

ESMF Regridding Update

CW2020

September 21, 2020

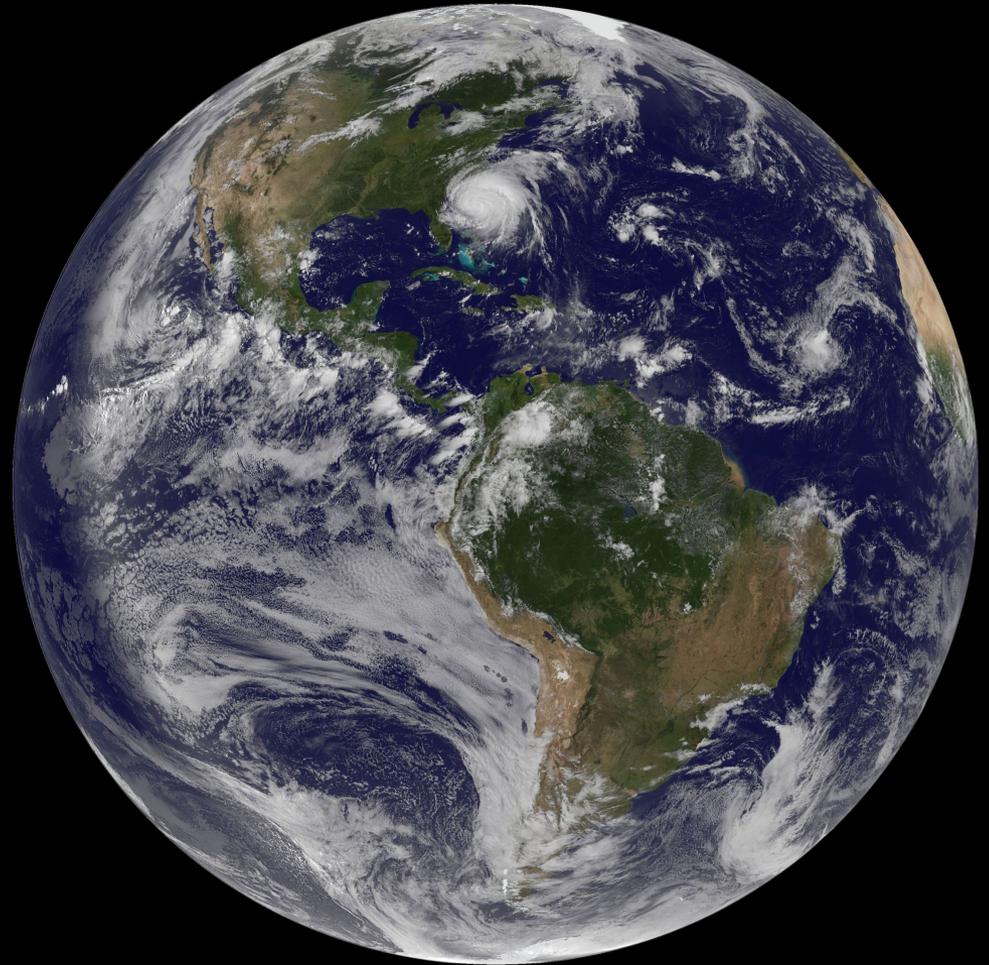
Robert Oehmke NCAR

Ryan O’Kuinghttons NCAR

Peggy Li NASA

Gerhard Theurich NRL

Ben Koziol NCAR



Hurricane Irene/NASA GOES-13 satellite image/August 26, 2011

Topics

- Brief overview of ESMF Regridding
- Updates on:
 - Extrapolation
 - New optimization capabilities useful for regridding
 - Other new regridding features
 - Integration of Mesh-Oriented datABase (MOAB) mesh library
 - Bit-for-bit testing
 - Upcoming features in next release (8.1.0)

This talk describes ESMF as of tag: `ESMF_8_1_0_beta_snapshot_27`

ESMF Regridding Overview

Fast, flexible, regridding engine with many options



High-performance

Interpolation weight matrix is generated in parallel in 3D space and applied in parallel

Options

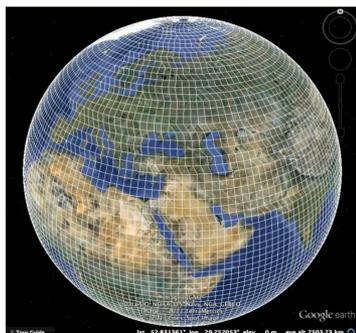
Masking, multiple pole treatments, straight or great circle distance measure, extrapolation

Wide range of supported grids

Logically rectangular connected tiles, unstructured meshes, observational data streams (point cloud), 2D and 3D, global and regional grids, Cartesian and spherical coordinates

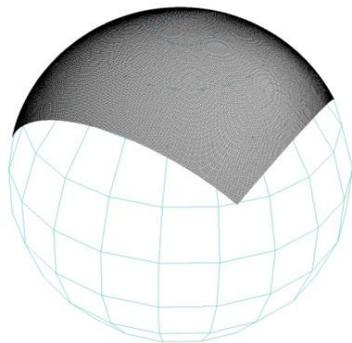
Multiple interfaces

- **Fortran API** - generate and apply weights during a model run
- **Python API** - generate and apply weights using ESMPy
- **Standalone tools** - generate and apply weights from grid files using ESMF command line utilities



Multiple interpolation methods

Bilinear, higher-order patch recovery[1][2], first-order conservative, nearest neighbor, second-order conservative[3]



<https://www.earthsystemcog.org/projects/esmf/>

Extrapolation

- Fills destination locations that aren't filled by an initial regridding
- **Important because:**
 - Having a gap is common due to mismatched masking, grid coverage, etc.
 - Before users would need to do a lot of work to implement their own solution
 - More efficient because it reuses parts of the regridding setup
 - Extrap. weights are incorporated into regridding weights, so there isn't a change in user's typical regridding flow. Makes it easy to experiment with
- Four types:
 - **Nearest neighbor** - closest point is used. **Useful when jumping across mesh gaps**
 - **Inverse dist. weighted avg.** - blend of N closest points. **Useful for smoother results**
 - **Creep fill** - destination data is spread a given number of neighbor levels into unmapped areas through the structure of the mesh. **Useful for flowing around obstacles**
 - **Creep fill plus nearest destination** - like creep fill, but then any remaining unmapped points are filled by mapping to closest filled location (either regridded or creeped)

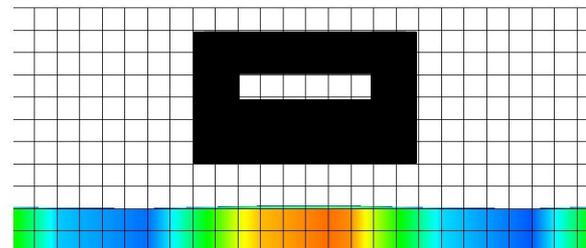
Extrapolation Examples



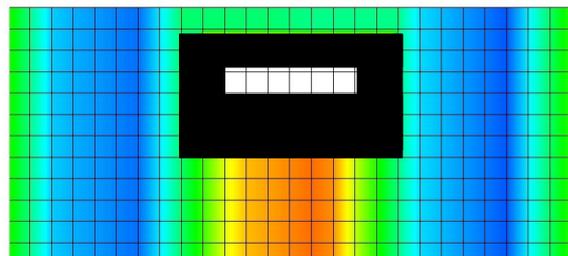
Masked Area



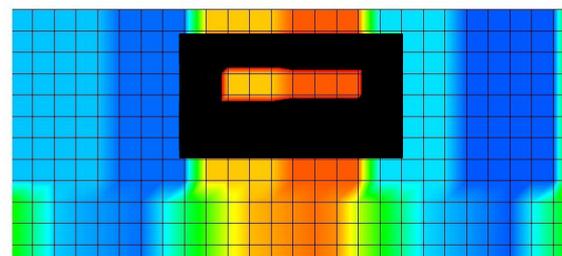
Source Grid and Field



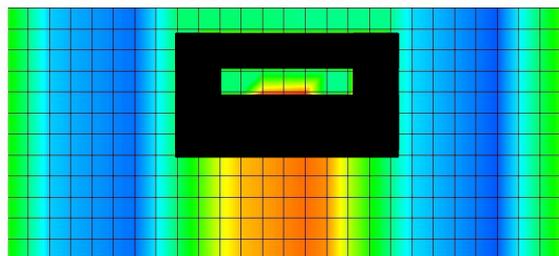
Dest: Without Extrapolation



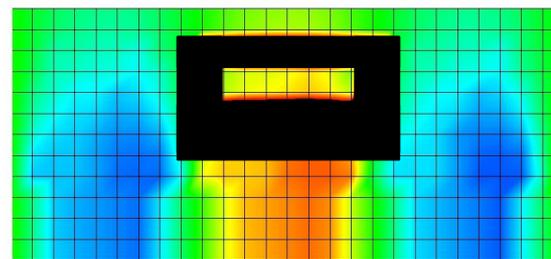
Dest: Creep Fill



Dest: Nearest Neighbor



Dest: Creep Fill with Nearest



Dest: Inverse Dist. Weighted Avg.

New Regridding Related Optimization Capabilities

- RouteHandle* I/O:
 - Routehandles can be written/read from a file to improve initialization speeds
 - This can be much faster than even reading precomputing regridding weights
 - Only works when routeHandle is written/read for the same number of procs/distribution
 - **Useful** for cases like short production runs where the same distribution is used repeatedly and initialization time can be a large part of the total run time

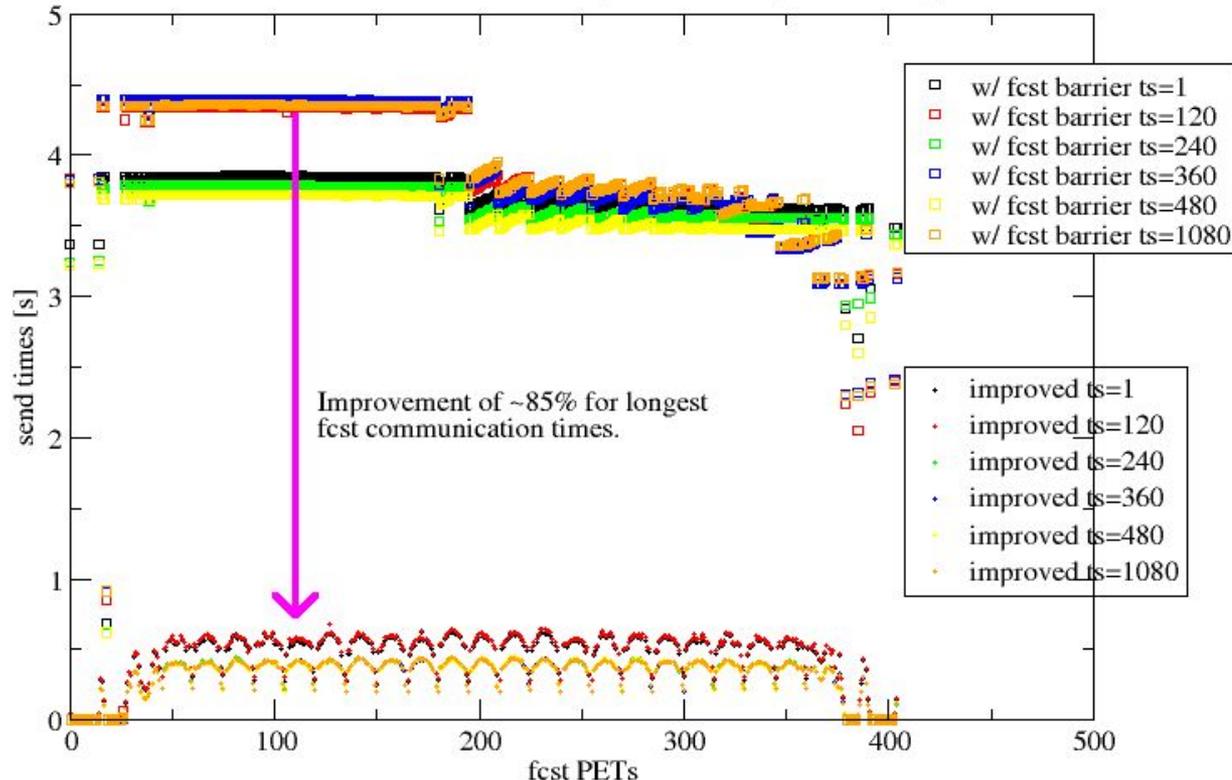
- VM Epochs:
 - Improve communication speed by allowing the user to tell ESMF to internally collect a set of routeHandle communication (e.g. ESMF_FieldRegrid()) and send them in one piece
 - Works for any “wild mix of routeHandles” (e.g. different distributions, different operations,...)
 - However, currently only works when send/receive processors are disjoint
 - **Useful** to trade memory (extra buffer space) for time (faster communication speed)

* A **RouteHandle** is an ESMF class used to apply sparse matrix multiply operations (e.g. regridding). They contain precomputed information, etc. in order to make a repeated sparse matrix multiply as efficient as possible.

VM Epoch Example

FV3_CAP fcst->wrt component send time optimization

Improvement 1: ESMF 8.0.0 + consolidated messages for src-dst pairs and delayed waits on sends



- Sparse mat. mul. of 191 Fields (total levels 1196) from forecast component to write component
- Forecast component on 405 processors. Write component on 27 processors.
- Run on Hera at NOAA (Cray Compute Cluster/2.4 Ghz Intel SkyLake/2.4 GB per core)
- Compiled with -O2, intel/18.0.5.274, impi/2018.0.4

Other New Regridding Related Functionality

- Dynamic masking:
 - User can mask or otherwise alter what happens to dst. value during application of regridding
 - Routehandle / regridding weights stay the same
 - User sets function that is triggered for specific destination or source values, function can then alter the destination value
 - **Useful** for cases where usable source locations change frequently (e.g. ice)
- Destination status Field:
 - Gives an indication of what happened to each destination location during weight generation
 - Fills integer Field the same size as destination Field with flags indicating states like successfully mapped, source masked, destination masked, extrapolated, etc.
 - **Useful** for debugging and when a destination Field needs to be modified based on regridding behavior
- Full Mesh get capability:
 - Allows user to get full information about an ESMF Mesh
 - **Useful** for regridding because user can create a modified version of an input Mesh
 - For example, a recent case where a user needed to extend a 2D Mesh provided by one component to a 3D one in the mediator to allow regridding with a 3D field line grid

Integration of Mesh-Oriented datABase (MOAB)



- **Important because:** ESMF regridding is an **essential** capability for major modeling codes. **It must be:** maintainable, highly efficient/scalable for future grids, and anticipate new capabilities for future science requirements
- Underneath ESMF regridding is a custom built 3D finite element code to provide low level capabilities like representing points, cells, connections, etc.
- The goal is to replace that with a new externally developed code (MOAB) to:
 - Get new capabilities:
 - More efficient representation of some types of cells (e.g. >4 sides)
 - Representation of data on edges, higher-order elements, ...
 - More flexible/efficient internal fields for coords, masking, etc. ...
 - Get to share community benefits of an externally developed library (as others get to do with ESMF)
 - Get improved efficiency in some areas (e.g. memory use for large meshes)
- ESMF and MOAB are a good match. ESMF provides high level data structures and abstractions for model coupling, while using MOAB for some of its low level internal mesh representation needs.

Status of MOAB Integration



Currently finished capabilities:

- Can optionally create an ESMF_Mesh built on MOAB internally
- Can compute ESMF regridting weights using MOAB for conservative, bilinear, nearest-neighbor, using most options (e.g. masking), and most extrapolation.
- Reconciling a MOAB mesh (so it can be used in multi-component system)

To do:

- **Patch, 2nd order conservative, and creep fill extrapolation**
- More **testing**, and then even more testing...

User perspective on integration:

- MOAB is an internal capability, so **no user code changes** will be required
- MOAB builds automatically with ESMF, so **does not add another dependency**
- **Will give users/application groups a lot of time to verify correctness:**
 - ESMF 8.1.0 - MOAB off by default, but can be switched on for testing
 - ESMF 8.2.0 - MOAB on by default, but can be switched off
 - ESMF 8.3.0 - Native (non-MOAB) implementation no longer supported

Bit-For-Bit Testing



- **Important because:** Answer changes are a big deal for operational models, so we want to let them know when ESMF answers have changed and why
- Bit-for-bit changes are uncommon, but they may happen due to bug fixes or improvements (e.g. in accuracy)
- Our new testing watches for **any** change in regridding weights as development progresses
- The testing is done using regrid weight generation application:
 - Bit-for-bit checks between new weights and baseline versions
 - Run nightly on three platforms:
 - Cheyenne (NCAR), Discover (NASA), Hera (NOAA)
 - Comparisons done on 165 regrid cases (model grids/methods/options)
- Bit-for-bit changes during development are now tracked along with explanation/justification:
 - We hope to have a page with this information available to users soon
- Changes across a release are now reported as part of release output

Scheduled for Next Release (ESMF 8.1.0)

Anticipated March 2021

- Dynamic creep fill (requested by US Navy):
 - Fills all reachable locations without needing to specify num. levels
 - This is a significant improvement over current approach which requires a manual process to determine the creep-fill depth
- Improved regridding diagnostic/debugging output:
 - **Important because:** It saves time, by allowing users to find subtle errors quickly
 - Finding these kinds of errors can often hold up development for a long time
 - New parameter that allows user to easily turn on more expensive checks:
 - Folded grid, self intersecting cells, non unique mesh ids, etc...
 - Too expensive to run in operations, but useful for diagnosing problems
 - Goal is to give users more ability to diagnose regridding issues themselves
- Completed MOAB version of Mesh/regridding

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. Meurdesoif Y., Kritsikis E., Aechtner M., Dubos T. Conservative interpolation between general spherical meshes. *Geoscientific Model Development*, 10, 2017.

Thanks!

If you have questions or requests
ask me or email:

esmf_support@ucar.edu

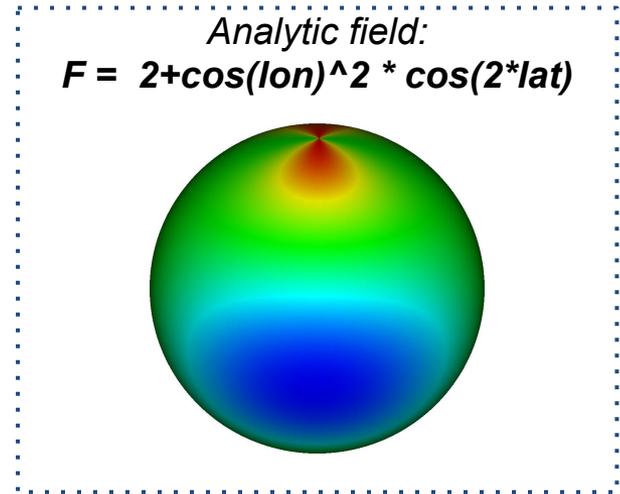
Extra Slides

Start of Extra Slides

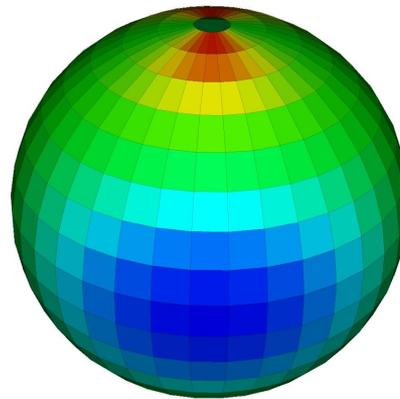
Second-Order Conservative

This regrid method preserves the integral of a field across regridding, but uses the source gradient to give smoother results than first order.

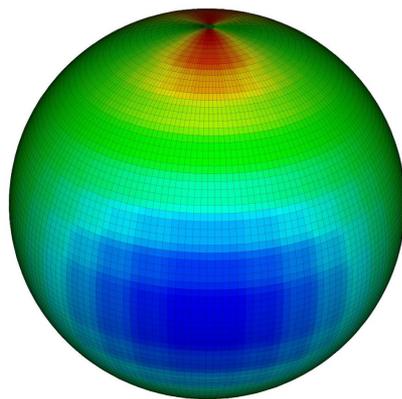
Important because: Gives smoother results for conservative regridding, NASA/GMAO requested this to regrid ocn/atm fluxes in GEOS.



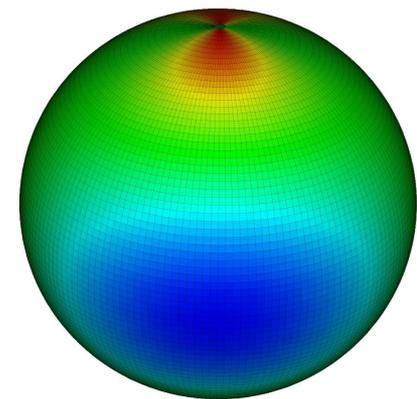
Source:
10 degree uniform global



Destinations:
2 degree uniform global



**First-Order
Conservative**



**Second-Order
Conservative**

Second-order Conservative Regrid Method

- Destination cell value is combination of values of intersecting source cells modified to take into account **source cell gradient**:

$$d = \sum (s_i + \nabla s_i \cdot (c_{si} - c_d))$$

Where:

d = destination value

s_i = intersecting source cell value

c_d = destination centroid

c_{si} = intersecting source cell centroid

∇s_i = intersecting 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)

Generation and Application of Regrid Weights

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
srcFld = ESMF_FieldCreate(srcGrid,...)
dstFld = ESMF_FieldCreate(dstGrid,...)

! compute regrid weight matrix
call ESMF_FieldRegridStore(srcFld, dstFld, routehandle, ...)

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

    ! compute new srcFld

    ! apply regrid weight matrix in parallel
    call ESMF_FieldRegrid(srcFld, dstFld, 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.

Python Interface

Access to parallel ESMF regridding from Python

A Python API to ESMF regridding and related classes

Transforms data from one grid to another by generating and applying remapping weights.

Supports structured and unstructured, global and regional, 2D and 3D grids, created from file or in memory, with many options.

Fully parallel and highly scalable.

Visit the ESMPy home page for user documentation and installation instructions:

<https://www.earthsystemcog.org/projects/esmpy/>

```
# create source and destination grid and mesh
grid = ESMF.Grid(filename=grid1,
filetype=ESMF.FileFormat.SCRIP, ...)
mesh = ESMF.Mesh(filename=grid2,
iletype=ESMF.FileFormat.ESMFESH, ...)

# create source and destination fields
srcfield = ESMF.Field(grid, name='srcfield', ...)
dstfield = ESMF.Field(mesh, name='dstfield', ...)

# create object to regrid data from the source to the
destination field
regrid = ESMF.Regrid(srcfield, dstfield,
regrid_method=ESMF.RegridMethod.CONSERVE,
unmapped_action=ESMF.UnmappedAction.IGNORE)

# Loop over time
for t in range[...]

    # Fill source field
    ...
```

File-based Regridding

Command line tools work on NetCDF files

ESMF_RegridWeightGen

Reads in two NetCDF grid files and outputs a NetCDF file containing interpolation weights.

Input formats: SCRIP, ESMFMESH, UGRID, GRIDSPEC

Output format: SCRIP weight file

Arguments for interp. method, pole treatment, regional grids, ignore unmapped points and degenerate cells.

```
$ mpirun -np 4 ./ESMF_RegridWeightGen
  -s src.nc \      # source grid file
  -d dst.nc \      # destination grid file
  -m conserve \   # interpolation method
  -w w.nc         # output weight file
```

ESMF_Regrid

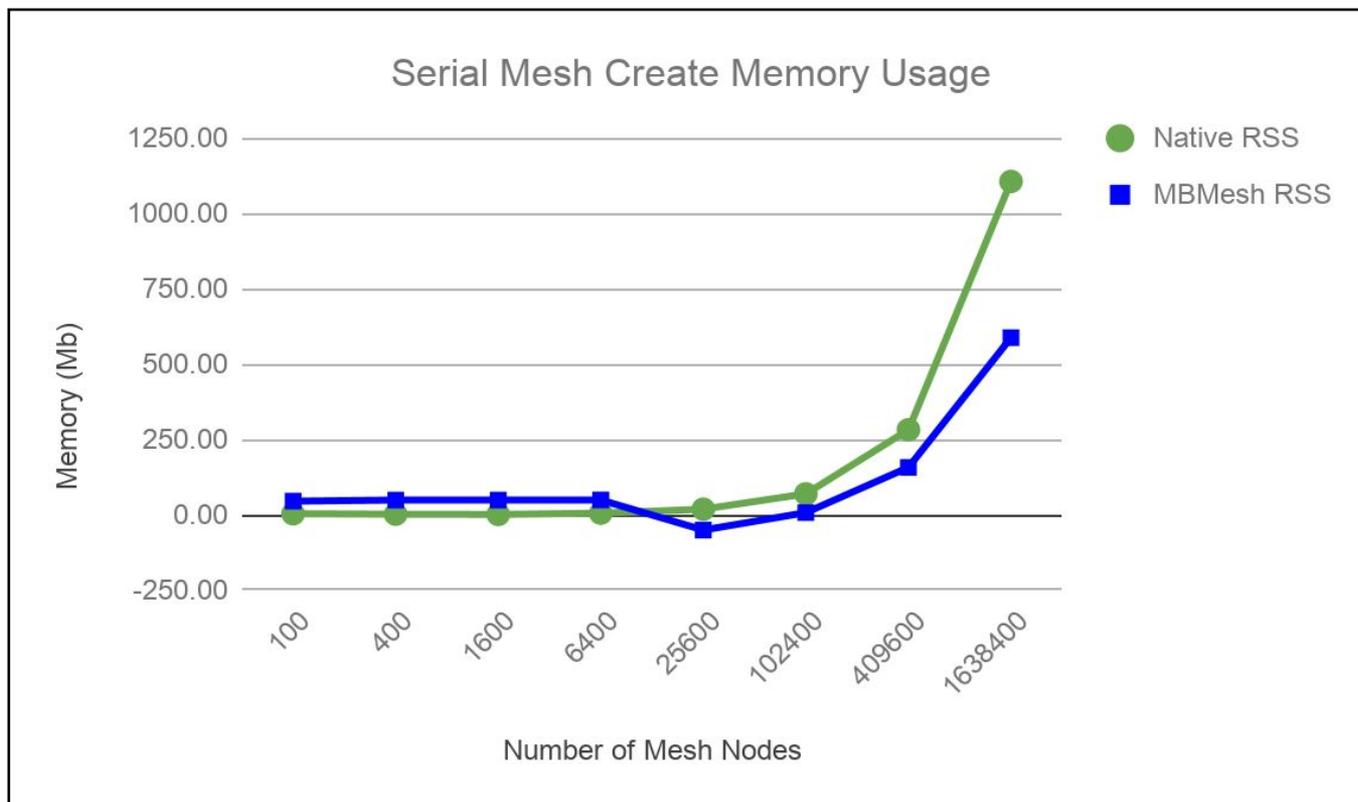
Generates and applies interpolation weights from a source to destination grid file.

Formats: GRIDSPEC for structured grids and UGRID for unstructured. Currently limited to 2D grids.

Arguments for interp. method, pole treatment, regional grids, ignore unmapped points and degenerate cells.

```
$ mpirun -np 4 ./ESMF_Regrid \
  -s simple_ugrid.nc \      # source file
  -d simple_gridspec.nc \   # dest. file
  --src_var zeta \         # source
variable
  --dst_var zeta          # dest. variable
```

MOAB vs. Native Mesh Memory Example



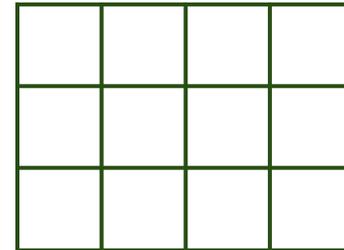
Memory Use for Square Latitude-longitude Grids

(Measured on Cheyenne (NCAR HPC platform) using MOAB 4.9.2 and ESMF version ESMF_8_0_0_betasnapshot_25. Both were compiled with Intel 17.0.1.)

Supported Geometries

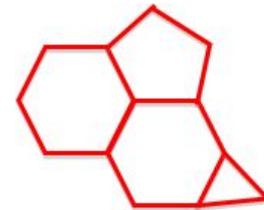
Grid

A **structured** representation of a region consisting of one or more logically rectangular tiles (e.g., a uniform global grid or a cubed sphere grid)



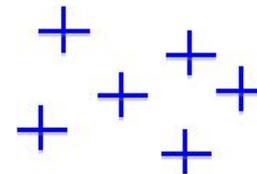
Mesh

An **unstructured** representation of a region including 2D polygons with any number of sides and 3D tetrahedra and hexahedra

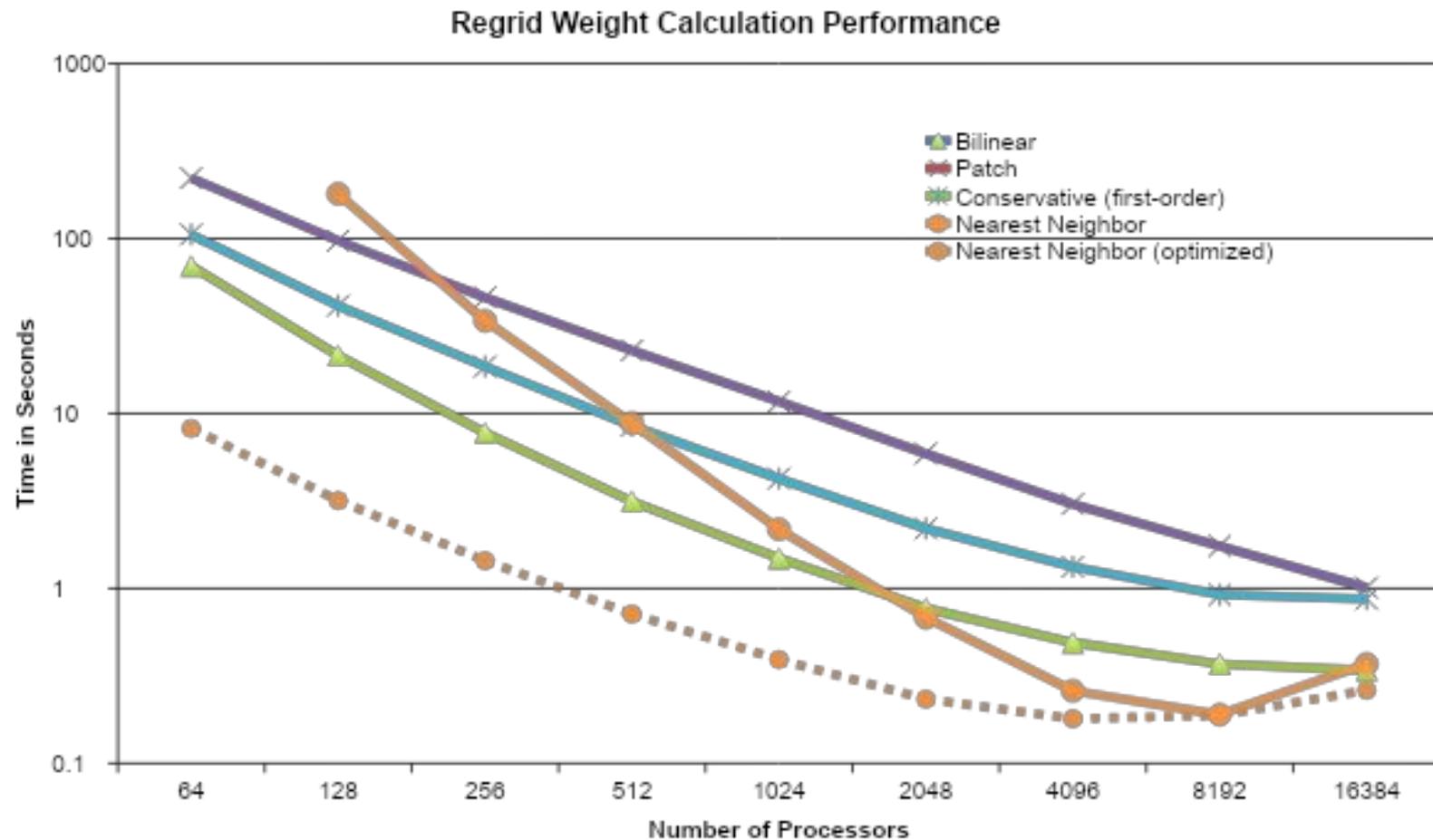


LocStream

A set of **disconnected points** such as locations of observations



Performance



Source: cubed sphere grid (~25 million cells)

Destination: uniform latitude longitude grid (~17 million cells)

Platform: IBM iDataPlex cluster (Yellowstone at NCAR)