Cleanup of FV dycore interface

Notes on cleanup of FV dycore interface

Below are notes on what I think needs to be done to cleanup the FV dycore interfaces in CAM. Both short and long term changes are discussed. Some cleanup depends on things being fixed on the physics side of the model. I've discussed both the public interfaces that are visible to CAM's top level control and utility layers, and interfaces that should only be visible within the dycore interface. The public interfaces need to be generic in the sense of being dycore independent since CAM supports multiple dycores. This is a first cut just to get things going. Feel free to comment, criticize, whatever...
Initial version: B. Eaton, 21 February 2007
sawyer.070222 – Comments by Will Sawyer
eaton.070228 – Comments by Eaton

General notes

The SPMD CPP token should be eliminated in favor of using the mpi-serial lib provided by MCT.
Here is the list of files in dynamics/fv that should be considered to be part of the portable FV dycore. These files should not contain any CAM specific code.

```
benergy.F90
cd_core.F90
diag_module.F90
dynamics_vars.F90
epvd.F90
fill_module.F90
FVperf_module.F90
geopk.F90
mapz_module.F90
par_vecsum.F90
par_xsum.F90
pft_module.F90
pkez.F90
sw_core.F90
te_map.F90
tp_core.F90
trac2d.F90
```

The special mode flag for full physics should be hardcoded to .true. in the calls to the FV dycore. We want CAM to be responsible for supplying whatever physics forcings are specified by the user and not have this exercise different code paths in the dycore. The dycore should not care what kind of forcings are being applied.

pmgrid

Contains global grid dimensions that are set via CPP macros. Historically this module has provided the global grid (assumed to be a rectangular lat/lon grid) to any other module. We'd like to transition this module to being an internal part of the CAM specific dycore interface, or completely eliminate it. That is not currently possible because there are still a large number of non-dynamics modules that use it.
sawyer.070222 – Note that it has been completely removed from the model-independent FVCORE code, and, frankly, it should be eliminated entirely: the information it contains is in the T_FVDYCORE_GRID type.
Outside the dynamics the only uses of pmgrid are for the global dimensions plon, plat, and plev. We've eliminated the use of plon and plat from the standard CAM physics, but it's still used by optional aerosol code and by the chemistry packages.
pmgrid also contains variables that describe the dynamics grid decomposition and contains the variables used by pilgrim to do communications. I suspect that this should all be removed from pmgrid since the fvcore's grid object contains all this info.
sawyer.070222 – This is the wrong place for pilgrim-related variables. Your suspicion is right.

spmd_dyn

Computes the dynamics decomposition. It stores info in pmgrid. Probably should be storing info in dyn_state%grid.
Initializes PILGRIM and uses its methods to create the PILGRIM decompositions. It stores info in pmgrid. Probably should be storing info in dyn_state%grid.
sawyer.070222 – Right.
I don't know the details of decompositions and communicators well enough to be sure about what's going on here. There appear to be duplicated calls to decompcreate between spmd_dyn and spmd_vars_init in the dynamics_vars module. I'm not sure where these calls belong. If this is part of the CAM specific code then they belong (I think) in spmd_dyn, because dynamics_vars is part of the portable FV dycore.
sawyer.070222 – The PILGRIM related stuff in spmd_dyn,

```
  type (ghosttype), save  :: ghostpe_yz, ghostpe1_yz
  type (parpatterntype)   :: ikj_xy_to_yz, ijk_yz_to_xy, &
                             ijk_xy_to_yz, pexy_to_pe, pkxy_to_pkc
```

and all the pilgrim calls (such as decompcreate), should not be there (I don't think this variables are even used; the active ones are in dyn_state%grid). My oversight. I'll remove them during the next iteration.
The subroutine compute_gsfactors is only used by the offline driver. This should be eliminated and a pilgrim communication used instead.

dyn_init

the prototype currently (cam3_3_50) looks like this:

```
dyn_init(... im, jm, km, ...         ! global grid
         ak, bk,                     ! vertical grid
         nq, ntotq, ...              ! constituents
         ifirstxy, ilastxy, ...      ! local grid
         cp, rair, ae, ...           ! constants
         npr_yzxy, ...               ! namelist
         dyn_state,                  ! fvcore internal state
         dyn_in, dyn_out)            ! dynamics import and export states
```

dyn_init is the CAM specific initialization for the dycore. Because it is CAM specific there is no need to pass all these arguments that can be obtained from CAM's control/utility layer. The goal is that this interface should look like:

```
dyn_init(dyn_in, dyn_out)
```

sawyer.070222 – Agreed. This is possible now that we have decided that dyn_comp should be on the CAM-specific side.
Modifications to move dyn_init towards it's generic interface:

- the global grid should come from the pmgrid module (or eventually its replacement).
- the vertical grid is in the hycoef module
- constituent number is in constituent module (the distinction between nq and ntotq has been removed from CAM)
  sawyer.070222 – There used to be a distinction between advected (1..nq) and non-advected (nq+1:ntotq) tracers. What are examples of non-advected tracers? No idea. Only having NQ would be preferable, if it is possible.
  eaton.070228 – Yes, having only NQ is possible. I removed the little used and often misunderstood "feature" of non-advected tracers from the source code at cam3_3_47.
- the local dynamics grid is being computed in spmd_dyn, stored in pmgrid, and passed through this interface by cam_initial. Probably spmd_dyn should be getting called inside dyn_init and assign these values directly into dyn_state%grid
- the constants should come from physconst (which should be renamed cam_const).
- the namelist variables should be read directly from an fvcore specific namelist. The name of the file containing the namelist should come from the cam_namelist module (still needs to be created).
- dyn_state doesn't need to be passed through this arg list. CAM is only supporting one instantiation of the dycore, and until there is a need to seriously consider changing that then dyn_state should be able to be module data of the dynamics interface. We tried making it module data in dyn_comp but ended up moving it into its own module (dyn_internal_state) because of a circular dependency that we ran into with the history module.
  sawyer.070222 – Right, dyn_state should be removed from the arg list, unless you want to support multiple dyn_states (Max's goal in GEOS5). The following should be removed from dyn_comp:

  ```
  type (T_FVDYCORE_STATE), save, target :: dyn_state
  ```

  since, as you point out, the real one is in dyn_internal_state.
  dyn_init is not currently responsible for reading either the initial file or the restart file. I think it should be.
  sawyer.070222 – Good point. But as you point out, some work is involved.
- Since the initial file is random access (netcdf) it should be possible to have dyn_init read what it needs from that file. We need a cam_initial_file module to manage the name of the file and be responsible at least to make sure the file exists and can be opened for reading. Then any other module that needs to read initial data can get a filehandle from cam_initial_file.
- The restart file needs to be converted to netcdf before it's reasonable for the dyn_init method to read it. Currently the dycore just provides methods that are called by a higher level control when it's time for the dycore the read its restart data from the binary sequential access file.

  ### commap

  Contains global grid lats, lons, and area weights. Assumes rectangular lat/lon grid. Uses of this module have been eliminated from the standard CAM physics, but there are still some uses in optional aerosol/chemistry packages. So must be maintained for the time being.
  sawyer.070222 – Except for initcom, FVcore only uses the weights "w". Why can't the FVcore maintain its own set of weights (initialized in dyn_init)?

initcom

There is a bunch of global grid info being set in the commap module. This should be eliminated and whatever needs to be set in commap for backwards compatibility purposes should be set by the fvcore method responsible for the global grid (or just be taken from the dyn_state%grid object after it's initialized.
sawyer.070222 – Agreed. Who should do this (I've never even looked at the routine before today)?

phys_grid_init

Initializes the dynamics/physics coupling.
This method is currently accessing the spmd_dyn module. This should be replaced by an access to the dyn_grid module (and if something is missing from dyn_grid then just add it). The dyn_grid module is what supplies anything that needs to be known about the dynamics grid to the outside world – both global and decomposed grid info.
sawyer.070222 – Sounds good.

dyn_grid

Supply information about the global and decomposed dynamics grid to rest of model.
Once all accesses to pmgrid from outside the dycore are eliminated, the global grid dimensions that it contains should be moved here.
Currently information about the decomposed grid is being accessed from the pmgrid, commap, and spmd_dyn modules. These accesses should be replaced by accesses to the dyn_state%grid object.
sawyer.070222 – Right!
All accesses to the rgrid module should be eliminated. The FV core doesn't support a reduced grid capability. The current reduced grid capability in the spectral cores is slated to be removed.
sawyer.070222 – The one CAM has does not, though there is another FVcore (revised by Kevin Yeh) which does. Probably no one wants this capability now that the cubed sphere is coming out. Removing rgrid requires changes to dyn_grid.F90. Who should do this? (I'll be happy to take a shot at it...)

inidat

Currently responsible for reading both dynamics and physics fields from the initial file.
The part of this module that's responsible for reading fields that are copied or scattered to the dynamics grid decomp should be split out into a dynamics only module. Then the code for reading physics fields doesn't need to be duplicated in every dycore directory.
Reading dynamics fields from the initial file should be able to be done from the dyn_init method.
There are too many 3D temporary arrays being declared. We should only need one 3D temporary to read in a global field that will be scattered.
The utility code that does the reading from the netCDF file may need to allocate an additional 3D array to transpose data if the file ordering doesn't match the internal ordering. Right now it looks like 6 3D arrays are being allocated.
sawyer.070222 – Agree on all points.

stepon_init

stepon_init currently exists to hide the fact that generic dynamics interfaces have not been implemented. It should go away.
Everything that's currently being done in stepon_init should be done in dyn_init. Before that can happen dyn_init needs to be made responsible for reading data from the initial file (and restart file?).
sawyer.070222 – Right, but as you pointed out, some work is involved here (restart file needs to be converted to netcdf).

stepon_run1

The stepon module should eventually be eliminated. It exists for historical reasons, and persists to hide the facts that 1) generic dycore interfaces haven't been implemented, and 2) the calling order of physics and dynamics is different for the different dycores. The immediate goal is just to have as little code in the stepon methods as possible.
Currently diag_dynvar_ic and compute_adv_tends_xyz are being called outside the dyn_run call. There's no need for this since dyn_run is a CAM specific interface. Move these calls inside dyn_run.
sawyer.070222 – For other readers who have got this far, some historical background: originally I had planned that dyn_comp (dyn_run) would be the interface for the **model-independent** FVcore
interface. Brian convinced me that we need to provide a CAM-specific interface (dyn_comp) and a model-independent interface (will be: fvcore_comp or something similar).

stepon module

After completing my review of these interfaces I realized that the stepon module also maintains the value of dtime used for the physics timestep. This is not always the same as the value of dtime maintained in CAM's time manager which is the dynamics/physics coupling interval, so let's refer to it instead as phys_dtime. In the case of the Eulerian dycore phys_dtime is equal to dtime for the first call to physics, then is increased to 2☆dtime for subsequent calls to physics due to the leapfrog timestepping scheme. So elimination of stepon also requires a new way to manage the physics timestep. Since the physics timestep is dycore dependent there should probably be a module in the CAM specific dycore interface to manage this, although this seems a bit strange.

dyn_run

The prototype currently (cam3_3_50) looks like this:

```
subroutine dyn_run( ptop,             ! vertical grid
                    ndt,              ! dynamics/physics coupling interval
                    ns0, convt,       ! namelist
                    te0,              ! fvcore internal state
                    dyn_state,        ! fvcore internal state
                    dyn_in, dyn_out,  ! dynamics import/export states
                    rc )              ! return code
```

dyn_run is the CAM specific dynamics run method. Because CAM must support multiple dycores this interface should also be dycore independent. And since CAM is not supporting multiple instantiations of a dycore, the internal state doesn't need to be passed as an argument. So the goal for the dyn_run interface is:

```
dyn_run(ndt, dyn_in, dyn_out)
```

sawyer.070222 – Right. See historical background above.
Changes to go from current interface to generic one:

- ptop is available for the hycoef module
- ns0 is by default setting or from the namelist
- convt should be hardwired to .true. in the dycore. The dycore shouldn't know anything about the nature of the physics forcings
- te0 is internal to the dycore
- dyn_state should be accessed from the dyn_internal_state module
- dycore return codes should be checked inside dyn_run

d_p_coupling

The prototype currently (cam3_3_50) looks like this:

```
subroutine d_p_coupling(grid,                   ! for grid dimensions
                        phys_state, phys_tend,  ! physics import/export states
                        pbuf,                   ! physics buffer
                        full_phys,              ! full physics logical flag
                        dyn_out )               ! dynamics export state
```

d_p_coupling is the CAM specific dynamics to physics coupler. Because CAM supports multiple dycores this interface must be generic. The desired generic (i.e., dycore independent) interface is

```
d_p_coupling(dyn_out, phys_state)
```

sawyer.070222 – dyn_out does not currently contain the grid. Should it? The grid carries all sorts of information needed in dp_coupling.
Changes to get to the generic interface:

- the dimensions that are coming from the grid arg need to come from the dyn_internal_state module which contains dyn_state.
  sawyer.070222 – I think I get it: we can just import dyn_state from dyn_internal_state and reference dyn_state%grid inside dp_coupling. Right?
- full_phys can be accessed from the CAM control module (ideal_phys and adiabatic logicals in cam_control_mod).
- energy conservation variables are being modified in phys_tend and pbuf. Look to see if this can be done in the physics package.

p_d_coupling

The prototype currently (cam3_3_50) looks like this:

```
subroutine p_d_coupling(grid,
                        phys_state, phys_tend,
                        full_phys,  &
                        adiabatic,
                        dyn_in,
                        dtime,
                        zvir, cappa,
                        ptop )
```

p_d_coupling is the CAM specific physics to dynamics coupler. The desired dycore independent interface is

```
p_d_coupling(phys_state, phys_tend, dyn_in)
```

Changes to get to the generic interface:

- get grid dimensions from dyn_internal_state module
- full_phys and adiabatic can be accessed from cam_control_mod (should also evaluate whether these logicals are really needed)
- dtime can be accessed from CAM's time manager (it's the physics/dynamics coupling interval)
- constants come from physconst module
- ptop comes from hycoef module