Enhancing GIS Capabilities for High Resolution Earth Science Grids (IN24B-05)

Benjamin Koziol (ben.koziol@noaa.gov)\textsuperscript{1}, Robert Oehmke\textsuperscript{1}, Peggy Li\textsuperscript{2}, Ryan O’Kuinghttons\textsuperscript{3}, Gerhard Theurich\textsuperscript{4}, Cecelia DeLuca\textsuperscript{1}, Rocky Dunlap\textsuperscript{1}

\textsuperscript{1}NESII/CIRES/NOAA-ESRL, \textsuperscript{2}NASA-JPL, \textsuperscript{3}Cherokee Nation Technologies, \textsuperscript{4}Source Spring, Inc.

AGU December 2017
A Presentation in Two Parts

- **Part 1 - National Hydrography Dataset Conservative Regridding**
  - **Motivation:** Develop and profile a high performance regridding solution for complex, irregular meshes and fine resolution rectangular grids over large scales.
    - Data discussion
    - Implementation
    - Performance

- **Part 2 - Chunked Regridding**
  - **Motivation:** Develop software infrastructure for grid manipulations that scale to high spatial resolutions and accuracies while accommodating arbitrary compute environments and grid structures.
    - Implementation
    - Next Steps
Software Stack

Earth System Modeling Framework (ESMF)

C / Fortran
- NetCDF/HDF
- LAPACK
- PIO
- mpich / OpenMPI
- GEOS
- PROJ.4
- GDAL

OpenClimateGIS (OCGIS)

Python
- NumPy
- netCDF4-python
- mpi4py
- Shapely
- pyproj
- Fiona
- osgeo/osr
- nose
Part 1: National Hydrography Dataset
Conservative Regridding
Here’s the Catch-ments

- **Conservatively regrid** to hydrologic catchment polygons (unstructured) from an arbitrarily gridded field (structured)
  - High performance requirements
  - Reusability

- **Source Grid:** CONUS exact test field - 250-meter (~2e-3 degrees), three timesteps
  \[
  f(lat, lon) = 2 + \cos^2(rad(lat)) + \cos(2 \times rad(lon))
  \]

- **Destination Grid:** NHDPlus hydrologic catchments for CONUS [1]
  - 7.7 GB of ESRI-based vector data
  - 2,647,454 Mesh Elements
  - 485,638,947 Nodes

Structure of an Element

- **Holes/Interiors** → Split geometry on interior center and use multi-geometry implementation
- **Multi-Geometries** (Multiple Geometry Parts Counted as One Unique Feature) → Create weights for geometry parts then normalize for entire geometry
- **High Node Count** → Split geometries based on node counts and use multi-geometry regridding
Results and Performance

Conservative regridding result with CONUS NHDPlus catchments overlaid on analytical source field.

- **OCGIS Format Conversion** - 8 cores, minutes
- **ESMF Regridding**
  - Yellowstone Supercomputer - 512 cores, ~17 mins for weight generation
  - Weight application - ~17 seconds initialization, ~0.01 seconds SMM
- **Root-Mean-Square Error**: $1.710e-3$
- **Normalized Root-Mean-Square Error**: 0.2 %
Part 2: Chunked Regridding
Motivation for Chunked Approach

- ESMF memory requirement for weight calculation and sparse matrix multiplication increases linearly with \textit{factor count}.
- As grid resolution and complexity increases, factor counts will increase in-step resulting in potentially \textit{outsized memory requirements} for regridding operations.
- Provided \textit{spatial mapping} may be maintained for weight calculation, source and destination grids may be chunked (split).
  - "Maintaining spatial mapping" implies spatial relationships are indistinguishable inside the local chunks from the spatial relationships present in their global parent grids.
- An \textit{iterative, offline approach} to weight generation and sparse matrix multiplication lifts grid resolution limitations on machine memory at the expense of computational time and ease of use.
- Ultimate goal is to wrap index-based decompositions within a \textit{spatial decomposition} framework.
- Increasing the \textit{number of processors}, and hence increasing the available total memory, will not always be a feasible solution.
Red = Destination grid slices
Green = Example buffered bounding box used to subset source grid
Development Pathway

- Integrate chunked regridding into the `ESMF_RegridWeightGen` CLI
- Add spatial decomposition capability to the core ESMF library and expose in the ESMF Python interface ESMPy
- “Out-of-core” memory paradigm (lazy evaluation) is a useful analog and there is considerable ongoing work in the Python community on how to address it (biggus, dask, lama/cf-python)
- ESMF Homepage: [https://www.earthsystemcog.org/projects/esmf/](https://www.earthsystemcog.org/projects/esmf/)
- OCGIS Homepage: [https://www.earthsystemcog.org/projects/openclimategis/](https://www.earthsystemcog.org/projects/openclimategis/)
- Chunked Regridding Demo: [https://sourceforge.net/p/esmf/external_demos/ci/master/tree/ESMF_FileRegridWFDemo/](https://sourceforge.net/p/esmf/external_demos/ci/master/tree/ESMF_FileRegridWFDemo/)
- Suggestions?
- Please email [esmf_support@list.woc.noaa.gov](mailto:esmf_support@list.woc.noaa.gov) if you’d like to try these workflows or have any questions
Backup Slide(s)
Workflow 1

NHD ESRI File Geodatabase → OpenClimateGIS (OCGIS) → Create Exact Test Field (Source Grid) → Prepare Geometries → Convert to Spherical Coordinate System → Convert to UGRID w/ Ragged Arrays → Convert to ESMF Unstructured Format (Destination Grid) → Earth System Modeling Framework (ESMF) → Weight Generation (ESMF_RegridWeightGen) → Weight Application / Sparse Matrix Multiplication (ESMF_ArraySMM) → OpenClimateGIS (OCGIS) → Error Evaluation → Convert Back to ESRI Formats
Workflow 2

MPI Parallelism + Asynchronous I/O

ESMF_RegridWeightGen

- Source Chunk (1...i)
- Destination Chunk (1...i)
- ESMF Recursive Weight Generation
- Weight File Chunk (1...i)
- Optional ESMF Sparse Matrix Multiplication
- Optional OCGIS Global Weight File Creation (Serial)
How to store?

- Unstructured data stores use **coordinate indexing (indirection)**
  
  ```
  node_connectivity = 0 1 2 3 4 2
  coordinate_value = 10. 20. 30. 40. 50.
  ```

- **Multi-geometry breaks** may be included using a standard flag (may also be used for interiors)
  - Minimize indirection and indexing to accommodate parallelism

  ```
  node_connectivity = 0 1 2 -8 3 4 2
  ```

- This approach used the ESMF Unstructured Format as opposed to UGRID. ESMF format uses a **vector** for coordinate indirection where UGRID uses a **rectangular array**.

  Masked, empty space in rectangular array

- Node thresholding, interior splitting, and multi-geometry flagging done in data conversion step
Subsetting / Spatial Decomposition Challenges

- Maintain **independent spatial mask** - is it a masked value because of data issues or spatial position?
- Coordinate indices must be **re-indexed** to persist a spatial subset - requires a flurry of communication in parallel
- **Label-based slicing and mask cascades** is critical → Once a spatial slice and associated mask is created this must be applied across all subset targets with shared dimensions
- Delayed loading of "**payload**" data greatly increases performance and lower memory usage
- Difficult to calculate **memory requirements** except in very controlled conditions
- MPI → More difficult to implement but offers the **necessary communication solution** for distributed slicing (slicing-in-parallel), re-indexing, and asynchronous IO