DAO's suggestions to the software design of CAM

S.-J. Lin, S. Nebuda, B.-W. Shen, J.-D. Chern, W. Sawyer, and A. DaSilva

March 16, 2001

Data Assimilation Office, NASA Goddard Space Flight Center

1 Code structure & Array layout

1.1 Modular physical packages

All physics packages should be modularized and put in separate directories. For example, under the main directory "physics" there could be sub-directories specifically for moist convection, PBL parameterization, Gravity wave Drag, cloud parameterization(s), and long- and short-wave radiations. This will enable easeir swapping of individual physics package.

1.2 Array layout

For efficiency, there should be separate arrays for advected and non-advected tracers. The non-advected tracers should be declared in a way that is most efficient for physics/chemistry computation. But the advected tracers should be dimensioned in the way that is most efficient for the particular dynamics. For example, for the finite-volume dynamical core (*fvcore* hereafter) it should be as follows:

```
real (r8) q_dyna(plon,beglat:endlat,plev,pcnst)     ! advected tracers
real (r8) q_phys(plon,plev,beglat:endlat,pnats)     ! non-advected tracers
```

Currently a transpose of the array **q3** to **q_dyna** is required inside **dynpkg,** which not only slows the model down but also requires extra memory, particularly when there are many tracers.

It is also highly desirable to have separate restart files for dynamics and physics (see also section 32.). For example, the non-advected tracers (including the prognostic cloud water) should be put into the physical restart file(s). Once the separation is done we actually don't need to have the so-called "initial dataset" for the model. For fvcore, there shouldn't be any difference between an "initial run" and a "restart run". We can eliminate the code segment in routine "stepon" for the *fvcore* initialization run (**nstep=0** case).

The array **t3** can also be eliminated from the codes. It is redundant when *fvcore* is used. The thermodynamic variable for *fvcore* is the potential temperature (**pt** in the code).

There are also duplicated 3D arrays in phys_state and phys_tend (*eg*, **pint**, **pdel**, **omga**, etc) when fvcore is used.

1.3 Dynamics-Physics interface

It is unclear if the introduction of the "**phys_state**" and "**phys_tend**" data structure made the interface cleaner and more efficient. [Max Suarez/GESO-GCM uses a similar idea. It is cleaner looking. But I don't know if it would help produce a cleaner code]. We can also introduce a data structure for the dynamics to make the interface even more abstract and cleaner looking. But we don't know the impact to the computational efficiency.

For fvcore, we believe for optimal computational efficiency, we probably should combine "**d_p_coupling**" with "**tphysbc**" into a single routine which we may call "**d_p_bc**" (before coupling) where the mapping from the Lagrangian to the Eulerian (ETA) coordinate can also be performed. This will save some 3D to 3D data copying/transposition and reduce the **OPENMP** overhead. [This is what we did in the **NASCAR** model]. After "**d_p_bc**" is called we then call the LSM/ocean models, which is then followed by a "**p_d_ac**", which is a combination of **tphysac** and "**p_d_coupling**".

1.4 Code consolidation

Currently there are many cases of nearly identical code for all three dycores in CCM3_11_XX. Perhaps not all cores will make it into CAM (*aka*, CCM4), but an effort has to be made to consolidate codes. This requires consensus, which is always challenging to achieve. There will always be code with is core-specific. This can be localized. Example: there were some deficiencies with the **SPMDINIT** routine for *fvcore*. The current solution (having an fvcore-specific **SPMDINI** routine) is not necessary. There should be a global **SPMDINIT** routine which calls a dycore-specific initialization routine, *e.g.*, **SPMDINIT_LR**, **SPMDINIT_EUL**, etc.. Of course, one may question the wisdom of supporting three dynamical cores.

1.5 Exception handling (W. Sawyer)

Exceptions as such are not supported in F90/95 (like they are in Java, for example), but we need more solid ways to isolate errors. Lately there have been many complaints that "LR does not work", but it is very difficult to determine the problem because information is lacking. Of course, this leads into the whole discussion of testing. Things are currently going in the right direction: everyone is slowly getting access to all platforms, and the test scripts appear to be converging.

For data assimilation system, there should exist no "stop" statement anywhere in the code (DaSilva). An error code should be returned instead.

1.6 Inline documentation (W. Sawyer)

We recommend that there be a way to document code, create implementation documents in HTML and/or LaTeX. Arlindo Dasilva's "protex" is a very good example. (Contact W. Sawyer and/or A. DaSilva for details).

2 Calendar and update of prognostic variables

This issue has been beaten to death, many times. But it has yet to be done. It is absolutely critical for data assimilation and NWP applications that the model's calendar be capable of handling leap years. The calendar itself, the initialization of the land model as well as other physics packages should not depend on "**nstep**", which could be dynamics dependent.

Currently (ccm3.11.37) the update of prognostic variables is still incorrect (off by one hour) when the finite-volume dynamics is used. To cleanly fix this problem we first need to have new calendar code(s), removing the "nstep" dependency from the lsm and all physics.

3 I/O System

There are simply too much I./O activities in the CCM. Calls to mswrite are currently executed in an asynchronous mode via **ioFileMod**. For a run with 6-hour output frequency (minimum for data assimilation), there may be 5+ individual mswrite processes concurrently. This will require more processors in a queue system (*eg*, PBS) than total number of physical CPUs allocated. We have found this is one of the main reason the model does not scale on DAO's SGI. It would be critical to the throughput that the calls to mswrite be reduced as much as possible. This should be easily achivable by calling it less frequent but increasing the file size. Perhaps the model can already do this. But there is no up to date documentation.

Currently the logic for I/O is too messy and too difficult to understand and modify. A new design is called for. All the I/O needs to be abstracted away to perhaps a separate library/module like GFIO or GPIOS (*e.g.*, DAO's design for I/O for MP-GEOSGCM; *ref*: A. Dasilva).

3.1 Boundary conditions (S. Nebuda)

- For data assimilation, there is a need to separate SST from Sea Ice.
- Allow arbitrary frequency of data for SST, Sea Ice and Ozone. Example: monthly, weekly, and non-uniform.
- Make climatology or cyclic data format consistent, either the year information in the metadata is ignored or must be zero.
- SST and Ozone currently use opposite standards.
- The time interpolation code(s) must support a leap year calendar.

3.2 Initial/restart files

- Separate dyn_fields from the phys_fields in the restart.
- Specifiable frequency (from namelist) for writing restarts (dyn, phys, lsm). For example, writing restarts every 6 or 12 hr in data assimilation or NWP applications.
- Restart program should check if the rpointer files and restart files have the same date. Because rpointer files are written earlier than the restart files, this might cause problems if computer crashed and resubmitted the job at this point.
- Should reduce the files needed for restart run, Currently, it needs almost all of the files (ccmr_*, ccmr_*.A, lsmr, ha, hb, etc.).
- No need to keep the initial run files (ccmi and lsmi) and restart run files (ccmr and lsmr). Should combine these files into one set of restart files.

3.3 History files

- To save memory (so as to allow ultra-high-resolion NWP-type runs), use 32-bit precision for the histry data buffer. To save mass storage space, the output history files should be 32-bit precision or even less.
- Change the file names of daily and 6-hourly history files (ha***.nc and hb***.nc) to include time/date information. Currently it is very difficult to figure out which date it is.
- If we want to put one month of data into "ha" and "hb" files. the number of data should change according to the number of day of each month. Currently it is fixed (30 for ha and 120 for hb).
- Allow variable number of records in each output history file.
- Do not require the existence of an output history file to continue a restart run (lsmh and ha, hb even monthly means). Warn the user instead or tag files as incomplete.
- Review diagnostic names and/or descriptions, have enough diagnostics to compute temperature and water vapor budgets.
- Add 2m and 10m winds and temperature, surface albedo, clear sky heating rates. These are required diagnostics for data assimilation/re-analyses.

4 Parallel framework (W. Sawyer)

We need a general parallel framework (e.g., Sawyer's pilgrim communication library; see dir **dynamics/lr**) that can hide explicit MPI calls from the science developers. The initial difficulty in implementing the messaging passing version of the fvcore into CAM is mostly due to the limitation of the primitive MPI calls in the CAM.

Similar to the I/O, complex message passing tasks should be abstracted away from the code. We realize this is controversial, and a common opinion is that calls to generic MPI routines maybe general enough, so why not just put MPI calls right into the code? Here two examples: J. Rosinski's initial reaction to solving the equal distribution limitation was to say "there are calls to MPI_GATHER in a lot of places which have to be changed". If a standardized routine had been used (like "Gather" or "GatherAndWrite") it would not have been an issue. Second example: certain camps have been emphasizing other message-passing technologies, e.g., shared memory *"arenas"* or the so-called *Multi-Level Parallelism* (MLP, Taft 2000). This may or may not be the way to go, but it is very hard to impossible to use another technology if lots of MPI calls are strewn through the code. [illegible] [illegible] [illegible] performance

5 Utilities

5.1 Grid to grid interpolation/mapping

- There is a need for a utility code to interpolate/map lsm restarts from one horizontal grid to another.
- DAO (S.-J. Lin) can provide a conservative and high order accurate mapping utility codes for transforming dynamic restart file to different vertical/horizontal resolutions.

5.2 Model scripts and environment variables (B. Shen)

The 'test-model.pl' and its super classes are very usefulfor model testing. Especially, they also provide lots of information which were not discussed in the current User's guide. ~~So far, the test-model.pl can perform two major tests: 1) [illegible] and 2) perturbation run~~ several tests for LR, EUL, and SLD dynamics. It will be very helpful if we have a more user-friendly interface, which allows us to do ~~either~~ any one of the ~~above~~ tests with a specific dynamics only. This can save time during the developing stage, since we could test the model without re-running all cases for minor modifications.

Comparison tests among OMP only, SPMD only and Hybrid OMP/SPMD runs should [illegible] be included in the test scripts.

On DAO SGI platforms, environment variables **_DSM_PLACEMENT** in the scripts should be set to **FIRST_TOUCH** for better performance when the fvcore is chosen.

5.3 Makefiles

- It is highly desirable if **-O3** can be used on the SGI to compile the full model. In addition, fast_math_library can be used to speed up the model. Combined with the **FIRST_TOUCH** environment variable mentioned above we found that the model can be up to two times faster when using 32-CPU on the SGI.
- DAO could help modify the Makefile in order to run the model on Linux box with the Lahey/Fujitsu F95 compiler, which appears to be a more robust and stricter compiler than the one on the SGI or the pgf90 on the Linux.

6 Documentations

The current CCM3 user's guide has been useful in guiding us to run the CCM3 model on DAO platforms. However, there is some critical information deserved to be updated or added in the new version. For example,

1. ~~The usage of **empty_htape** in the namelist needs to be added. This is very helpful to perform a test run, since it can reduce the I/O tremendously.~~

2. More detailed description in the naming of the outputs will be helpful for model developers as well as users. So far, we have problems knowing which files will be created and which files will be needed for restart runs when different namelists are used.

3. The mechanisms in the scripts (test-model.pl) used to test the model need be addressed. A brief discussion on how to diagnose the outputs of the perturbation test from test-model.pl will be very helpful.

We need a up to date full description of all the namelist variables. Since this is a community model to be developed by the community, it may also be very useful to have a ***developer's guide***. A developer's workshop could also be very helpful.

7 Data assimilation specific requirements (A. DaSilva)

7.1 There must be a leap-year capable calendar in the model

7.2 The dynamics restart should be separated from the physics

7.3 The I/O efficiency must be improved

For DAO applications, the main parallel bottleneck of the current model appears to be I/O. It is critically important that the I/O handling be more efficient. There appears to be too many open/close operations when the data frequency is 6-hr.

The file formats (netcdf, hdf, grib, etc) should be abstracted out of the code. GFIO is DAO's solution to this problem, but this is not the only solution. Other constraint is that there should be no STOP statement whatsoever in the code. Rather, the system should terminate with well defined (and documented) return codes - this is crucial in an operational environment.

7.4 Scripts

The data assimilation system uses its own scripts. The main requirement is that the f90 code should name its own files (no write to generic fortran unit) and be able to handle **GrADS** templates for file names, *e.g.*, d_rst.%y4%m2%d2_%h2Z.bin (DaSilva). File name conventions shall not be hardwired in the code but rather namelist/resource file specified. Also, output files should be **GrADS** readable. At the architectural level, the system should have well defined (and separate) building, experiment setup, and running steps (sources shall not be required to run the model). Scripts should be governed by the same rules of abstraction and modularity as the f90 code; perl or phython should be used in place of csh.