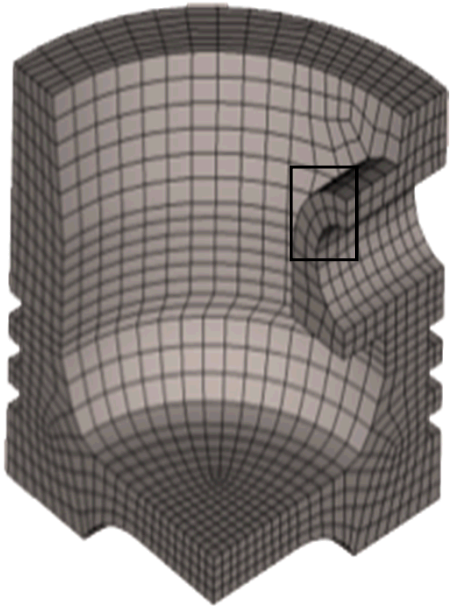
SAND2017-8825PE



Scaling Post-Meshing Operations on Next Generation Platforms

Roshan Quadros, Brian Carnes, Madison Brewer, and Byron Hanks

DOE Centers of Excellence Performance Portability Meeting

Aug 22-24, 2017

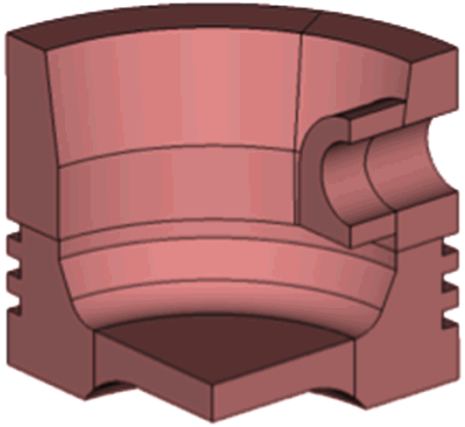
Denver

Sandia National Laboratories is a multi mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Layout of Presentation

- Geometry & Meshing Efforts at SNL
- Scaling Post-Meshing Operators
  - Refinement
  - Smoothing
  - Projection
- Future Work
- Conclusion

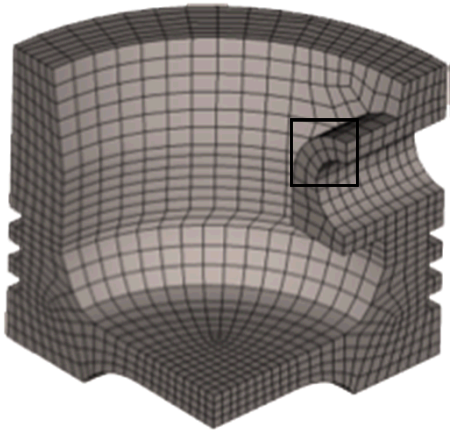
CUBIT and Percept combined workflow for generating large meshes



CUBIT provides extensive capabilities for preparing geometry and generating an initial mesh

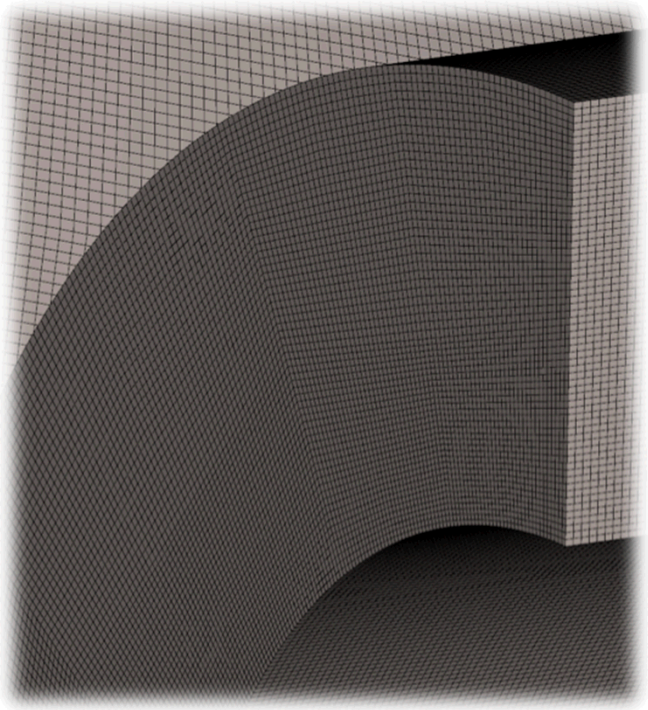
https://cubit.sandia.gov

CUBIT



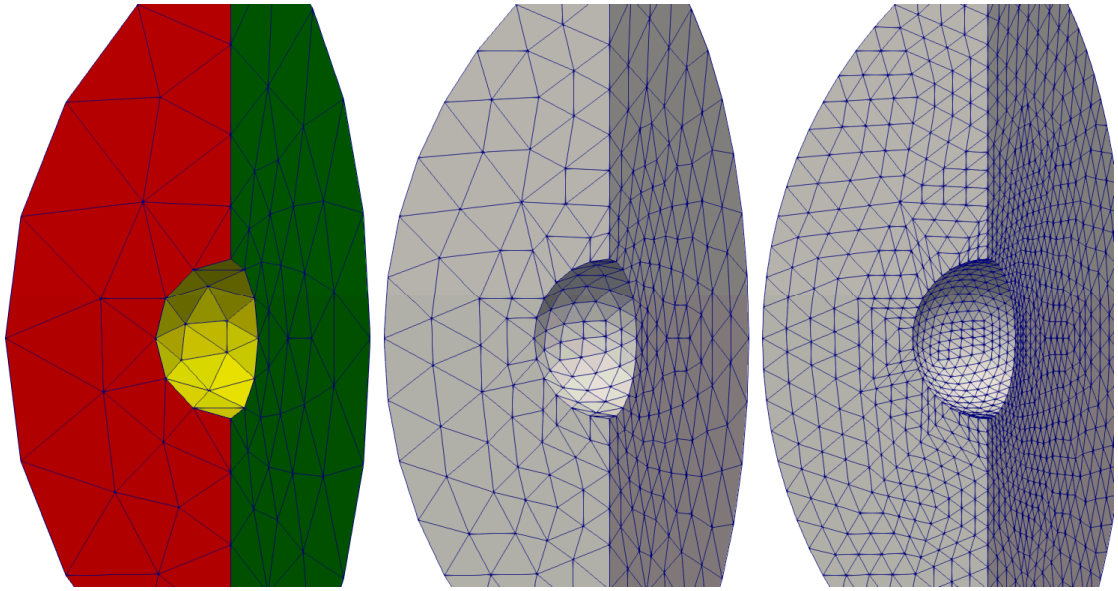
Initial Mesh Generation: advanced meshing algorithms for Tri, Quad, Tet, and Hex mesh generation

Percept



Parallel decomposition, refinement, smoothing, & projection

Post-Meshing Operations: Refinement->Projection->Smoothing



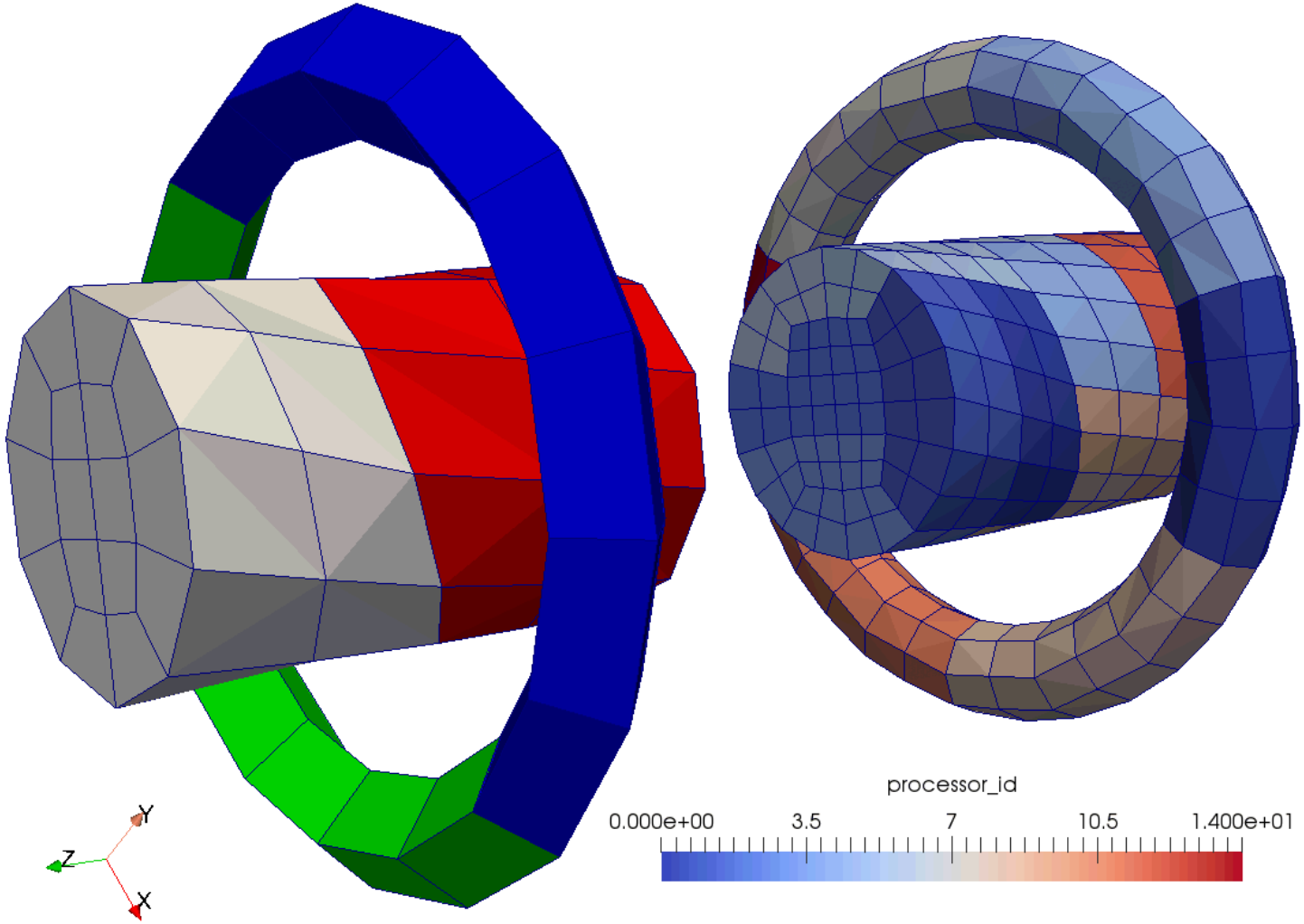
Mesh refinement workflow:

- Generate refined meshes in memory from an existing mesh
- Project new boundary nodes onto geometry
- Smooth interior mesh nodes to improve mesh quality

Supported mesh types:

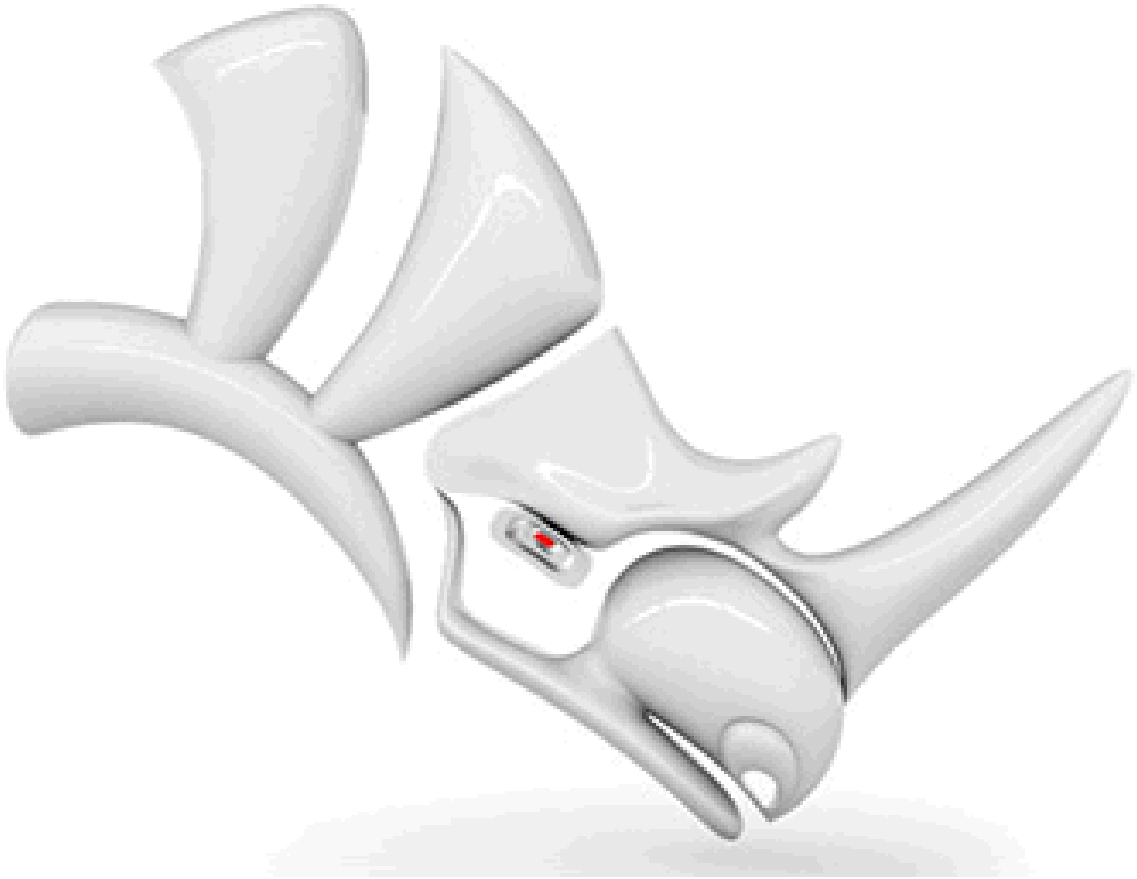
- block-structured
- unstructured
- hybrid

Post-Meshing Operator #1: Projection

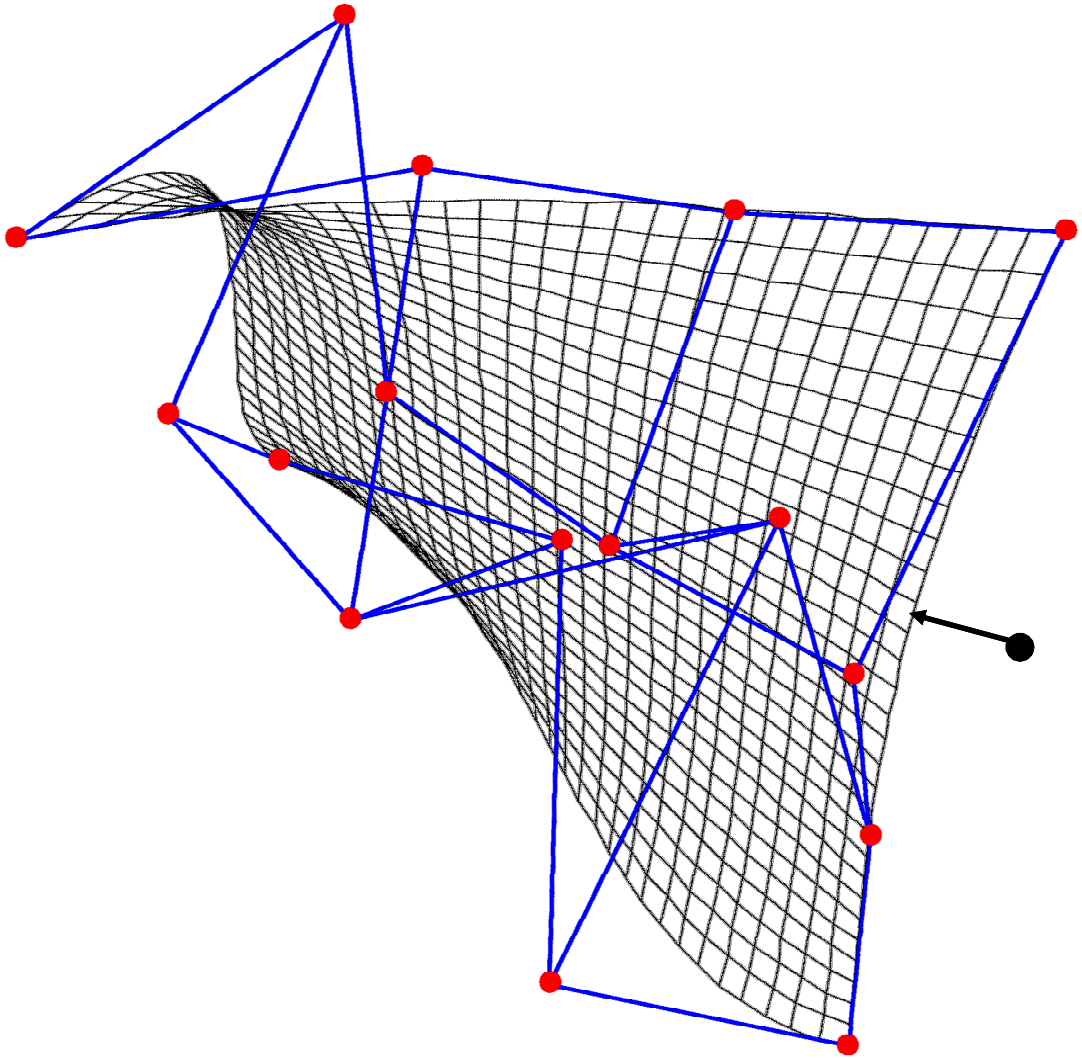


Geometry Kernel: OpenNURBS

- Open Source from www.rhino3d.com
- Lightweight & easy to Port
- Query operations are thread safe
- Supports various curves & surface definitions



Parallel Kernel:
Project Points on a NURBS Surface



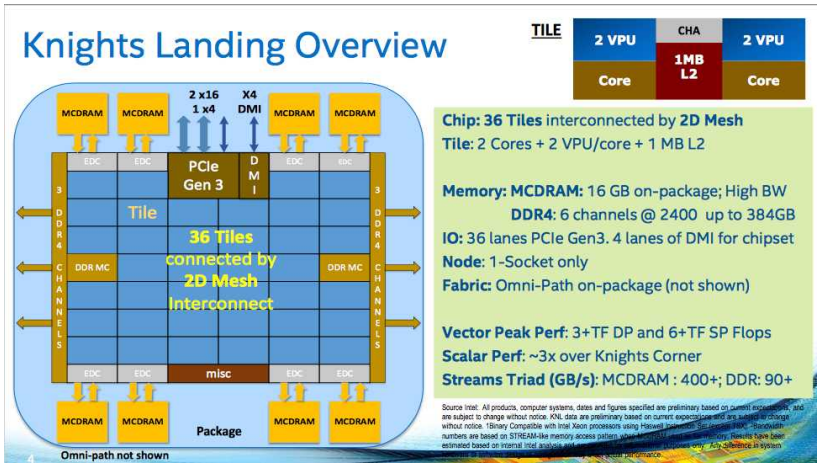
Programming Model: MPI + Kokkos (OpenMP)

Three levels of parallelism is required:

1) Distributed memory parallelism via MPI

2) Shared memory thread level parallelism on the MIC device using Kokkos with OpenMP runtime

3) Vectorization for Vector Processing Unit (VPU)

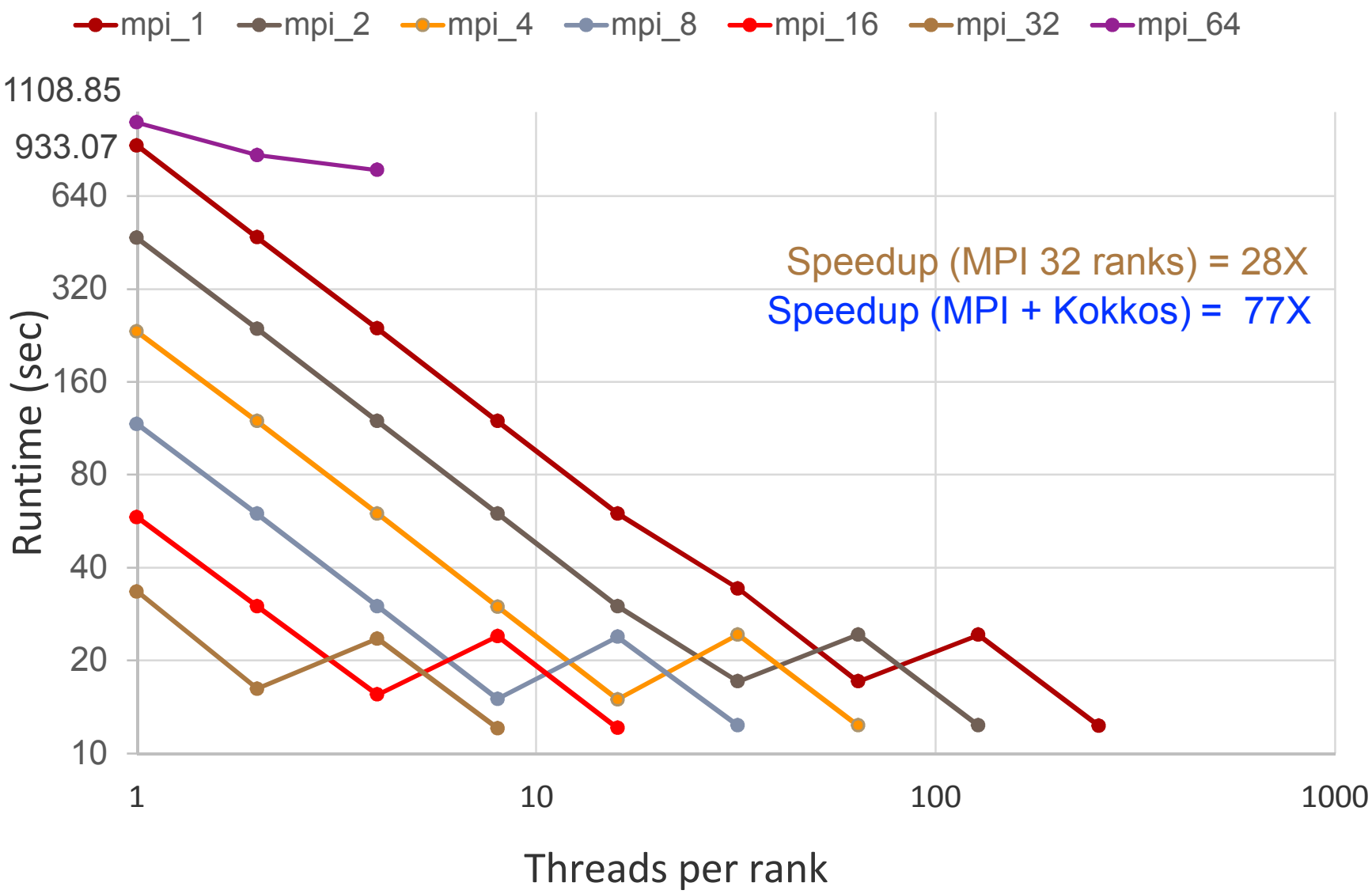


Hardware:
Trinity testbed containing 72 core KNL
Image courtesy of http://www.hotchips.org

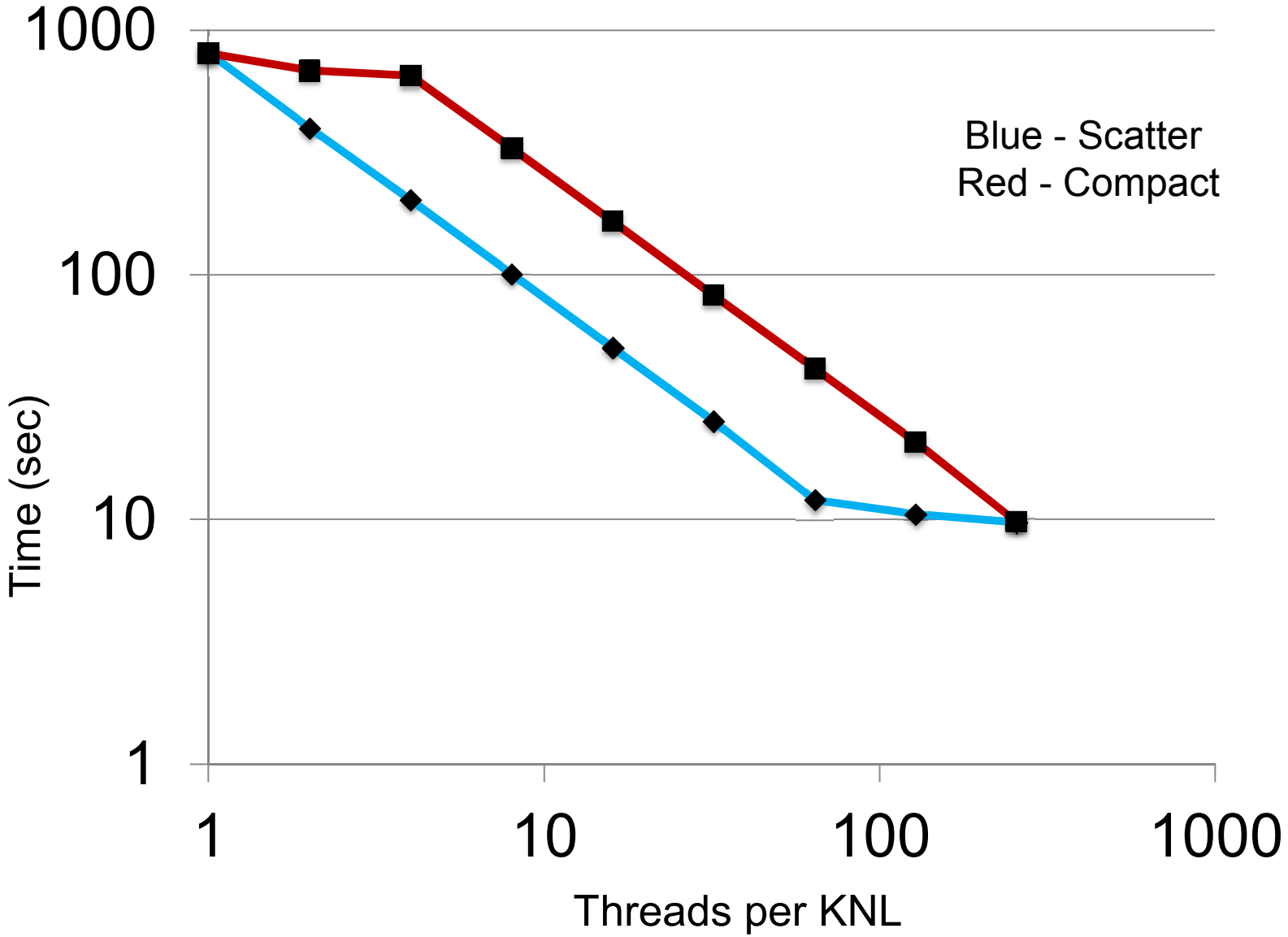
Programming Model: MPI + Kokkos (OpenMP)

```
{  // MPI distributes data to n processes
    ON_3dPoint *buff_p;
    MPI_Comm_size( MPI_COMM_WORLD, &numtasks);
    …
    if ( rank == 0 ){
        for( int r=1; r < numtasks; r++){
            …
            ierr = MPI_Send ( p_start, num_pnts*3, MPI_DOUBLE, r, Tag, MPI_COMM_WORLD);
        }
    } else{
        ierr = MPI_Recv ( buff_p, num_pnts*3, MPI_DOUBLE, MPI_ANY_SOURCE, Tag, MPI_COMM_WORLD,
&status );
    }
    projection_method( buff_p, num_pnts );
}

void projection_method( ON_3dPoint *buff_p,  const int num_pnts ){

    // Kokkos handles thread level parallelism
    Kokkos::parallel_for ( num_pnts, KOKKOS_LAMBDA(const int i ){

        // OpenNURBS API for projecting a point on a surface
        double u, v;
        surface->GetClosestPoint( buff_p[i], &u, &v );
        ON_3dPoint projected_pt = surface->PointAt( u, v );
     });
}
```

Point Projection Scaling Results on KNL



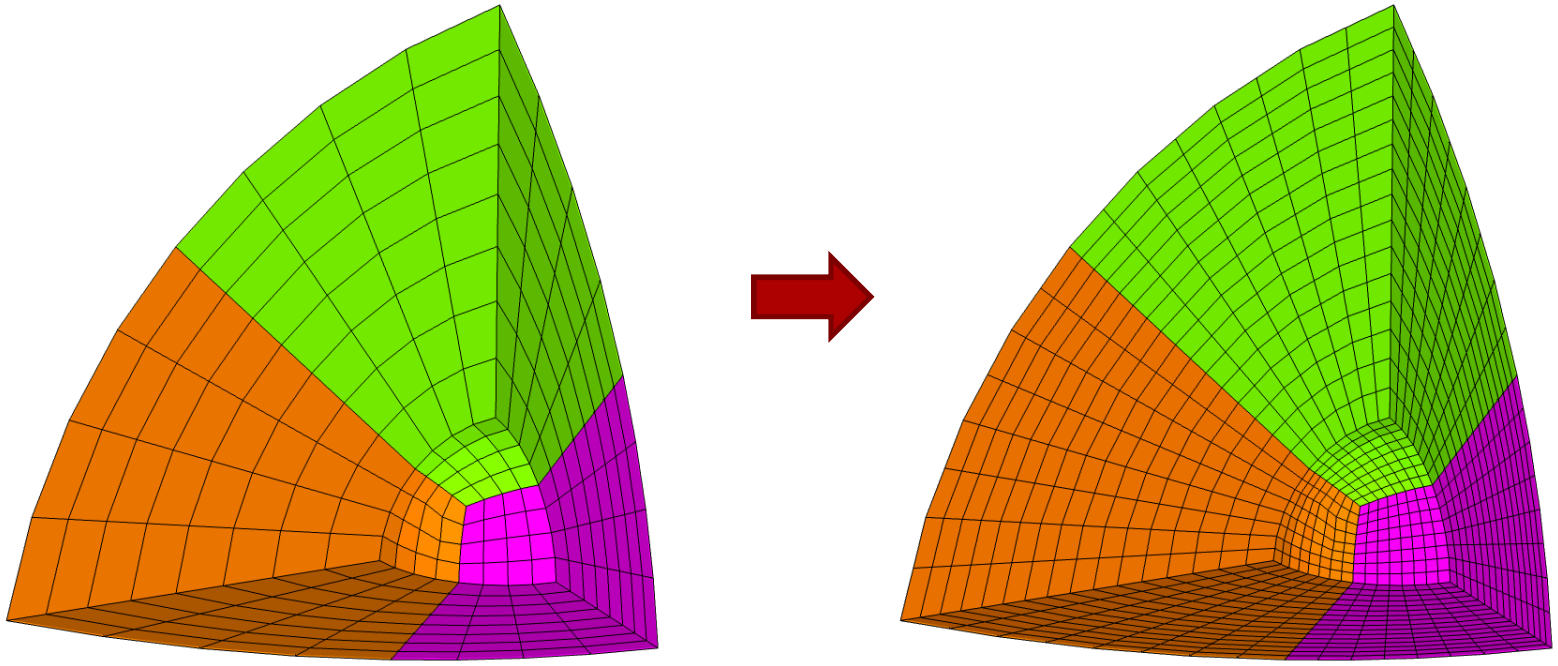
Thread Affinity on KNL



Post-meshing Operator #2: Refinement

Structured grid refinement was a relatively simple process

- Removed pointers and references to main memory objects
- Replaced structured grid data structures with Kokkos views
- Algorithm :
    - Allocate new mesh (view)
    - For each block (in serial)
        - For each element in old mesh (in parallel)
            - Interpolate coordinates for new mesh
            - Transfer existing node coordinates



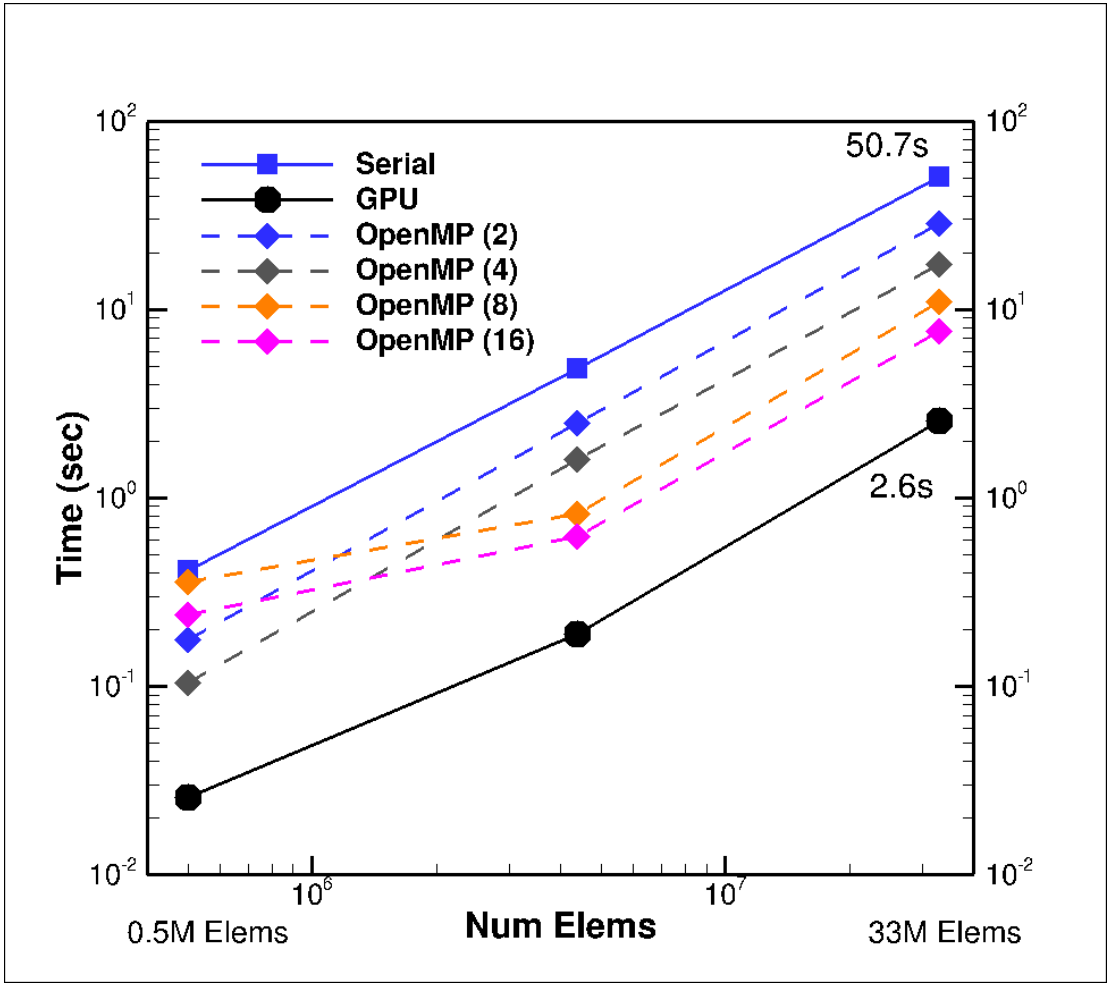
Scalability of Refinement

Sequence of meshes
- 0.5M, 4M, 33M elements
- multiple blocks (12)

Compare
- serial
- GPU
- threading (OpenMP)

Better scalability with increasing problem size

Blocks were refined sequentially

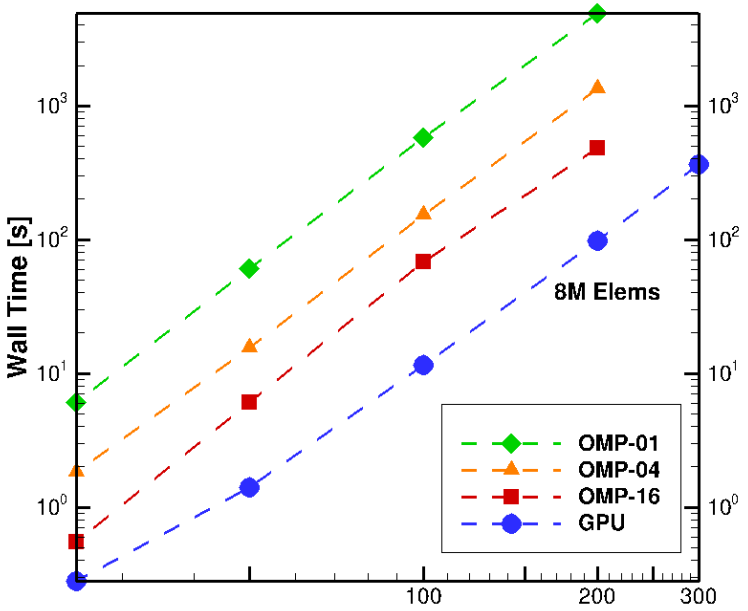


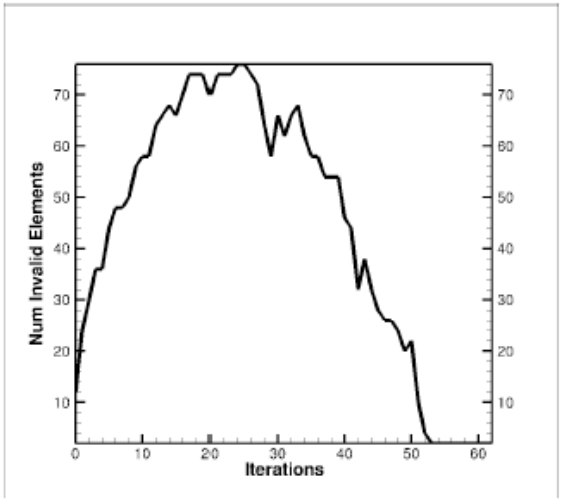
Post-Meshing Operator #3: Smoothing

Smoothing structured grids was more complex process

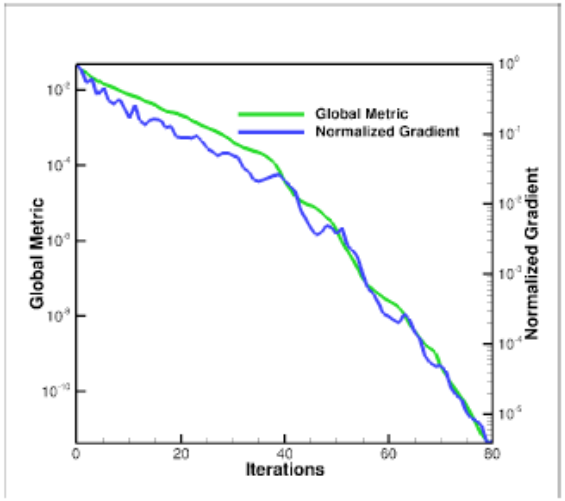
- compute global quality metric and gradient
- nonlinear CG optimization with line search
- communication between structured blocks (gradients)

Example: smoothing a large cube with initial randomly perturbed nodes

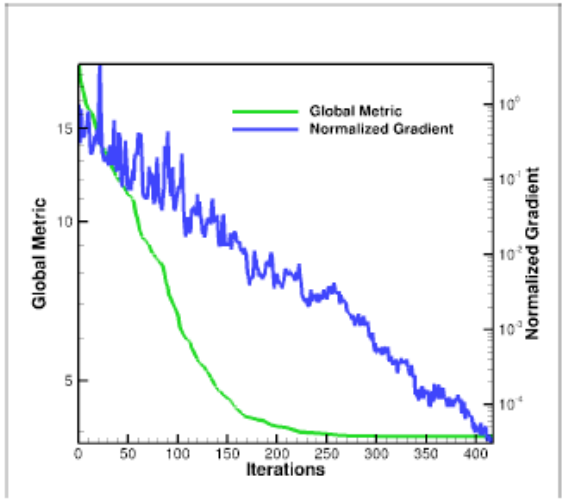




Untangling: invalid elements



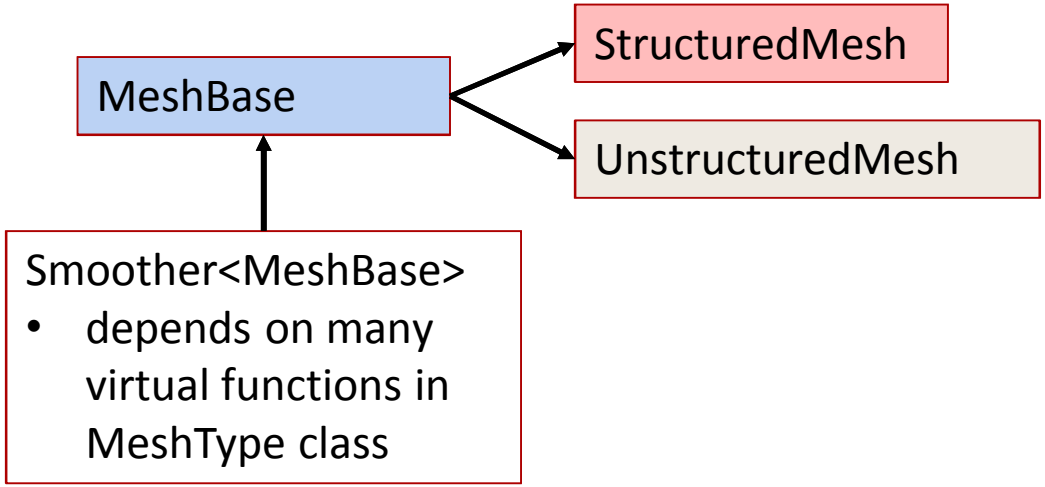
Untangling: metric/gradient



Smoothing: metric/gradient

Smoothing Pitfall: Abstract Interfaces

- Abstract mesh interface for both structured and unstructured
  - high cost of kernel calls
  - sub-optimal interface to structured grid
  - functions not safe for threads or GPU
- Suggestion for abstract interfaces:
  - designed with shared memory (Kokkos) from start
  - otherwise, opt for specialized interfaces.



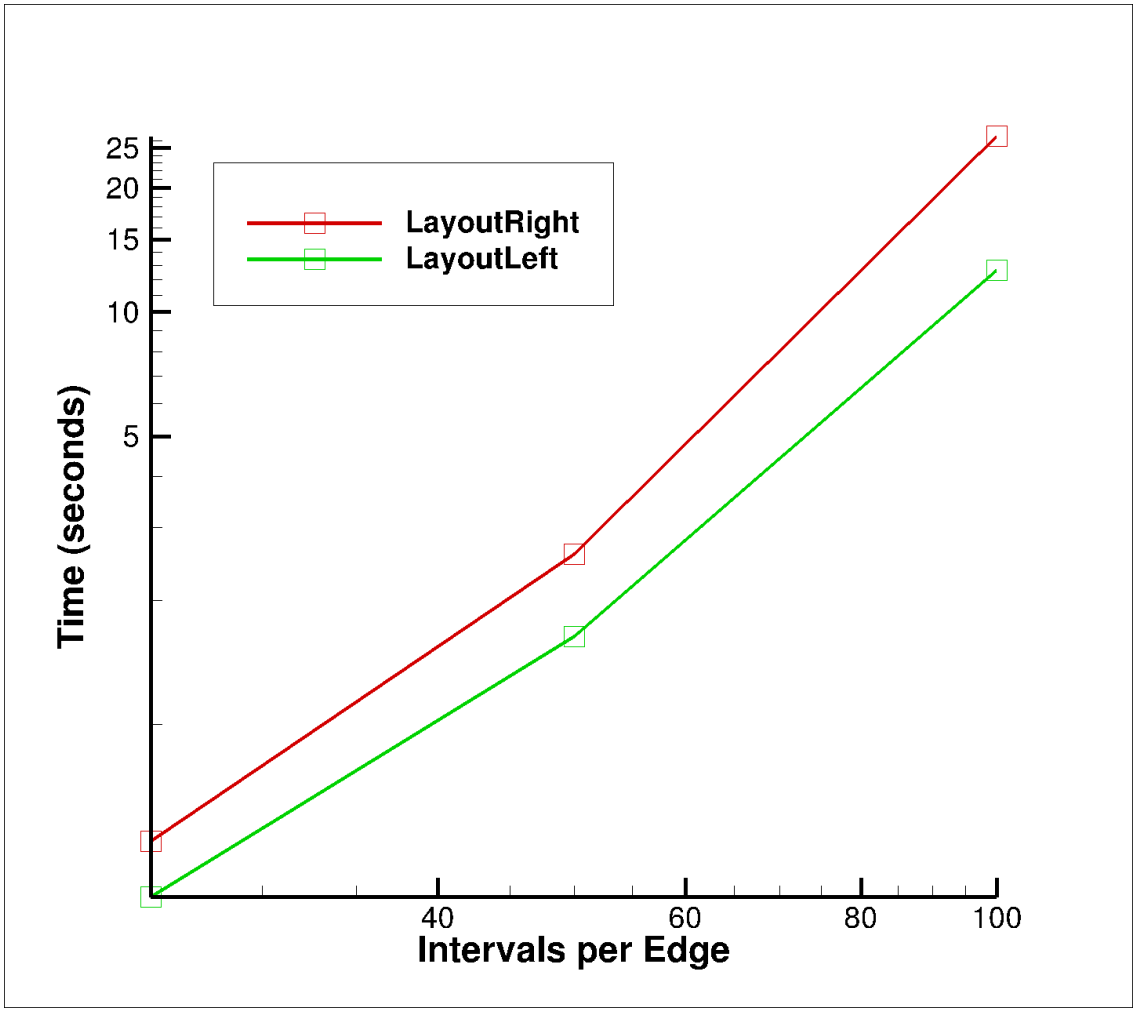
Smoothing Pitfall: Hardware Differences

- Total metric (long double) was sum of individual metrics (double)
- Problematic for GPU builds as CUDA only uses up to 64 bits (double precision) for floating point representation.
- Causes illegal memory access:
  - **cudaDeviceSynchronize() error(cudaErrorIllegalAddress): an illegal memory access was encountered**

- Certain STL classes and functions are problematic on GPUs
  - array, unorderedmap, vector, set, …
    - array => Kokkos::Array
    - unorderedmap => Kokkos::UnorderedMap
  - std::max was rewritten locally

Smoothing Pitfall: Memory Layout

- Memory layout initially caused very poor performance on the GPU
- Smoothing test case: perturbed cube followed by mesh smoothing



Future Work

- **Projection:**
  - Study high water mark of memory usage for different combination of MPI ranks and Threads per rank
- **Unstructured Refinement:**
  - More complicated algorithm than structured
  - Example: determine number of new nodes
    - use Kokkos map to store needed nodes
    - values stored for every mesh edge/face
    - map also used to interpolate new coordinates
- **Smoothing:**
  - Investigate other smoothing algorithms (elliptic smoother)

Conclusion

- MPI+ Kokkos programming model was used for scaling post-meshing operators such as refinement, smoothing, and projection
- Kokkos performance portability library assisted in supporting heterogeneous architectures. Scaling studies were performed on both KNL and GPUs

Thank You