Adding MOAB to CIME’s MCT driver

Robert Jacob, Iulian Grindeanu, Vijay Mahadevan, Jason Sarich

CESM SEWG winter meeting
February 27, 2018

Support: DOE BER “Climate Model Development and Validation” project
Biggest development since June 2017 meeting...
The CIME mct driver/coupler

MCT datatypes used in coupler, driver and “caps” for models.

cime/src/drivers/mct/ (driver and coupler code)
cime/src/externals/mct (MCT library)
Recall: MCT’s idea of the mesh (grid)

- MCT only knows about the number of grid points or “degrees of freedom”, dof.
  - Each grid must have its points numbered with integers. One unique integer per unique point.
    - Numbering up to user/model.
    - Must be consistent if the same grid appears in more than one model.
  - MCT GlobalSegmentMap records how grid points are assigned to processors.
    - The GSMap is duplicated on every processor a model grid is decomposed on.
    - At a minimum GSMap size grows with P, number of processors
    - At a maximum can grow with N, total number of points.
      - Optimal grid numbering: nearby points are sequentially numbered
      - Sub-optimal grid numbering: randomly number the points.
Example of sub-optimal grid numbering

NOTE: Figures made possible by MOAB Visit plugin!
MCT driver/coupler regridding workflow: offline/online

**offline phase**

- **Generate grid**
  - MPAS grid-gen, HOMME: internal or GenerateCSMesh

- **Generate dual**
  - HOMME only. Iterative method in Matlab

- **Convert to SCRIP**
  - (some of the above can output SCRIP)

- **Feed grids to program that finds intersection and calculates weights.**
  - SCRIP or ESMF_Regridder or TempestRemap

- **Obtain weights file in SCRIP format**
  - Ex: map_ne30np4_to_oEC60to30v3_conserve.161222.nc

- **Update config file in source code**
  - cime/config/($model)/config_grid.xml
# MCT driver/coupler regridding workflow: offline/online

## online phase: init

<table>
<thead>
<tr>
<th>Task</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define GSMaps</td>
<td>Separately in each model. Reflects model-preferred decomposition</td>
</tr>
<tr>
<td>Communicate grids to coupler</td>
<td>Only the total number of dofs.</td>
</tr>
<tr>
<td>Define coupler GSMaps</td>
<td>Trivial decomposition</td>
</tr>
<tr>
<td>Read in weight files.</td>
<td>Read a block and scatter. Each node picks out what it needs.</td>
</tr>
<tr>
<td>Initialize MCT Routers/Rearranglers</td>
<td>Figure out communication needed to complete a mapping. Between model and coupler. Within coupler.</td>
</tr>
</tbody>
</table>
MCT driver/coupler regridding workflow: offline/online

**online phase: run**

- Receive new data from source model
  - Stored in an MCT AttributeVector

- Possibly rearrange data prior to mapping
  - Runtime option “X” mapping type

- Perform local sparse-matrix-vector multiple
  - Use MCT sMatAvMult method

- Possibly rearrange data after mapping
  - Runtime option “Y” mapping type.

- Send remapped data to destination model
  - Stored in an MCT AttributeVector
MCT driver/coupler regridding workflow: offline/online

Problems

• Total offline workflow poorly documented.
• Easy to lose provenance at each step.
• Need to generate map sets for different resolution combinations.
  – E3SM 1 degree fully coupled case (no ice sheet) has 6 maps.
  – 100s of mapping files in https://svn-ccsm-inputdata.cgd.ucar.edu/trunk/inputdata/cpl/gridmaps/
• Bottleneck to trying simulations with different grids.

Possible Solution

• Support online interpolation by replacing MCT with MOAB!
Mesh-Oriented datABase (MOAB)

- Library for representing, manipulating structured, unstructured mesh models

- Supported mesh types:
  - FE zoo (vertices, edges, tri, quad, tet, pyramid, wedge, knife, hex)
  - Polygons/polyhedra
  - Structured mesh

- Optimized for memory usage first, speed second

- Implemented in C++, but uses array-based storage model
  - Avoids C++ object-based allocation/deallocation
  - Allows access in contiguous arrays of data

- Mesh I/O from/to various formats
  - HDF5 (custom), vtk, CCMIO (Star CD/CCM+), Abaqus, CGM, Exodus, NetCDF

- Main parts:
  - Core representation
  - Tool classes (skinner, kdtree, OBBtree, ParallelComm, …)
  - Tools (mbsize, mbconvert, mbzoltan, mbcoupler, …)
Mesh-Oriented datABase (MOAB)

- Library for representing, manipulating structured, unstructured mesh models
- Supported mesh types: (not just dof numbers)
  - FE zoo (vertices, edges, tri, quad, tet, pyramid, wedge, knife, hex)
  - Polygons/polyhedra
  - Structured mesh
- Optimized for memory usage first, speed second
- Implemented in C++, but uses array-based storage model
  - Avoids C++ object-based allocation/deallocation
  - Allows access in contiguous arrays of data
- Mesh I/O from/to various formats
  - HDF5 (custom), vtk, CCMIO (Star CD/CCM+), Abaqus, CGM, Exodus, NetCDF
- Main parts:
  - Core representation
  - Tool classes (skinner, kdtree, OBBtree, ParallelComm, …)
  - Tools (mbsize, mbconvert, mbzoltan, mbcoupler, …)
Online MOAB regridder coupling workflow

- **Offline phase:** only the mesh generation step

- **Online phase init:**
  - For each component in E3SM
    - Replicate component meshes with MOAB database
    - Migrate MOAB mesh from component PEs to coupler PEs
  - Compute pair-wise mesh intersections in parallel
  - Compute projection weights with TempestRemap
  - Store weights in a parallel SparseMatrix datastructure (not MCT)

- **Online phase run:**
  - Send solution data from a source component to coupler
  - Execute parallel matvec operation to compute remapped solution
  - Communicate solution to target component from coupler
Replicate component meshes

- MOAB (Fortran) interfaces available for both HOMME and MPAS within E3SM
- Partition aware, in-memory copy of the full component mesh

HOMME (left) MPAS (right) meshes on 16 processes, replicated and visualized through MOAB
Migrate component meshes

- Use a custom partition to transfer from component PEs to coupler PEs
- Currently we use trivial partition but can integrate ParMetis or Zoltan for optimal redistribution for better coupler scalability

Migrated HOMME (left) MPAS (right) meshes on 8 processes in coupler PEs, replicated and visualized through MOAB
Compute mesh intersection

- Compute the "coverage", and migrate the source mesh to cover the target elements
- MOAB uses an advancing-front algorithm with fast adjacency queries for intersection

Compute the coverage source (HOMME) mesh to cover the local MPAS elements

Compute the mesh intersection between HOMME and MPAS meshes
Advancing front intersection

• Computing mesh intersection with the advancing front algorithm is an independent task in parallel once we have the full coverage mesh
• Several opportunities to accelerate with fine-grained parallelism (threads)
Implementation in CIME/E3SM

- Add MOAB methods alongside MCT methods (for verification)
- Perform all MOAB based operations in parallel with MCT operations.

```c
# Initialize MOAB in driver

void cime_driver()
{
    cime_pre_init();
    seq_comm_init();
    call mct_world_init(ncomps, DRIVER_COMM, comms, comps);
    #ifdef HAVE_MOAB
        ierr = iMOAB_InitializeFortran();
    #endif
}
```
Implementation in CIME/E3SM

cime_driver (Initialize mesh in component)
cime_init
component_init_cx
  ocn_mct_init (in MPAS)
  #ifdef HAVE_MOAB
    call mpas_moab_instance(domain_ptr) ! should return MPOID ..
    call mpas_log_write('initialized MOAB MPAS ocean instance...')
  #endif

---

cime_driver (Initialize mesh in MPAS)
cime_init
component_init_cc
  ocn_mct Init (in MPAS)
  #ifdef HAVE_MOAB
    call mpas_moab_instance(domain_ptr)
    call mpas_log_write('initialized MOAB MPAS ocean instance...')
  #endif
Complexities in parallel weight generation

• The intersection is computed with coverage mesh and not coupler view of the source mesh; DoF mismatch from component view.

• So we need to maintain decomposition mapping between
  – Component and coupler PEs for meshes
  – Original source and coverage mesh in coupler PEs

• Parallel computation of weight matrix can utilize 2 different algorithms
  – Decouple SparseMatrix objects with only DoF contribution from local process
  – Assemble full parallel SparseMatrix for the weights (similar to FEM assembly)

• The first strategy will involve a reduction during each matrix-vector product, while the other strategy only involves reduction during setup.

• Which strategy is appropriate? Matrix size, nnz’s, Ω_{source} \cap Ω_{coverage}

• Note: Such issues do not exist with offline regridding.
Current status and next steps

- Can compile and build E3SM that
  - Instantiates MOAB
  - Defines full meshes in HOMME and MPAS
  - Migrates full meshes to coupler
  - Computes intersection in coupler
  - Computes weights
  - Migrates data from atmosphere to coupler
  - Applies weights
  - Migrates regridded data to ocean
MOAB stores the data to be transferred between models and provides methods for efficiently transferring the data between the components, accounting for the different partitions and grids. TempestRemap provides remapping tools. HYPRE provides scalable linear algebra routines. All these libraries are built upon existing scientific computation libraries, such as NetCDF, LAPACK, MPI.
• More info: http://sigma.mcs.anl.gov/moab-library/

• Soon in http://github.com/ESMCI/cime

cime/src/drivers/mct/
cime/src/drivers/moab/
Extra slides after this
Resolving Interface Mesh

- Start: local mesh, with global ids on vertices
- Find: vertices & other entities shared by other procs; for each shared entity, need remote processor(s), remote entity handle(s)
- Solve in 2 stages:
  - Shared vertices (using TL.gs_init on global ids)
  - Shared edges/faces (based on connectivity, after remote vertices known)
Parallel mapping problem example

Case: ne11 SE atmosphere, QU240 MPAS ocean
Atmosphere on 2 processors, ocean on 4

Decomposition in the models.
Parallel mapping problem example

Case: ne11 SE atmosphere, QU240 MPAS ocean
Atmosphere on 2 processors, ocean on 4

Decomposition in the coupler.