CAM History and I/O Infrastructure Changes

Steve Goldhaber

National Center for Atmospheric Research
CGD: Atmospheric Modeling & Predictability

June 16, 2015

Outline

- What is the problem?
- Introdution to new structures
- Creating CAM grids
- Using CAM grids for input and history – UI changes
- Advantages of new infrastructure
- Status and upcoming changes

Acknowledgments: Brian Eaton
This work was funded by the National Science Foundation and the Department of Energy

What is the problem?

- CAM history has grown haphazardly over the last 25+ years

What is the problem?

- CAM history has grown haphazardly over the last 25+ years
- Module dependencies prevent intuitive usage patterns
  – so –

What is the problem?

- CAM history has grown haphazardly over the last 25+ years
- Module dependencies prevent intuitive usage patterns

  – so –
- CAM History has ended up being the repository for hardcoded dycore and grid-specific information

What is the problem?

- CAM history has grown haphazardly over the last 25+ years
- Module dependencies prevent intuitive usage patterns

  – so –
- CAM History has ended up being the repository for hardcoded dycore and grid-specific information
- This has led to geometric code complexity

What is the problem?

- CAM history has grown haphazardly over the last 25+ years
- Module dependencies prevent intuitive usage patterns

  – so –
- CAM History has ended up being the repository for hardcoded dycore and grid-specific information
- This has led to geometric code complexity
- Hardcoded assumptions prevent requested features such as variables defined on multiple grids in a single file

What is the problem?

- CAM history has grown haphazardly over the last 25+ years
- Module dependencies prevent intuitive usage patterns

  – so –

- CAM History has ended up being the repository for hardcoded dycore and grid-specific information
- This has led to geometric code complexity
- Hardcoded assumptions prevent requested features such as variables defined on multiple grids in a single file

---

What does this look like?

cam_history (e.g. addfld) and ncdio_atm (infld)

- Special case code for FV (staggered grids), by variable name or magic value

cam_history (e.g. addfld) and ncdio_atm (infld)

- Special case code for FV (staggered grids), by variable name or magic value
- Special case code for SE (and other unstructured grids?)

cam_history (e.g. addfld) and ncdio_atm (infld)

- Special case code for FV (staggered grids), by variable name or magic value
- Special case code for SE (and other unstructured grids?)
- Special case code for dycore-specific NetCDF attributes (but not specific enough as some attributes leak into other dycore's files).

cam_history (e.g. addfld) and ncdio_atm (infld)

- Special case code for FV (staggered grids), by variable name or magic value
- Special case code for SE (and other unstructured grids?)
- Special case code for dycore-specific NetCDF attributes (but not specific enough as some attributes leak into other dycore's files).
- Convoluted user interfaces (example below)

cam_pio_utils

The utility interface between CAM and the Parallel I/O library (PIO). However:

cam_pio_utils

The utility interface between CAM and the Parallel I/O library (PIO). However:

- Special case code for FV (staggered grids), by magic value
- Special case code for SE (and other unstructured grids?)
- Special case code for column (regional) output
- More special case code for column output with new collected-column code
- Special case code for physics decomposition
- Special case code for physics decomposition and regional output
- Special case code for different variable file ordering

cam_pio_utils

The utility interface between CAM and the Parallel I/O library (PIO). However:

- Special case code for FV (staggered grids), by magic value
- Special case code for SE (and other unstructured grids?)
- Special case code for column (regional) output
- More special case code for column output with new collected-column code
- Special case code for physics decomposition
- Special case code for physics decomposition and regional output
- Special case code for different variable file ordering
- Regional output still doesn't work for SE

Introduction to five new structures

Goal: Centralize information about distributed data to facilitate clean and extensible parallel I/O and history output code.

Introduction to five new structures

Goal: Centralize information about distributed data to facilitate clean and extensible parallel I/O and history output code.

1. `cam_filemap_t`
   Contains a map between a distributed 2-D array and that array in NetCDF file order along with methods to create maps for higher-dimensional arrays.
   For every element in the array, the map shows where this element will show up in the NetCDF file representation of that array

Introduction to five new structures

Goal: Centralize information about distributed data to facilitate clean and extensible parallel I/O and history output code.

1. `cam_filemap_t`
   Contains a map between a distributed 2-D array and that array in NetCDF file order along with methods to create maps for higher-dimensional arrays.
   For every element in the array, the map shows where this element will show up in the NetCDF file representation of that array
2. `horiz_coord_t`
   Contains information about a horizontal coordinate (i.e., lat, lon). Coordinates can be distributed across processors.

Introduction to new structures (cont.)

3. `cam_grid_t`
   Contains information about a distributed 2-D grid along with methods for reading and writing distributed arrays defined on that grid.

Introduction to new structures (cont.)

3. `cam_grid_t`
   Contains information about a distributed 2-D grid along with methods for reading and writing distributed arrays defined on that grid.
4. `cam_grid_patch_t`
   Contains a subset of the points of a distributed 2-D grid. This type is useful for regional output.

Introduction to new structures (cont.)

3. `cam_grid_t`
   Contains information about a distributed 2-D grid along with methods for reading and writing distributed arrays defined on that grid.
4. `cam_grid_patch_t`
   Contains a subset of the points of a distributed 2-D grid. This type is useful for regional output.
5. `cam_grid_attribute_t`
   Each attribute is a NetCDF attribute associated with a `cam_grid_t` and is output into any file which contains arrays defined on that grid.

Steps for creating a CAM grid

1. Create horizontal coordinates, usually including mapping (`horiz_coord_register`)
2. Create map from basic 2-D array to file order
3. Add grid (`cam_grid_register`)
4. Add grid attributes (`cam_grid_attribute_register`)

Creating CAM grids – Coordinates

1. Create horizontal coordinates, usually including mapping. Example for unstructured grid:

```
call horiz_coord_register('lat', 'ncol', ngcols_d,   &
                          'latitude', 'degrees_north',   &
                          pelat_deg, pemap)
```

'lat' – The name of the coordinate

'ncol' – The name of the coordinate's dimension. This will be the same as the coordinate name for a rectangular (lat/lon) grid.

ngcols_d – The global size of the coordinate

'latitude' – Coordinate long name

'degrees_north' – Coordinate units

pelat_deg – Values for the coordinate on this PE

pemap – 1-D map between local coordinate values and NetCDF order. May be omitted for a non-distributed coordinate

Creating CAM grids – Grid

3. Create CAM grid. Example for unstructured grid

```
call cam_grid_register('GLL', dyn_decomp, 'lat',     &
                  pelat_deg, 'lon', pelon_deg,        &
                  grid_map, unstruct=.true.)
```

'GLL'	–	The name of the grid
dyn_decomp	–	An integer ID for the grid
'lat'	–	The name of the grid's latitude coordinate
pelat_deg	–	Grid latitude values for this PE (may be the same as associated coordinate)
'lon'	–	The name of the grid's longitude coordinate
pelon_deg	–	Grid longitude values for this PE (may be the same as associated coordinate)
pemap	–	2-D map between local array elements and their NetCDF order.

Using CAM grids for input

infld – old interface

```
call get_dyn_decomp(elem, nlev, pio_double, iodesc)
lsize = pio_get_local_array_size(iodesc)
tlncols = lsize/nlev
allocate(tmp(tlncols,nlev))
call infld(’U’, ncid_ini, iodesc, tlncols,’ lev’, &
      tmp, found)
```

Using CAM grids for input

infld – old interface

```
call get_dyn_decomp(elem, nlev, pio_double, iodesc)
lsize = pio_get_local_array_size(iodesc)
tlncols = lsize/nlev
allocate(tmp(tlncols,nlev))
call infld('U', ncid_ini, ~~iodesc, tlncols~~,' lev', &
      tmp, found)
```

Using CAM grids for input

infld – old interface

```
call get_dyn_decomp(elem, nlev, pio_double, iodesc)
lsize = pio_get_local_array_size(iodesc)
tlncols = lsize/nlev
allocate(tmp(tlncols,nlev))
call infld(’U’, ncid_ini, iodesc, tlncols,’ lev’, &
      tmp, found)
```

infld – new interface

```
allocate(tmp(npsq, nlev, nelemd))
call infld(’U’, ncid_ini, ’ncol’, ’lev’,     &
      1, npsq, 1, nlev, 1, nelemd,           &
      tmp, found, gridname=’GLL’)
```

Using CAM grids for input

infld – new interface

```
allocate(tmp(npsq, nlev, nelemd))
call infld('U', ncid_ini, 'ncol', 'lev',       &
      1, npsq, 1, nlev, 1, nelemd,             &
      tmp, found, gridname='GLL')
```

'ncol', 'lev'	–	NetCDF names of array dimensions
1, npsq, 1, nlev, 1, nelemd	–	Array dimension sizes
gridname='GLL'	–	Grid name

Using CAM grids for input

infld – new interface

```
allocate(tmp(npsq, nlev, nelemd))
call infld('U', ncid_ini, 'ncol', 'lev',         &
      1, npsq, 1, nlev, 1, nelemd,               &
      tmp, found, gridname='GLL')
```

'ncol', 'lev'	–	NetCDF names of array dimensions
1, npsq, 1, nlev, 1, nelemd	–	Array dimension sizes
gridname='GLL'	–	Grid name

Using CAM grids for input

infld – new interface

```
allocate(tmp(npsq, nlev, nelemd))
call infld('U', ncid_ini, 'ncol', 'lev',      &
     1, npsq, 1, nlev, 1, nelemd,             &
     tmp, found, gridname='GLL')
```

'ncol', 'lev'	–	NetCDF names of array dimensions
1, npsq, 1, nlev, 1, nelemd	–	Array dimension sizes
gridname='GLL'	–	Grid name

Using CAM grids for input

infld – new interface

```
allocate(tmp(npsq, nlev, nelemd))
call infld(’U’, ncid_ini, ’ncol’, ’lev’,       &
      1, npsq, 1, nlev, 1, nelemd,             &
      tmp, found, gridname=’GLL’)
```

'ncol', 'lev'	–	NetCDF names of array dimensions
1, npsq, 1, nlev, 1, nelemd	–	Array dimension sizes
gridname='GLL'	–	Grid name

Using CAM grids for output

addfld – old interface

```
call addfld ('FU', 'm/s2', nlev, 'A',               &
       'Zonal wind forcing term', dyn_decomp,  &
       begdim1=1, enddim1=npsq,               &
       begdim3=1,enddim3=nelemd)
```

Using CAM grids for output

addfld – old interface

```
call addfld (’FU’, ’m/s2’, nlev, ’A’,            &
        ’Zonal wind forcing term’, dyn_decomp,  &
        ~~begdim1=1, enddim1=npsq,~~                 &
        ~~begdim3=1,enddim3=nelemd~~)
```

Using CAM grids for output

addfld – old interface

```
call addfld ('FU', 'm/s2', nlev, 'A',                 &
        'Zonal wind forcing term', dyn_decomp,  &
        ~~begdim1=1, enddim1=npsq,~~                 &
        ~~begdim3=1,enddim3=nelemd~~)
```

Using CAM grids for output

addfld – old interface

```
call addfld ('FU', 'm/s2', nlev, 'A',              &
       'Zonal wind forcing term', dyn_decomp,  &
       begdim1=1, enddim1=npsq,               &
       begdim3=1,enddim3=nelemd)
```

addfld – new interface

```
call addfld ('FU', (/ 'lev' /), 'A', 'm/s2',       &
       'Zonal wind forcing term', gridname='GLL')
```

Using CAM grids for output

addfld – new interface

```
call addfld ('FU', (/ 'lev' /), 'A', 'm/s2',         &
       'Zonal wind forcing term', gridname='GLL')
```

(/ 'lev' /)	–	Names of non-grid array dimensions
gridname='GLL'	–	Grid name

Using CAM grids for output

addfld – new interface

```
call addfld ('FU', (/ 'lev' /), 'A', 'm/s2',         &
       'Zonal wind forcing term', gridname='GLL')
```

(/ 'lev' /)	–	Names of non-grid array dimensions
gridname='GLL'	–	Grid name

Using CAM grids for output

addfld – new interface

```
call addfld ('FU', (/ 'lev' /), 'A', 'm/s2',          &
      'Zonal wind forcing term', gridname='GLL')
```

(/ 'lev' /)	–	Names of non-grid array dimensions
gridname='GLL'	–	Grid name

CAM history namelist UI changes

- New namelist, **cam_history_nl**, for all history items

CAM history namelist UI changes

- New namelist, **cam_history_nl**, for all history items
- SE-specific analysis_nl namelist folded into cam_history_nl

CAM history namelist UI changes

- New namelist, **cam_history_nl**, for all history items
- SE-specific analysis_nl namelist folded into cam_history_nl
- interpolation resolution and type can vary by history file

CAM history namelist UI changes

- New namelist, **cam_history_nl**, for all history items
- SE-specific analysis_nl namelist folded into cam_history_nl
- interpolation resolution and type can vary by history file
- NB: Any items in analysis_nl namelists will have to be moved to cam_history_nl or camexp

CAM history namelist UI changes

- New namelist, **cam_history_nl**, for all history items
- SE-specific analysis_nl namelist folded into cam_history_nl
- interpolation resolution and type can vary by history file
- NB: Any items in analysis_nl namelists will have to be moved to cam_history_nl or camexp

---

How does all this improve things?

Advantages of new infrastructure

- All dycore specific information compact and local to dycore code

Advantages of new infrastructure

- All dycore specific information compact and local to dycore code
- Physics package gets column locations and areas from dycore but defines its own grid (decomposition)

Advantages of new infrastructure

- All dycore specific information compact and local to dycore code
- Physics package gets column locations and areas from dycore but defines its own grid (decomposition)
- History and I/O infrastructure does not need any dycore specific information

Advantages of new infrastructure

- All dycore specific information compact and local to dycore code
- Physics package gets column locations and areas from dycore but defines its own grid (decomposition)
- History and I/O infrastructure does not need any dycore specific information
- Adding or modifying a new dycore becomes a much easier task

Advantages of new infrastructure

- All dycore specific information compact and local to dycore code
- Physics package gets column locations and areas from dycore but defines its own grid (decomposition)
- History and I/O infrastructure does not need any dycore specific information
- Adding or modifying a new dycore becomes a much easier task
- Grids manage their own coordinates and variables

Impact on cam_pio_utils

- No special case code for dycore or physics (no code dependency)

Impact on cam_pio_utils

- No special case code for dycore or physics (no code dependency)
- No special case code for column (regional) output

Impact on cam_pio_utils

- No special case code for dycore or physics (no code dependency)
- No special case code for column (regional) output
- Column output is a parallel operation

Impact on cam_pio_utils

- No special case code for dycore or physics (no code dependency)
- No special case code for column (regional) output
- Column output is a parallel operation
- SE column-output fix is automatic

Status and upcoming changes

- Code review complete
- Going through final testing

Status and upcoming changes

- Code review complete
- Going through final testing

Coming soon (not the infrastructure trunk tag):

- Support for output of zonal means (probably only FV and SE)
- Separate grid for CAM physics package (physgrid)

Status and upcoming changes

- Code review complete
- Going through final testing

Coming soon (not the infrastructure trunk tag):

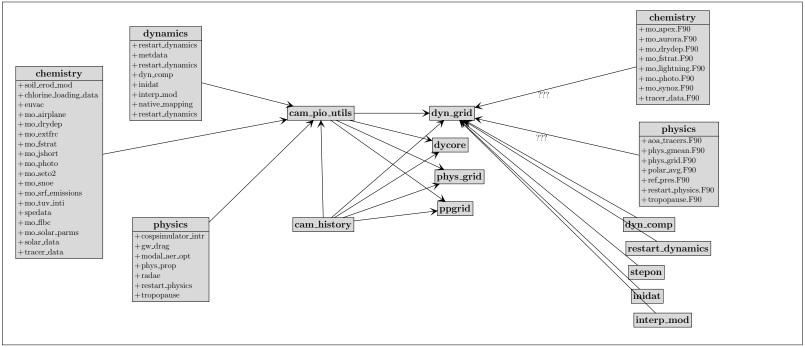
- Support for output of zonal means (probably only FV and SE)
- Separate grid for CAM physics package (physgrid)

Possible future developments

- Online mapping between grids?
- Convert more of CAM to use grid infrastructure
- Move new grid infrastructure to public CIME infrastructure for use by other components or models?

Questions?

Current CAM I/O and history interface



dynamics
- +restart_dynamics
- +metdata
- +restart_dynamics
- +dyn_comp
- +inidat
- +interp_mod
- +native_mapping
- +restart_dynamics

chemistry
- +soil_erod_mod
- +chlorine_loading_data
- +euvac
- +mo_airplane
- +mo_drydep
- +mo_extfrc
- +mo_fstrat
- +mo_jshort
- +mo_photo
- +mo_seto2
- +mo_snoe
- +mo_srf_emissions
- +mo_tuv_inti
- +spedata
- +mo_flbc
- +mo_solar_parms
- +solar_data
- +tracer_data

physics
- +cospsimulator_intr
- +gw_drag
- +modal_aer_opt
- +phys_prop
- +radae
- +restart_physics
- +tropopause

cam_pio_utils

dyn_grid

dycore

phys_grid

ppgrid

cam_history

chemistry
- +mo_apex.F90
- +mo_aurora.F90
- +mo_drydep.F90
- +mo_fstrat.F90
- +mo_lightning.F90
- +mo_photo.F90
- +mo_synoz.F90
- +tracer_data.F90

???

physics
- +aoa_tracers.F90
- +phys_gmean.F90
- +phys_grid.F90
- +polar_avg.F90
- +ref_pres.F90
- +restart_physics.F90
- +tropopause.F90

???

dyn_comp

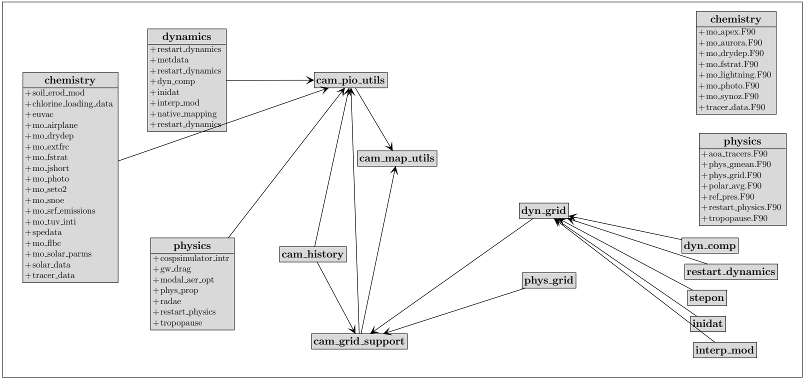
restart_dynamics

stepon

inidat

interp_mod

Proposed CAM I/O and history interface



dynamics
+restart_dynamics
+metdata
+restart_dynamics
+dyn_comp
+inidat
+interp_mod
+native_mapping
+restart_dynamics

chemistry
+soil_erod_mod
+chlorine_loading_data
+euvac
+mo_airplane
+mo_drydep
+mo_extfrc
+mo_fstrat
+mo_jshort
+mo_photo
+mo_seto2
+mo_snoe
+mo_srf_emissions
+mo_tuv_inti
+spedata
+mo_flbc
+mo_solar_parms
+solar_data
+tracer_data

physics
+cospsimulator_intr
+gw_drag
+modal_aer_opt
+phys_prop
+radae
+restart_physics
+tropopause

cam_pio_utils

cam_map_utils

cam_history

cam_grid_support

dyn_grid

phys_grid

dyn_comp

restart_dynamics

stepon

inidat

interp_mod

chemistry
+mo_apex.F90
+mo_aurora.F90
+mo_drydep.F90
+mo_fstrat.F90
+mo_lightning.F90
+mo_photo.F90
+mo_synoz.F90
+tracer_data.F90

physics
+aoa_tracers.F90
+phys_gmean.F90
+phys_grid.F90
+polar_avg.F90
+ref_pres.F90
+restart_physics.F90
+tropopause.F90

Infrastructure user interface – Horiz coords

- horiz_coord_register
- horiz_coord_get_index
- horiz_coord_get_dim_name

Infrastructure user interface – Grids

- cam_grid_register
- cam_grid_attribute_register
- cam_grid_write_attrs
- cam_grid_write_vars
- cam_grid_read_dist_array
- cam_grid_write_dist_array

Infrastructure user interface – Grids (cont.)

- cam_grid_dimensions
- cam_grid_num_grids
- cam_grid_check ! T/F if grid ID exists
- cam_grid_id ! Grid ID (decomp) or -1 if error
- cam_grid_get_local_size
- cam_grid_get_file_dimids
- cam_grid_get_decomp
- cam_grid_get_gcid
- cam_grid_get_array_bounds

Infrastructure user interface – cam_pio_utils

- cam_pio_createfile
- cam_pio_openfile
- cam_pio_closefile
- cam_pio_newdecomp
- init_pio_subsystem ! called from cam_comp
- cam_pio_get_decomp
- cam_pio_handle_error
- cam_permute_array
- calc_permutation

Infrastructure user interface – cam_pio_utils

! Convenience interfaces

- cam_pio_def_dim
- cam_pio_def_var
- cam_pio_get_var

Infrastructure user interface – cam_pio_utils

! General utility

- cam_pio_var_info
- cam_pio_find_var
- cam_pio_check_var

Creating CAM grids – Coordinates

```
call horiz_coord_register(’lat’, ’ncol’, ngcols_d,  &
                          ’latitude’, ’degrees_north’,   &
                          pelat_deg, pemap)


call horiz_coord_register(’lon’, ’ncol’, ngcols_d,  &
                          ’longitude’, ’degrees_east’,   &
                          pelon_deg, pemap)
```

Creating CAM grids – Coordinates

```
call horiz_coord_register(’lat’, ’ncol’, ngcols_d,  &
                     ’latitude’, ’degrees_north’,   &
                     pelat_deg, pemap)


call horiz_coord_register(’lon’, ’ncol’, ngcols_d,  &
                     ’longitude’, ’degrees_east’,   &
                     pelon_deg, pemap)
```

```
call horiz_coord_register(’slat’, ’’, (plat - 1),  &
                     ’staggered latitude’,         &
                     ’degrees_north’, slatvals)

call horiz_coord_register(’lon’, ’lon’, plon,     &
                     ’longitude’, ’degrees_north’,  &
                     lonvals, coord_map)
```

Creating CAM grids – Grid

```
call cam_grid_register('GLL', dyn_decomp, 'lat',     &
                       pelat_deg, 'lon', pelon_deg,    &
                       grid_map, unstruct=.true.)
```

Creating CAM grids – Grid

```
call cam_grid_register('GLL', dyn_decomp, 'lat',    &
                  pelat_deg, 'lon', pelon_deg,      &
                  grid_map, unstruct=.true.)
```

```
call cam_grid_register('fv_centers', dyn_decomp,   &
                  'lat', latvals, 'lon', lonvals, &
                  grid_map)
```

```
call cam_grid_register('fv_u_stagger',               &
                  dyn_ustag_decomp,                  &
                  'slat', slatvals, 'lon', lonvals,  &
                  grid_map)
```