ESMF Regridding

Robert Oehmke, Peggy Li, Ryan O'Kuinghttons
NOAA Cooperative Institute for Research in Environmental Sciences
University of Colorado, Boulder
robert.oehmke@noaa.gov

January 30, 2017

Context

- NEMS uses the Earth System Modeling Framework (ESMF), which provides:
  - Fundamental coupling operations such as fast parallel regridding of fields between models, parallel communication, and model time management
  - Parallel data structures for representing fields, grids, and model components in a standard way
- This talk is about the regridding part of ESMF
- Except where noted, the work presented here is finished and we expect it to come out in ESMF 7.1.0 due late spring/early summer 2017
- Also available before that as a development snapshot

ESMF

Basic Regridding Flow

- Setup grids to represent geometry
  - Cubed sphere for FV3
  - Tripole for MOM, Hycom, CICE
  - Gaussian for GSM
- Build Fields on grids to hold data

↓

- Setup regridding matrix between grids
  - Chose regrid method
  - Chose other regrid options

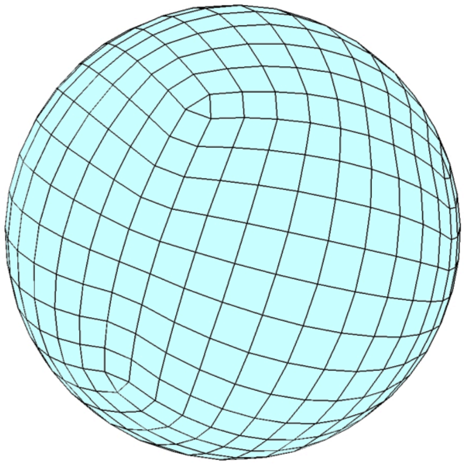
OR

- Build Exchange grid between grids
- Setup regridding matrix using exchange grid

↓

- Apply regridding matrix to move data from source to destination

ESMF Cubed Sphere Support



- Two ways cubed spheres are supported in ESMF:
  1. Unstructured Mesh
     - Data fields are 1D
     - Somewhat more efficient for calculating regridding weights
  2. Multi-tile Grid:
     - Data fields are 2D which more naturally matches shape of tiles

- Both representations can be regridded to other ESMF geometry types (i.e. Grids, Meshes, and Location Streams)

- We have recently added **three new APIs** to allow easier creation of cubed spheres in ESMF

Cubed Sphere Mesh Generation API

ESMF_MeshCreateCubedSphere(tileSize, nx, ny, rc)

Create a ESMF_Mesh object for a cubed sphere grid using identical regular decomposition for every tile. The grid coordinates are generated based on the algorithm used by GEOS-5. The tile resolution is defined by tileSize. The total number of PETs has to be nx x ny x 6 (i.e. the smallest number of PETS it can run on is 6). We expect this restriction to be lifted by February 2017.

tilesize - the number of elements on each side of the tile of the cubed sphere grid
nx - the number of processors on the horizontal size of each tile
ny - the number of processors on the vertical size of each tile
[rc] - return code; equals ESMF_SUCCESS if there are no errors

```
! Calculate decomposition
nx = petCount/6
ny = 1

! Create Mesh
mesh = ESMF_MeshCreateCubedSphere(tileSize=45, nx=nx,ny=ny, rc=localrc)
```

Cubed Sphere Grid Generation API

ESMF_GridCreateCubedSphere(tileSize,regDecompPTile, & decompflagPTile, deLabelList, delayout, name, rc)

Create a six-tile ESMF_Grid for a Cubed Sphere grid using regular decomposition. Each tile can have different decomposition. The grid coordinates are generated based on the algorithm used by GEOS-5. The tile resolution is defined by tileSize.

tilesize - the number of elements on each side of the tile of the cubed sphere grid
[regDecompPTile] - list of DE counts for each dimension. The second index steps through the tiles.
[decompflagPTile] - list of decomposition flags indicating how each dimension of each tile is to be divided between the DEs.
[deLabelList] - list assigning DE labels to the default sequence of DEs.
[delayout] - optional ESMF_DELayout object to be used. By default a new DELayout object will be created with as many DEs as there are PETs.
[name] -ESMF_Grid name.
[rc] - Return code; equals ESMF_SUCCESS if there are no errors.

```
! Setup Decomposition
decompTile(:,1)=(/2,2/)  ! Tile 1
decompTile(:,2)=(/2,2/)  ! Tile 2
decompTile(:,3)=(/2,2/)  ! Tile 3
decompTile(:,4)=(/1,3/)  ! Tile 4
decompTile(:,5)=(/1,3/)  ! Tile 5
decompTile(:,6)=(/1,3/)  ! Tile 6

! Create cubed sphere grid
grid = ESMF_GridCreateCubedSphere(tileSize=20, regDecompPTile=decomptile, rc=rc)
```

Cubed Sphere Read From GridSpec File API

**ESMF_GridCreateMosaic(filename,regDecompPTile,
decompflagPTile, &
deLabelList, delayout, &
name, tileFilePath, rc)**

Create a six-tile ESMF_Grid for a cubed sphere grid using regular decomposition. Each tile can have different decomposition. The tile connections are defined in a GRIDSPEC format mosaic file.

filename - The name of the GRIDSPEC Mosaic file

[regDecompPTile] - List of DE counts for each dimension

[decompflagPTile] - List of decomposition flags indicating how dimension of each tile is to be divided between DEs

[deLabelList] - List assigning DE labels to the default sequence of DEs

[delayout] - Optional ESMF_DELayout object to be used

[name] - ESMF_Grid name

[tileFilePath] - Optional argument to define the path where the tile files reside

[rc] - Return code; equals ESMF_SUCCESS if there are no errors.

```
! Create cubed sphere grid from file using default decomposition
 grid = ESMF_GridCreateMosaic(filename='data/C48_mosaic.nc’, rc=rc)
```

Periodic Grid Create (Tripole or Gaussian)

ESMF_GridCreate1PeriDim(regDecomp, decompflag, & minIndex, maxIndex, & polekindflag, …., rc)

Create a grid with one periodic dimension using regular decomposition. The grid will consist of one tile. After creation the user needs to set the coordinates in the grid using ESMF_GridAddCoord() and then ESMF_GridGetCoord().

[regDecomp] – DE counts for each dimension

[decompflag] – Decomposition flags indicating how each dimension is to be divided between the DEs

[minIndex] – The lowest index in the grid for each dimension.

maxIndex- The maximum index in the grid for each dimension.

[polekindflag] – flag specifying the type of connection that occurs at each pole. Defaults to monopole.

[rc] - Return code; equals ESMF_SUCCESS if there are no errors.

```
! Use defaults to create 100x100 spherical grid with a monopole at each pole
! (e.g. for Gaussian)
grid = ESMF_GridCreate1PeriDim(maxIndex=(/100,100/),  rc=rc)
```

```
! Create 100x100 spherical grid with bipole and monopole
! (e.g. for Tripole)
grid = ESMF_GridCreate1PeriDim(maxIndex=(/100,100/),  &
         polekindflag=(/ESMF_ POLEKIND_MONOPOLE, ESMF_POLEKIND_BIPOLE/),  &
         rc=rc)
```

Creating a Field on a Grid

An ESMF_Field wraps model variables

Additional metadata is stored along with the field data, such as the associated grid, stagger, etc.

Field options

-data types: int*4, int*8, real*4, real*8
-memory allocated by user or ESMF
-stagger locations: center, corner, edges
-local or global indexing
-ungridded dimensions
-halo region (“ghost” cells)

```
! create a grid
grid = ESMF_GridCreate1PeriDim(minIndex=(/1,1/), &
   maxIndex=(/10,20/), &
   regDecomp=(/2,2/), name="atmgrid", rc=rc)


 ! create a field from the grid and typekind
! this allocates memory for you
field1 = ESMF_FieldCreate(grid,   &
      typekind=ESMF_TYPEKIND_R4, &
      indexflag=ESMF_INDEX_DELOCAL, &
      staggerloc=ESMF_STAGGERLOC_CENTER, &
      name="pressure", rc=rc)


! get local bounds, assuming one local DE
call ESMF_FieldGet(field1, localDe=0, farrayPtr=farray2d, &
   computationalLBound=clb, computationalUBound=cub, &
   totalCount=ftc)

do i = clb(1), cub(1)
  do j = clb(2), cub(2)
                    farray2d(i,j) = …  ! computation over local DE
  enddo
enddo
```

Code that creates an ESMF_Field on center stagger with local indexing. Memory is allocated by ESMF. The local bounds are retrieved.

Calculate Regridding Weights and Apply Them

Regrid operation computed in two phases

The first phase **computes an interpolation weight matrix** which is efficiently stored in an ESMF_RouteHandle.

The weights **only need to be computed once.**

The second phase **applies the weight matrix** to a source field resulting in a destination field.

This same pattern is used for other operations such as **redistribution** and **halo**.

```
! create source and destination grids
srcGrid = ESMF_GridCreateCubedSphere(...)
dstGrid = ESMF_GridCreate1PeriDim(...)

! Create Fields to hold data
srcField = ESMF_FieldCreate(srcGrid,...)
dstField = ESMF_FieldCreate(dstGrid,...)

! compute regrid weight matrix
call ESMF_FieldRegridStore(srcField, dstField, routehandle, ...)

! loop over time
do t=1,...

            ! compute new srcField

      ! apply regrid weight matrix in parallel
            call ESMF_FieldRegrid(srcField, dstField,
routehandle,  ...)
enddo

! release resources
call ESMF_FieldRegridRelease(routehandle, ...)
```

Typical code pattern for executing an ESMF communications operations. Once computed, a RouteHandle can be reused for multiple calls.

Regrid Methods

- Bilinear:
  - Destination is a linear combination of source cell corners
  - Weights based distance from corners
  - Typically used to regrid model state variables (e.g. temperature)

- Higher order patch recovery:
  - Multiple polynomial patches represent region around source cell
  - Destination is linear combination of patch values
  - Yields better derivatives/smoother results than bilinear
  - Based on “patch recovery” used in finite element modeling [1][2]

- Nearest neighbor:
  - Destination is equal to closest source point (or vise versa)
  - Useful for extrapolating data outside of source grid, or categorical data

Conservative Regrid Methods

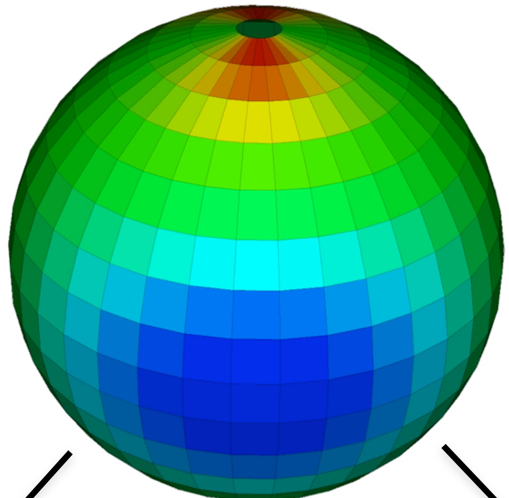
- First-order conservative:
  - –Destination is combination of intersecting source cells
  - –Preserves integral of data across interpolation

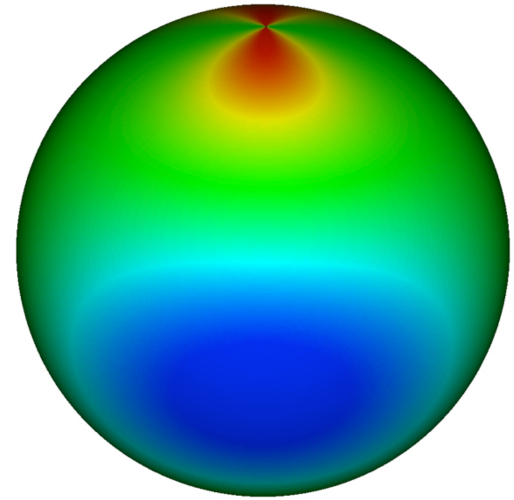
- Higher-order conservative (in progress):
  - –Destination is combination of intersection source cells modified to take into account source cell gradient
  - –Requires a wider stencil and more computation, so more expensive in terms of memory and time than first-order
  - –Preserves integral of field across interpolation, but gives smoother results than first-order (especially when going from coarser to finer grids

ESMF

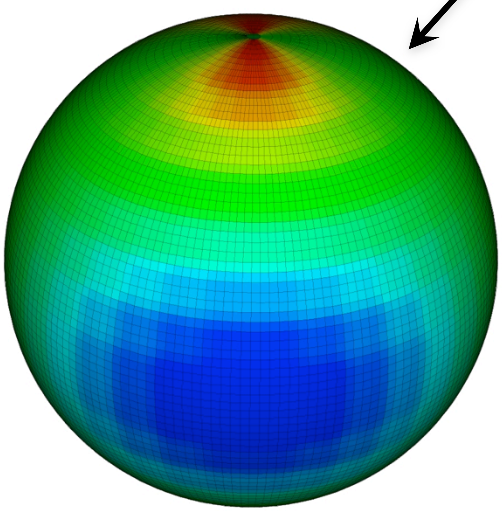
Conservative Methods Example

Source:
•10 degree uniform global
•F = 2+cos(lon)^2 * cos(2*lat)



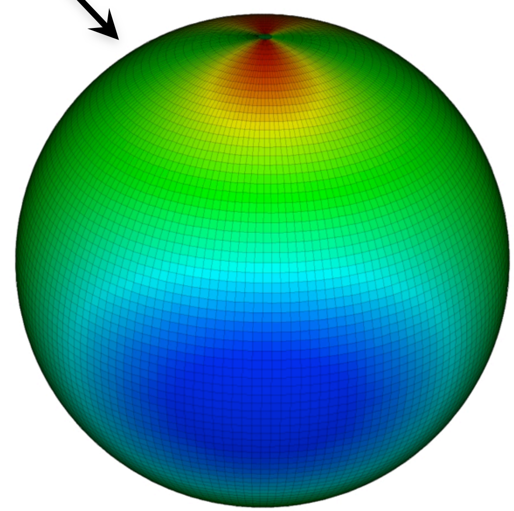


F = 2+cos(lon)^2 * cos(2*lat)



First-Order Conservative

Destinations:
•2 degree uniform global

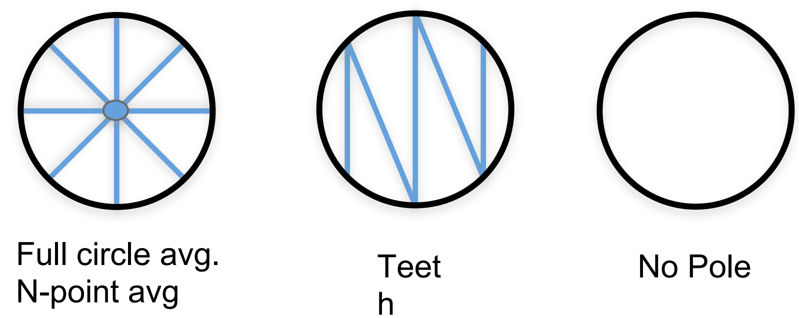


Higher-Order Conservative
(Preliminary)

Other Regrid Options

- Path between points in bilinear on a sphere:
  - Straight line
  - Great circle

- Options for extrapolating across pole region:
  - Full circle average
  - N-point average
  - Teeth
  - No pole



- Others:
  - Source and Destination Masking
  - Information about what happened to each destination location during regridding (e.g. outside source grid, masked, etc.)
  - User area
  - Ignore unmapped, Ignore degenerate

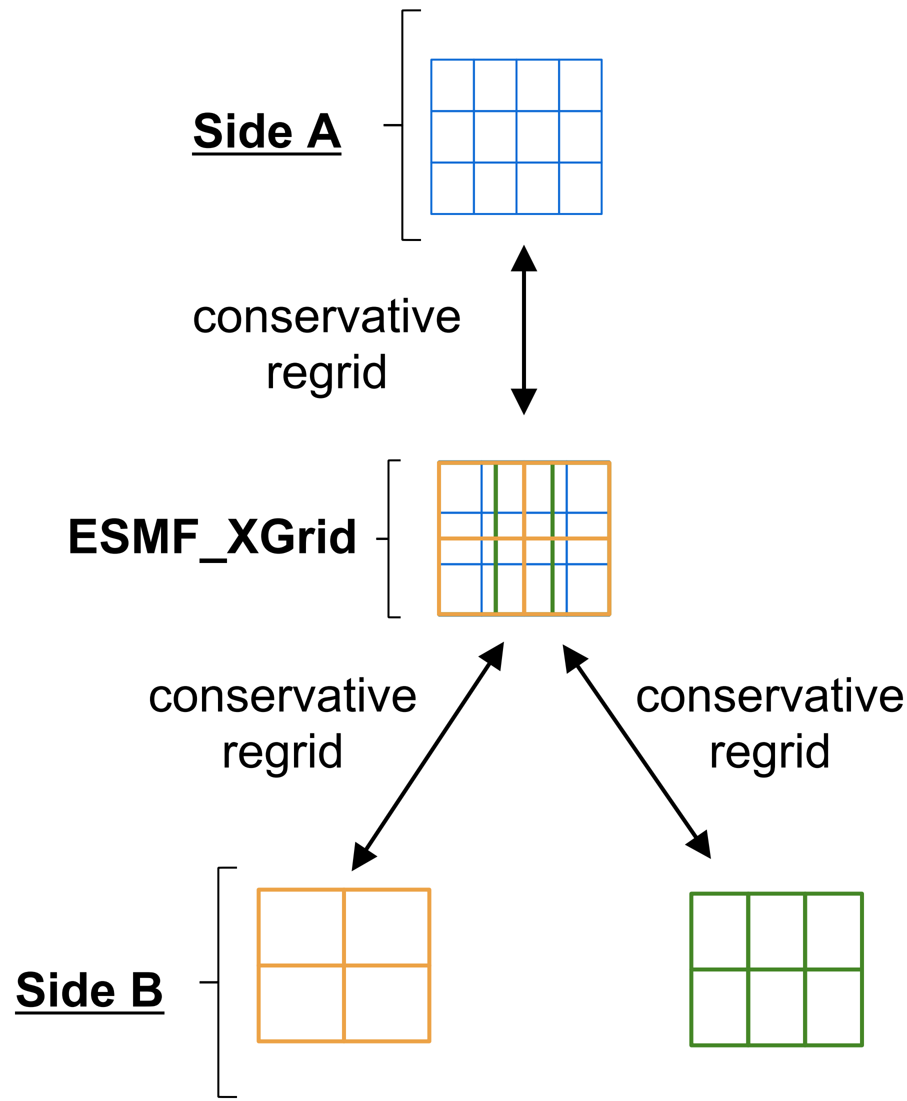
Exchange Grid

Modeled on GFDL Exchange Grid (V. Balaji et al. [3])

ESMF_XGrid **generated from two sets of source grids/meshes** (sideA & sideB).
-**merge** process in which higher priority grids clip into lower priority grids
-**masks** are respected

XGrid supports **first-order conservative regridding** - will support higher-order conservative method next release

ESMF_Field is constructed on the XGrid mesh and used as source/ destination of regridding operations.



Exchange Grid Create API

ESMF_XGridCreate(sideAGrid, sideAMesh, sideBGrid, sideBMesh, & …, sideAMaskValues, sideBMaskValues, …, rc)

Create an exchange grid between a set of Grids and Meshes on one side with a set of Grids and Meshes on another. Once the exchange grid has been created data can be interpolated between the two sides and the center using ESMF_FieldRegridStore().

[sideAGrid] - List of Grids on side A of the exchange grid.

[sideAMesh] – List of Meshes on side A of the exchange grid.

[sideBGrid] - List of Grids on side B of the exchange grid.

[sideBMesh] – List of Meshes on side B of the exchange grid.

[sideAMaskValues] – List of values which indicates a cell should be masked on side A.

[sideBMaskValues] – List of values which indicates a cell should be masked on side B.

```
! Create a Mesh for side A
atmMesh=ESMF_MeshCreateCubedSphere(…)

! Create two Grids for side B
landGrid=ESMF_GridCreate1PeriDim(…)
oceanGrid=ESMF_GridCreate1PeriDim(…)

! Create an exchange grid between the atmosphere Mesh and both land and ocean Grids.
! Mask out any cells in the land and ocean grids with a mask value of 1.
xgrid = ESMF_XGridCreate(sideAMesh=(/atmmesh), sideBGrid=(/landGrid, oceanGrid/),
                         sideBMaskValues=(/1/), rc=rc)
```

Regridding Using an Exchange Grid

```
! Start with atmMesh, landGrid, and oceanGrid

! Create exchange grid
xgrid = ESMF_XGridCreate(sideAMesh=(/atmmesh), sideBGrid=(/landGrid, oceanGrid/),  ….)

! Create Fields to hold data
atmField = ESMF_FieldCreate(atmMesh,...)
landField = ESMF_FieldCreate(landGrid,...)
oceanField = ESMF_FieldCreate(oceanGrid,...)
xField = ESMF_FieldCreate(xgrid,...)

! compute regrid weight matrix  from atm to xgrid
call ESMF_FieldRegridStore(xgrid, atmField, xField, a2x_rhandle, ...)

! compute regrid weight matrix  from  xgrid to ocean
call ESMF_FieldRegridStore(xgrid, xField, oceanField,  x2o_rhandle, ...)

! loop over time moving data from atm to ocean through xgrid
do t=1,…
          ! compute new atmField

     ! apply regrid weight matrix moving data from atm to xgrid
          call ESMF_FieldRegrid(atmField, xField, a2x_rhandle,  ...)

     ! apply regrid weight matrix moving data from xgrid to ocean
          call ESMF_FieldRegrid(xField, oceanField, x2o_rhandle,  ...)

enddo
```

Scheduled for Next Release

- Cubed sphere creation interfaces (7.1.0) ← already working
- Higher-order conservative regridding (7.1.0) ← in progress
- Extrapolation of points that lie outside the source grid (7.1.0)
- Dynamic masking during sparse matrix multiply (7.1.0)

References

1. Khoei S.A., Gharehbaghi A. R. The superconvergent patch recovery technique and data transfer operators in 3d plasticity problems. *Finite Elements in Analysis and Design*, 43(8), 2007.
2. Hung K.C, Gu H., Zong Z. A modified superconvergent patch recovery method and its application to large deformation problems. *Finite Elements in Analysis and Design*, 40(5-6), 2004.
3. Balaji, V., Anderson, J., Held, I. Winton, M. Malyshev, S., Souffer, R. The FMS Exchange Grid: a mechanism for data exchange between Earth System components on independent grids. 2007.

If you have questions or requests, come talk to me, or email: esmf_support@list.woc.noaa.gov

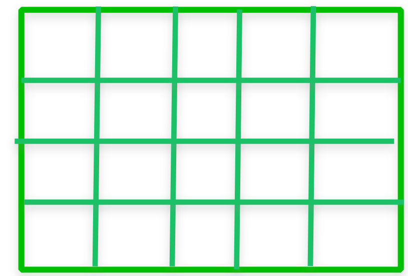
End of Presentation

F95 Regridding Example

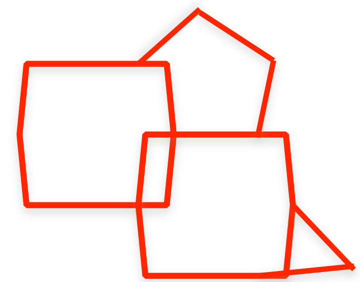
```
! Create Geometry Classes
srcGrid=  ESMF_GridCreateCubedSphere(…)
dstMesh=ESMF_MeshCreate(…)

! Create Fields
srcField=ESMF_FieldCreate(srcGrid, …)
dstField=ESMF_FieldCreate(dstMesh, …)

! Calc regrid sparse matrix (routeHandle)
ESMF_FieldRegridStore(srcField, dstField, …routeHandle, …)

! Loop applying regrid sparse matrix (routeHandle) whenever source data changes
do i=1,…
        ! Compute new srcField
               ….

        ! Apply regrid sparse matrix (routeHandle)
               ESMF_FieldRegrid(srcField, dstField, …routeHandle, …)

       ! dstField contains regridded data here
enddo
```

Supported Geometry Types

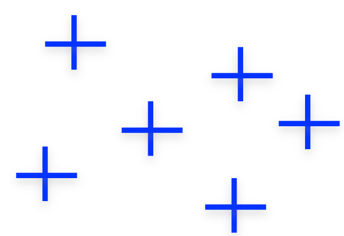
- Grid:
  - Structured representation of a region
  - Consists of one or more logically rectangular tiles (e.g. a uniform global grid or a cubed sphere grid)



- Mesh:
  - Unstructured representation of a region
  - In 2D: polygons with any number of sides
    - A single mesh cell can consist of multiple pieces (e.g. Hawaii)
  - In 3D: tetrahedrons & hexahedrons



- LocStream (Location Stream):
  - Set of disconnected points
    - E.g. locations of observations
  - Very flexible and efficient
  - Can't be used with every regrid method



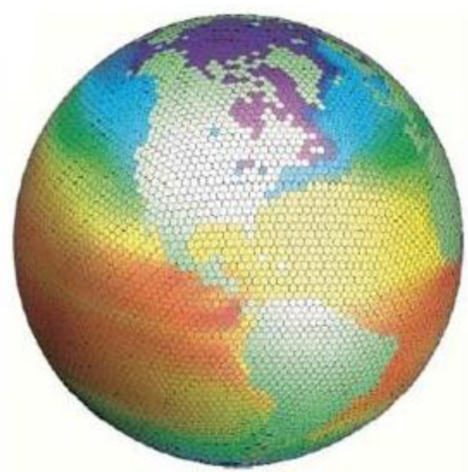
Spherical Regrid Support

- Regrid works with spherical (lon, lat, radius) coordinates
- All regrid methods supported between any pair of:
  - 2D Global or 2D regional logically rectangular Grids
  - 2D Unstructured Meshes composed of polygons with any number of sides
  - 2D Multi-tile grids (e.g. cubed spheres)

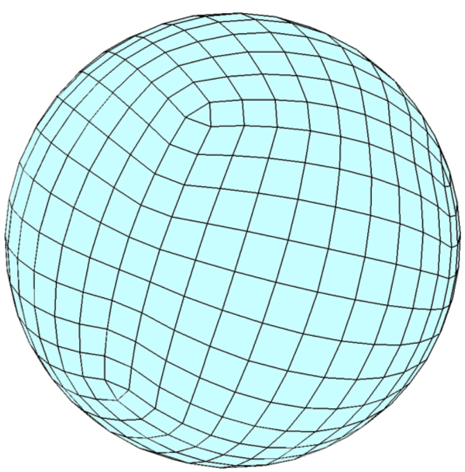
- Bilinear supported between any pair of:
  - 3D Meshes composed of hexahedrons
  - 3D Global or regional logically rectangular Grids

- LocStreams supported for above depending on regrid method

3D Global Spherical Grid



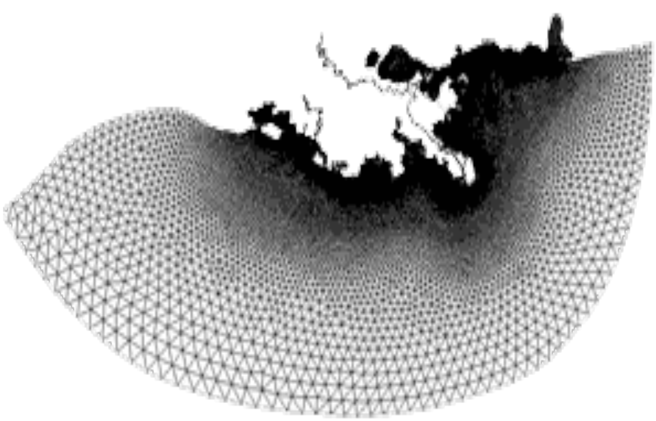
Unstructured Grid



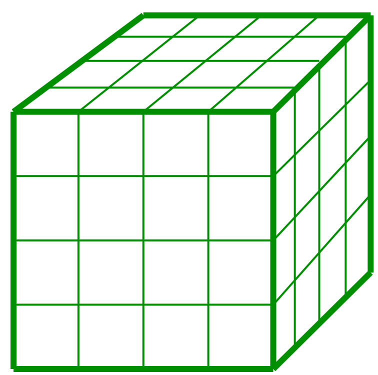
Multi-tile Grid

Cartesian Regrid Support

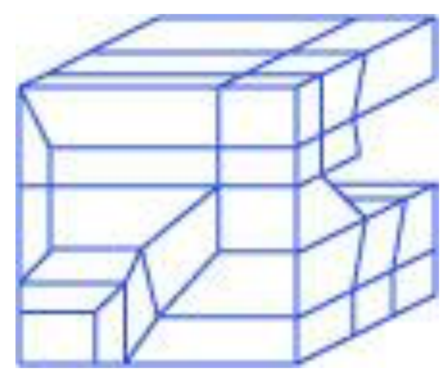
- Regrid works with Cartesian (x,y,z) coordinates
- All regrid methods between any pair of:
  - 2D Meshes composed of polygons with any number of sides
  - 2D logically rectangular Grids
- Bilinear, conservative, or nearest neighbor between any pair of:
  - 3D Meshes composed of hexahedrons
  - 3D logically rectangular Grids
- LocStreams supported for above depending on regrid method



2D Unstructured Mesh
From www.ngdc.noaa.gov



3D Grid



3D Unstructured Mesh

Interfaces

- Complete F95 API:
  - use ESMF
  - Derived types and methods
  - Investigating moving to Fortran 2003
- C API:
  - #include “ESMC.h”
  - Structs and methods
- Python API:
  - Import ESMPy
  - Classes with methods
- Applications:
  - File-based regrid weight generation:
    mpirun –np <N> ESMF_RegridWeightGen –s ….
  - File-based weight generation AND application of weights:
    mpirun –np <N> ESMF_Regrid –s…

Regridding Application Examples

- Regrid weight generation:

```
mpirun –np 16 ESMF_RegridWeightGen -s src_grid_file.nc –d dst_grid_file.nc
                                   -m regrid method  … Other options ….
                                   -w weight_file.nc
```

  - src_grid_file.nc – file describing source grid
  - dst_grid_file.nc – file describing destination grid
  - regrid_method – the regrid method used to calculate the weights
  - weight_file.nc – after running contains the regrid sparse matrix

- Regridding data between variables in two files:

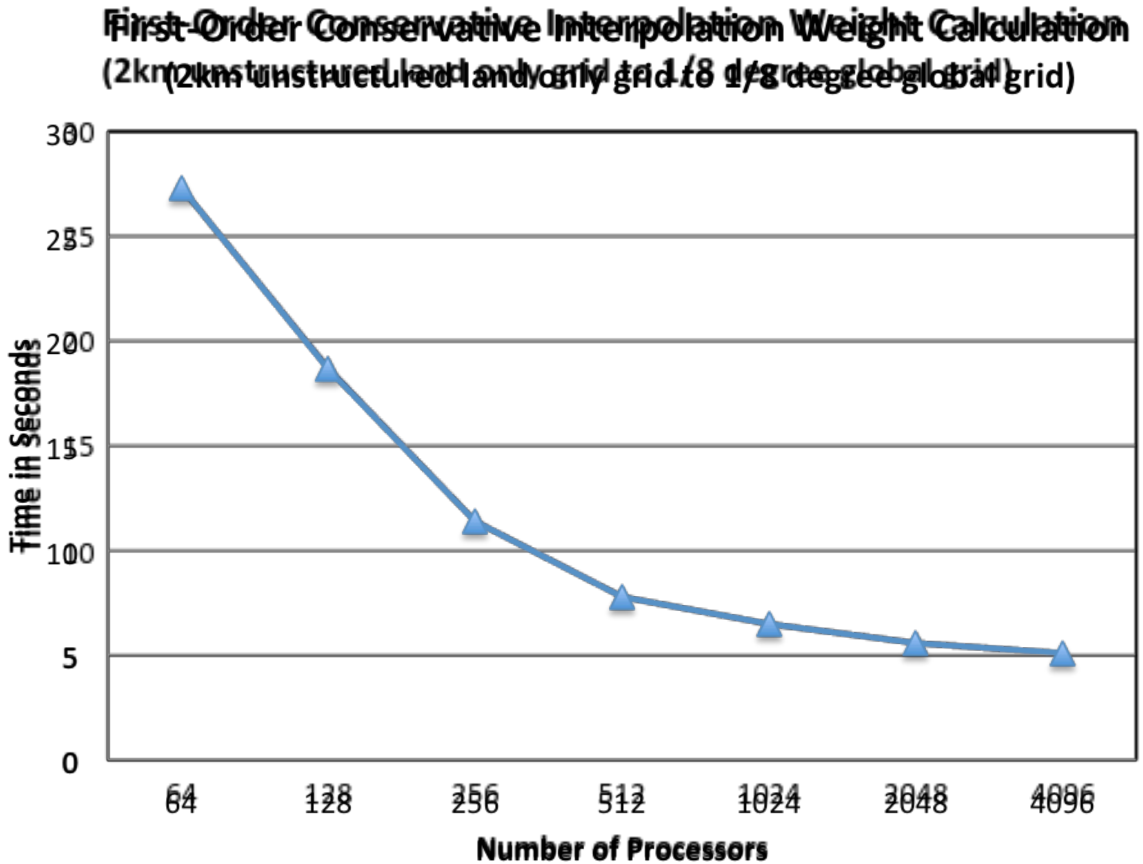
```
mpirun –np 16 ESMF_Regrid –s src_file.nc –d dst_file.nc
                          -m regrid method  … Other options ….
```

  - src_file.nc – file containing source grid and data
  - dst_file.nc - file containing destination grid
  - regrid_method – the regrid method to use

Supported Grid File Formats

- SCRIP:
  - Format used by SCRIP regridding tool
  - 2D spherical logically rectangular Grids or unstructured Meshes
- ESMF unstructured:
  - Custom ESMF format
  - 2D or 3D / spherical or Cartesian unstructured Meshes
- UGRID:
  - Proposed CF convention
  - 2D or 3D / spherical or Cartesian unstructured Meshes
- CF Grid:
  - CF convention
  - 2D spherical logically rectangular Grid
- GRIDSPEC mosaic:
  - Format from GFDL
  - 2D spherical set of logically rectangular tiles with connections between them

Regrid Weight Calculation Performance



Platform: IBM IDataPlex cluster (Yellowstone at NCAR)
Grid size: ~30 million cells and ~4 millions cells

Other Tools Using ESMF Regrid

- Ultrascale Visualization Climate Data Analysis Tool (UV-CDAT):
  - Package designed for analyzing large climate data sets
  - Uses ESMF regridding via ESMPy
  - Won Federal Laboratory Consortium technology transfer award

- Cf-python:
  - Python package for manipulating cf data and files
  - Uses ESMF regridding via ESMPy

- NCAR Command Language (NCL):
  - Language for scientific data analysis and visualization
  - Uses ESMF regridding via ESMF_RegridWeightGen application