Parallel Infrastructure in MOAB for Climate Data

Iulian Grindeanu, Danqing Wu, Robert Jacob

Mathematics and Computer Science Division

Argonne National Laboratory

19th Annual CESM Workshop

Community Earth System Model CESM

Outline

- What is MOAB?
- Parallel framework in MOAB
- Climate-driven features
  - Structured interface
  - Climate data readers/writers
  - Distributed parallel intersection
- Examples

MOAB : Mesh Oriented datABase

- Lightweight C++ library, used to represent and evaluate mesh data
- Functional interface is simple, yet powerful
- Based on access to mesh in chunks rather than on individual entities: *array based representation*
- Data model consists of four fundamental types:

  mesh interface instance

  mesh entities : vertex, edge, quad, polygon, hexahedron, ...

  Sets : arbitrary groupings of mesh entities and other sets

  - Support also parent-child relation between sets
  - Partition sets for distributing meshes
  - Topological relations between different entities
  - Different search trees are implemented using sets

  Tags : named data which can be assigned to the mesh as a whole, to individual entities, or to sets

  - Field data represented with dense tags
  - Attributes represented with sparse tags

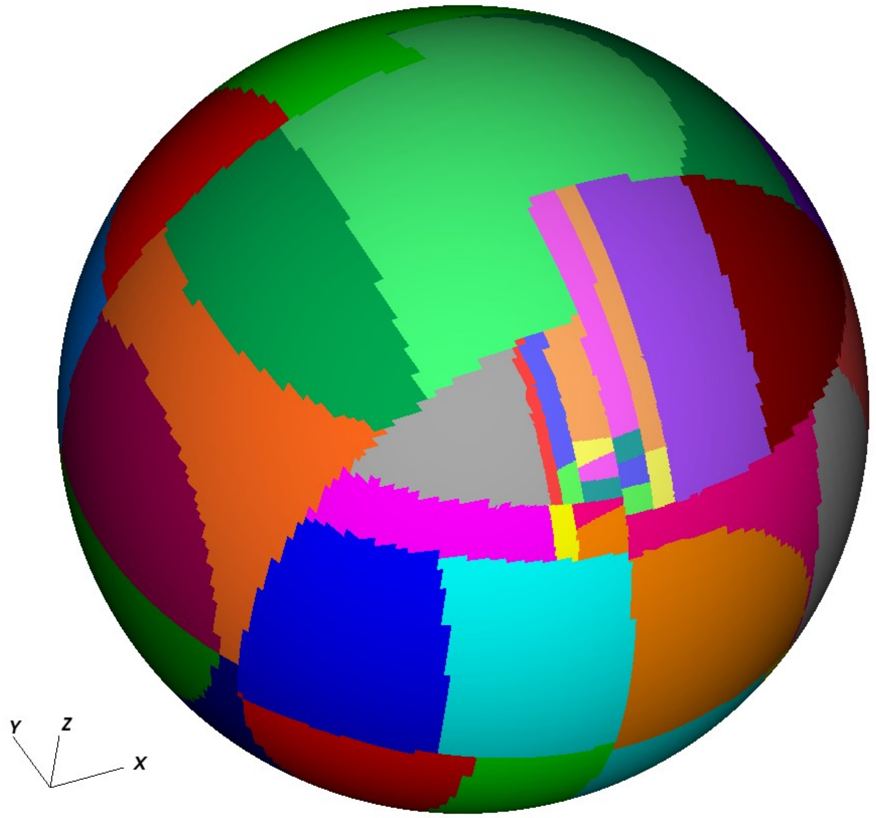
How to use MOAB?

MOAB can be used in several ways in applications:

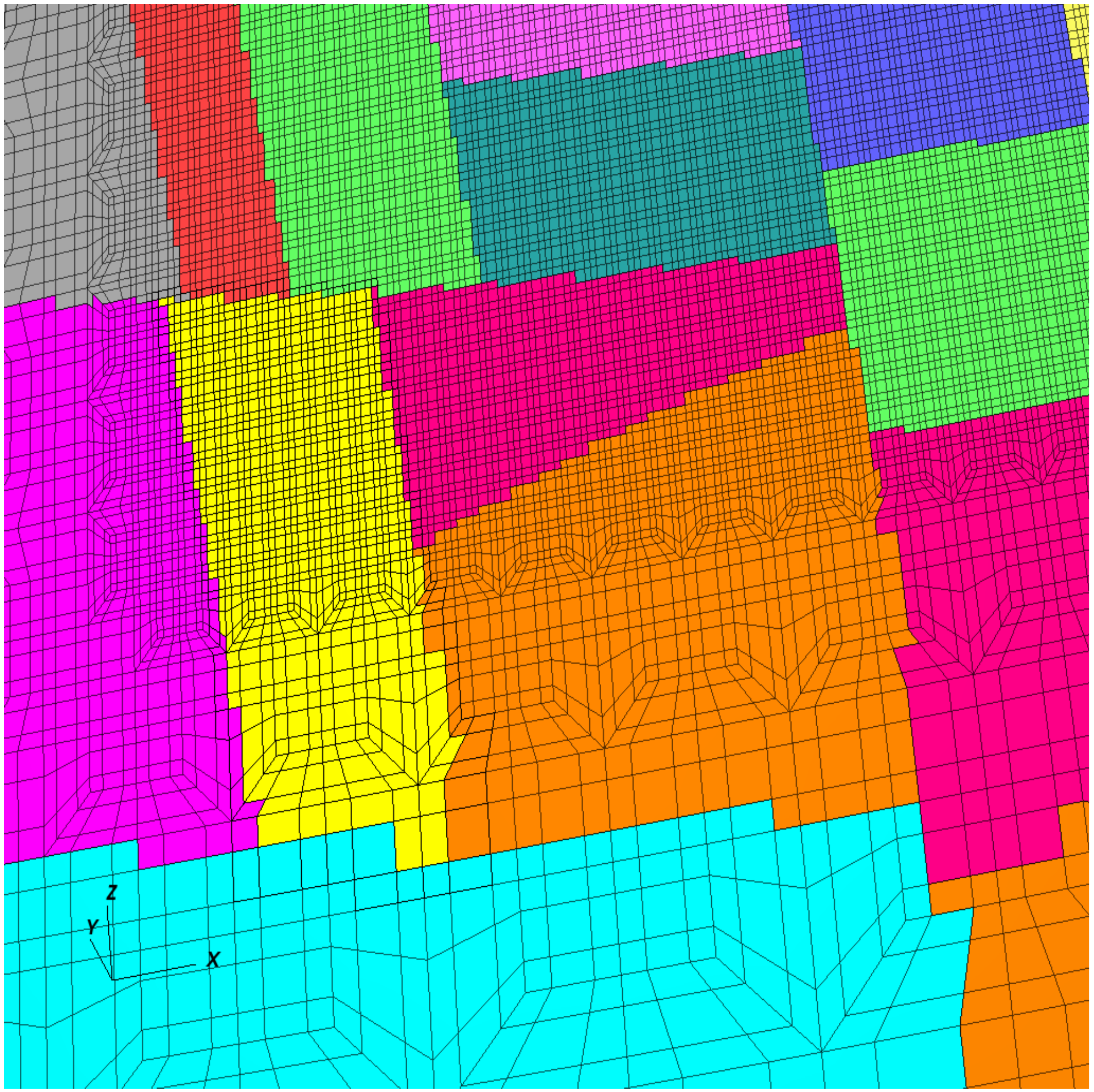
1) Underlying mesh data structure : linked directly to the scientific code
   - Pargal/Parvis, MeshKit/RGG, DAGMC, ...
2) Mesh + metadata translator : various mesh readers/writers
   - MOAB is configurable with HDF5, NETCDF, PNETCDF, CGNS, CCMIO, VTK
3) Coupling between different physics : mbcoupler tool
   - Nek5000 + Proteus + Diablo coupling
   - COUPE uses mbcoupler for non-intrusive coupling

MOAB Parallel Model

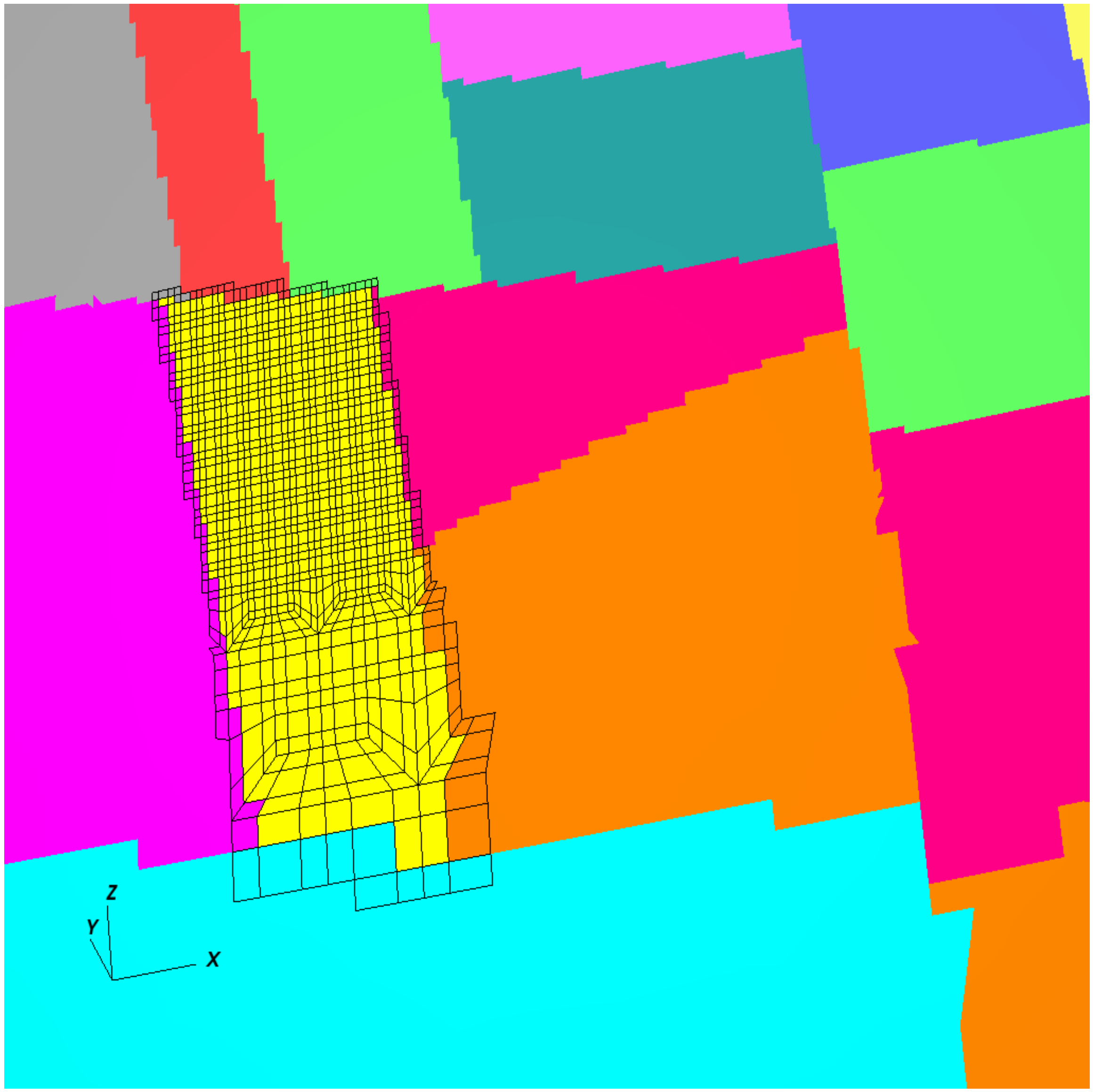
- Each processor has one MOAB instance, which appears locally as a serial mesh
- All serial MOAB function calls available locally
- Parallel model depends on element-based partition
- Neighbors communicate
- Each element assigned to exactly one part, with vertices shared between parts
- Parallel model usually initialized by loading from some decompositions in file
- Material set, geometric volume, trivial or Zoltan-generated partitioning
- Ghosting is supported
- Once the parallel model is resolved, the user can use “exchange_tags” for communication



MOAB Parallel Model

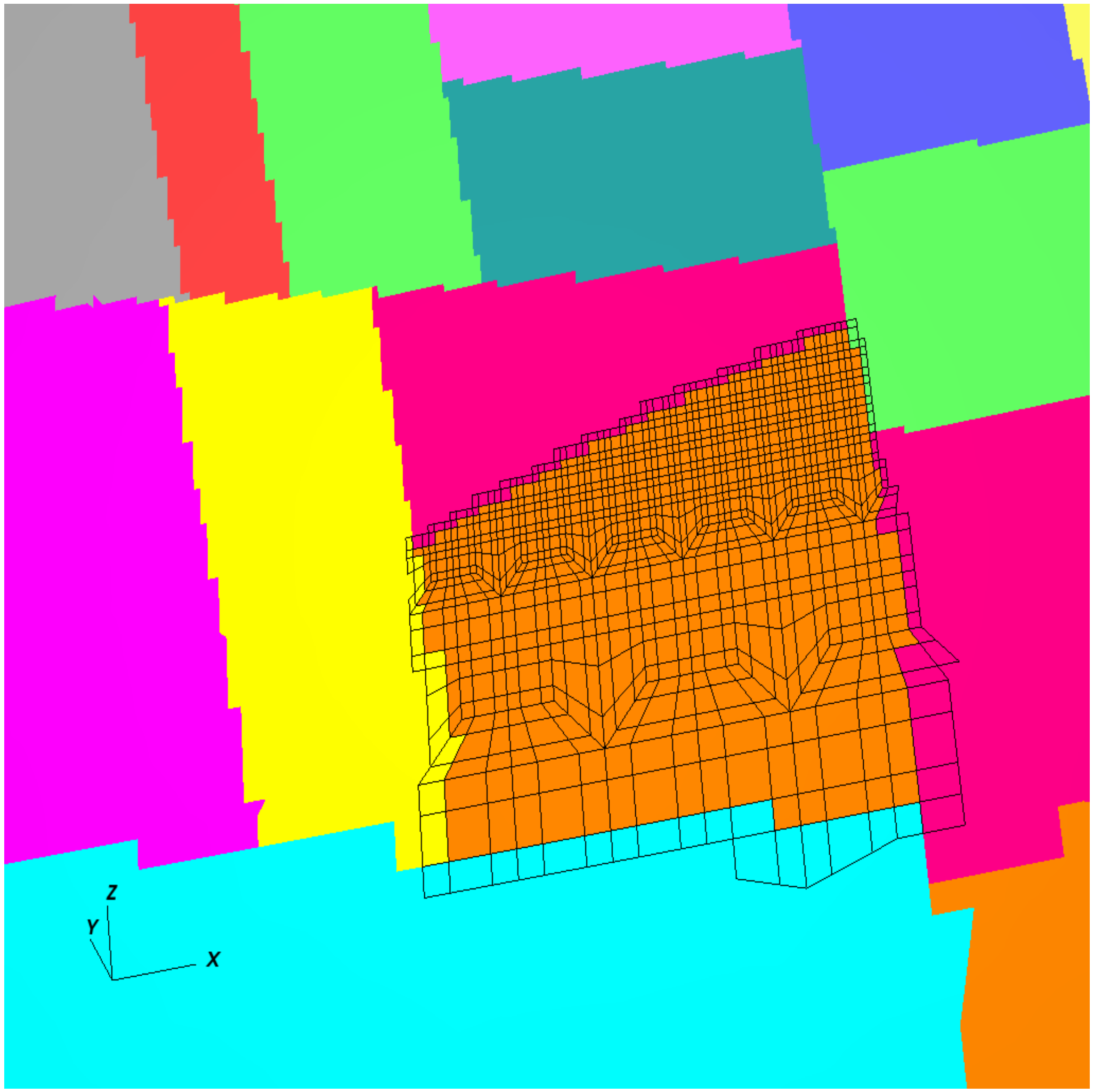


MOAB Parallel Model

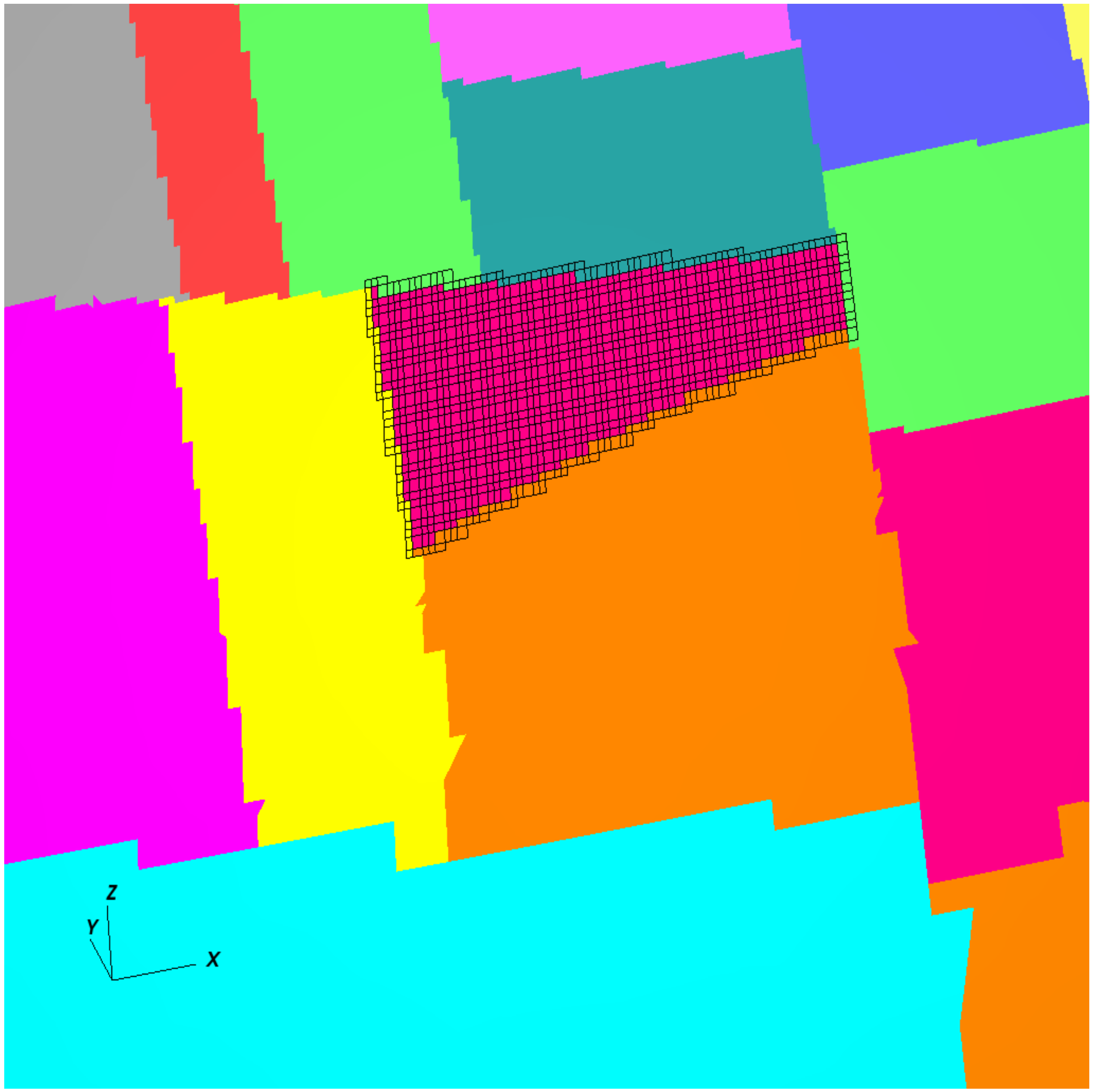


Ghost layer(s) easy to obtain

MOAB Parallel Model



MOAB Parallel Model

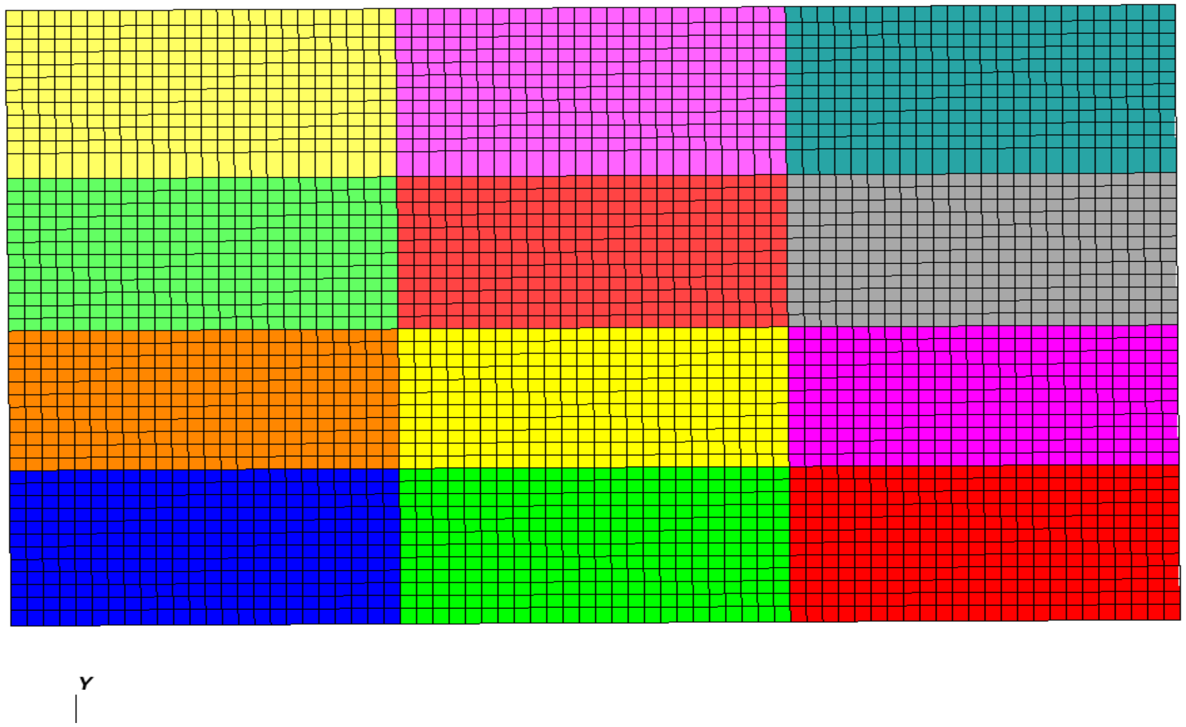


How to partition meshes, for a distributed parallel read ?

CAM Euler/FV files: lat-lon grid.

Use ScdInterface, a special class in moab, to handle structured grid.

It handles periodicity too.



How to partition/distribute meshes, for a distributed parallel read ?

MPAS files : typical data look like that:

…

dimensions:

    nCells = 642 ;
    nEdges = 1920 ;
    nVertices = 1280 ;
    maxEdges = 10 ;

…

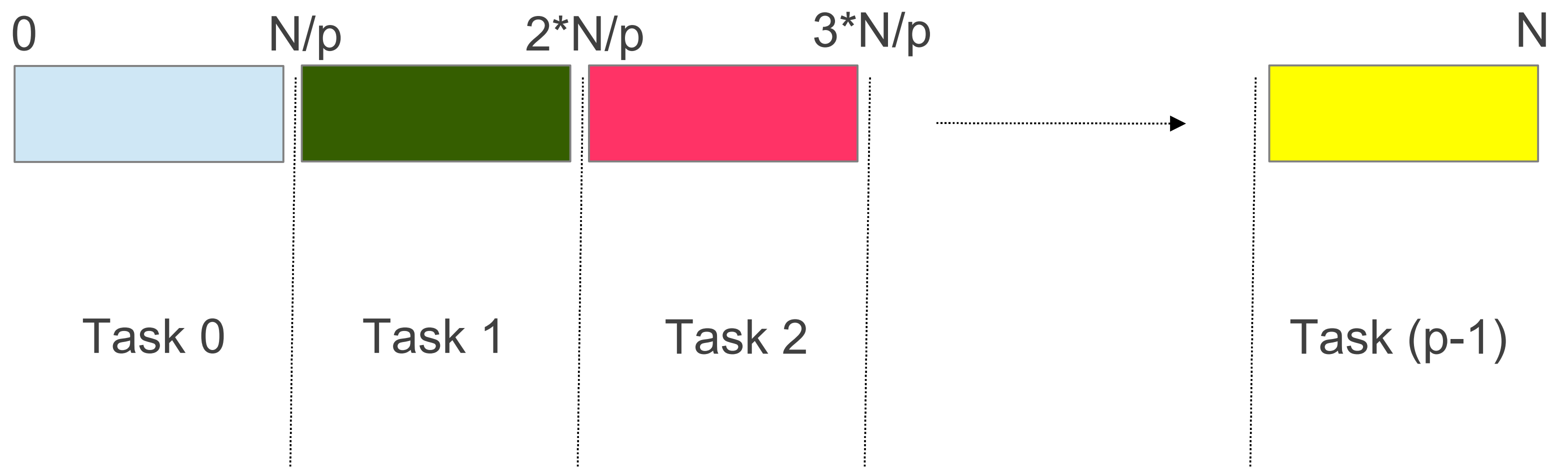
variables:

    double u(Time, **nEdges**, nVertLevels) ;
    double v(Time, **nEdges**, nVertLevels) ;
    double h(Time, **nCells**, nVertLevels) ;
    double vh(Time, **nEdges**, nVertLevels) ;
    double circulation(Time, **nVertices**, nVertLevels) ;

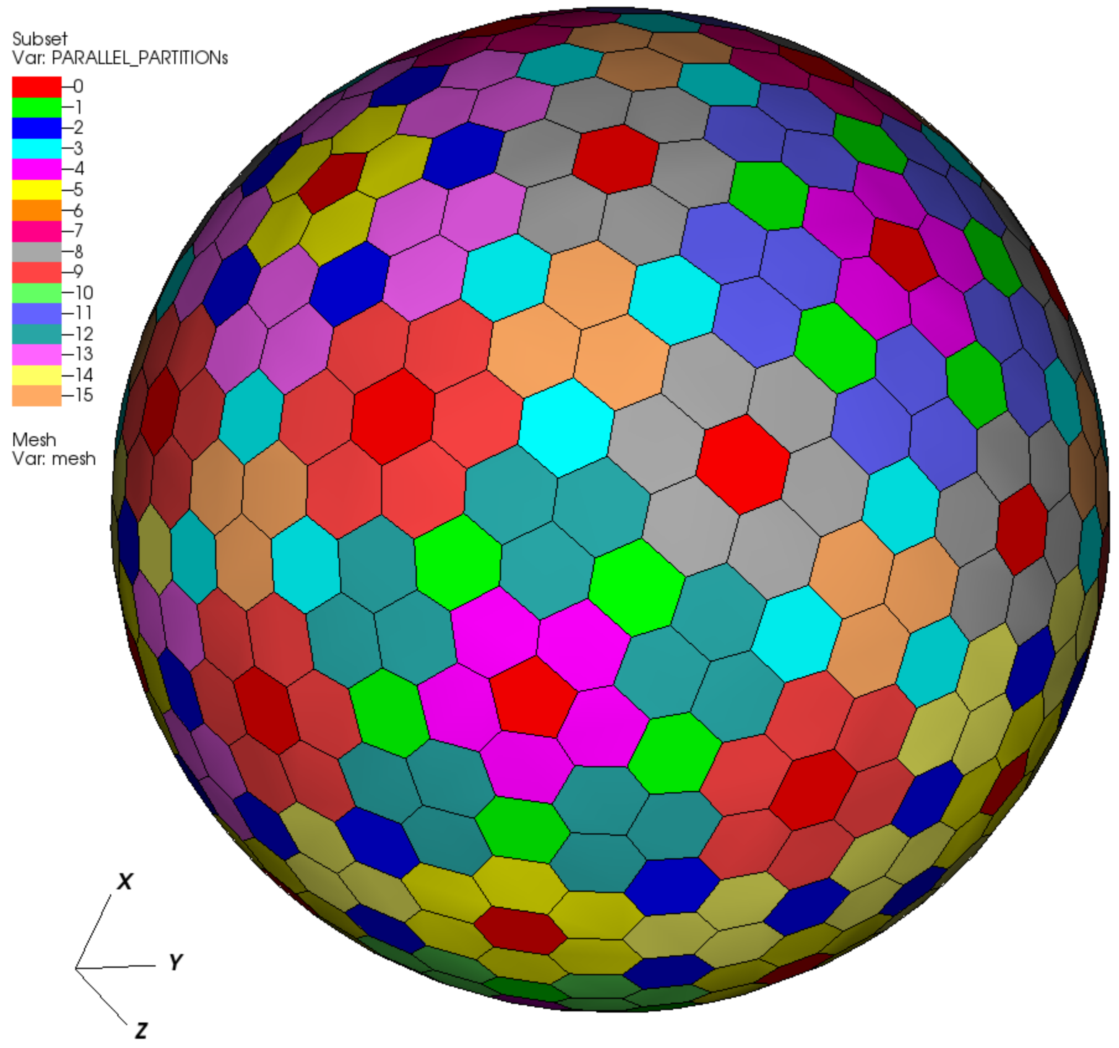
First method: Trivial Partition

- Classic method, natural
- Get the number of cells N, divide by number of processors p, then every processor reads its “slice”, in parallel

Processor k reads values for cells [k*[N/p] .. (k+1)*[N/p])



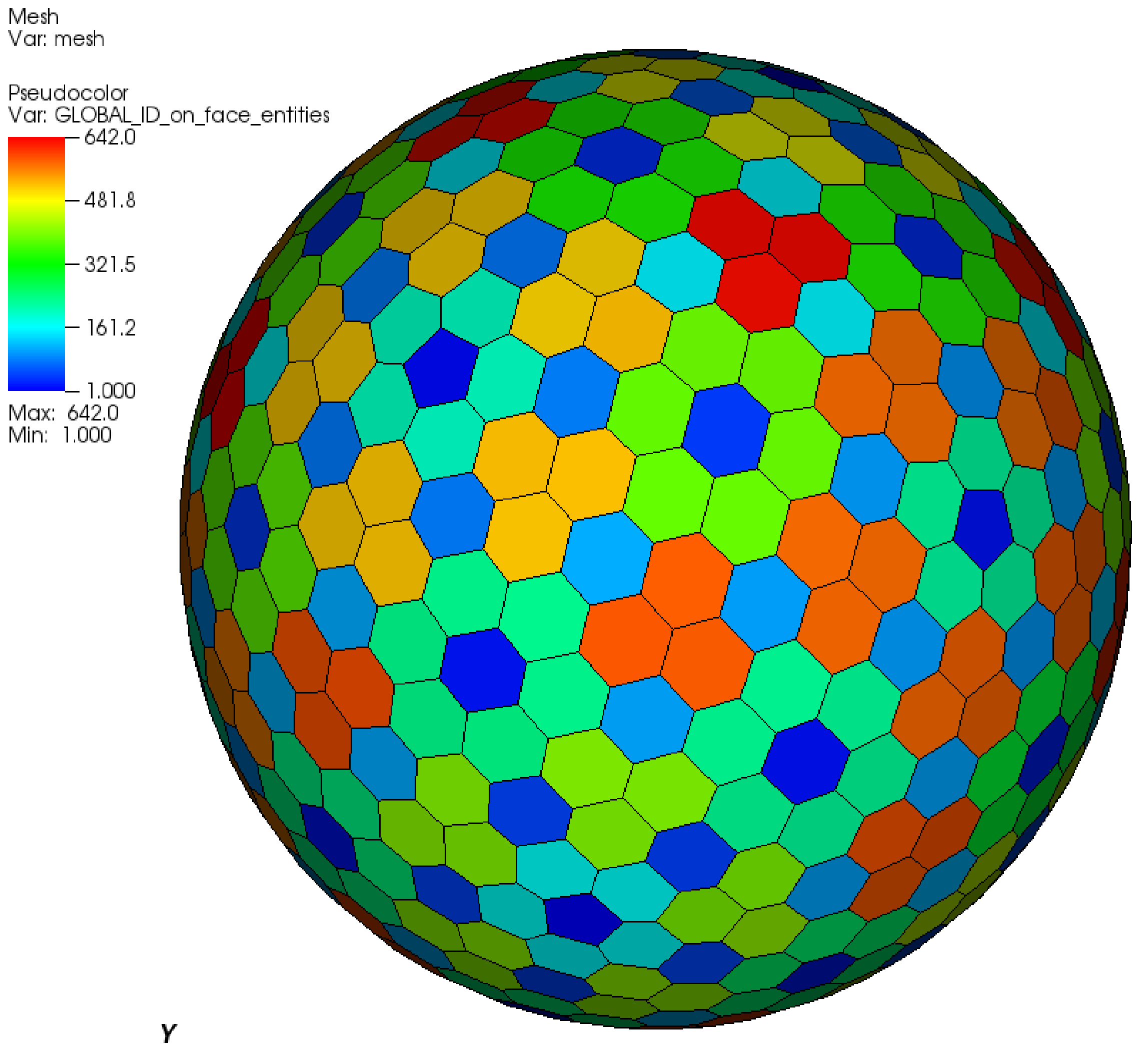
Small MPAS file, 16 parts, 642 cells, trivial partition



Every color means a different part

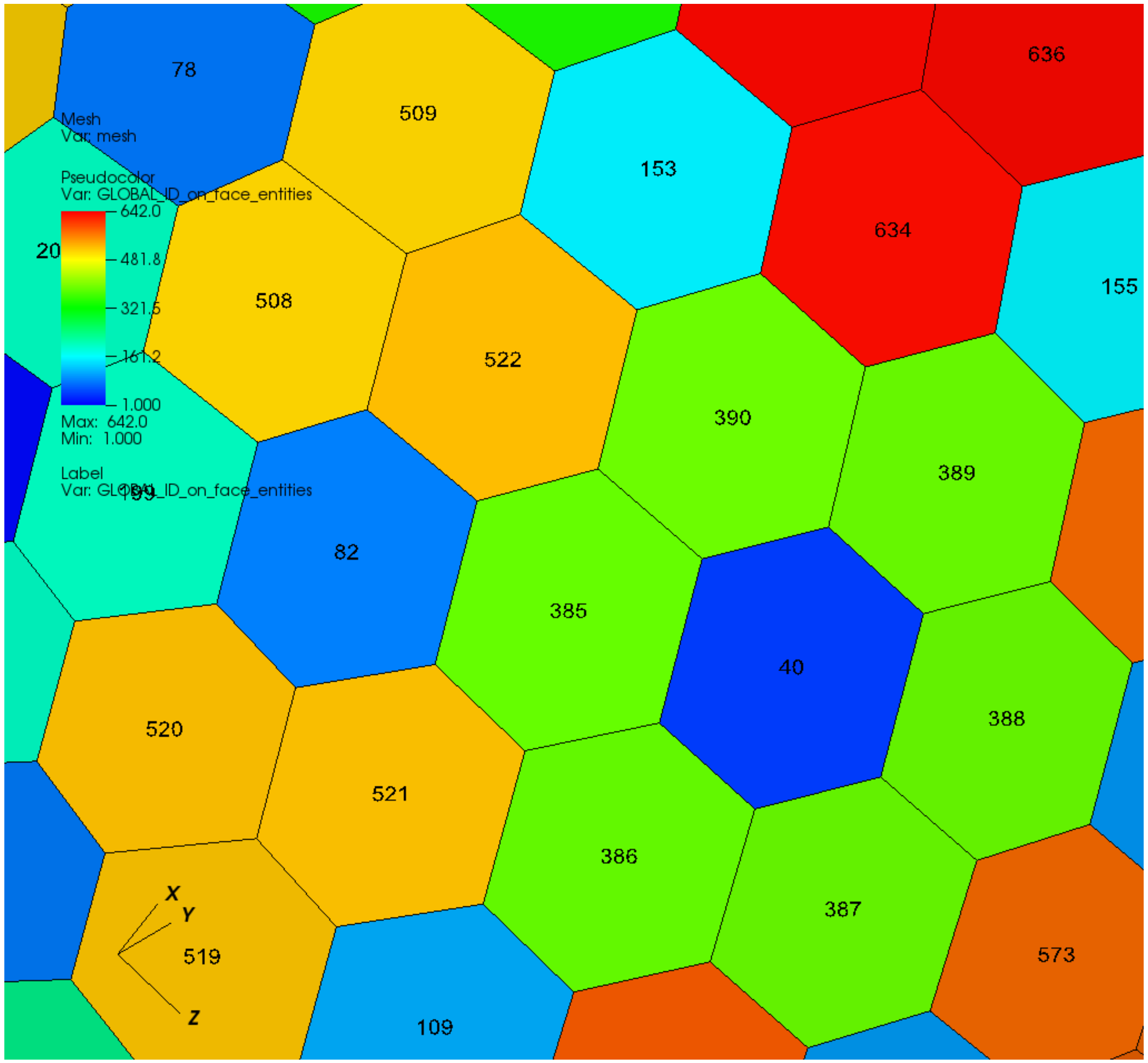
Why are the parts in partition so disjoint?

Small MPAS file, global ID plot



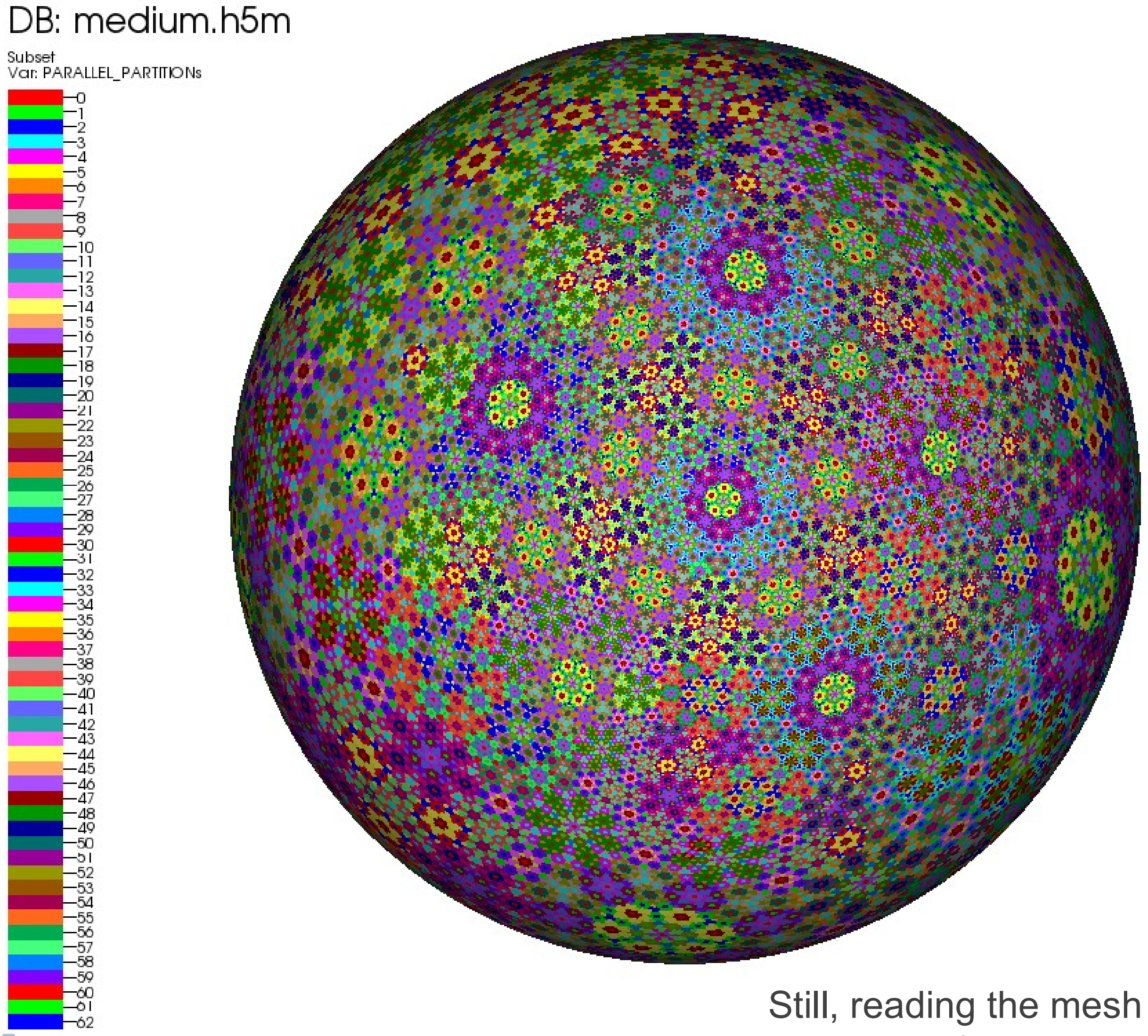
Color shows the order in the file, from 1 to 642

Small MPAS file, 16 parts, global ID, zoom in



Color shows the order in the file, from 1 to 642

Medium MPAS file, 64 parts, 163K cells (92km refined to 25 km over NA)



Color shows partition tag

Pretty picture, but: communication load

Still, reading the mesh on 64 tasks took 1.4s (including resolving sharing)

Large MPAS file, 65M polygons

Trivial partition: MOAB had a problem :(

The reader broke down. It read the mesh, but it was not able to resolve the shared entities, on 1024 tasks.

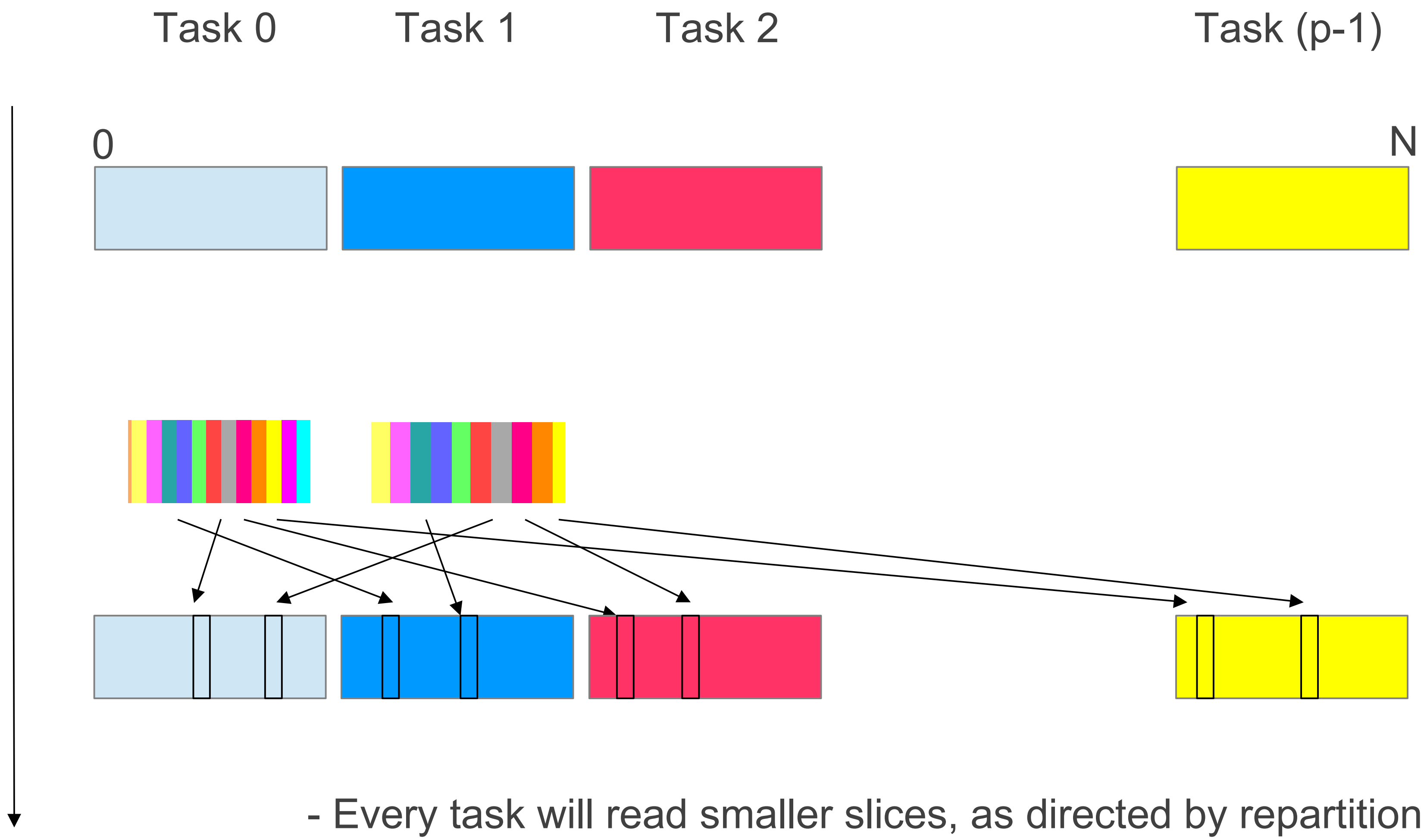
Solution: repartition, using Zoltan.

Read in the "trivial" step, just the centers of the cells;

Then, repartition the cells, to be more compact in space.

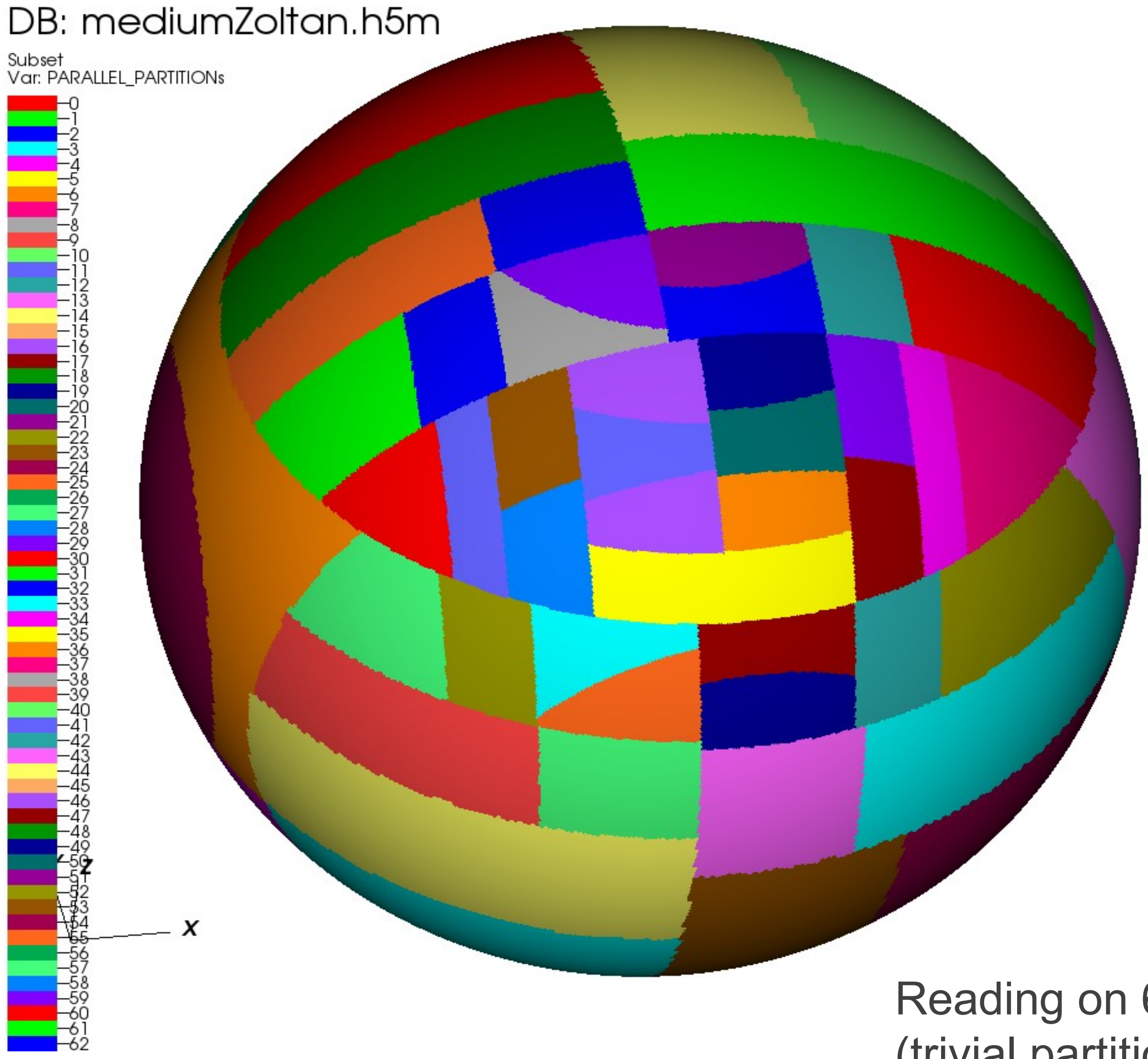
Each processor reads (cherry-picks) the connectivity and vertex positions of the cells in its partition: all these calls are nonblocking reads, with pnetcdf.

Repartition with Zoltan (cont)



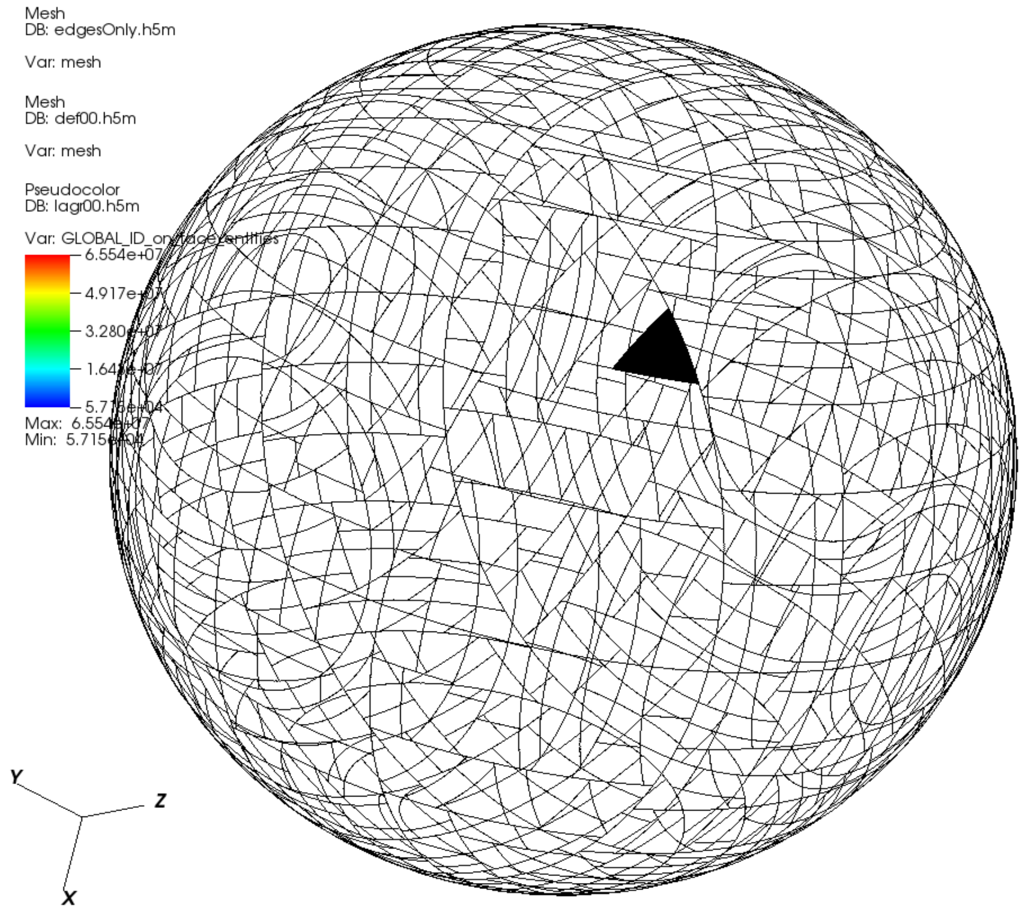
- Every task will read smaller slices, as directed by repartitioning
- All reads are nonblocking

Repartition, medium MPAS file, 64 parts



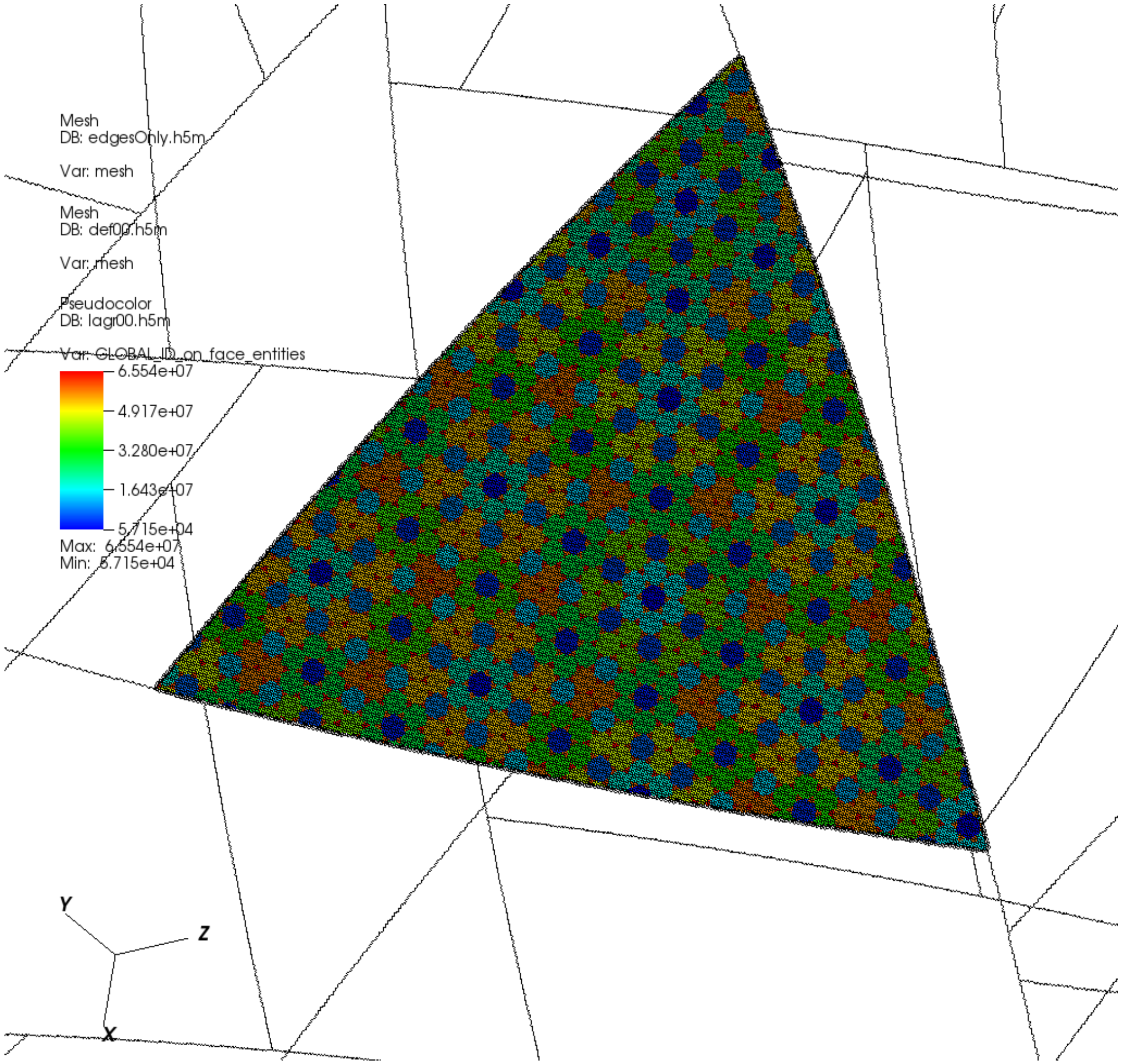
Reading on 64 tasks: 0.6s
(trivial partition took 1.4s)

Partition of large file, 1024 parts



Show just the boundary between parts, and the mesh on one part

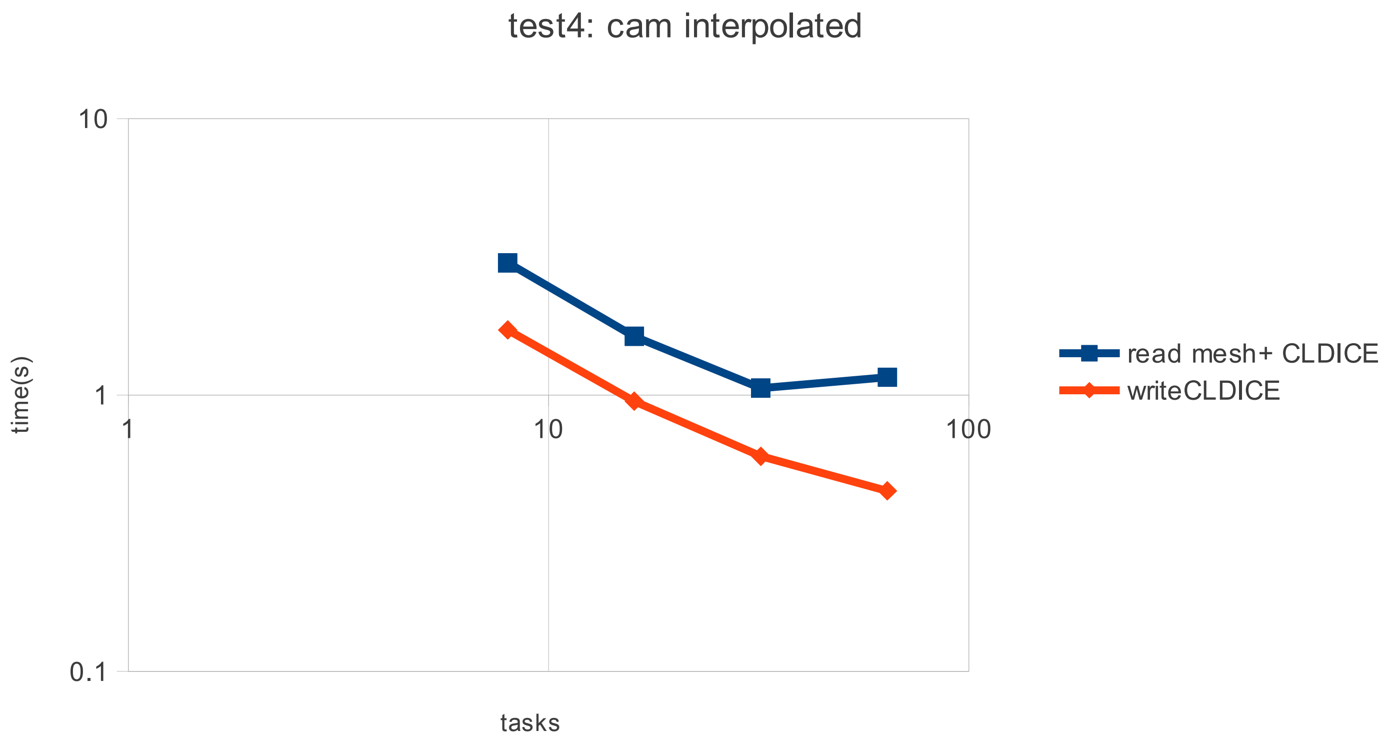
Partition of large file, 1024 parts, zoom in



Show just the boundary between parts, and the mesh on one part : ~65K cells in each part

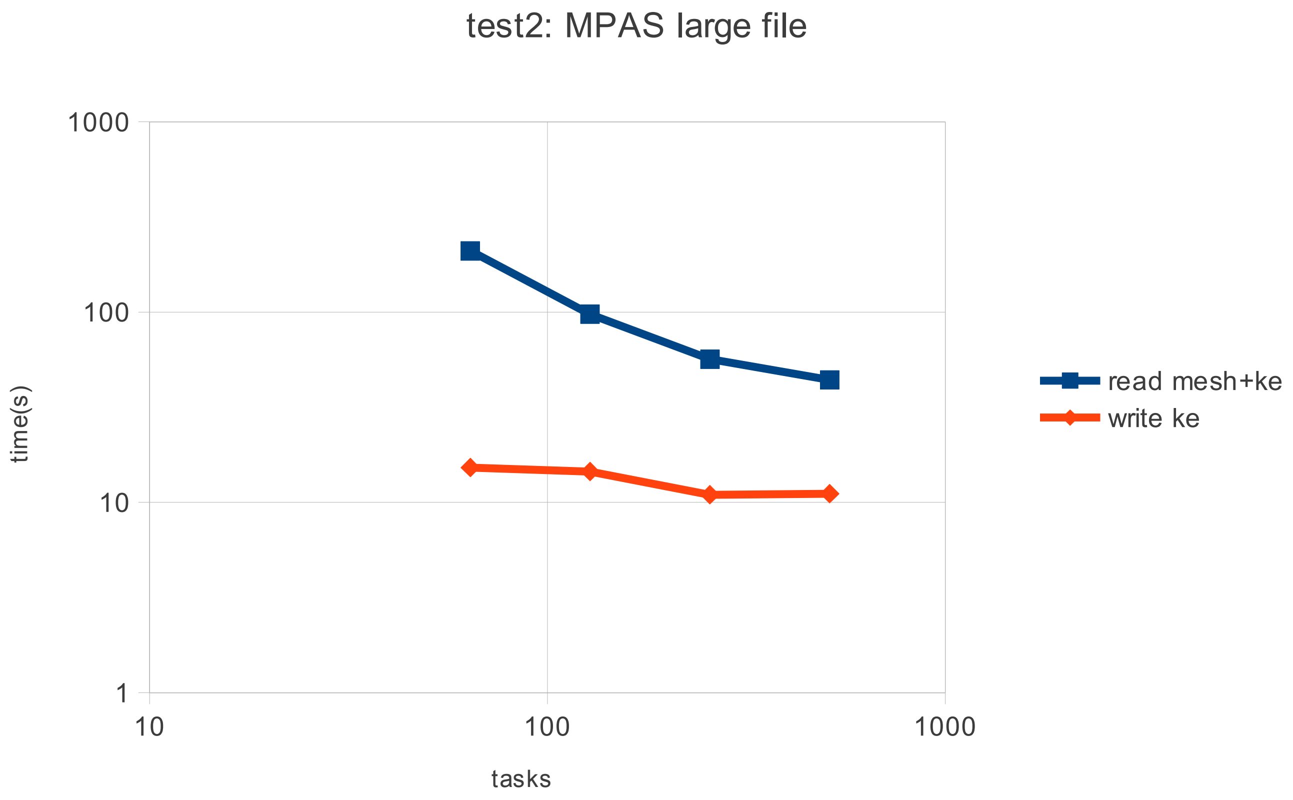
Some scalability results for reading / writing climate data files with MOAB

Example 1: 0.25 degree CAM-SE cubed sphere; 777602x26 f1850_ne120tx01.cam2.h0.0006- 01.nc



More results

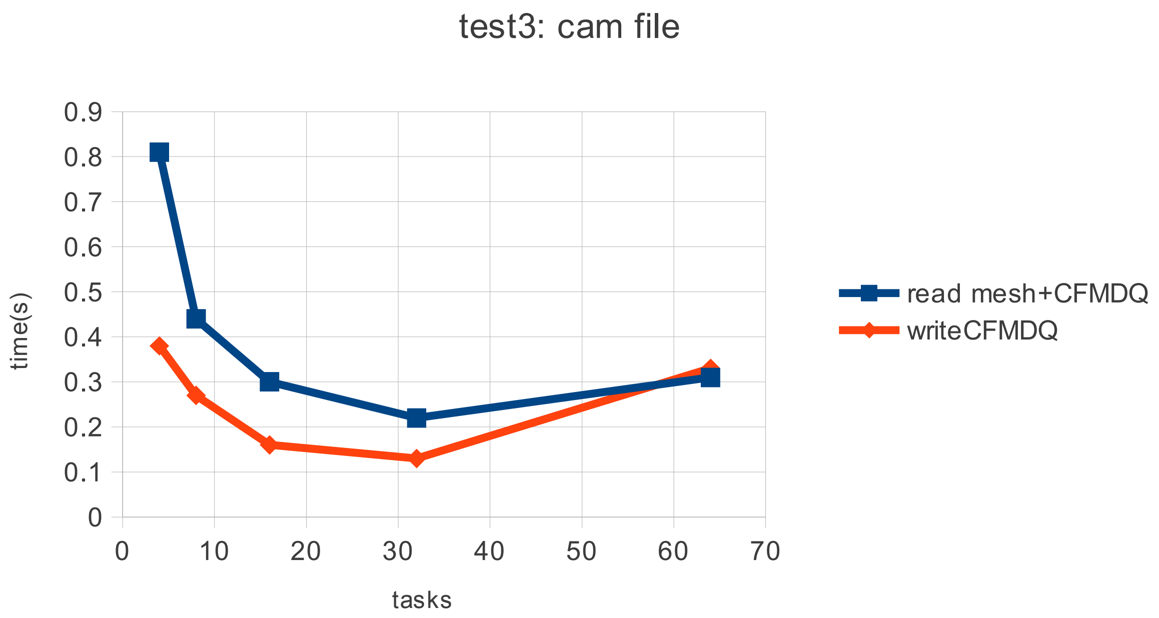
Example 2: 3Km MPAS grid x1.65536002.output.2009-01-28_00.00.00.nc (65M cells)



More results

Example 3: 0.25 degree CAM-FV; 768x1152x26

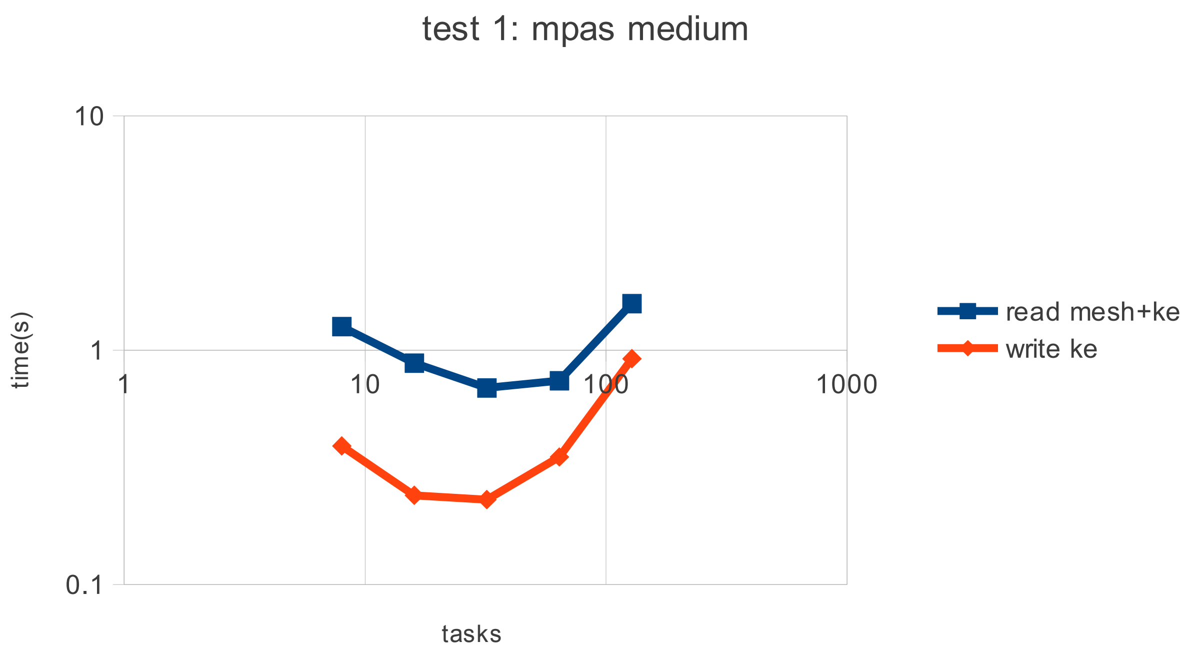
f40_amip_025d_b06c4_207jp.cam2.h0.1986-04.nc (4.1 Gb)



More results

Example 2:

MPAS 92-km global resolution refined down to 25-km resolution over North America; 163842x42   x1.163842.2010102800.nc



Intersection of meshes (CSLAM)

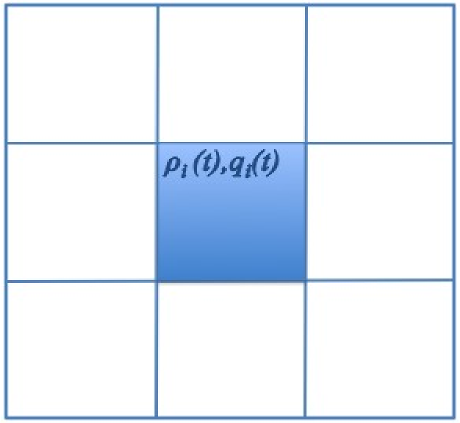
For transport/advection problems, one method is ALE

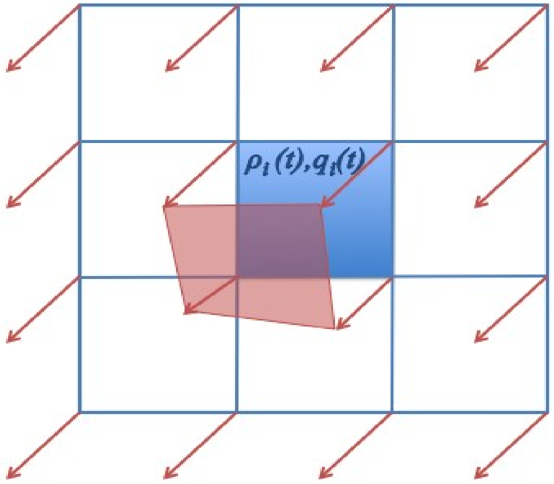
It was developed in an attempt to combine the advantages of the Eulerian and the Lagrangian approaches, by letting the mesh move in a prescribed manner as an extra independent degree of freedom

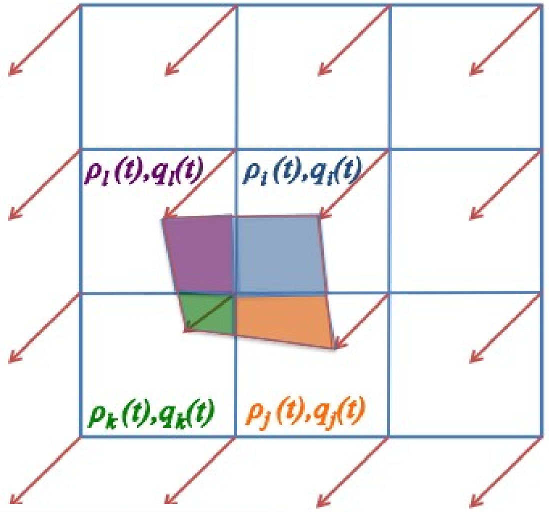
Run in Lagrangian mode for one time-step and then interpolate back to the static and regular (Eulerian) mesh. This interpolation step involves intersecting the Lagrangian mesh (also called departure mesh) and Eulerian mesh, and is known as semi-Lagrangian method

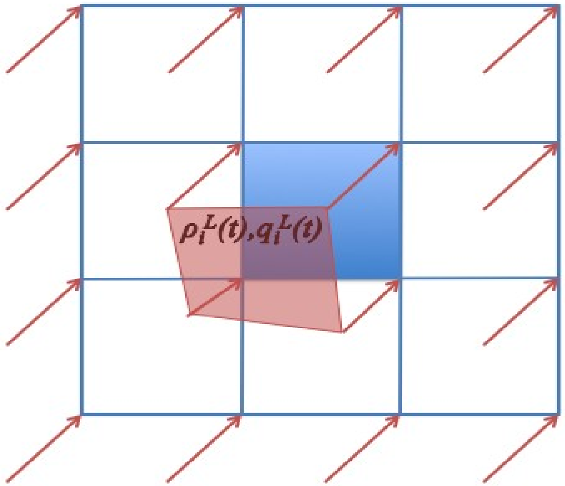
For correct intersection resolution, mesh need to be communicated to neighboring processors

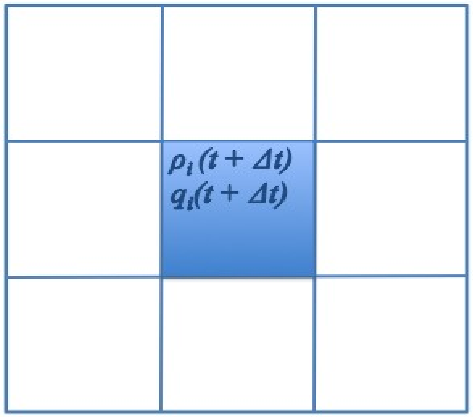
Intersection of meshes





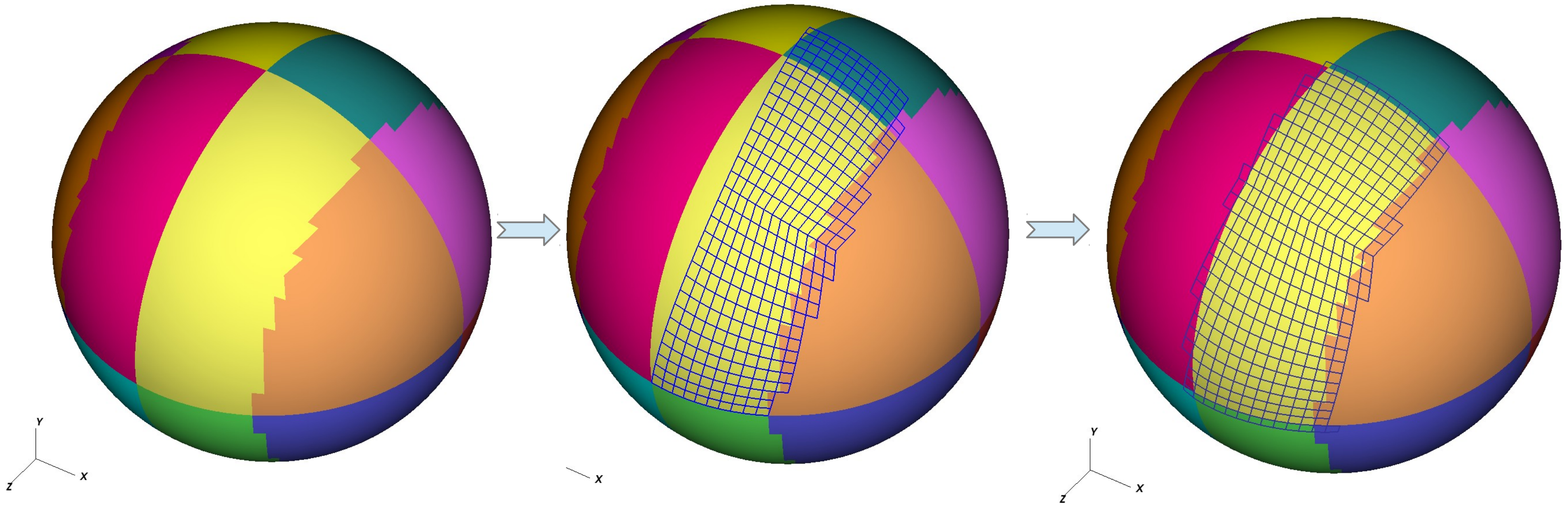




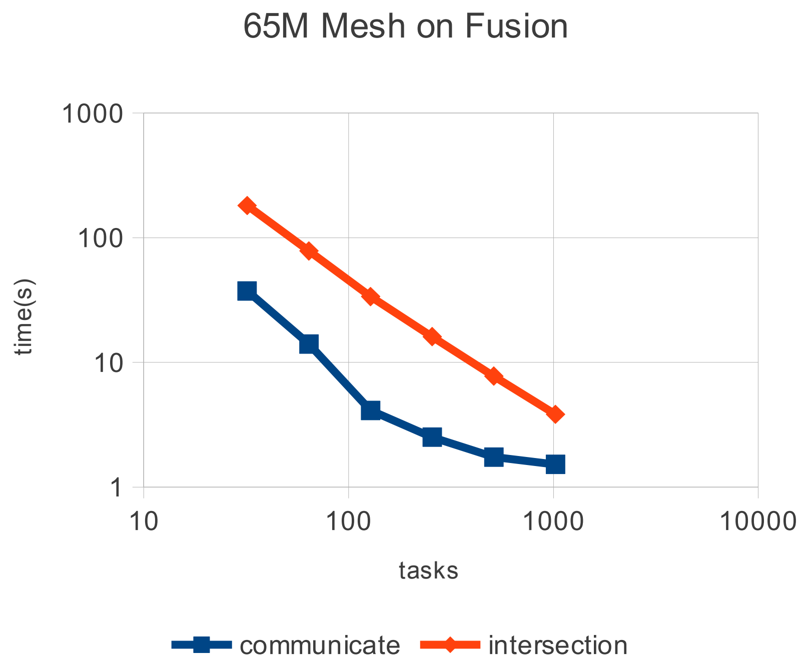


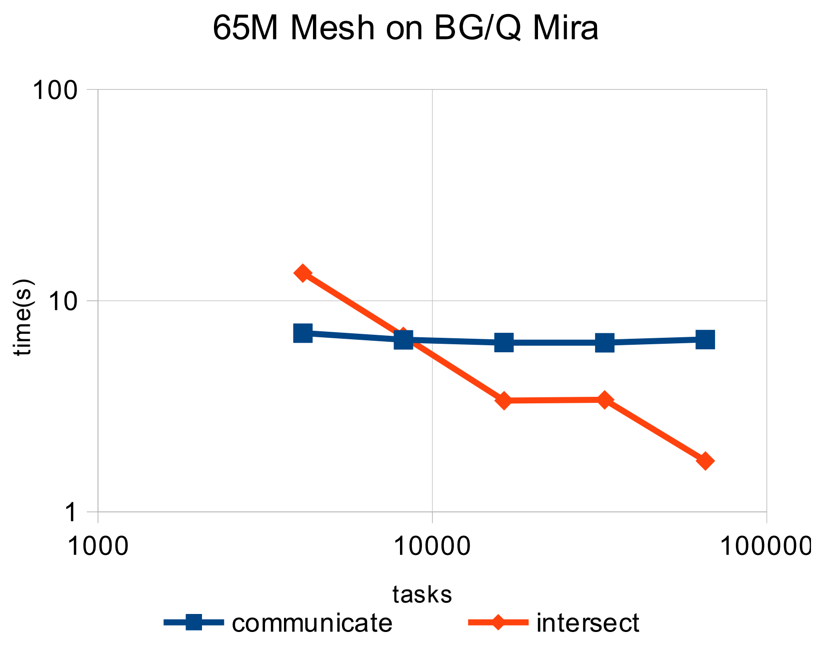
Intersection of distributed meshes

- Partitioning + departure mesh + covering



Results for Parallel Intersection MPAS 3km - 65M cells





Future Development

- Improvements in parallel framework: local communicators, topology aware communicators
- Coupling: different communicators for different physics/solvers
- New readers/writers for different climate formats
- Direct link with CAM/FV for CSLAM-related intersection

Extra Slides

References

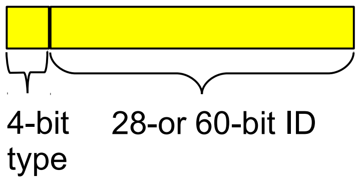
Robert Jacob et. al. , “ParNCL and ParGAL: Data-parallel Tools for Postprocessing of Large-scale Earth Science Data”, Procedia Computer Science, Vol 18, 2013

Tim Tautges et. al. , “Mesh interface resolution and ghost exchange in a parallel mesh representation”, Parallel and Distributed Processing Symposium Workshop, 2012

T. J. Tautges, et. al. . MOAB: a mesh-oriented database. SAND2004-1592, Sandia National Laboratories, 2004. Report.

MOAB Entity Storage

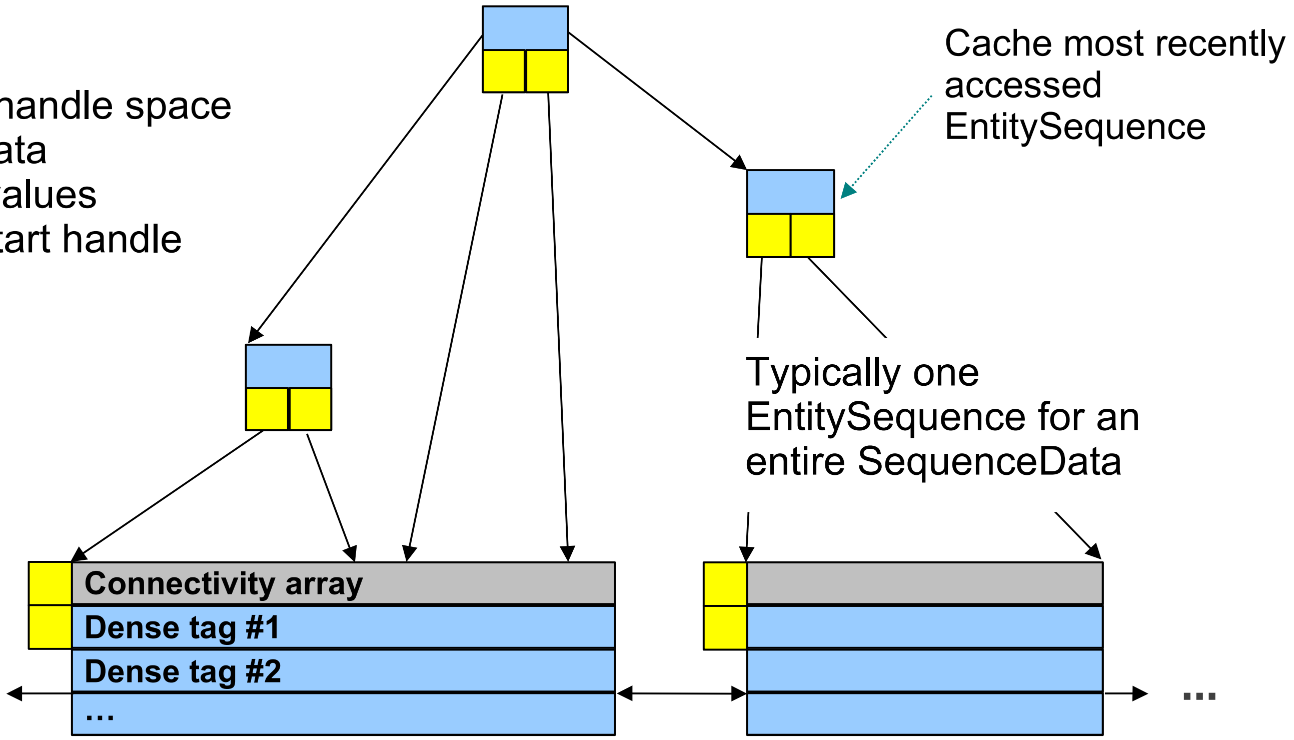
Entity Handle:
- Unsigned long type
- Bitmask
- Sorts by dimension, type



Range:
- Container of handles
- Constant-size if contiguous handles

EntitySequences:
- Represent *used* portions of handle space
- Have pointer to SequenceData
- Have start and end handle values
- Arranged in binary tree by start handle

SequenceData:
- Represent *allocated* portions of handle space
- Have start and end handle
- Coordinates or Connectivity
- Dense Tag Data


Cache most recently accessed EntitySequence
Typically one EntitySequence for an entire SequenceData
Connectivity array
Dense tag #1
Dense tag #2
…
…