SAND2018-1531PE

STK Mesh Modification on GPU

Applied Computer Science (ACS) Technical Exchange Meeting

Feb 16, 2018

Presenter Alan Williams

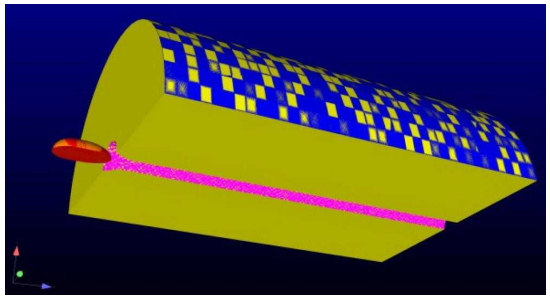
STK-Mesh Dev. Team: Manoj Bhardwaj, Dave Glaze, Tolu Okusanya, Johnathan Vo, Riley Wilson

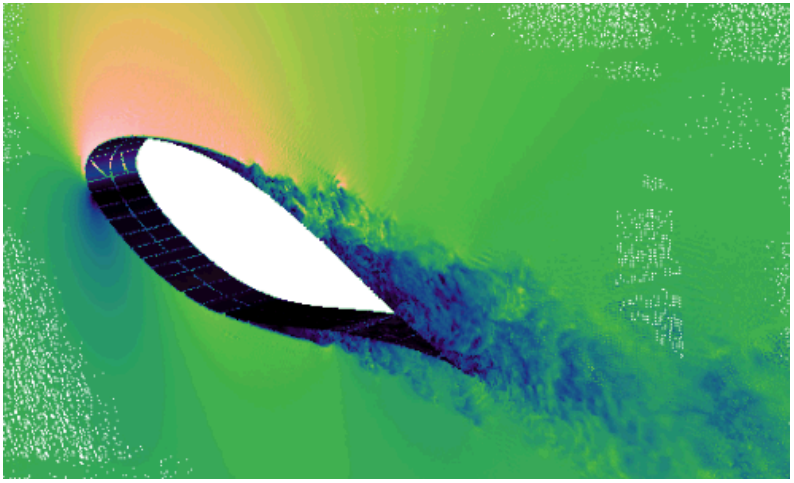
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

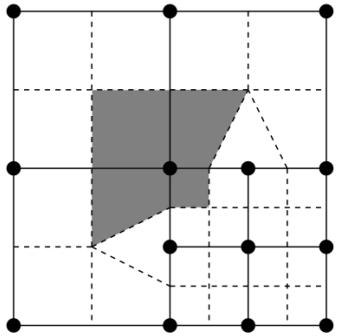
Introduction to STK-Mesh
(STK – Sierra Tool Kit)

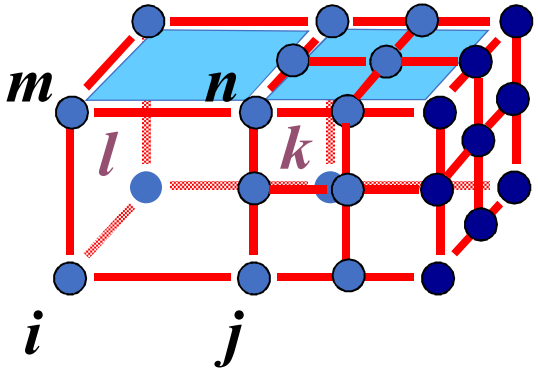
Parallel, unstructured Mesh database

- Heterogeneous element types
- Field-data on subsets (Parts)
- Dynamic modification (refinement, remeshing, rebalancing, …)

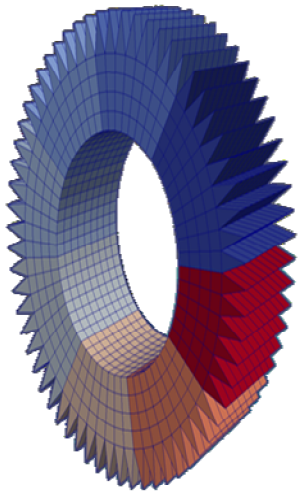






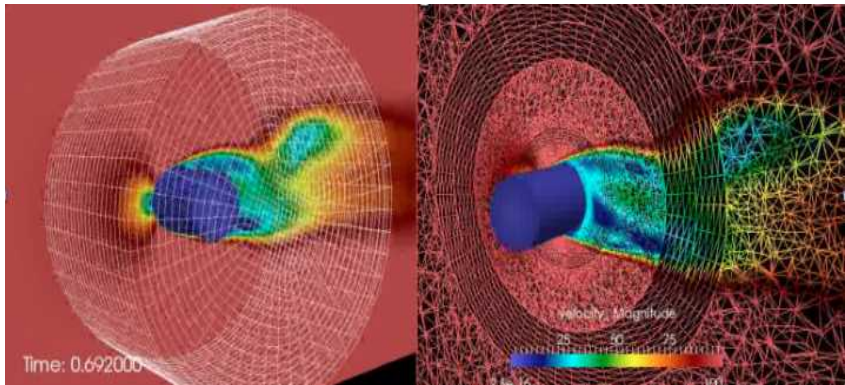


What is Mesh Modification?



Structural/Topological changes to the mesh

- Creating or deleting mesh entities
  - Adaptive Refinement
- Moving mesh entities from one Part to anoth
  - Moving front calculations
- Creating or deleting connections between entities (e.g., element-to-node connectivity)
  - Adaptive Refinement
- ‘Ghosting’ mesh entities from one MPI rank to others
  - Contact, Sliding Interface
- Moving mesh entities from one MPI rank to others
  - Dynamic rebalancing

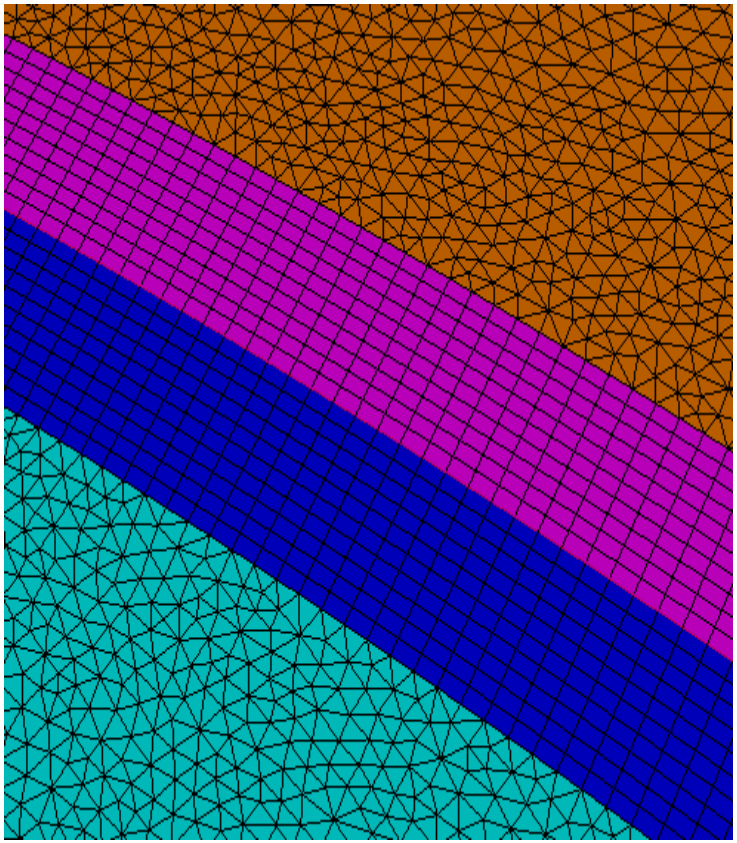


Why is mesh modification difficult on GPU?

1. Memory management:
   - Mesh modifications typically involve heap memory allocation/deallocation, which isn't possible on GPUs.
   - ➢ Can we use memory-pool-like approaches with a pre-allocation to simulate dynamic heap allocations?

2. MPI communication:
   - Modification requires communication to ensure global mesh consistency
   - ➢ Can MPI reach directly into GPU (device) memory?

3. Will on-GPU modifications be "thread" safe and high-performing?



We're just getting started. So…
Progression of unit tests to explore issues

1. GPU-aware MPI communication
   - simple unit-test to send/recv GPU memory directly

2. Single Local Mesh modification
   - GPU memory management
   - Single MPI-rank, single GPU initially, to avoid MPI questions

3. Tests for more mesh-modifications, in parallel, etc

GPU Mesh Modification

A core concept in STK-Mesh is the Bucket –
a contiguous allocation holding a subset of the mesh.

CPU (host)

```
class Mesh {
  …
  std::vector<Bucket*> buckets;
  …
};
```

```
class Bucket {
  …
  std::vector<DataType> data;
  …
};
```

GPU (device)

```
class Mesh {
  …
  Kokkos::View<Bucket*> buckets;
  …HostMirror hostBuckets;
  …
};
```

```
class Bucket {
  …
  Kokkos::View<DataType*> data;
  …HostMirror hostData;
  …
};
```

GPU Mesh Modification

Simulate heap memory allocation using Kokkos::MemoryPool

GPU (device)

```
class Mesh {
  Kokkos::MemoryPool<ExecSpace> pool;
  Kokkos::View<Bucket*> buckets;
  …HostMirror hostBuckets;
  …
};
```

Need to size the pool at construction, i.e., set max number of buckets, and max capacity per bucket.

```
class Bucket {
  …
  Kokkos::View<DataType*,MemoryUnmanaged> data;
  …HostMirror hostData;
  …
};
```

Bucket memory is tagged as Unmanaged, since it will be ‘views’ into pool memory.

GPU Mesh Modification

Create outer array (View) on host, create inner arrays (Views) on device.

CPU (host)

```
//during mesh construction
buckets = Kokkos::View<..>(numBuckets);

//later, during mesh modification…
Kokkos::parallel_for(numBuckets, [..](..)
{
  mesh.add_bucket(…);
}
…
```

GPU (device)

```
KOKKOS_FUNCTION
void add_bucket(…)
{
  buckets(index).data =
      Kokkos::View<..>(static_cast<..>(pool.allocate(bytesPerBucket)));
  …
}
```

Conclusion, Path Forward

1. Technical issues such as memory management and MPI communication with GPU memory contain challenges but are not road-blocks.

2. Haven’t yet explored performance and thread-safety issues.

3. Developing robust production-quality mesh modification on the GPU would be expensive. We will need to balance this with other capability development priorities.