Regridding ESMF: A Parallel Rendezvous Interpolation Algorithm

David Neckels

National Center for

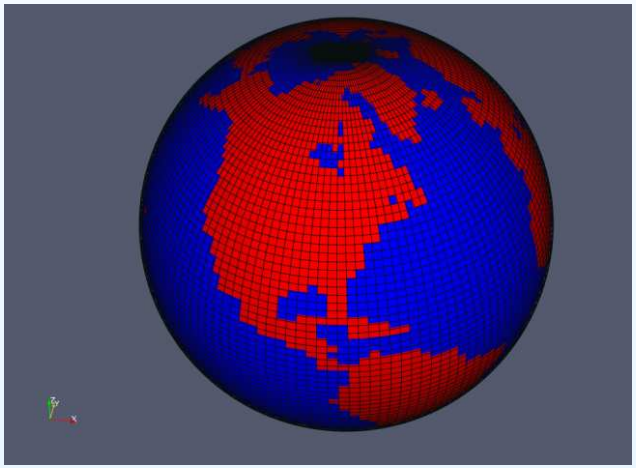
Atmospheric Research

Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.

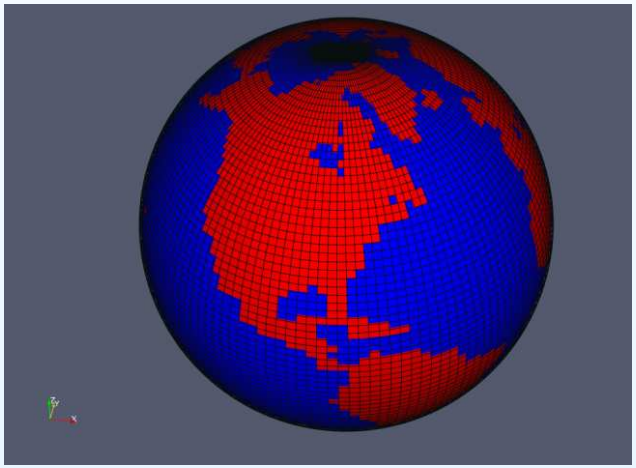
Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



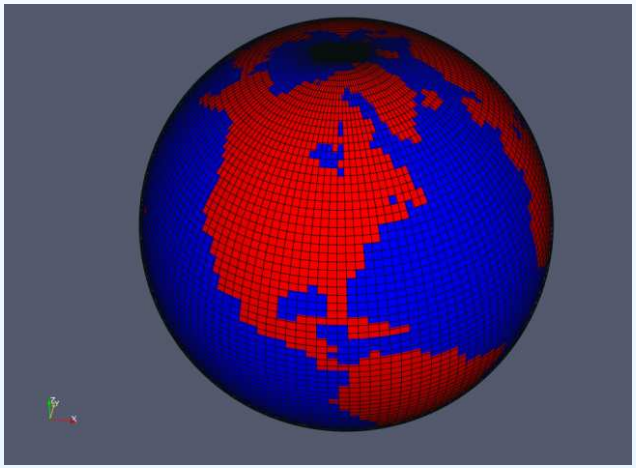
Description of the Problem

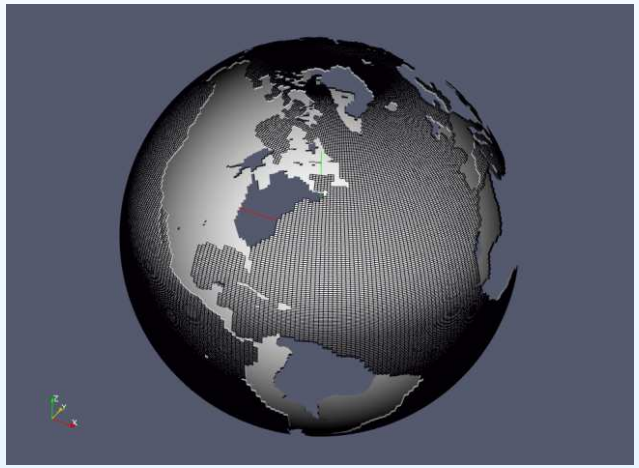
To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



Description of the Problem

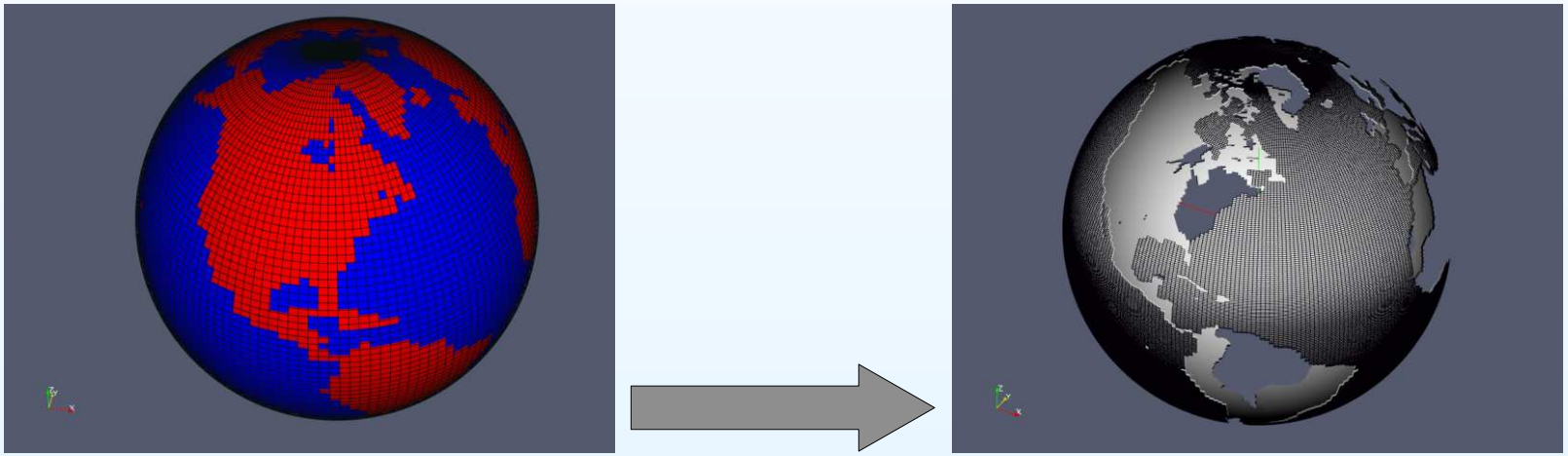
To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.





Description of the Problem

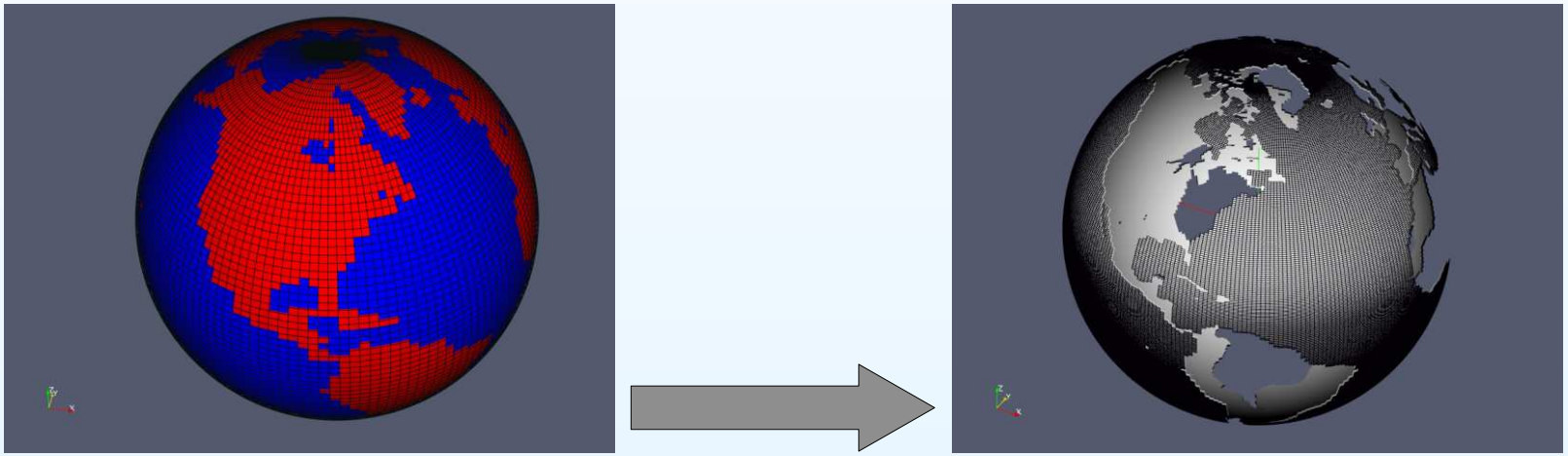
To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



Within ESMF, how to calculate weights in an efficient/load balanced manner?

Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



Within ESMF, how to calculate weights in an efficient/load balanced manner?

How to perform the interpolation in an efficient/load balanced manner?

Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

Destination points are shipped to the source grid decomposition for the search and weight calculation.

Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

Destination points are shipped to the source grid decomposition for the search and weight calculation.

These bounding boxes depend on a fixed coordinate system (which the two grids must negotiate), and optimal performance requires the parallel decomposition be roughly aligned with this coordinate system.

Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

Destination points are shipped to the source grid decomposition for the search and weight calculation.

These bounding boxes depend on a fixed coordinate system (which the two grids must negotiate), and optimal performance requires the parallel decomposition be roughly aligned with this coordinate system.

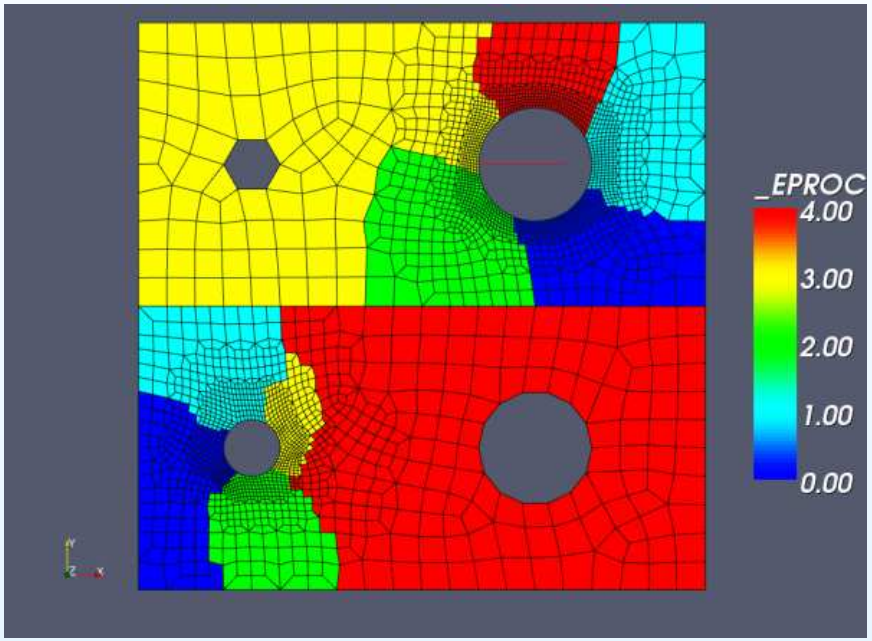
This condition is rarely satisfied.

Load in balance problem

The interpolation problem is by nature geometric, but the grid decomposition is not necessary so

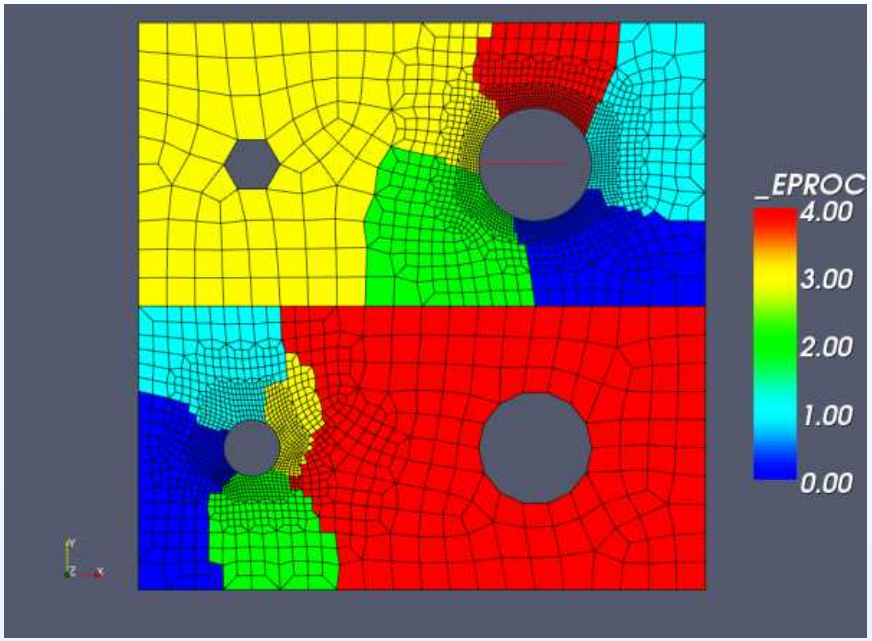
Load in balance problem

The interpolation problem is by nature geometric, but the grid decomposition is not necessary so



Load in balance problem

The interpolation problem is by nature geometric, but the grid decomposition is not necessary so



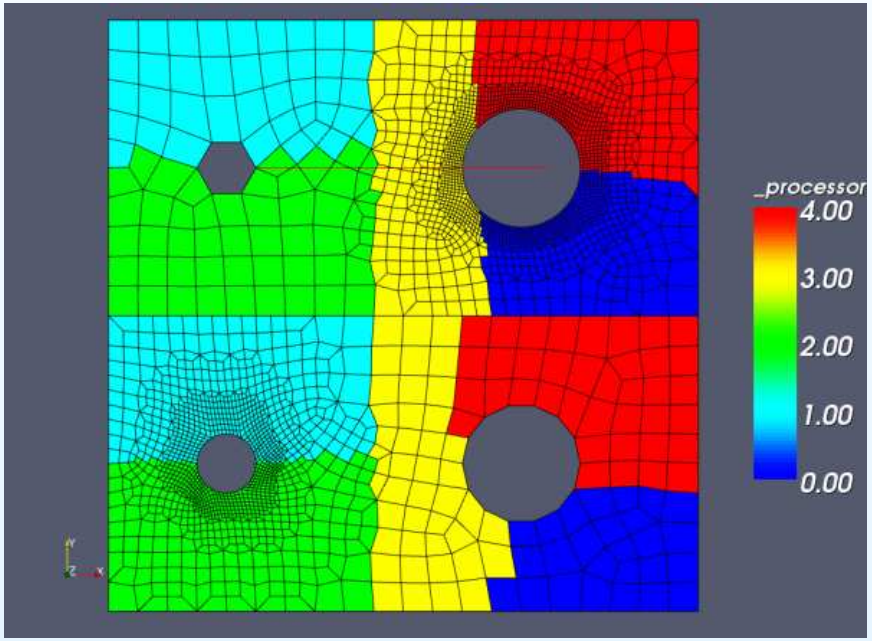
In the worst case, the entire source mesh may be shipped to one processor! This standard approach lacks robustness.

A Geometric solution

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!

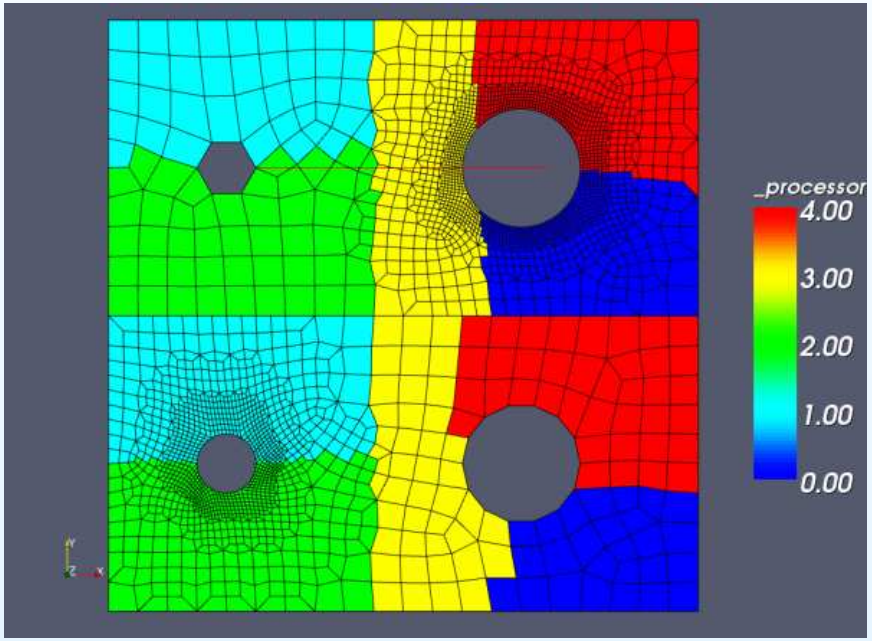
A Geometric solution

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!



A Geometric solution

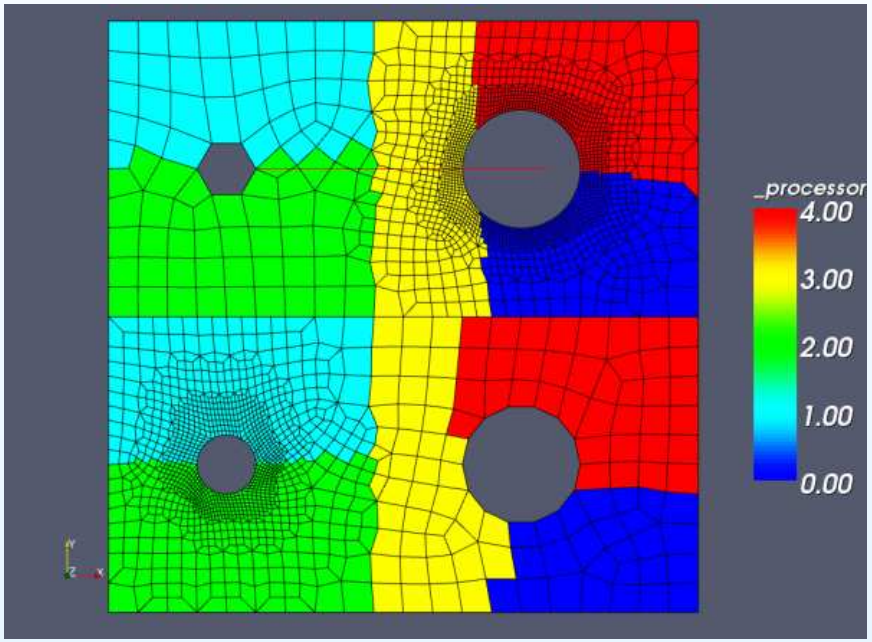
We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!



Also, the union of meshes is load balanced!

Exchange grids, anyone?

This new partition could easily be used to compute the exchange grid fractional areas, in parallel.



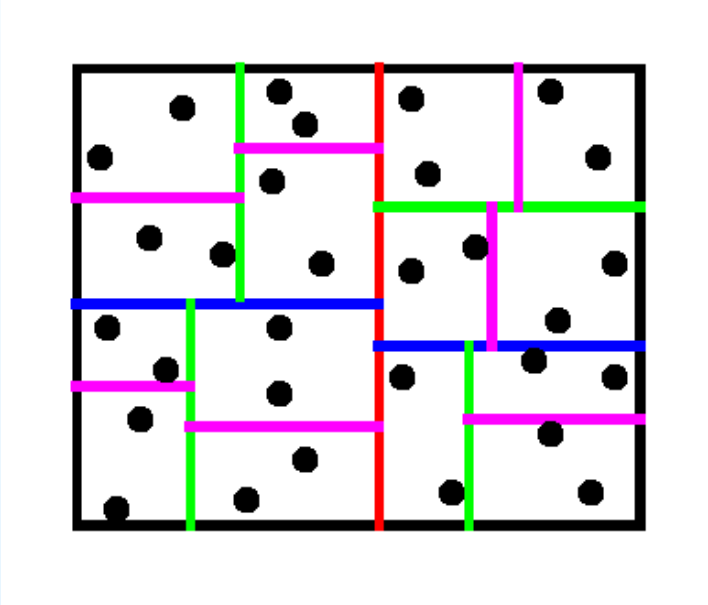
Just choose your favorite serial algorithm for calculating fractional cell areas.

RCB to the rescue

The Recursive Coordinate Bisection algorithm is a parallel algorithm for partitioning a set of geometric entities (possibly with weights). The package Zoltan provides this.

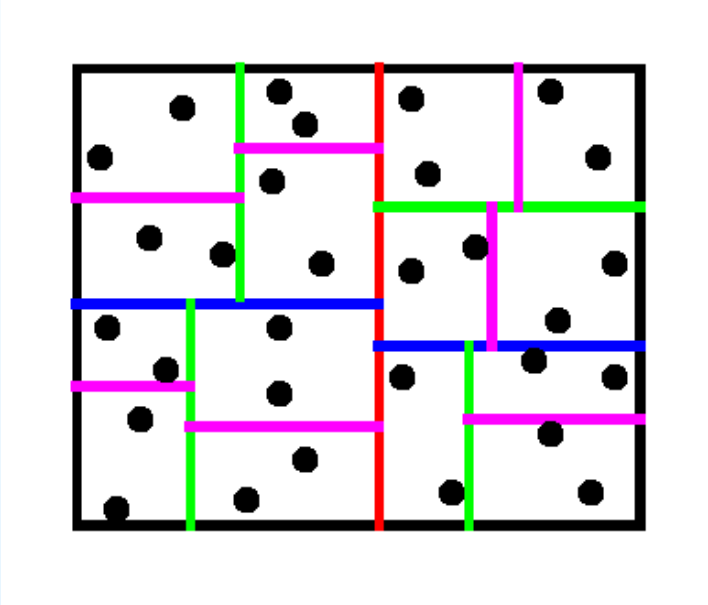
RCB to the rescue

The Recursive Coordinate Bisection algorithm is a parallel algorithm for partitioning a set of geometric entities (possibly with weights). The package Zoltan provides this.



RCB to the rescue

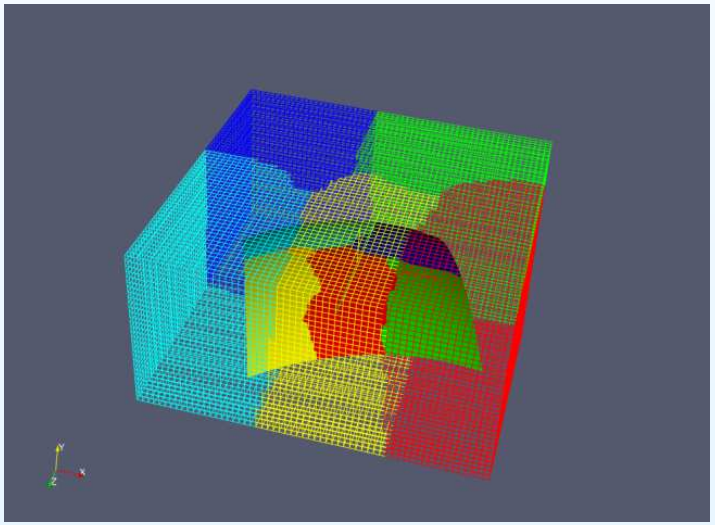
The Recursive Coordinate Bisection algorithm is a parallel algorithm for partitioning a set of geometric entities (possibly with weights). The package Zoltan provides this.

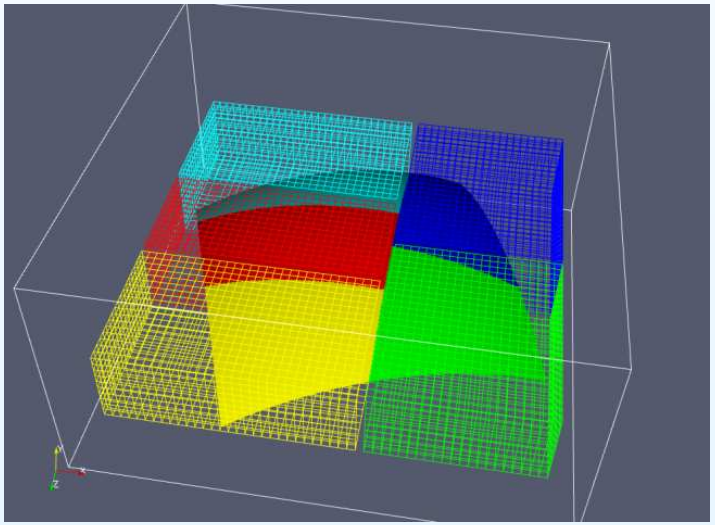


A parallel median-finding kernel is at the core of the algorithm.

Intersecting grids

This algorithm is only applied to the geometric intersection of the meshes.



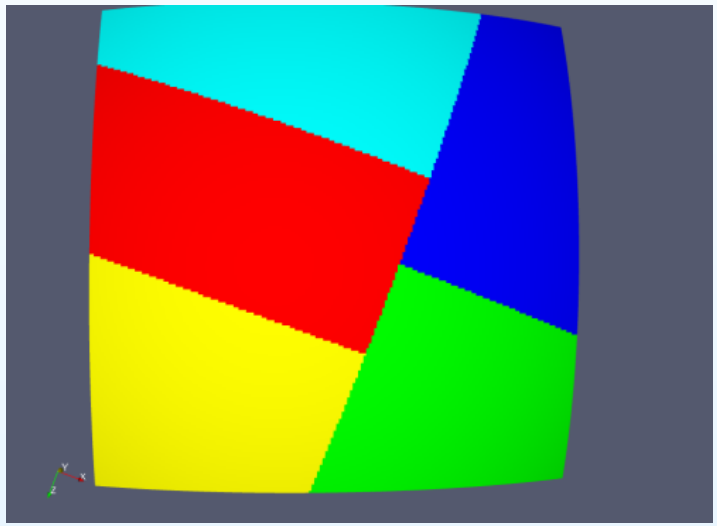


Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.

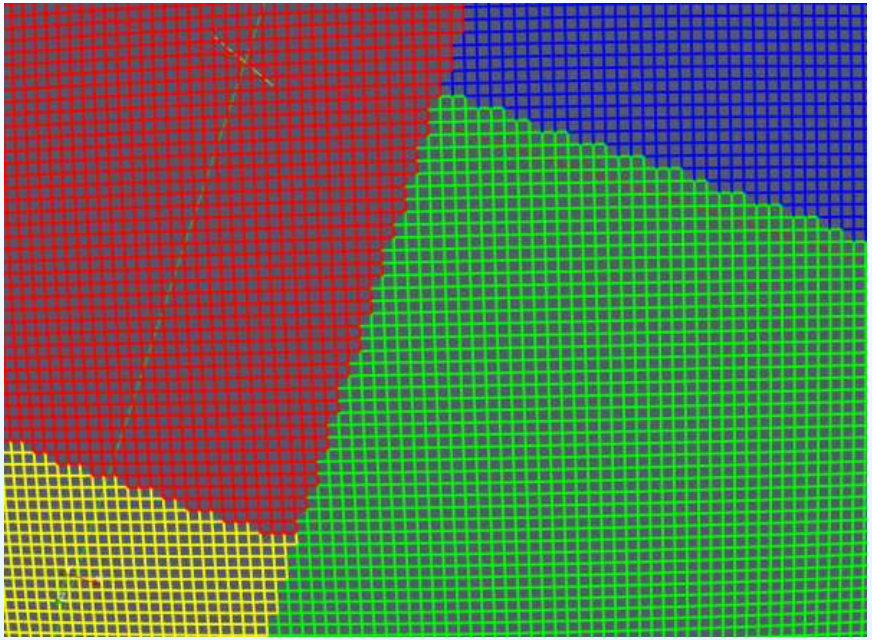
Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.



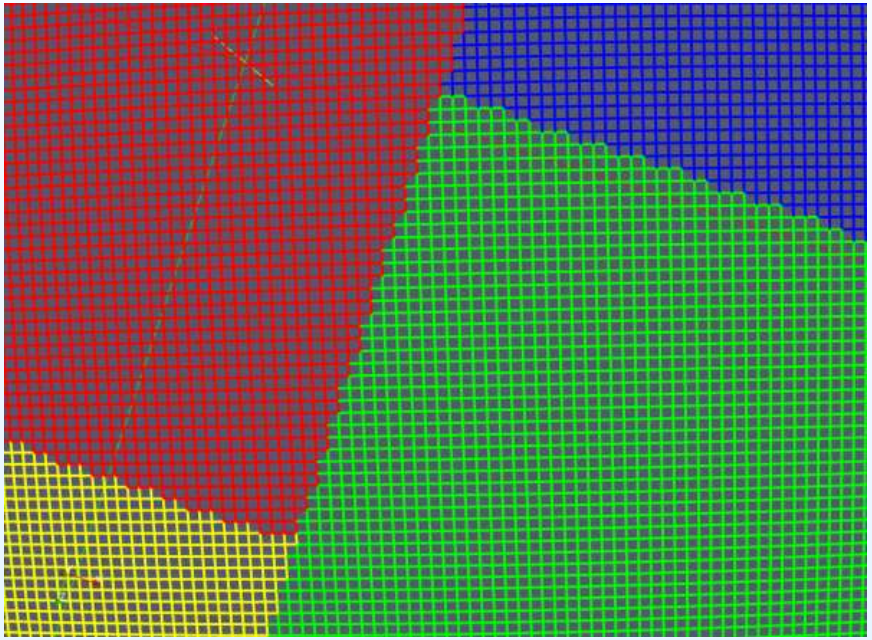
Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.



Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.



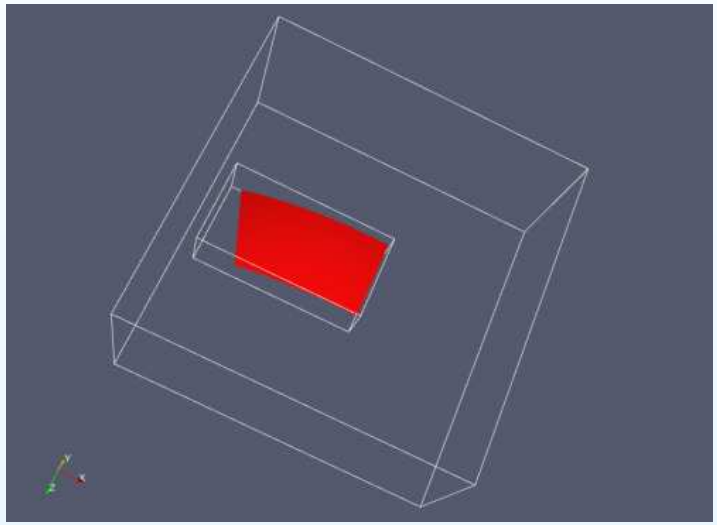
For the exchange grid calculation or for two way regridding, we need a representation for such decompositions.

Local search

For the local search algorithm we use either the Swegle method, or a more modern octree method.

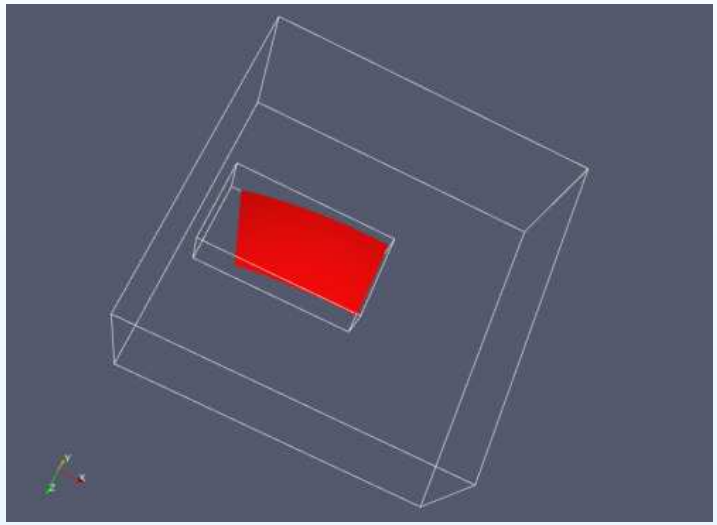
Local search

For the local search algorithm we use either the Swegle method, or a more modern octree method.



Local search

For the local search algorithm we use either the Swegle method, or a more modern octree method.



We calculate the weights using whatever local scheme is appropriate (bi-linear or conservative for finite difference codes, or by master element interpolation rules for finite element.

Load balancing the regrid

Once we form a matrix on the rendezvous grid, we have three choices for storing and applying the matrix.

Load balancing the regrid

Once we form a matrix on the rendezvous grid, we have three choices for storing and applying the matrix.

1) Send the matrix to a row decomposition (destination owned).

Load balancing the regrid

Once we form a matrix on the rendezvous grid, we have three choices for storing and applying the matrix.

1) Send the matrix to a row decomposition (destination owned).

2) Send the matrix to a column decomposition (source owned).

Load balancing the regrid

Once we form a matrix on the rendezvous grid, we have three choices for storing and applying the matrix.

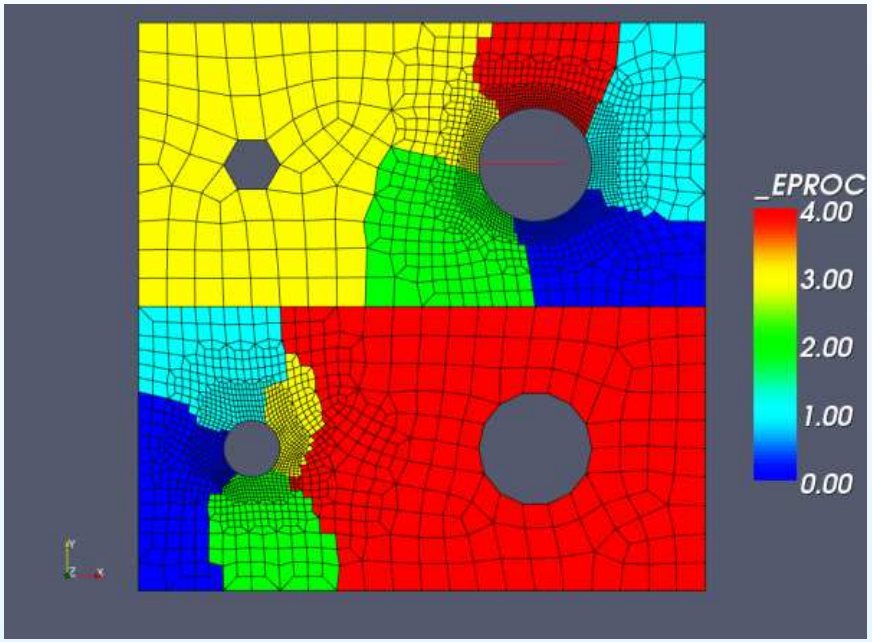
1) Send the matrix to a row decomposition (destination owned).

2) Send the matrix to a column decomposition (source owned).

3) Keep the matrix on the rendezvous decomposition.

Load imbalance problem

Only the rendezvous decomposition guarantees a load balance for the matrix application and storage.



Both the row and column decompositions may swamp a processor with incoming communication and memory requirements.

The rendezvous matrix application

The interpolation forms a commutative diagram

The rendezvous matrix application

The interpolation forms a commutative diagram

$$\begin{tikzcd}
Src_R \arrow[r, "\mathcal{C}"] & Dst_R \\
Src_s \arrow[u, "\mathcal{A}"] \arrow[r, "\mathcal{I}"] & Dst_d \arrow[u, "\mathcal{B}"']
\end{tikzcd}$$

The rendezvous matrix application

The interpolation forms a commutative diagram

$$\begin{array}{ccc} Src_R & \xrightarrow{\mathcal{C}} & Dst_R \\ \mathcal{A}\uparrow & & \mathcal{B}\uparrow \\ Src_s & \xrightarrow{\mathcal{I}} & Dst_d \end{array}$$

Where $\mathcal{A}$ and $\mathcal{B}$ are the mesh migration communication spec's and $\mathcal{C}$ is the local interpolation operator. The subscripts $_{s,d,R}$ are the source,destination and rendezvous decompositions. We have $\mathcal{I} = \mathcal{B}^{\top} \circ \mathcal{C} \circ \mathcal{A}$.

The rendezvous matrix application

The interpolation forms a commutative diagram

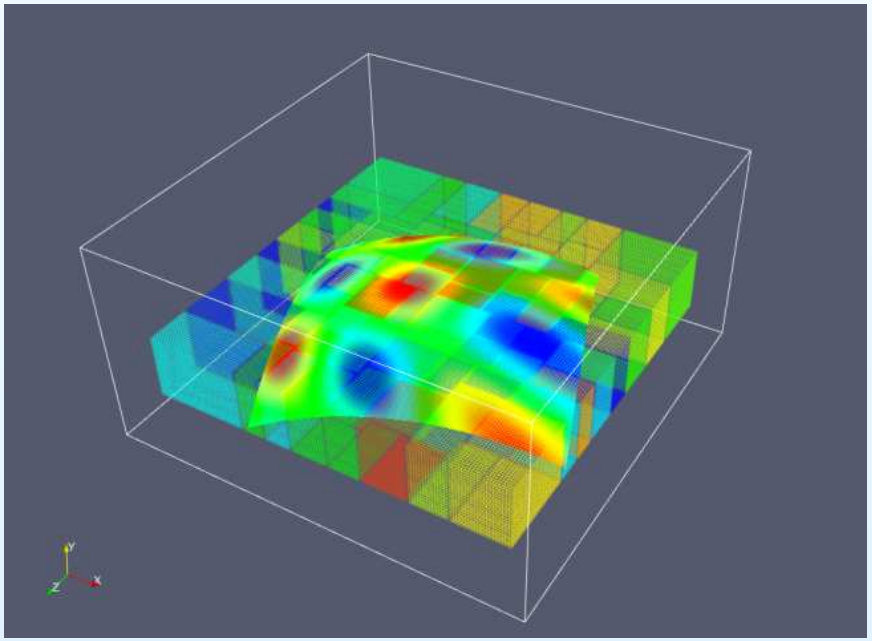
$$\begin{array}{ccc} Src_R & \xrightarrow{\mathcal{C}} & Dst_R \\ \mathcal{A}\uparrow & & \mathcal{B}\uparrow \\ Src_s & \xrightarrow{\mathcal{I}} & Dst_d \end{array}$$

Where $\mathcal{A}$ and $\mathcal{B}$ are the mesh migration communication spec's and $\mathcal{C}$ is the local interpolation operator. The subscripts $_{s,d,R}$ are the source,destination and rendezvous decompositions. We have $\mathcal{I} = \mathcal{B}^\top \circ \mathcal{C} \circ \mathcal{A}$.

We ship fields and results using the mesh migration comm spec's $\mathcal{A}$ and $\mathcal{B}^\top$.

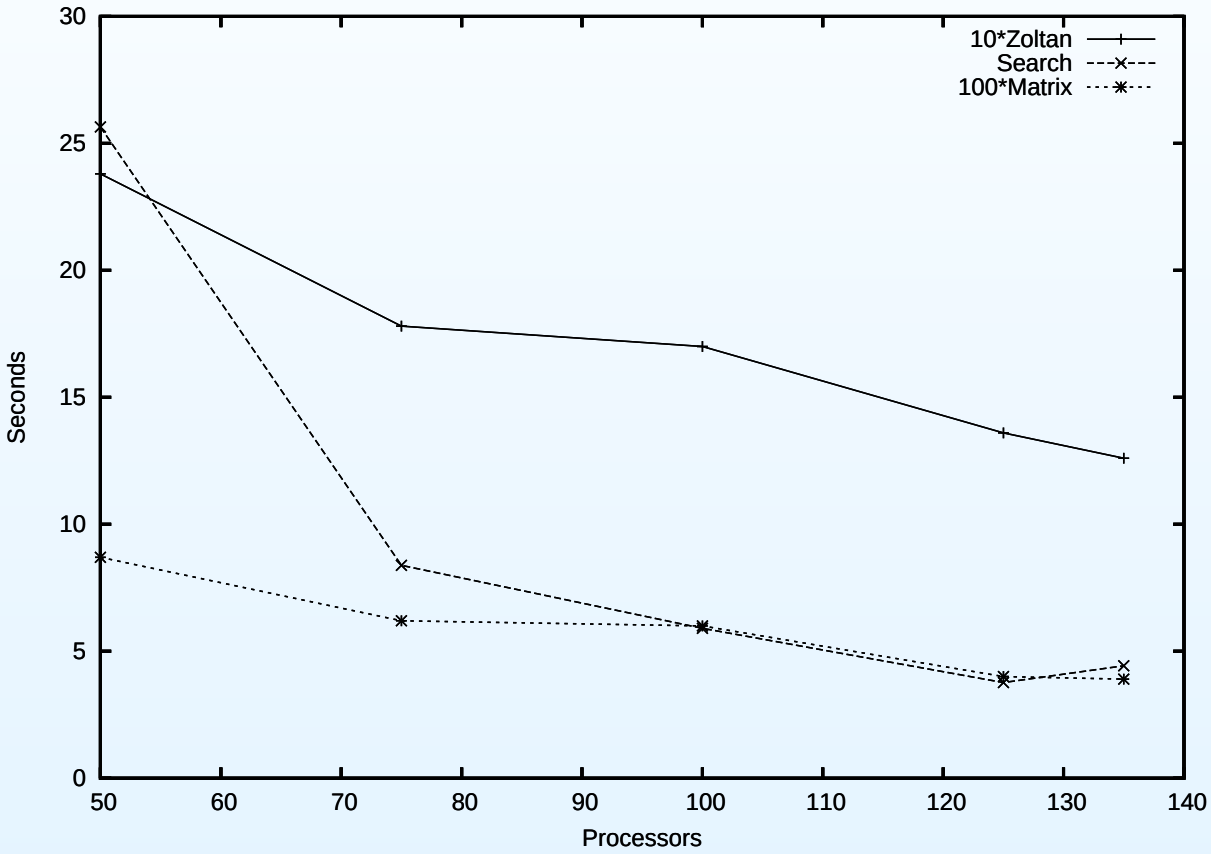
Results

We interpolate from a 3d volume to a 2d manifold (bilinear). The volume contains 4M cells, the surface contains 1.9M cells. Only 984K source cells intersect the destination bounding box. Using UCAR's lightning cluster. 128 nodes, each with two 2.2GHz AMD Opteron processors, 4GB memory shared. 128-port Myrinet switch through single-port Myrinet PCI adaptor.



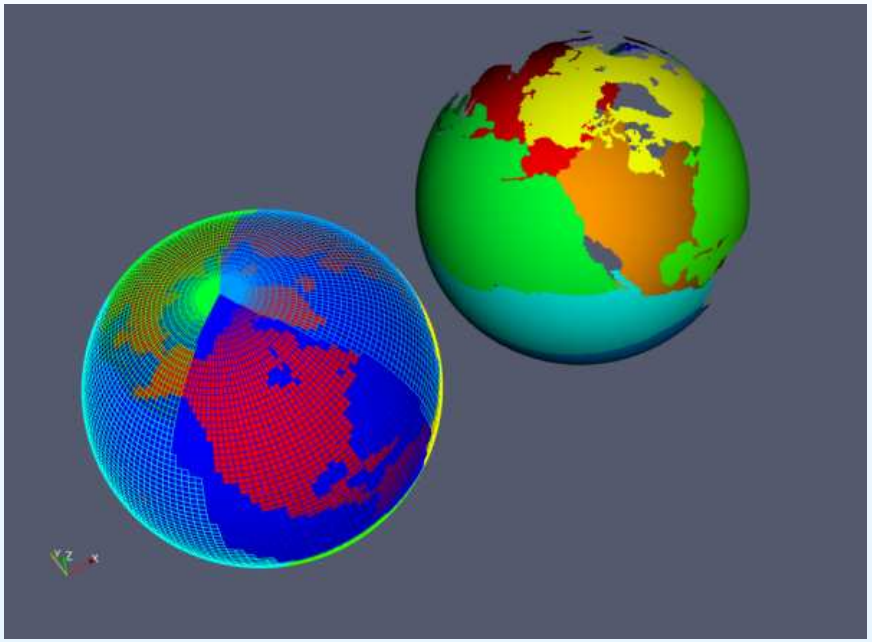
Results, timing

Timings:



Example, analytic wind field

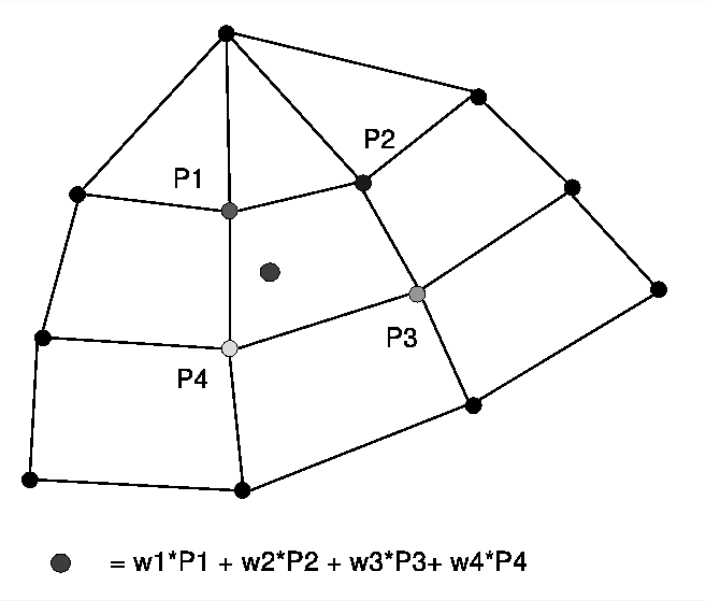
We interpolate an analytic wind field from a standard lat/lon earth grid to the POP ocean grid.



We use both bilinear and a patch-interpolation method.

Patch interpolation

The patch method averages the results of least squares polynomial fit around each node.



The value at a point within an element is a weighted average of the values of each patch.

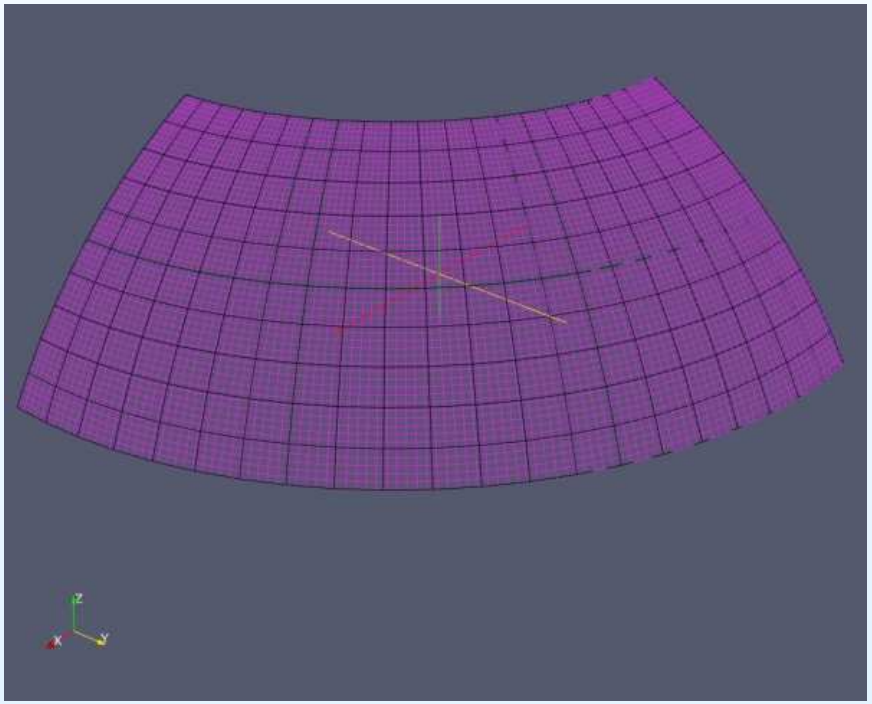
Improved derivatives

The patch method results in better approximation to derivatives, such as curl when compared to bilinear interpolation.
Example, analytic wind stress:

Improved derivatives

The patch method results in better approximation to derivatives, such as curl when compared to bilinear interpolation.
Example, analytic wind stress:

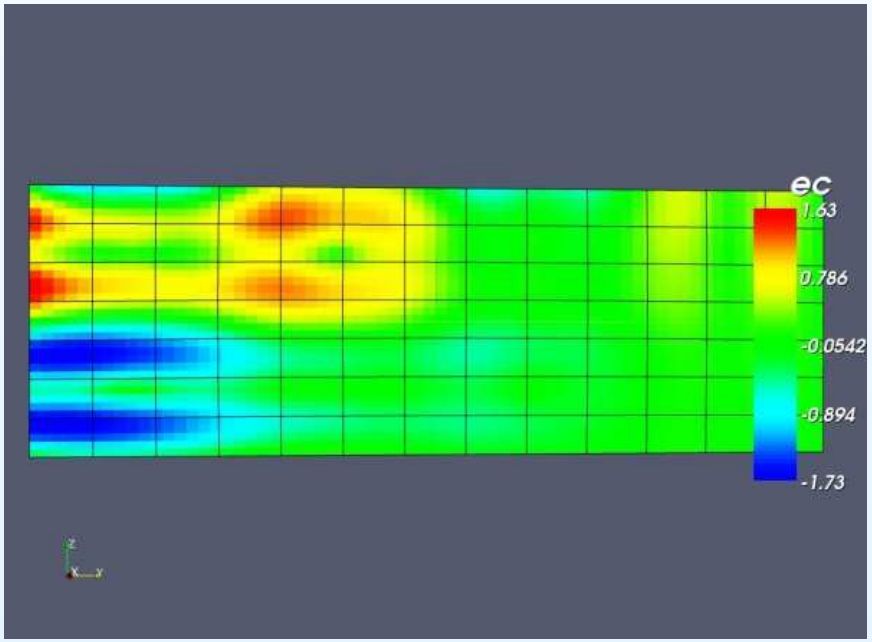
Mesh scales



Improved derivatives

The patch method results in better approximation to derivatives, such as curl when compared to bilinear interpolation.
Example, analytic wind stress:

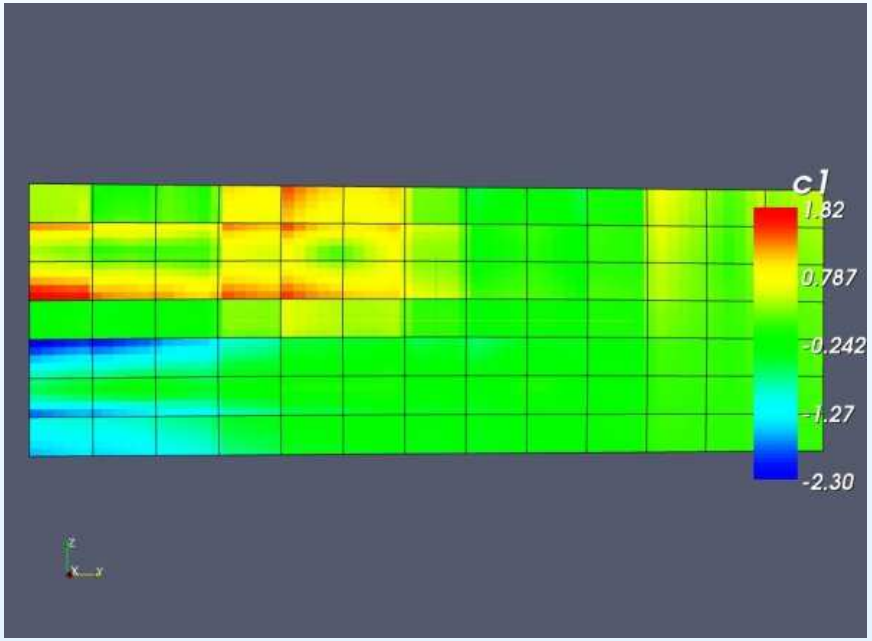
Exact curl



Improved derivatives

The patch method results in better approximation to derivatives, such as curl when compared to bilinear interpolation.
Example, analytic wind stress:

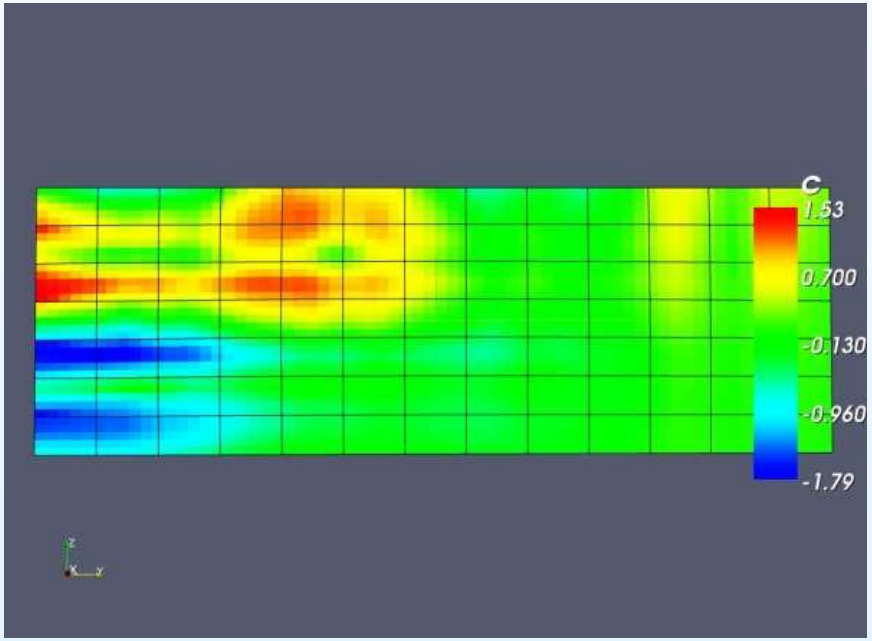
Curl of bilinear interpolant



Improved derivatives

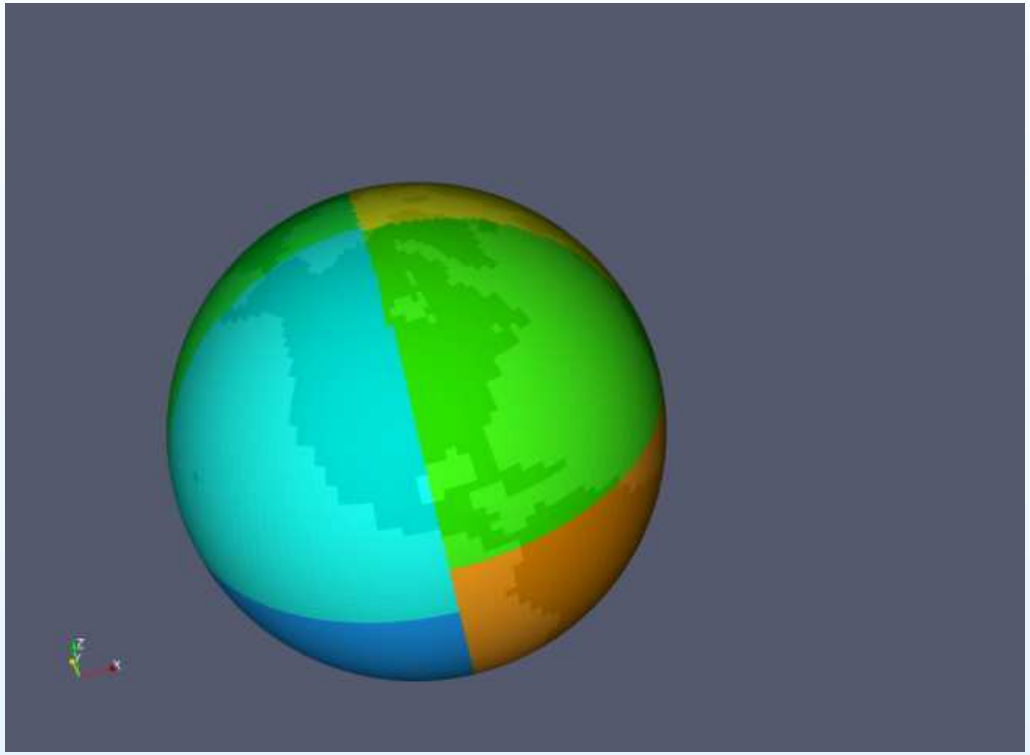
The patch method results in better approximation to derivatives, such as curl when compared to bilinear interpolation.
Example, analytic wind stress:

Curl of patch recovered interpolant



Rendezvous grid

The rendezvous grid decomposition for these meshes



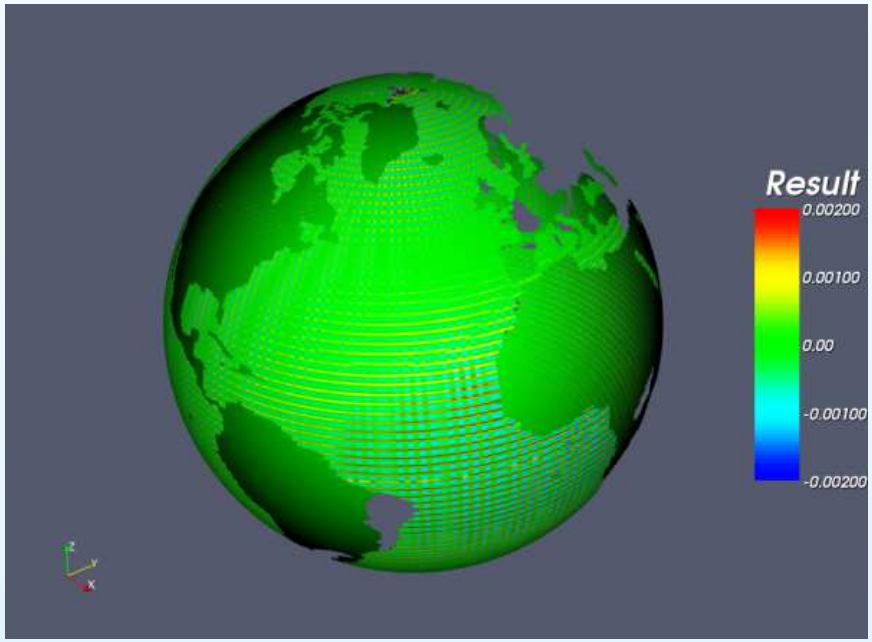
Patch vs Bilinear gradients

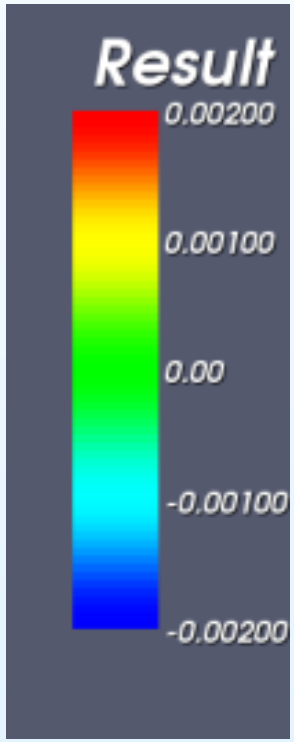
The patch method produces much more accurate derivatives, curl is shown here.

Patch vs Bilinear gradients

The patch method produces much more accurate derivatives, curl is shown here.

Patch recovered Error

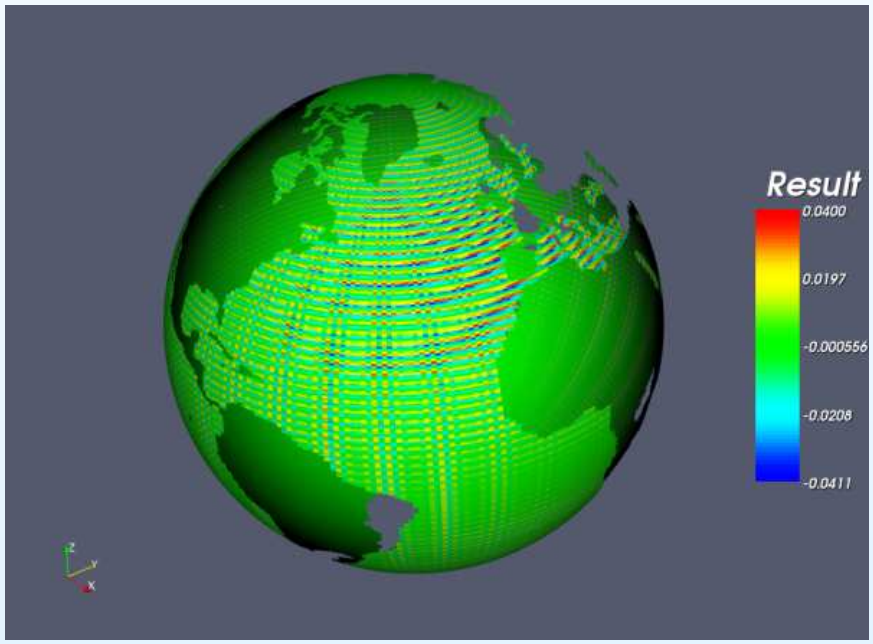


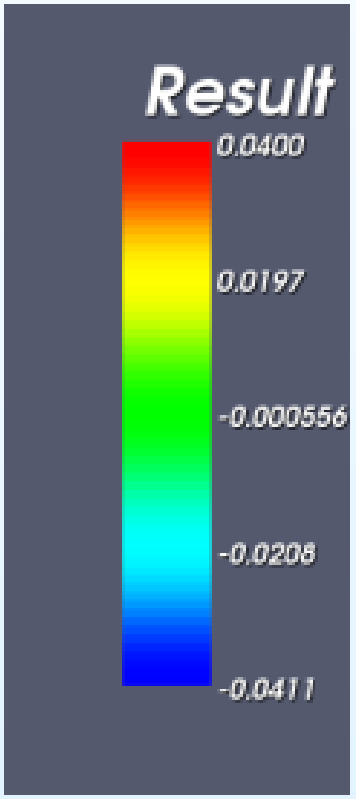


Patch vs Bilinear gradients

The patch method produces much more accurate derivatives, curl is shown here.

Bilinear Error





Conclusions

The Rendezvous algorithm presents a straightforward and robust method to perform parallel regridding and (with the addition of a fractional area kernel) to compute the exchange grid.

Conclusions

The Rendezvous algorithm presents a straightforward and robust method to perform parallel regridding and (with the addition of a fractional area kernel) to compute the exchange grid.

Due to non-regular decompositions in rendezvous space, an unstructured mesh representation is appropriate there.

Conclusions

The Rendezvous algorithm presents a straightforward and robust method to perform parallel regridding and (with the addition of a fractional area kernel) to compute the exchange grid.

Due to non-regular decompositions in rendezvous space, an unstructured mesh representation is appropriate there.

For some meshes, the matrix multiplication should be performed in rendezvous space, to combat load imbalance.

References

- "A parallel rendezvous algorithm for interpolation between multiple grids," Steve Plimpton, Bruce Henderickson, James Stewart. Proceedings of the 1998 ACM/IEEE conference on Supercomputing, 1998.
- "SIERRA Framework Version 3: Core Services Theory and Design," H. Carter Edwards. Sandia National Laboratories, report SAND2002-2616, 2002.
- "Architecture of the Earth System Modeling Framework," Hill, C., C. DeLuca, V. Balaji, M. Suarez, and A. da Silva. Computing in Science and Engineering, Volume 6, Number 1 (2004).
- "Zoltan: Data Management Services for Parallel Dynamic Applications," Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson and Courtenay Vaughan. Computing in Science and Engineering, Vol 4 Number 2, 2002.