PORTAGE - A Flexible Conservative Remapping Framework for Modern HPC Architectures

P. Crossman, G. Dilts, R. Erramilli, C. Ferenbaugh, R. Garimella, A. Herring, E. Kikinzon, C. Malone, N. Ray, M. Rogers

Los Alamos National Laboratory

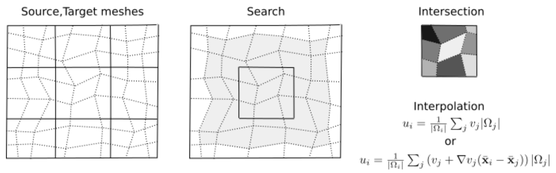
1. Summary

We demonstrate the design and performance of Portage, a flexible, conservative, parallel remapping framework designed for modern HPC platforms.

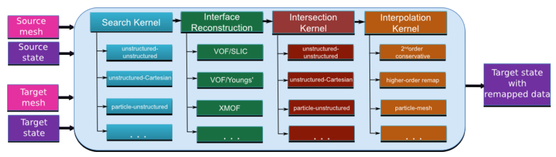
2. Introduction

- ***Remapping*** - transfer of field data between two computational meshes (or point clouds)
- Remapping is used to transfer data between
  - two computational codes and their meshes,
  - two physics modules of a single code, or
  - poor quality and good quality meshes in Arbitrary-Lagrangian Eulerian (ALE) methods
- Conservative, Accurate, Bounds-preserving, Fast, Scalable

3. Schematic of Exact Intersection Remapping



4. Portage Framework Design



- Portage is templated on most components - mesh, state, search, intersect and interpolate classes.
- Allows clients to use custom components or default ones
- Portage takes care of executing the algorithm with distributed, threaded and GPU parallelism
- Functional design will allow use task parallelism (e.g., FleCSI/Legion)

5. Supplied Component Classes

- Mesh/State: Simple Mesh (serial, cartesian mesh, scalar fields) Jali (`github.com/lanl/Jali`), FleCSI (`github.com/laristra/flecsi`)
- Search: *kd*-tree search
- Intersect: R2D/R3D (`github.com/devonmpowell/r3d`)
- Interpolation: 1st-, 2nd- and 3rd-order accurate interpolation with Barth-Jespersen gradient limiting for bounds-preservation

6. Handling Unstructured Polyhedral Meshes

- Default mesh interface assumes unstructured polyhedral meshes, possibly non-convex with curved faces
- Supplied search, intersect, interpolation handle such meshes
- Can replace with more efficient components for specialized meshes
- Some basic functionality expected from supplied mesh class (node coordinates, cell $\rightarrow$ nodes, node $\rightarrow$ cells, etc.)
- Rest can be constructed by Portage (if not supplied)

7. Distributed memory parallelism

- If source cells overlapping target cell are not on MPI rank, one has to fetch them
- Portage copies and transmits mesh/field data so that clients don't have to redistribute
- Check if bounding boxes of target mesh on rank i overlaps with source mesh from rank j
- If there is overlap, get source mesh and field data from rank i to rank j - no communication needed subsequently

8. On-node parallelism

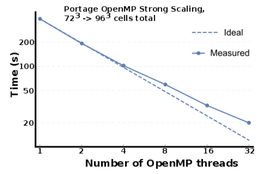
- On-node parallelism achieved through NVidia Thrust parallel constructs like thrust::transform or thrust::for_each
- Similar constructs in Kokkos (Trilinos) and C++17 standard
- Abstracted out as Portage::transform so that we can call std::transform if Thrust not enabled
- Thrust can be directed to run the parallel constructs using OpenMP, Intel TBB or CUDA backends
- Requires search, intersect and interpolate to be written in functional style - be functors with no side effects
- Code is fully tested with OpenMP - early version of 1st order remap tested with CUDA

9. Test Machine - ASC Snow at Los Alamos National Lab

Machine: Snow
Specs: Based on Penguin Computing Inc.'s TundraES racks 368 Intel Broadwell nodes, 2 sockets/node, 18 cores/socket (13248 cores)
Memory/Node: 128GB **Interconnect**: Intel Omnipath
Peak performance: 445 TFlop/s

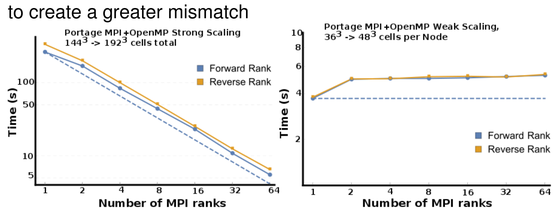
10. Scaling Results

10.1 OpenMP scaling on single node



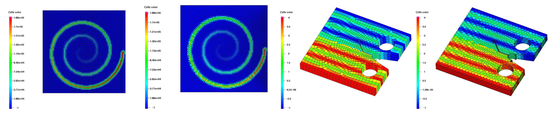
10.2 Results - MPI+OpenMP Strong, Weak Scaling

Standard partitioning refers to closely aligned partitioning for source and target mesh
Reverse partitioning reverses the MPI ranks for the source mesh to create a greater mismatch



11. Results - Mesh-Mesh Remapping

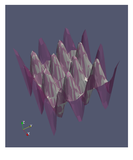
2nd order accurate with Barth-Jespersen

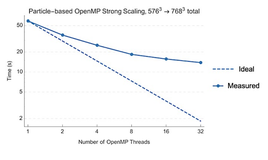


Source field 2D Cartesian mesh | Target field 2D Polygonal mesh | Source field 3D tetrahedral mesh | Target field 3D polyhedral mesh

12. Remapping between Particle Swarms (Point Clouds)

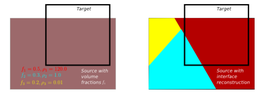
- Portage can transfer data between two particle swarms
- Estimation of field at target particle from Local Regression Estimate (LRE)[Dilts, G.A.] on source swarm
- Currently only on-node parallelism. Distributed parallel implementation coming soon.
- Lack of scaling on-node likely due to poor search scaling
- Portage can also remap from mesh-mesh using particles as an intermediary



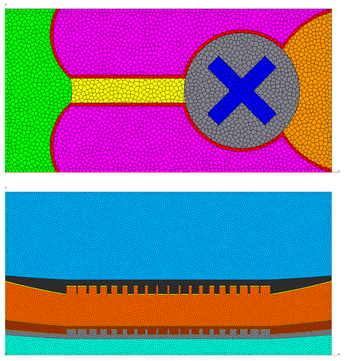


13. Multi-material Remapping and Interface Reconstruction

- Remapping of quantities when souce cells may have multiple materials
- Only material values of field and volume fractions of materials in each cell known on source mesh
- Need to temporarily subdivide source cells into pure material polyhedra by *interface reconstruction*
- Interface reconstruction package **Tangram** in the works



14. Tangram Results with XMOF2D Plugin



15. Other ongoing improvements

- Remapping of vector and tensor fields
- Improvement of intersection efficiency (most time spent here)
- Minimizing data exchange between processors
- Automatic data dependency resolution using FleCSI/Legion
- Automatic task parallelism through FleCSI/Legion
- Cartesian mesh, Spherical mesh and other specializations

16. Availability

Open Source at http://www.github.com/laristra/portage

17. Acknowledgments

This work was performed under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396, and was supported by the DOE Advanced Simulation and Computing (ASC) program.