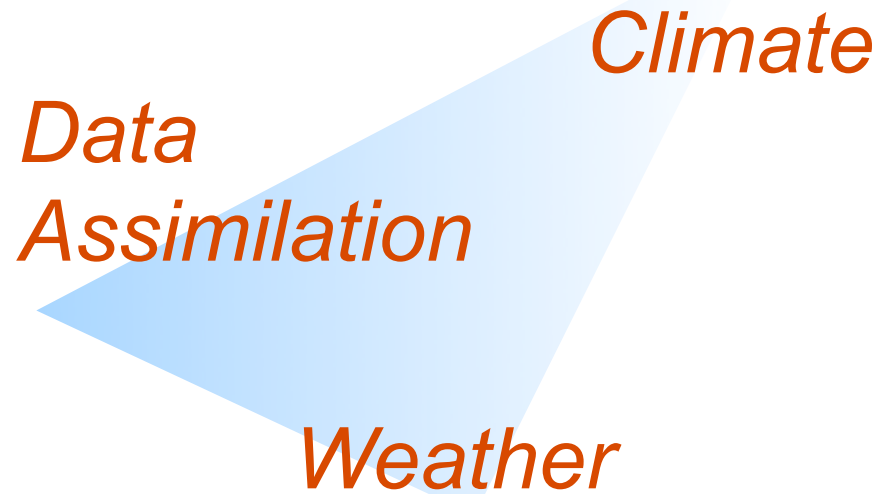


Support for non-rectilinear Grids in ESMF



Donald Stark stark@ucar.edu

Thursday June 22, 2006



Status



- Internal Release ESMF Version 3.0.0 (May 2006)
 - Very general multi-patch data structures
 - Sparse matrix multiply for regridding can handle general structured & unstructured grids
 - New data structures will be the building blocks for a variety of methods on grids such as cubed sphere, tripole, geodesic, etc.
 - Improved build by adding better link stage support for external libraries.
- Greater user satisfaction with build
- Growing interest from groups working in related physical domains
- Increase in user contributions to source



ESMF Platform Support

- IBM AIX (32 and 64 bit addressing)
- SGI IRIX64 (32 and 64 bit addressing)
- SGI Altix (64 bit addressing)
- Cray X1 (64 bit addressing)
- Compaq OSF1 (64 bit addressing)
- Linux Intel (32 and 64 bit addressing, with mpich and lam)
- Linux PGI (32 and 64 bit addressing, with mpich)
- Linux NAG (32 bit addressing, with mpich)
- Linux Absoft (32 bit addressing, with mpich)
- Linux Lahey (32 bit addressing, with mpich)
- Linux g95 (32 bit addressing, with mpich)
- Mac OS X with xlf (32 bit addressing, with lam)
- Mac OS X with absoft (32 bit addressing, with lam)
- Mac OS X with NAG (32 bit addressing, with lam)
- Mac OS X (both G5 & Intel processors) with g95 (32 bit addressing, with lam)

- Currently porting to NEC
- Currently porting to Cray XT3



ESMF Code Adoption



| Scope | Description | Status |
|---------------------------|--|--|
| Atmosphere | (1) NCAR Community Atmosphere Model (CAM), (2) NASA-GMAO GEOS-5 Atmosphere, (3) NOAA-GFDL Atmosphere-land-ice, (4) NOAA-NCEP Global Forecast System (GFS), (5) MIT MITgcm Atmosphere, (6) DoD-NRL COAMPS, (7) UCLA AGCM, (8) GISS Model-E, (9) NSF-NOAA Weather Research and Forecast Model (WRF) | (1) TD, (2) Prod, (3) TD, (4) TD, (5) Prod opt., (6) TD, (7) Prod opt., (8) TD, (9) TD |
| Atmospheric dynamics/Phys | (1) NASA-GMAO fvCore, (2) NOAA-NCEP non-hydrostatic mesoscale dynamics (NMM), (3) NASA-GMAO gravity wave drag, (4) GEOS-5 turbulence, (5) GEOS-5 atmospheric chemistry, (6) GEOS-5 moist processes, (7) NSF-NCAR CAM physics, (8) GEOS-5 radiation. | (1) Prod, (2) TD, (3)(4)(5)(6) Prod, (7) TD, (8) Prod |
| Atmospheric Analysis | (1) NASA-GMAO GSI, (2) NOAA_NCEP SSI | (1) Opt, (2) TD |



ESMF Code Adoption



| Scope | Description | Status |
|-------------------|---|--|
| Ocean | (1) DoD-NRL Hycom, (2) MIT MITgcm Ocean, (3) NOAA-GFDL MOM4 Ocean, (4) DoD-NRL NCOM, (5) DOE-LANL NSF-NCAR POP, (6) GMU Poseidon (7) ROMS | (1) TD, (2) Prod opt., (3) TD, (4) TD, (5) TD, (6) Prod |
| Ocean Physics | (1) NASA-GMAO Biogeochemistry, (2) NASA-GMAO Ocean Radiation | (1)(2) Prod |
| Coastal & Sea-ice | (1) DoD-ERDC ADCIRC, (2) DOE-LANL CICE, (3) NASA-GMAO data sea-ice | (1)(2) TD, (3) Prod |
| Land surface | (1) NASA-GMAO Land, (2) NASA-GMAO Catchment, (3) NASA-GMAO vegetation, (4) NASA-GMAO lake, (5) NASA-GMAO land ice, (6) NASA-GMAO salt water, (7) DoD ERDC WASH123 | (1)(2)(3)(4)(5) (6) Prod, (7) TD |



New Directions



- Extension into new domains and coupling to other frameworks
 - Space weather (SWMF, CISM, AFWA)
 - Air quality
 - Hydrology
- Increased support for C/C++ (*from DoD/NRL*)
- Extension into a new coupling paradigm (*Alan Sussman, University of Maryland*)
 - Multiple executable
 - Direct intercomponent data transfers via State put & get
- Observational data streams (*Will Sawyer, NASA GMAO*)
- 3D grids (*Bob Oehmke, University of Michigan*)



New Support for Grids

ESMF 3_0_0 beta provides support for general structured grids through three new classes.

- DELayout class
- DistGrid class
- Array class

Currently this support requires that the grid and remapping weights for remapping are user provided objects.

This functionality is modeled on that used by MCT.

The ESMF implementation differs from that of MCT in that MCT requires a 1D data structure. The attribute vector, unless it is 1D, must be packed and reshaped for each communication. ESMF avoids the additional copy and reshape by using multidimensional ESMF_Array objects to reference native arrays.

DELayout Class

DELayout class provides an additional layer of abstraction to the Virtual Machine (VM).

- Virtual machine (VM) is defined in terms of the persistent execution threads (PETs), specified by the machine architecture
- Machine resources defined in terms of Decomposition Elements (DEs)
- the DELayout object keeps track of the relationship between DEs and VM.
- number of DEs chosen to best match computational requirements (load balancing)
 - # of DEs < PETs, user controlled multithreading - e.g. multicore CPUs
 - # of DEs > PETs, finer granularity for load balancing.
 - # of DEs = PETs, reorder resources to optimize communication paths.

ESMF_DELayoutCreate

- **deCount**, e.g. deCount=4*petcount
- **petlist & petMap**, associate mappings between specific DEs & PETs.
- **compWeights & commWeights**, to balance heterogeneous system.



DistGrid Class

The DistGrid object describes the index space topology & its decomposition in terms of the DEs.

- Allows the specification of the global domain as the union any number of logically rectangular patches.
- Allows the creation of complex index space topologies by specifying contact relationships between patches.

ESMF_DistGridCreate

- global index space dimensions **minCorner & maxCorner**
- decomposition topology descriptor **regDecomp**
- patch contact supports periodic interconnect.

Combined with DELayout and VM, the DistGrid object defines the data distribution of a domain decomposition across a component's computational resources.

New Array Class

- Distributed data class describing decomposed array data
 - Array object decomposition defined by **DistGrid**.
 - Supports pointers, and explicit shape & allocable F90 arrays.
 - Local array bounds obtained with **ArrayGet**.
- Current support for
 - **ESMF_ArraySparseMatMul**
 - **ESMF_ArrayScatter**
- Future support for
 - **ESMF_ArrayBundle**
 - **ESMF_ArrayHalo**
 - **ESMF_ArrayRedist**
 - **ESMF_ArrayReduce**



Remapping/Interpolation

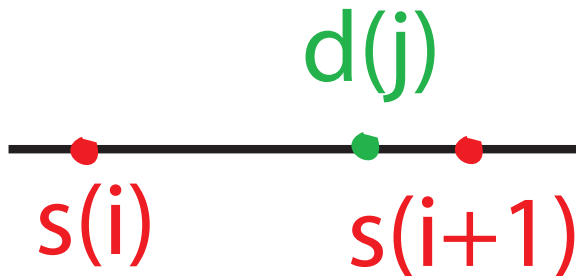
Consider the problem of remapping a 1D field Γ . Let the source grid be defined at the locations $s(i)$ and the destination grid at the points $d(j)$. Specify the source points bounding each destination point with the indirect addressing $BND(j)$ on the left and $BND(j+1)$ on the right. Thus

$$s(BND(j)) < d(j) < s(BND(j+1))$$

Then the interpolated field at the destination points is

$$\Gamma(d(j)) = w(j) * \Gamma(s(BND(j))) + w(j+1) * \Gamma(s(BND(j+1)))$$

where the weights must satisfy $w(j) + w(j+1) = 1$.



Remapping as a Sparse Matrix

In matrix form the interpolation scheme becomes a sparse matrix problem. For a large number of points, most of the matrix is filled with zeros.

$$\begin{bmatrix} \Gamma(d_1) \\ \Gamma(d_2) \\ \vdots \\ \Gamma(d_j) \\ \vdots \end{bmatrix} = \begin{bmatrix} w_1 w_2 & 0 & 0 & 0 \dots \\ 0 & w_2 w_3 & 0 & 0 \dots \\ \vdots & \vdots & \vdots & \vdots \dots \\ \dots & 0 & w_j w_{j+1} & 0 \dots \\ \vdots & \vdots & \vdots & \vdots \ddots \end{bmatrix} \begin{bmatrix} \Gamma(BND_1) \\ \Gamma(BND_2) \\ \vdots \\ \Gamma(BND_j) \\ \vdots \end{bmatrix}$$

More efficient to solve this system by recasting it as a sparse matrix multiplication.

Sparse Matrix Multiplication

Indirect addressing is used to eliminate the need to store nonzero matrix elements.

```
dst_array = 0.0
do k=1,num_links
  dst_array(dst_address(k)) = dst_array(dst_address(k)) +
    remap_matrix(k)*src_array(src_address(k))
end do
```

Indirect addressing reduces the matrix multiplication to a single loop.

ESMF_ArraySparseMatMul

ESMF 3_0_0 Beta provides routines to conduct the sparse matrix multiplication as a parallel process employing the routines

- **ESMF_ArraySparseMatMulStore**

to set up the multiply and

- **ESMF_ArraySparseMatMul**

to conduct the multiplication.

A fully realized example of the processes of remapping a field using the **SparseMatMul** routines can be found in the **Use_Test_Case** directory at **SourceForge** under **ESMF_SparseMatMul**.





ESMF_ArraySparseMatMul

The first stage scatters the weights and the indirect addresses for the multiplication, and constructs a **routehandle** for each DE to locate the transform information.

```
call ESMF_ArraySparseMatMulStore( srcArray=srcArray,           &
                                   dstArray=dstArray,           &
                                   factorList=remap_weights,    &
                                   factorIndexList=remap_addresses, &
                                   rootPet=0,                   &
                                   routehandle=sparseMatMulHandle, ... )
```

Currently the **remap_weights** and the **remap_addresses** must initially reside on the **rootPET**, where they are scattered by the **Store** command. The public version of the routines will also support *pre-distributed* transformation information. The **routehandle** provides the map to this transform information for each DE. The information from the **Store** call persists so that multiple calls can be made to the **SparseMatMul** routine.





ESMF_ArraySparseMatMul

The second call conducts the actual matrix multiplication, placing the interpolated values in the array object `dstArray`.

```
call ESMF_ArraySparseMatMul( srcArray=srcArray,           &  
                             dstArray=dstArray,          &  
                             routehandle=sparseMatMulHandle, rc=rc)
```

The `routehandle`, which is constructed by the `ArraySparseMatMulStore` call, points to the necessary transform values for the remapping.

Summary

- ESMF now supports general structured grids through the new
 - DELayout class
 - DistGrid class
 - Array class
- This more general grid support requires that the user supply
 - both the source and destination grids
 - and the remapping weights.
- Future support for regridding will allow the option of internally generated grids and remapping weights.

