CAM-decomp

Information on the CAM decomposition, for ESMF folks to verify:

Location of files to describe decomposition:
/fis/cgd/home/erik/CAM_decomps
(NOTE: This archive was moved to MSS under /ERIK/archive/CAM_decomps.c060612.tar.gz on June/12/2006)
Subdirectories. exist for each grid. These directories contain the decomposition descriptions for each case, and softlinks to input files that contain the CAM and CLM input grids.
What the Physics Grid Create would look like:

```
integer, parameter :: numdims = 2
type(ESMF_LOGICAL) :: periodic(numdims)= (/ESMF_TRUE, ESMF_FALSE/)
character(len=64) :: dimnames(numdims) = (/"longitude","latitude "/)
character(len=64) :: dimunits(numdims) = (/"degrees","degrees"/)
!
! pcols is the max number of columns in a chunk:
! For straight latitude decomposition, pcols=plat
! Otherwise, pcols = 16
! On vector platforms pcols = large value optimized for vectorization loops
!
coord1(1) = 0.0
do lon = 2, plon
     coord1(lon) = (londeg(lon,1) + londeg(lon-1,1))*0.5
end do
coord1(plon+1) = 360.0
coord2(1) = -90.0
do lat = 2, plat
    coord2(lat) = (latdeg(lat) + latdeg(lat-1))*0.5
end do
coord2(plat+1) = 90.0

MyCount = 0
n            = 0
do c = begchunk, endchunk  ! Loop over chunks in this processor
    MyCount = get_ncols_p(c) + Mycount
    do i = 1, get_ncols_p(c)   ! Number of columns in a chunk
        n = n + 1
        myIndices(n,1) = get_lat_p(c,i)
        myIndices(n,2) = get_lon_p(c,i)
    end do
end do
!
! NOTE: Optionally a DELayout could be created with DE's for each chunk, with chunks
! begchunk to endchunk on virtual DE's that are on the same processor.
! Create and distribute ESMF grid
phys_grid = ESMF_GridCreateHorzLatLon(
                     coord1, coord2,                      &
                     horzstagger=ESMF_GRID_HORZ_STAGGER_A, &
                     dimnames=dimnames,dimunits=dimunits,  &
                     name="Physics Grid", rc=rc)
call ESMF_GridDistribute(phys_grid, delayout=layout,       &
                         myCount=myCount, &
                         myIndices=myIndices, rc=rc)
```

What are the different physics chunking options that will be used?

- Option == -1 Latitude decomposition
- Option == 0: split local longitude/latitude blocks into chunks, while attempting to create load-balanced chunks
- Option == 1: load balance chunks and assignment, attempting to also minimize communication costs
- Option == 2: split local longitude/latitude blocks into chunk assigning columns using block ordering
- Option == 3: split indiviudal longitude/latitude blocks into chunks, assigning columns using block ordering (default)
  What about the Land model decomposition?
  The Land model clumped decomposition turns the land points into a 1D vector and thus has a different decomposition than the physics chunked decomposition.
  The land grid:

```
   use clmtype,     only: clm3, gridcell_type
   integer, parameter :: numdims = 2
   type(ESMF_LOGICAL) :: periodic(numdims)= (/ESMF_TRUE, ESMF_FALSE/)
   character(len=64) :: dimnames(numdims) = (/"longitude","latitude "/)
   character(len=64) :: dimunits(numdims) = (/"degrees","degrees"/)
   type(gridcell_type), pointer  :: gptr ! pointer to gridcell derived subtype
   coord1(1) = 0.0
   do lon = 2, lsmlon
        coord1(lon) = (longxy(lon,1) + longxy(lon-1,1))*0.5
   end do
   coord1(lsmlon+1) = 360.0
   coord2(1) = -90.0
   do lat = 2, lsmlat
       coord2(lat) = (latixy(lat) + latixy(lat-1))*0.5
   end do
   coord2(lsmlat+1) = 90.0
   call get_proc_bounds( begg, endg, begl, endl, begc, endc, &
                                     begp, endp)
   gptr => clm3%g
   !
   ! For CLM NOT all of the grid points will be filled (only roughly 1/3rd)
   !
   MyCount = endg - begg + 1
   n = 0
   do g = begg, endg    ! Loop over grid points used on this processor, get their global indices
        n = n + 1
       myIndices(n,1) = gptr%ixy(g)
       myIndices(n,2) = gptr%jxy(g)
   end do
   lnd_grid = ESMF_GridCreateHorzLatLon(
                        coord1, coord2,
&
                        horzstagger=ESMF_GRID_HORZ_STAGGER_A, &
                        dimnames=dimnames,dimunits=dimunits,  &
                        name="CLM Land model Grid", rc=rc)
   call ESMF_GridDistribute(lnd_grid, delayout=layout,       &
                            myCount=myCount, &
                            myIndices=myIndices, rc=rc)
```

What about the redistribute between land and atmosphere?
This redistribution goes both directions.

```
   call ESMF_BundleRedistStore(atmBundle, lndBundle, parentVM, a2l_route, &
          routeOptions=ESMF_ROUTE_OPTION_SYNC+ESMF_ROUTE_OPTION_PACK_PET, rc)
   call  ESMF_BundleRedist( atmBundle, lndBundle, a2l_route, rc
```

What are the grids and decompositions that will be used for the performance evaluation?
According to the Evaluation plan there are 2 different grids and a total of 8 different decompositions/grids that will be evaluated. Those are:

- 64x128.noomp 16,32,64 tasks
- 64x128 8 tasks, 8 threads
- 128x256.noomp 32, 64, 128 tasks
- 128x256 16 tasks, 8 threads
  What are the grids that will be used?:
  T5 8x16 Grid:
  lat = -73.7992136285632, -52.8129431899943, -31.704091745008,
  -10.5698823125761, 10.5698823125761, 31.704091745008, 52.8129431899943,
  73.7992136285632 ;
  lon = 0, 22.5, 45, 67.5, 90, 112.5, 135, 157.5, 180, 202.5, 225, 247.5, 270,
  292.5, 315, 337.5 ;
  T21 32x64 Grid:
  lat = -85.7605871204438, -80.26877907225, -74.7445403686358,
  -69.2129761693708, -63.6786355610969, -58.1429540492033,
  -52.6065260343453, -47.0696420596877, -41.5324612466561,
  -35.9950784112716, -30.4575539611521, -24.9199286299486,
  -19.3822313464344, -13.8444837343849, -8.30670285651881,
  -2.76890300773601, 2.76890300773601, 8.30670285651881, 13.8444837343849,

19.3822313464344, 24.9199286299486, 30.4575539611521, 35.9950784112716,
41.5324612466561, 47.0696420596877, 52.6065260343453, 58.1429540492033,
63.6786355610969, 69.2129761693708, 74.7445403686358, 80.26877907225,
85.7605871204438 ;
lon = 0, 5.625, 11.25, 16.875, 22.5, 28.125, 33.75, 39.375, 45, 50.625,
56.25, 61.875, 67.5, 73.125, 78.75, 84.375, 90, 95.625, 101.25, 106.875,
112.5, 118.125, 123.75, 129.375, 135, 140.625, 146.25, 151.875, 157.5,
163.125, 168.75, 174.375, 180, 185.625, 191.25, 196.875, 202.5, 208.125,
213.75, 219.375, 225, 230.625, 236.25, 241.875, 247.5, 253.125, 258.75,
264.375, 270, 275.625, 281.25, 286.875, 292.5, 298.125, 303.75, 309.375,
315, 320.625, 326.25, 331.875, 337.5, 343.125, 348.75, 354.375 ;
T42 64x128 Grid:
lat = -87.8637988392326, -85.0965269883174, -82.3129129478863,
-79.5256065726594, -76.7368996803683, -73.9475151539897,
-71.1577520115873, -68.3677561083132, -65.5776070108278,
-62.7873517989631, -59.9970201084913, -57.2066315276432,
-54.4161995260862, -51.6257336749383, -48.8352409662506,
-46.0447266311017, -43.2541946653509, -40.463648178115,
-37.6730896290453, -34.8825209937735, -32.091943881744,
-29.3013596217627, -26.510769325211, -23.7201739335347,
-20.9295742544895, -18.1389709902394, -15.3483647594915,
-12.5577561152307, -9.76714555919557, -6.97653355394864,
-4.18592053318915, -1.3953069108195, 1.3953069108195, 4.18592053318915,
...., 87.8637988392326 ;
lon = 0, 2.8125, 5.625, 8.4375, ..., 357.1875 ;
T85 128x256 Grid:
lat = -88.9277353522959, -87.5387052130272, -86.1414721015279,
-84.7423855907142, -83.3425960440704, -81.9424662991732,
-80.5421464346171, -79.1417096486217, -77.7411958655138,
-76.3406287023715, -74.9400230196494, -73.5393886337675,
-72.1387322891624, -70.7380587725176, -69.3373715749609,
-67.9366733025785, -66.5359659401756, -65.1352510260352,
-63.7345297708429, -62.3338031405324, -60.9330719152074,
-59.5323367318266, -58.1315981156439, -56.7308565037137,
-55.3301122627028, -53.9293657025561, -52.5286170870997,
-51.1278666423533, -49.7271145631097, -48.3263610181882,
-46.9256061546646, -45.5248501013023, -44.1240929713558,
-42.723334864877, -41.3225758706231, -39.9218160676465,
-38.5210555266244, -37.1202943109788, -35.719532477824,
-34.3187700787707, -32.918007160614, -31.5172437659226,
-30.1164799335463, -28.7157156990552, -27.3149510951204,
-25.9141861518467, -24.5134208970629, -23.1126553565776,
-21.7118895544042, -20.3111235129604, -18.9103572532454,
-17.5095907949986, -16.1088241568413, -14.7080573564048,
-13.3072904104462, -11.9065233349538, -10.5057561452436,
-9.10498885604852, -7.70422148160049, -6.3034540357076,
-4.90268653182654, -3.5019189831313, -2.10115140257898,
-0.700383802973324, 0.700383802973324, 2.10115140257898, 3.5019189831313,
....,
88.9277353522959 ;
lon = 0, 1.40625, 2.8125, 4.21875, ... 358.59375 ;
FV 10x15 Grid (19x24)
lat = -90, -80, -70, -60, -50, -40, -30, -20, -10, 0, 10, 20, 30, 40, 50,
60, 70, 80, 90 ;
lon = 0, 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210,
225, 240, 255, 270, 285, 300, 315, 330, 345 ;
FV 4x5 Grid (46x72)
lat = -90, -86, -82, ... 90 ;
lon = 0, 5, 10, ..., 355 ;
FV 2x2.5 Grid (91x144 points)
lat = -90, -88, -86, -84, ..., 90 ;
lon = 0, 2.5, 5, 7.5, ..., 357.5 ;

Write physics chunked decomposition:
Code that does this is checked in on: cam3_2_38_brnch_wrtdecomps

```
subroutine cam_writedecomp()
   use pmgrid,     only: plat, plon
   use commap,     only: londeg, latdeg
   use phys_grid,  only: get_ncols_p, get_lat_p, get_lon_p, phys_grid_getopts
   use ppgrid,     only: pcols, begchunk, endchunk
   use spmd_utils, only: iam, npes
   use units,      only: getunit, freeunit
   integer, parameter :: numdims = 2
   real(r8) :: coord1(plon+1)
   real(r8) :: coord2(plat+1)
   integer :: MyCount
```

```
   integer :: MyIndices(pcols*(endchunk-begchunk+1),numdims)
   character(len=256) :: filename
   integer :: n, n1, c, j, lat, lon, i   ! Indices
   integer :: ncols
   integer :: unit
   integer :: phys_load
   !
   ! pcols is the max number of columns in a chunk:
   ! For straight latitude decomposition, pcols=plat
   ! Otherwise, pcols = 16
   ! On vector platforms pcols = large value optimized for vectorization loops
   ! Coord1, and coord2 are the coordinate vertices that would be entered on GridCreate
   !
   coord1(1) = 0.0
   do lon = 2, plon
        coord1(lon) = (londeg(lon,1) + londeg(lon-1,1))*0.5
   end do
   coord1(plon+1) = 360.0
   coord2(1) = -90.0
   do lat = 2, plat
       coord2(lat) = (latdeg(lat) + latdeg(lat-1))*0.5
   end do
   coord2(plat+1) = 90.0
   MyCount = 0
   n       = 0
   call phys_grid_getopts( phys_loadbalance_out=phys_load )
   unit = getunit( iu=10+iam)
   write(filename,1010) plat, plon, iam, phys_load
1010 format( 'CAM_decomp:',i3.3,'x',i3.3, 'task-', i2.2, 'load-',i1.1,'.txt')
   write(6,*) 'Write out decomposition on file: ', filename
   call shr_sys_flush(6)
          open(unit=unit,file=filename,status='REPLACE',form='FORMATTED',action='WRITE')
   do c = begchunk, endchunk  ! Loop over chunks in this processor
       ncols = get_ncols_p(c)
       MyCount = ncols + Mycount
   end do
   write(unit,fmt='(a,i6)') 'Total number of grid points on this PE = ', MyCount
   do c = begchunk, endchunk  ! Loop over chunks in this processor
       ncols = get_ncols_p(c)
       n1 = n + 1
       write(unit,fmt='(a, 4i6)') 'Chunk id, #columns-in-chunk, start-index, end-index', &
                                  c, ncols, n1, n1+ncols
       do i = 1, ncols  ! Number of columns in a chunk
           n = n + 1
           myIndices(n,1) = get_lat_p(c,i)
           myIndices(n,2) = get_lon_p(c,i)
       end do
       write(unit,fmt='(a)') 'Latitude: global indices then coordinate in degrees '
       write(unit,fmt='(20i6)') (myIndices(j,1), j = n1,n)
       write(unit,fmt='(20f6.1)') (latdeg(myIndices(j,1)), j = n1,n)
       write(unit,fmt='(a)') 'Longitude: global indices then coordinate in degrees '
       write(unit,fmt='(20i6)') (myIndices(j,2), j = n1,n)
       write(unit,fmt='(20f6.1)') (londeg(myIndices(j,2),myIndices(j,1)), j = n1,n)
   end do
   close(unit)
   call freeunit(unit)
   write(6,) 'Done writting decomp'
   call shr_sys_flush(6)
end subroutine cam_writedecomp
```

Land decomp:

```
  subroutine clm_camWriteDecomp()
   use clmtype         , only : clm3, gridcell_type
   use clm_varpar      , only : lsmlon, lsmlat
   use clm_varsur      , only : latixy, longxy, numlon
   use clmtype         , only : gridcell_type
   use spmd_utils      , only : iam, npes
   use units           , only : getunit, freeunit
```

```
   use phys_grid       , only : phys_grid_getopts
   use shr_sys_mod     , only : shr_sys_flush
   use decompMod       , only : get_proc_bounds
   integer, parameter :: numdims = 2
   type(gridcell_type), pointer  :: gptr ! pointer to gridcell derived subtype
   real(r8) :: coord1(lsmlon+1,lsmlat+1)   ! Longitude
   real(r8) :: coord2(lsmlon+1,lsmlat+1)   ! Latitude
   integer :: lat, lon
   integer :: begg, endg, begl, endl, begc, endc, begp, endp
   integer, allocatable :: myIndices(:,:)
   integer :: MyCount, n, g, j, n1, n2, stride, nn
   character(len=256) :: filename
   integer :: unit
   integer :: phys_load
   coord1(:,1) = -90.0
   coord2(1,:) =   0.0
   do lat = 2, lsmlat
     do lon = 2, numlon(lat)
        coord1(lon,lat) = (longxy(lon,lat) + longxy(lon,lat-1))*0.5
        coord2(lon,lat) = (latixy(lon,lat) + latixy(lon-1,lat))*0.5
     end do
     coord2(numlon(lat)+1,lat) = 360.0
   end do
   coord1(:,lsmlat+1) =  90.0
   call get_proc_bounds( begg, endg, begl, endl, begc, endc, begp, endp)
   gptr => clm3%g
   !
   ! Open file to write out to...
   !
   call phys_grid_getopts( phys_loadbalance_out=phys_load )
   unit = getunit( iu=40+iam)
   write(filename,1010) lsmlat, lsmlon, iam, phys_load
1010 format( 'CLM_decomp:',i3.3,'x',i3.3, 'task-', i2.2, 'load-',i1.1,'.txt')
   write(6,*) 'Write out decomposition on file: ', filename
   call shr_sys_flush(6)
   open(unit=unit,file=filename,status='REPLACE',form='FORMATTED',action='WRITE')
   !
   ! For CLM NOT all of the grid points will be filled (only roughly 1/3rd)
   !
   MyCount = endg - begg + 1
   write(unit,fmt='(a,i6)') 'Total number of grid points on this PE = ', MyCount
   allocate( myIndices(MyCount,numdims) )
   n = 0
   do g = begg, endg    ! Loop over grid points used on this processor, get their global indices
       n = n + 1
       myIndices(n,1) = gptr%ixy(g)   ! Longitude
       myIndices(n,2) = gptr%jxy(g)   ! Latitude
   end do
   !
   ! Write to file:
   !
   stride = 14
   n2 = 0
   do nn = 1, n, stride
      n1 = n2+1
      n2 = n1 + stride-1
      if ( n2 > n ) n2 = n
      write(unit,fmt='(a,i6,i6)')  'Write grid points between: n1,n2: ', n1, n2
      write(unit,fmt='(a)')      'Longitude: global indices then coordinate in degrees '
      write(unit,fmt='(20i7)')   (myIndices(j,1), j = n1,n2)
      write(unit,fmt='(20f7.1)') (longxy(myIndices(j,1),myIndices(j,2)), j = n1,n2)
      write(unit,fmt='(a)')      'Latitude: global indices then coordinate in degrees '
      write(unit,fmt='(20i7)')   (myIndices(j,2), j = n1,n2)
      write(unit,fmt='(20f7.1)') (latixy(myIndices(j,1),myIndices(j,2)), j = n1,n2)
   end do
   call shr_sys_flush(6)
   deallocate( myIndices )
   close(unit)
   call freeunit(unit)
   write(6,*) 'Done writting decomp'
  end subroutine clm_camWriteDecomp
```

Decomposition information in CODE

There is a directory called "cam3_2_46_brnch_mct" that has all of the CAM and CLM code in question. I soft-linked the main program to the top level, so you can see what the top level driver looks like. The CAM decomposition is figured out in the file
"models/atm/cam/src/physics/cam1/phys_grid.F90"
The land model decomposition is figured out in the file
"models/lnd/clm2/src/main/decompMod.F90"
Files that have MCT data-structures in them all have a "MCT_" prefix in them.
So
"models/lnd/clm2/src/main/MCT_lnd_comp.F90"
figures out how to map the CLM internal description of it's decomposition into MCT, for example.
"models/atm/cam/src/control/MCT_atm_comp.F90"
does the same for the atmosphere model.
Each of the previous MCT_ layer subroutines have functions that define a MCT Global Seg Map inside them for that particular component in question. But, they use phys_grid or decompMod methods to figure out how to do this. In your case you could either read the files, or try to use the decomposition methods. The problem with doing the later is that you are likely to need a lot of other code that those two files are dependent on.
Looking for use statements in phys_grid...

- use shr_kind_mod – kind parameters
- use ppgrid, only ------ physics size parameters
- use pmgrid, only ----- Dynamics size parameters
- use abortutils --------- Abort method
- use spmd_utils ------- SPMD description (would need this, and their two components below)
    - use spmd_phys
    - use spmddyn
    - The above is about 2k lines)
- use psect ---------- Spectral data sizes
- use dycore ---------- Short description of which dycore is running
- use rgrid ---------- Reduced grid stuff (could probably hard-code this)
- use commap ---------- Latitude and Longitude could read this from a file
- use mod_com -------- Pilgrim stuff – is this needed? This is probably lots of stuff with it
- use dyn_grid ---------- Pretty short description of the dynamics decomp
- use mpishorthand — Some MPI wrappers – short