE SGI Origin2000 system. While the actual requirement for the GCM (including its I/O!) is nebulous, we believe that the performance needs to be approximate 60 days of assimilation per day.
- All software adheres to the DAO F90 Software Standards [3]. Particularly applicable here is that the code must be portable to all DAO production platforms.

3 Proposed Work Plan

The following suggested work plan supersedes [4] and reflects all those tasks in that document which have already been performed.

3.1 Preliminary Modeling Group Tasks

The first set of tasks are for the modeling group. The following work may or may not be part of the AM-1 system, however it is necessary to provide a basis for the MPI system.

1. The AM-1 GEOS-2.x GCM code needs to be officially frozen and checked into a repository. In particular, the latest versions of the grid transformations need to be frozen.
2. The code should be restructured as described in Appendix A.1. This will allow MPI developers to concentrate on the routines which are communication dependent.
3. The F90 framework of the prototype needs to be reproduced for the production code. Zero differences with the existing production code (with common blocks) compiled with the F90 compiler are expected. In this step, the current F77 non-compliant features (extensions) should be rewritten in F90 compliant constructs, in particular, the "POINTER" feature (which is neither F77 nor F90 compliant) should be removed in favor of REAL, SAVE, ALLOCATABLE (F90) arrays, and array over-indexing

removed (at least for the "vanilla" baseline), and AMAX1, AMIN1, removed in favor of MAX and MIN.

During the above changes it is an opportune time to add the arguments with contain the MPI parallelization information (e.g., decomposition and lattice data) to high level subroutines specified in Appendix A.2.

4. Unit tests for the key components of the code need to be provided by the modeling group. The timetable can flexible to correspond to the MPI development, i.e., only unit tests pertaining to modules under MPI development (section 3.2) need to be implemented at once.

3.2 MPI Group Migration Tasks

After the completion of the tasks in section 3.1 the MPI developers will proceed incrementally from this version, using zero-difference or round-off difference checks at every stage to verify the correctness of their changes.

1. The interface and procedural design document for the GEOS-3.X GCM will be drafted using this document and [1]. This will include all subroutine interfaces which have changed due to the MPI migration, and a description of the underlying algorithms. It will be based largely on proTEXdocumentation.
2. A test plan will be drafted for the subroutines which have to be altered for MPI migration.
3. The existing sequential *dycore* module will be parallelized with MPI. Although a parallel MPI version of *dycore* exists in the prototype (an old version from Max Suarez's forecast model), it was deemed unreliable and outdated by the modeling group. Considering the effort needed to add fourth order advection, improve performance and, in particular, validate the prototype *dycore*, it is more expedient to re-parallelize the existing sequential version with incremental changes and continual checks for zero differences. Unit testing.
4. Existing I/O facilities in the MPI prototype will replace sequential I/O in the production code. Unit testing.
5. GCM_INIT will be replaced by MPI prototype GCM_INIT, using currently existing F90 MPI facilities. This will include facilities to distribute the entire lat-lon grid into blocks on each PE. Unit testing.
6. Changes will be made to all I/O routines READPHNX, PHNXWRT, GETO3, GET_H2O, GETBCS, GETSST, GETSICE, GETPHIS, and MKWRLD to support parallel read and distribute/broadcast. Modifications are necessary to the routines which call these, e.g., GAEA, POSEIDON, ATMOS. Unit testing.
7. The model history will be replaced by the parallel GPIOS history developed for the MPI prototype.
8. The final scientific changes to the transformations ATOC, CTOA, ROTATE_F and ROTATE_B will be integrated into the corresponding prototype modules (which are based on pre-v6.6 versions). These components will be unit tested for round-off differences with the sequential versions. Note that it is possible that the parallel version might not satisfy zero difference checks (see [2]). Round-off differences and their effects will to be discussed with the modeling group and subject to the latter's approval.

9. The parallel transforms ATOC, CTOA, POLEWND, SHAP, ROTATE_F ROTATE_B will be unit tested against the sequential versions.

10. The mini-driver DYNDRV needs to be amended to call the parallel *dycore*. Unit testing.

11. Although conceptually no changes to the Physics are required, LWRIO, SWRIO, MOISTIO and TURBIO should be turned on one-by-one and tested for zero difference using corresponding unit tests.

12. Software integration. At this point all the components of the GCM should be running in a fully parallel manner, and should have at most round-off differences with the sequential baseline. Integration, however always brings with it unusual and unexpected interactions, and therefore we add it as a separate task and assign it not a small estimated effort (see section 5). Integration testing.

4 Document Trail

The following documents are planned for this migration effort. The approximate date of completion is also listed. Unless otherwise stated, all documents will be available in HTML format on the DAO Intranet underneath `http://dao/Intranet/GEOS3/Software/Core/`.

- *GEOS-2.x (multitasked) to GEOS-3.x (MPI) GCM Migration Plan: Requirements and Architectural Design.*

  This document, in `http://dao/Intranet/GEOS3/Software/Core/GCM/Design/migration`.
- *Data and Procedural Design for the Message-Passing GEOS 3.x GCM*
- *GEOS 3.x GCM On-line Source Code Documentation.*

5 Approximate Timeline

The tasks specified in section 3.1 are scheduled for completion at the end of June, 1998.

The tasks specified in section 3.2 are largely independent of one another, and have the following estimated efforts.

Task	Task No.	Effort (man-months)
Proc./Interface design and test plan	1,2	1.0
dycore parallelization	3	2.0
I/O facilities to MPI	4,5,6	1.0
Model history to MPI	7	2.0
Rewrite transformations	8,9	1.0
Dynamics Driver	10	0.5
Physics testing	11	0.5
Software integration	12	1.0

Approximately 1.2 people (one person at 70% and another at 50%) are assigned to the the MPI GCM parallelization, we judge the overall effort to be roughly 7 months. Some of this work, e.g., the procedural and interface design and test plan can be started as soon as the AM-1 code is frozen. Thus a full-up scientifically valid system can be anticipated at the end of January, 1999, provided none of the risk items mentioned in section 6 come into play, and *assuming continued consulting support from the modeling group.*

6 Risks

The following is a list of risks associated with the MPI GCM development. The risks and their possible manifestations are explained as well as possible recourse.

6.1 Sequential GCM Code Not Frozen As Scheduled

Explanation: The tasks in section 3.1 are not completed and frozen as scheduled for the end of April.

Effect: The schedule for completion of the MPI GCM will be pushed back correspondingly.

Possible Recourse: Addition modeling personnel assigned to the tasks in section 3.1.

6.2 Insufficient Support from Modeling Group

Explanation: The modeling group does not have sufficient time or capacity to provide a consulting function for the GEOS-3.x developers.

Effect: The schedule for completion of the MPI GCM will be pushed back correspondingly.

Possible Recourse: Addition modeling personnel committed to support the MPI development effort.

6.3 MPI Development Does Not Proceed As Planned

Explanation: The tasks in section 3.2 cannot be completed in the time-frame proposed in section 5.

Effect: The schedule for completion of the MPI GCM will be pushed back correspondingly.

Possible Recourse: Addition MPI personnel assigned to the tasks in section 3.2.

6.4 System Software Problems on SGI Origin

Explanation: Bugs in the SGI Origin system software, e.g., in the MPI implementation or in the F90 compiler, are encountered which cannot be avoided by simple work-arounds.

Effect: Possible delays.

Possible Recourse: SGI or NAS personnel assigned to assist MPI developers overcome the problems.

6.5 GCM Does Not Attain Required Performance

Explanation: The GCM does not attain sufficient performance to support a 30 assimilated days per day end-to-end core system performance.

Effect: Possible delays. Additional costs associated with optimization.

Possible Recourse: SGI, NAS or third-party personnel assigned to assist MPI developers optimize the code. Depending on the areas isolated as bottlenecks, such assistance may be in cache optimization, in optimization of the MPI libraries, or in replacement of some key MPI calls with faster communication paradigms (e.g., SHMEM).

6.6 Physics Unit Testing Does Not Proceed as Scheduled

Explanation: The anticipated zero differences for the long- and short-wave radiation, turbulence and moist processes are not obtained when they are unit tested in parallel mode.

Effect: Possible delays.

Possible Recourse: A member of the modeling group might assist MPI developers to clarify the reason for non-zero differences.

Acknowledgments

We would like to thank the participants of the walkthrough of this design document on YY/MM/DD, for their patience and insightful comments/corrections.

References

[1] Peter Lyster, William Sawyer, and Lawrence Takacs. Requirements and Design of the Parallel GEOS-3 General Circulation Model Subsystem. DAO Office Note 97-xx, Data Assimilation Office, NASA, Code 910.3 Greenbelt MD, 20771, USA, 1997. Not yet approved by the Configuration Control Board.

[2] William Sawyer. Reproducibility Issues: Sequential vs. MPI GCM. `http://dao/Intranet/GEOS3/Software/Core/GCM/Design/reprod.html`, 1997.

[3] Staff. DAO Fortran 90 Software Standards. DAO Office Note 97-xx, Data Assimilation Office, NASA, Code 910.3 Greenbelt MD, 20771, USA, 1997. Not yet approved by the Configuration Control Board.

[4] William Sawyer. Recommended Path from GEOS 2.X GCM (multitasked) to GEOS 3.X GCM (MPI). `http://dao/Intranet/GEOS3/Software/Core/GCM/Design/Migration2Xto3X/`, 1997.

A APPENDIX

A.1 Core System Source Code Structure

A restructuring of the GCM source code is proposed in order to facilitate development. The aims of the restructuring are:

- Directories are used to isolate a conceptually similar bulk of software, which we refer to as a module (not necessarily in the Fortran 90 sense).
- Developers only need to alter the smallest amount of code. In general, source code in large files with many subroutines should be avoided, because any change will be reflected as a change to the entire file. This can be unnerving to other developers who do not immediately see the change and know its extent. For the MPI implementation, large changes need to be made only to a small number of subroutines, therefore the granularity of the source files should be as fine as possible.
- Cyclic dependencies between modules and libraries are avoided. These tend to make compilation, library archiving, and linking more difficult, and create havoc in dependency analysis.
- A clear connection between a module or library and its principle responsible individual is created.

The GEOS-2.x source structure has been followed as closely as possible. In GEOS-2.x at the core source level, the `history` software has been given an independent directory. This recent step reflects that the history is to be used by other components of GEOS DAS, e.g., the analysis, i.e., both the model and the analysis depend on the history.

In GEOS-3.x circular dependencies are avoided if possible, i.e.,, situations where two libraries mutually depend on each other. In order to achieve this, a larger number of directories at the core source level are proposed, each containing software used in multiple modules.

`aplug`	Analysis Plug (as in GEOS-2.x)
`appl`	Applications (as in GEOS-2.x)
`psas`	Analysis, (see PSAS migration plan)
`gcm`	Model, (see subsequent discussion)
`history`	History module (as in GEOS-2.x)
`gcmutil`	GCM-related utilities used by non-GCM modules
`anautil`	Analysis-related utilities used by non-Analysis modules
`lsmutil`	Land-surface-related utilities used by non-lsm modules
`util`	General utilities used by several modules
`hermes`	Data transformations used by several modules
`dycore`	Dynamical core

It is possible that the PSAS Migration Plan might propose other directories at this level. The new directories `gcmutil`, `anautil`, `lsmutil`, and `hermes` all contains software that is used by more than one other module. The `dycore` directory contains software which is only used by the model, but, due to its large size, importance, and the ease to isolate it, its addition at this level would be convenient.

The GEOS-2.x `gcm` directory contains 22 source files, each containing as many as 45 subroutines. To achieve the above-mentioned aims with only a conservative step, the files should be split (with the so-called `fsplit` utility) into several files, one for each subroutine. All these split files should be placed in a new directory with the same base-name as the original file, e.g., `turb` for `turb.f`, etc.. Each of these directories will get a separate Makefile which is called hierarchically from the GCM Makefile.

The creation of the following directories is proposed to go under the directory `gcm`:

gcm_init	GCM-initialization
atmos_model	Atmospheric model
chemistry_model	Chemistry model
interp_chemistry	Chemistry interpolation
earth_model	Earth model
ocean_model	Ocean model
ffsldrv	
time	GCM time facilities
lwrad	Long-wave radiation (physics)
swrad	Short-wave radiation (physics)
moist	Moist processes (physics)
turb	Turbulence (physics)
gwdrag	Gravity-wave drag
tpcore6m	Lin-Rood advection
diagdrv	Diagnostic driver
dyndrv	Dynamics driver and Shapiro Filter
physdrv	Physics driver

A.2 Migration plans for Stretched Grid software

Although not part of the AM-1 code, we do intend to integrate the recently finished stretched grid software of GCM `vc6.6` into the baseline *before* MPI development. We note that this code has been exhaustively tested and its use in the MPI code has been recommended by Ricky Rood.

First a list of the changes which apply to some of all of the modules is given, then the changes to individual subroutines are listed in independent sections.

- As suggested in section A.1, the stretched grid code currently in one file (`grids.f`) will be split into files each containing one subroutine. These files will carry the name of the subroutine with the suffix `.F`. They will be put into the subdirectory `grids`, to which also a makefile will be added.

- Prologues will be added to all routines as stipulated in [3].
- CPP directives in the code will be cleaned up to ensure acceptability by all CPP preprocessors.
- Array over-indexing and all other F90 non-compliant extensions (e.g., `mallocx`) will be removed in favor of F90 compliant constructs.
- Indentation will be added (if possible, such purely cosmetic cleaning of the code will be performed with `f90lint`.

At this point the code should give identical results to the original code. In the following sections, MPI changes to the individual routines will be outlined.

A.3 Changes to Routines `rotate_f` and `rotate_b`

The following changes apply to both `rotate_f` and `rotate_b`. Since these routines will be significantly altered, it will not be possible to test them continually against the sequential version. From the GCM prototype, however, we feel there will not be a great problem in taking this large step.

These routines will be called on all PEs with replicated data. The actually assembly will occur using library routines from `SparseModule` — thus avoiding any real parallel programming.

1. Each of these routines will be modified to create the transformation matrix only (routines are available to apply the transformation in parallel).
2. An argument `Matrix` of type `MatrixType` (imported from `SparseModule` will be added to the argument list. After discussions with Larry Takacs to find an alternative, `check` will be removed. The sections which assume `check .eq. .TRUE.` will be removed and the other sections will be made the default.
3. 2-D arrays of integers (e.g., `ip1`, `jp1` and will become scalars with an appropriate name. 2-D real arrays will be removed entirely (these data will be entered into `Matrix`.
4. The above-mentioned scalars will replace the corresponding arrays at the appropriate code locations, e.g., `im1(i,j)` becomes `im1_i_j`.
5. The weights will be added directly to the sparse matrix (`Matrix`) using the utility `SparseInsertEntries`.

A.4 Changes to Routines `ctoa` and `atoc`

These routines are so similar to the successfully parallelized routines that we propose that the new initializations (which are `dlam` and `dphi` vectors be placed directly into the prototype versions. The lines in question are those associated with `dxg` and `dyg`, and those which define `ap2, ap1, ap0, am1, bp2, bp1, bp0,` and `bm1`. The latter arrays are defined by the local `im` and `jm` so care must be taken to use the correct offsets in the global, replicated arrays `dxg` and `dyg`.

The arrays `dlam` and `dphi` are replicated on every PE (as is `itype`). The variables `im` and `jm` can naturally have different values on different PEs, and arrays `qa` and `qc` are local to the PE and are dimensioned as dictated by `im` and `jm`.