Parallel Mesh Generation by Distributing CAD Geometry

by Hong-Jun Kim and Tim J. Tautges

Argonne National Laboratory
Mesh generation
- Large portion of entire FEM analysis time
- Hard to reduce

Problem size is getting larger
- Too long mesh time
- Too large memory for one machine
Parallel Mesh Generation
Classification

• **Sequential technique**
  – Delaunay, Advancing Front, Edge Subdivision
  – Quadrilateral, Hexahedral …

• **Degree of coupling b/w subproblems**
  – Tightly, partially coupled and decoupled

• **Partitioning**
  – Task, data
  – Domain Decomposition
    • Continuous domain decomposition
      – Quadtree/Octree, Medial axis
    • Discrete domain decomposition
      – Surface mesh, Coarse background mesh

---

CAD based approach

• **Domain is CAD geometry**
  – OpenCASCADE, ACIS ....
  – Distribute solid model geometries

• **Partitioning unit**
  – Geometry entities
  – Body, Volume, Surface, Curve, Vertex
  – Multiple partitioning units

• **Interface**
  – Shared surface entities
  – Mesh communicated
CAD based approach

• Not limited in specific mesh algorithm
  – All tri/tet mesh algorithms are applicable
  – New tri/tet mesh algorithms added
  – Extended to quad/hex algorithms

• Analysis
  – Geometry evaluation
  – Adaptive mesh

• Component based
  – ITAPS interface
  – iGeom: CGM
  – iMesh: MOAB
  – iRel: Lasso
  – MesKit
Partitioning in CAD approach

- **Volume is partitioned**
  - # of vols > # of procs
  - Assembly model
  - Decomposed model
    - Webcut, automatic decomposition
  - If too larger, artificial boundaries

- **Good partitioning**
  - Minimum volume decomposition
  - Surface partitioning
    - Interface / ordinary

For 3 processors, it is too much
Optimize partitioning

• **Graph**
  – Volume and interface surfaces
  – Best partition with the min. edge cut
    • Data locality and min. communication
  – Non interface surfaces
    • max to min

• **Library**
  – Chaco → Zoltan

• **Stored as Attribute**
  – Application-specific data associated with specific geometry entities
  – No need to broadcast partitioning information
Geometry Distribution

- **Read**
  - Not bad for small geometry

- **Read and delete**
  - Easy to implement
  - Smallest overhead
  - Need disk in compute node

- **Broadcast**
  - Geometry evaluation of remotely partitioned entity
  - Largest overhead in time and memory

- **Broadcast and delete**
  - Frees up unneeded memory
  - Overhead in time

- **Scatter**
  - Complex to implement
  - Small overhead in time and memory

- **Scatter and delete**
  - Delete non-assigned part in root

- **Read parallel**
  - Not implemented yet
  - True parallel read
  - Reads only its portion of the geometry
Geometry Distribution

• ITAPS interface
  – void \texttt{iGeom\_load}( iGeom\_Instance, char const* name,
    char const* \texttt{options}, int* err, int name\_len,
    int options\_len );
  – char* \texttt{options} =
    ";PARALLEL=BCAST\_DELETE;PARTITION=GEOM\_DIMENSION;PARTITION\_VAL=3; PARTITION\_DISTRIBUTE;";

• \texttt{iGeom} interface
  – Component
  – Other parallel application with geometry
Parallel Mesh Procedure

- **Mesh shared edge and vertex entities**
  - Scattered by root proc
- **Mesh assigned interface surfaces**
  - Geometry entity $\Leftarrow \Rightarrow$ Meshset entity
  - Meshset has geometry unique id as tag
- **Send mesh in Meshset**
  - By asynchronous send
- **Mesh ordinary surfaces in parallel**
- **Receive interface surface mesh**
  - By asynchronous receive
- **Mesh volumes in parallel**
- **Export mesh in parallel**
MOAB’s Parallel Mesh Representation

• **Balance**
  – Mesh similar as if it was in serial
  – Parallel mesh-based operations efficiently
    • Parallel export
    • Linked by parallel analysis code

• **Tags, functions**
  – `PARALLEL_PART, PARALLEL_PARTITION, __PARALLEL_SHARED_PROC(S), __PARALLEL_SHARED_HANDLE(S), __PARALLEL_STATUS`
  – Resolve_shared_entities, exchange_ghost_cells

• **Shared entities**
  – Shared
    • `__PARALLEL_SHARED_PROC` : remote processor
    • `__PARALLEL_SHARED_HANDLE` : handle in remote processor
  – Multi-shared
    • `__PARALLEL_SHARED_PROCS` : local + remote processor(s)
    • `__PARALLEL_SHARED_HANDLES` : handles in local + remote processors
Results

- **Cosmea Linux cluster**
  - 32 nodes, 128 cores
  - 2 dual-core Intel Xeon 5130 cpus
  - 16GB memory per node (4GB per core)
  - 19TB shared low-speed NFS/GFF filesystem storage
  - 1 giga-bi ethernet

- **Bricks_48**
  - Serial: 16.8 M elements, 571 sec
  - 16 procs:
    - 79 sec, Speedup: 8, 50% efficiency
  - Not well load balanced
    - Surfaces
    - Communication time
  - Remote handle resending included
Conclusions & Future works

• Parallel mesh generation method is developed by distributing geometry
• Initial results obtained

• Load balancing
  – Tool with Zoltan: export partitioned geometry
  – Link with Zoltan directly

• Interface surface mesh
  – Reduce message communication

• Robust mesh generator
  – Good interval assignment
  – Mesh generation algorithm

• Remove artificial boundary
  – Send and receive ghost elements
  – Smooth layers of ghost elements
Additional slide 1

- Mesh volumes *and surfaces* in parallel
- Use graph partitioning to partition work
- Use of component-based mesh, geometry, partitioning
- 1st-generation tool developed before ITAPS parallel interface specification
- Parallel communication handled at application-level
- ITAPS
  - Parallel iGeom: Read/Delete, Broadcast, Broadcast/Delete, Scatter, Read Parallel
Additional slide 2

**Distribution time for 16 processors**

- **Bcast**
- **Bcast_delete**
- **Scatter**

**Memory usage of test code with various methods**

- by scatter in root
- by scatter in proc 2
- by scatter in proc 15
- by bcast in root
- by bcast in proc 2
- by bcast in proc 15
- by bcast_del in root
- by bcast_del in proc 2
- by bcast_del in proc 15
• **Cosmea Linux cluster**
  - 32 nodes, 128 cores
  - 2 dual-core Intel Xeon 5130 cpus
  - 16GB memory per node (4GB per core)
  - 19TB shared low-speed NFS/GFF filesystem storage

• **Bricks_48**
  - 48 consecutive bricks
  - 16 procs
    - 12.1 speedup
    - 76% efficiency