A Parallel Unstructured Mesh Infrastructure

Seegyoung Seol, Cameron Smith,
Daniel Ibanez, Mark S. Shephard
Scientific Computation Research Center
Rensselaer Polytechnic Institute

WOLFHPC 2012
Second International Workshop on Domain-Specific Languages and
High-Level Frameworks for High Performance Computing
November 16, 2012

Outline

Simulation Based Engineering Workflow

PUMI

- Geometric Model
- Mesh
- Parallel Control

ParMA

- Multi-criteria Partition Improvement
- PHASTA – Strong scaling
- Predictive Load Balancing

Closing Remarks

Simulation Based Engineering Workflow

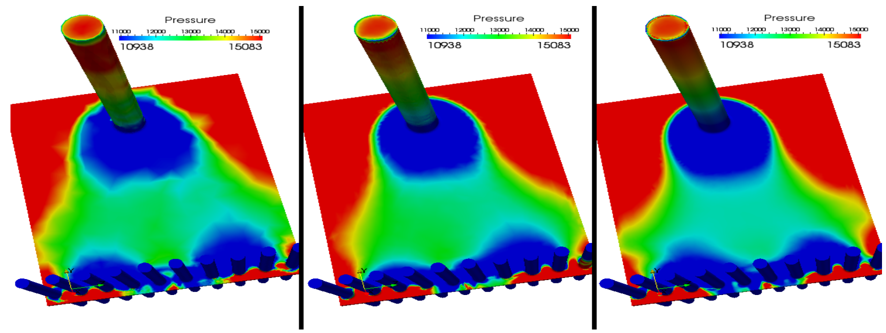
Problem Definition

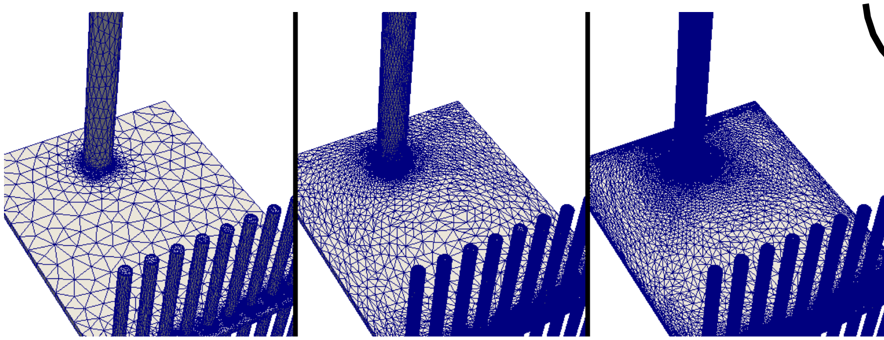
Mesh Generation

PDE Analysis

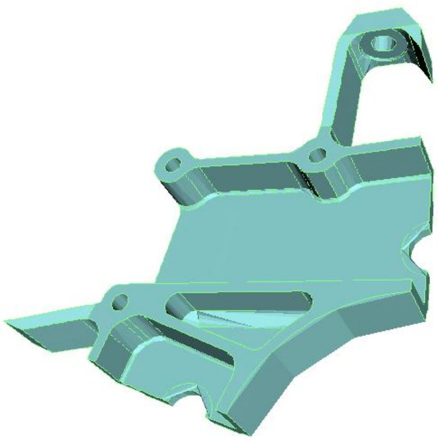
Adaptation

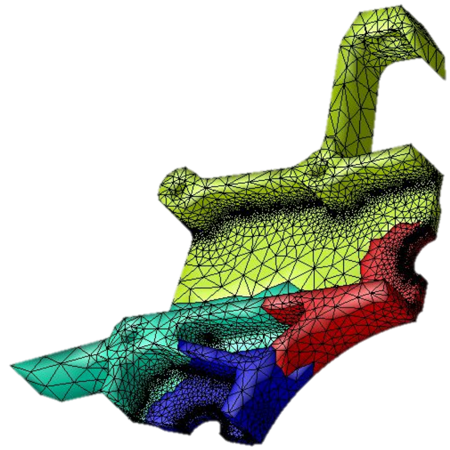
Post Processing





PUMI





Mesh

0-3D topological entities
and adjacencies

Geometric model

analysis domain

Fields

distribution of solution
over mesh

Parallel Control

communication utilities

Partition model

distribution of
mesh across
computing
resources

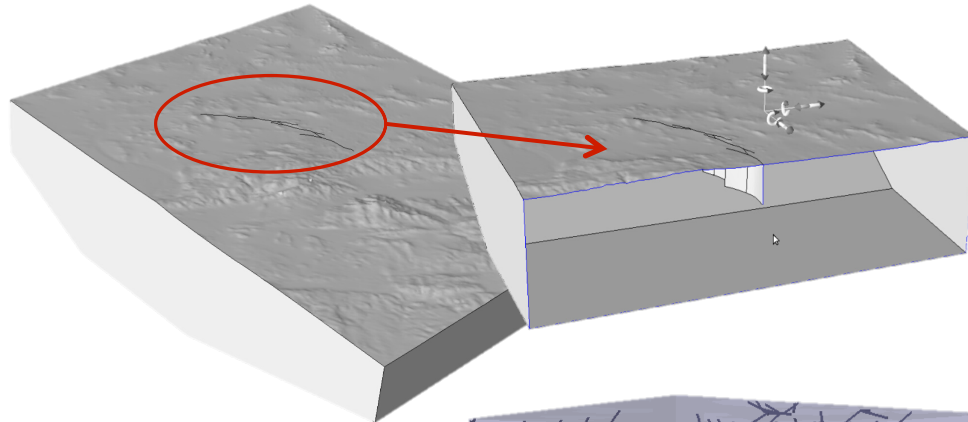
Geometric Model

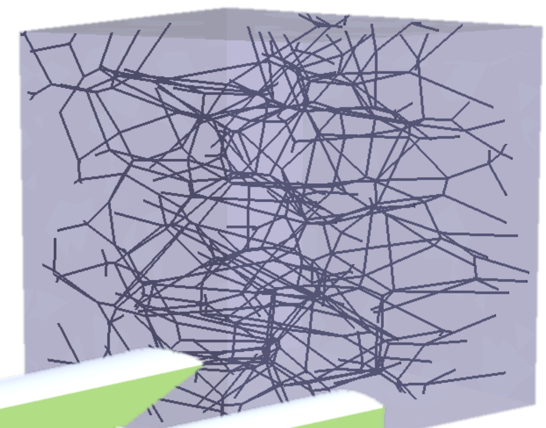
Non-manifold Representation
- Topological representation of any combination of volumes, surfaces, curves, and points

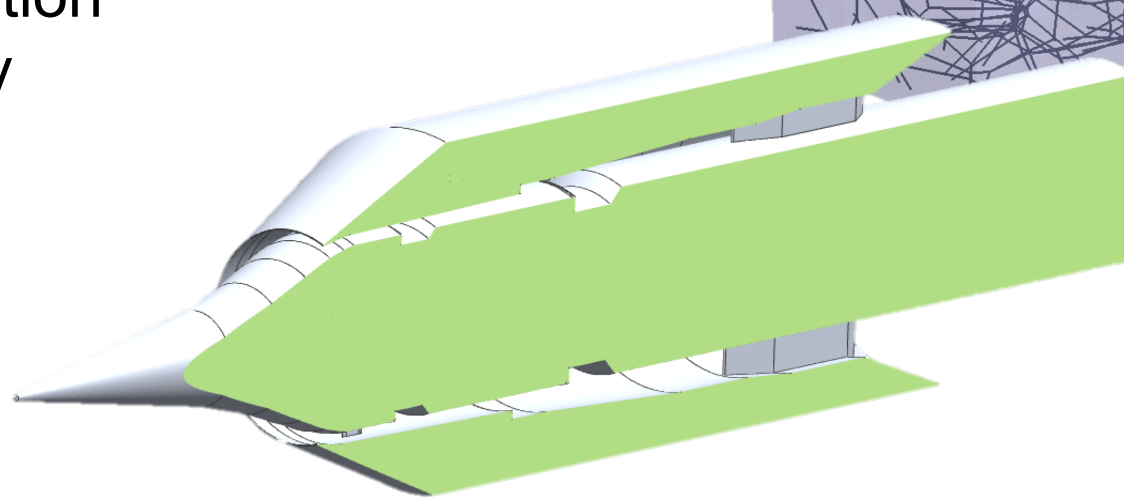
Geometric information
- Solid Modeling Kernels
- Coordinates on surface
- Tolerance

Topological information
- Entity adjacency

Shape information
- CAD geometry
- Mesh models
- Image data







Mesh - Representation

Mesh entities:

- vertex (0D), edge (1D), face (2D), or region (3D)

Adjacencies:

- How the mesh entities connect to each other
- Complete representation: store sufficient entities and adjacencies to get any adjacency in O(1) time

Geometric classification:

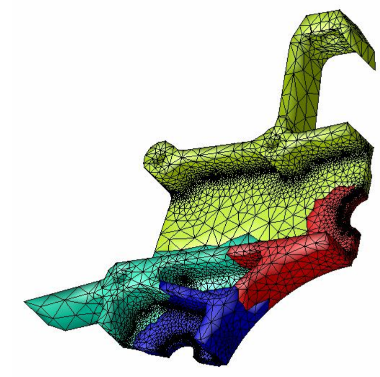
- A relation that each mesh entity maintains to a geometric model entity

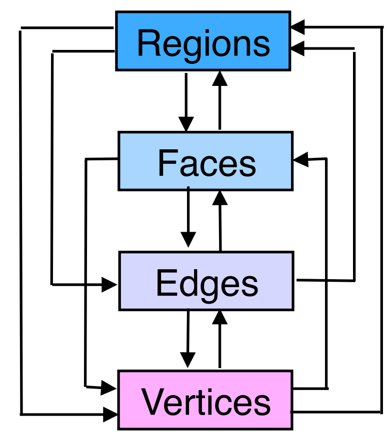
Entity set:

- Mechanism for grouping mesh entities

Tag:

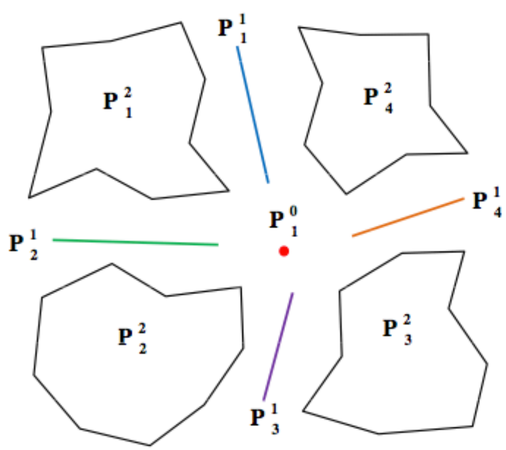
- Mechanism to attach arbitrary user data (tag data) to a part, entity set or mesh entity



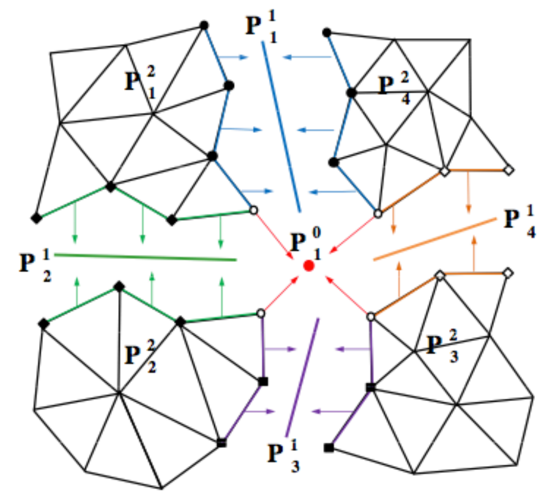


Mesh - Distribution

- Mesh partition defines parallel decomposition of applications.
- Mesh partitioning representation in topology for efficient mesh-based parallel operation support.
- Partition model: a conceptual model existing between a geometric model and distributed mesh
- Partition model entity: a topological entity in the partition model, P_i^d, representing a group of mesh entities of dimension *d with* the same residence parts.



Partition model entities

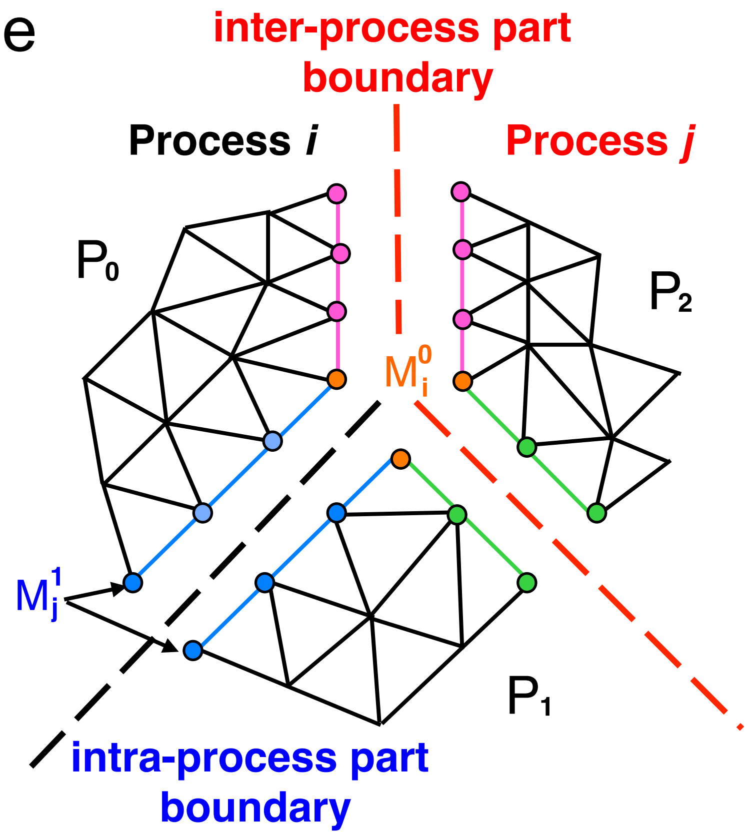


Partition classification in arrows

Mesh - Distribution

Part

- A unit of the mesh distribution
- Each part P_i assigned to a process
- Uniquely identified at part level by handle or id
- Consists of mesh entities assigned to i^{th} part.
- Treated as a serial mesh with the addition of *part boundaries*
  - *Part boundary:* groups of mesh entities common to multiple parts
  - *Part boundary entity*: duplicated entities on all parts for which they bound higher order mesh entities
  - *Remote copy*: entity copy in another part



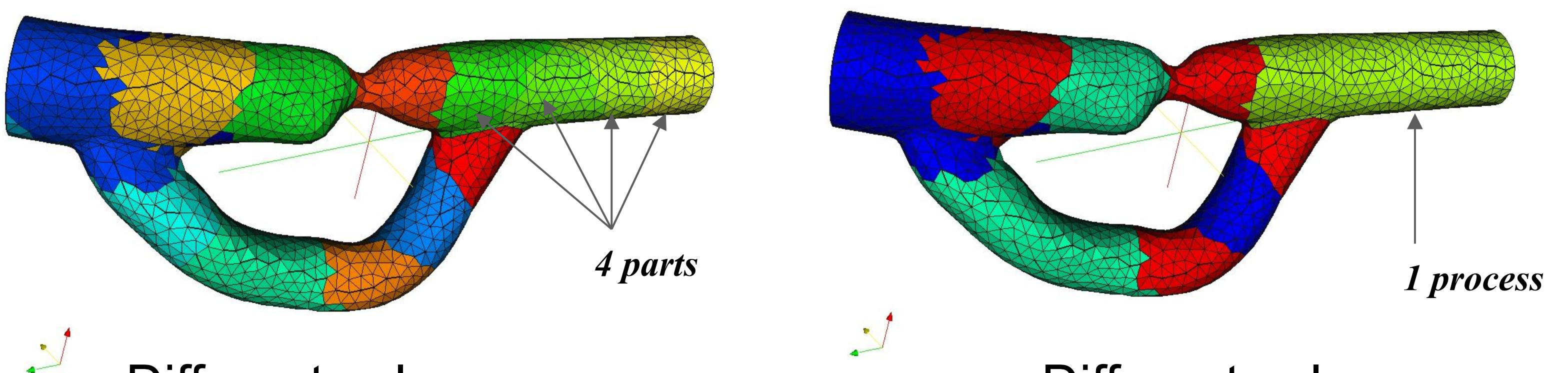
Mesh - Multiple Parts Per Process

Purpose

- Supports changing number of parts
- Dealing with problems with current graph-based partitioners on really large numbers of processors
- Architecture-aware two-level mesh partitioning

Multiple-Parts Per Process contained in Mesh Instance

- For effective manipulation, a mesh instance defined on each process contains part handles assigned to the process



Different color represents different **part**

Different color represents different **process**

A 3D mesh in 4 parts per process (16 parts total)

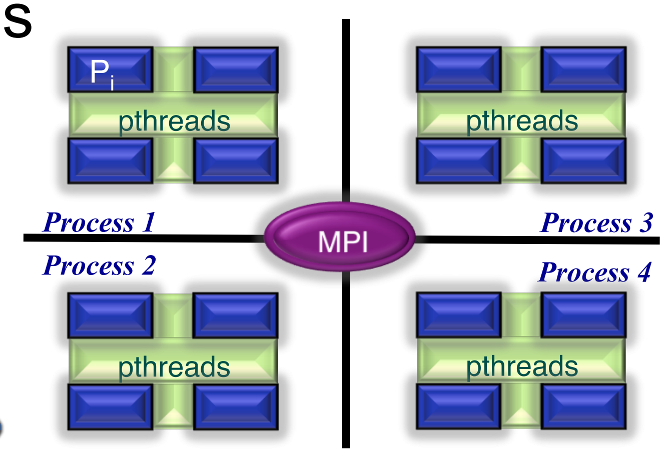
Mesh - Two-Level Partitioning

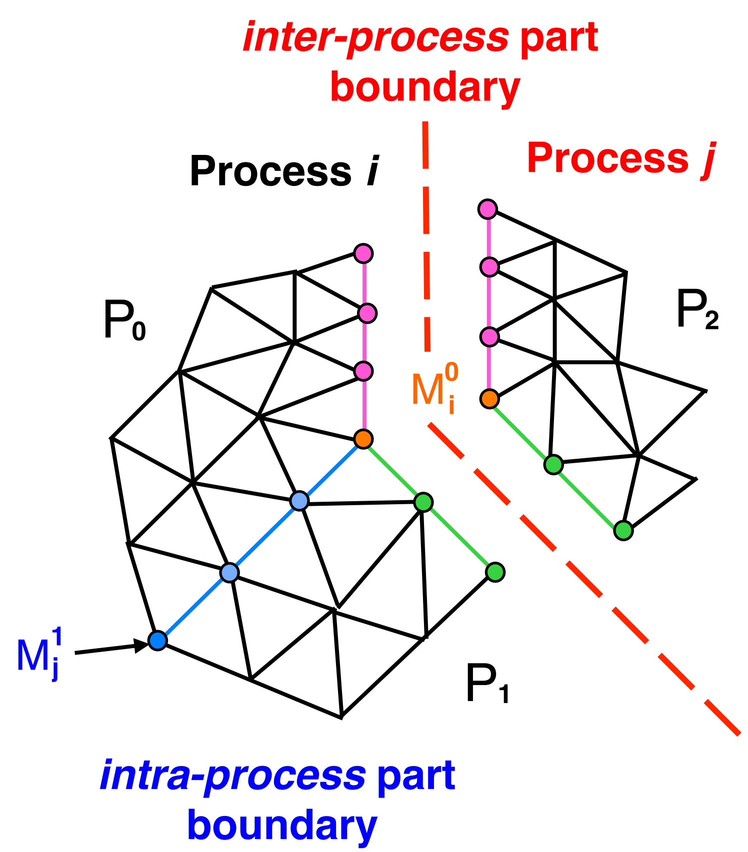
Exploit hybrid architecture of BG/Q, Cray XE6, etc…

- Reduced memory usage

Approach

- Partition mesh to processes, then partition to Pthreads
- Message passing, via MPI, between processes
- Shared memory, via Pthreads, within process
- Transparent-to-application use of Pthreads

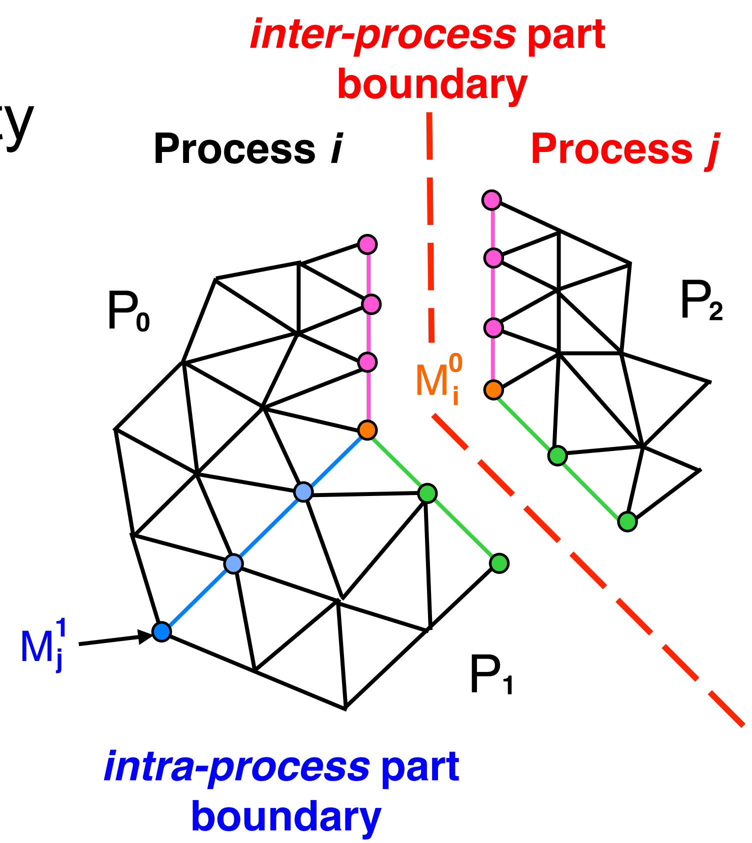




Mesh - Two-Level Partitioning

Entity is created at most once per process

- Part boundary entity is created at most once per process
- Part boundary entity on process *i* is shared by all on-process residence parts
- Only owning part can modify entity (no race condition guaranteed)
- Remote copy: entity copy on another *process*
- Parallel control utility provides architecture info to mesh, then the mesh is distributed accordingly.



** Authors thanks to Micah Corah and Ian Dunn (Dept. of Computer Science, RPI) for development and testing on RPI BG/Q.*

Parallel Control

Message Passing abstraction

- size, rank, send, receive

Present

- Architecture info collection via HWLOC*
- Communication rounds for termination detection
  - Local - Fixed neighborhoods
  - Global - Unknown neighborhoods

In Progress

- Hybrid MPI/Pthread communications
  - Hybrid rank = (MPI rank)*(#Pthread per process) + thread rank
  - Hybrid send/receive
- Pthread management – create, run, and join

** Portable Hardware Locality (http://www.open-mpi.org/projects/**hwloc**/)*

SCOREC

Mesh Partitioning

Parallel simulation requires that the mesh be distributed with equal work-load and minimum inter-part communications

Observations on graph-based dynamic balancing

- Parallel construction and balancing of graph with small cuts takes reasonable time
- Graph/hyper-graph partitions are powerful for unstructured meshes, however they use one order (as in 0,1,2,3) of mesh entity as the graph nodes, hence the balance of other mesh entities may not be optimal

Accounting for multiple criteria and or multiple interactions is not obvious

- Hypergraphs allows edges to connect more that two vertices – has been used to help account for migration communication costs
- Schloegel and Karypis (2002) discuss an effective optimization method for three, or fewer, constraints

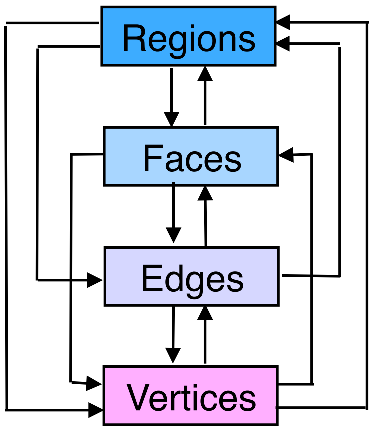
Partitioning using Mesh Adjacencies (ParMA)

Mesh adjacencies represent application data more completely then standard graph-partitioning models.

- All mesh entities can be considered, while graph-partitioning models use only a subset of mesh adjacency information.
- Any adjacency can be obtained in O(1) time (assuming use of a complete mesh adjacency structure).

Advantages

- Avoid graph construction (assuming you have complete representation)
- Directly account for multiple entity types – important for the solve process - typically the most computationally expensive step
- Easy to use with diffusive procedures



Disadvantage

- Lack of well developed algorithms for more global parallel partitioning operations directly from mesh adjacencies

SCOREC

ParMA – Multi-Criteria Partition Improvement

Improve scaling of applications by reducing imbalances through exchange of mesh regions between neighboring parts

- Current algorithm focused on improved scalability of the solve by accounting for balance of multiple entity types
  - Imbalance is limited to a small number of heavily loaded parts, referred to as spikes, which limit the scalability of applications
  - Application defined priority list of entity types such that imbalance of high priority types is not increased when balancing lower priority types
- Similar approaches can be used to:
  - Improve balance during mesh adaptation – likely want extensions past diffusive methods
  - Supporting Two-level partitioning – heterogeneous resources

ParMA – Multi-Criteria Partition Improvement

Input:

- Priority list of mesh entity types to be balanced (region, face, edge, vertex)
- Partitioned mesh with complete representation and communication, computation and migration weights for each entity

Algorithm:

- From high to low priority if separated by '>' (different groups)
  - From low to high dimension entity types if separated by '=' (same group)
    - *Compute migration schedule (Collective)*
    - *Select regions for migration (Embarrassingly Parallel)*
    - *Migrate selected regions (Collective)*

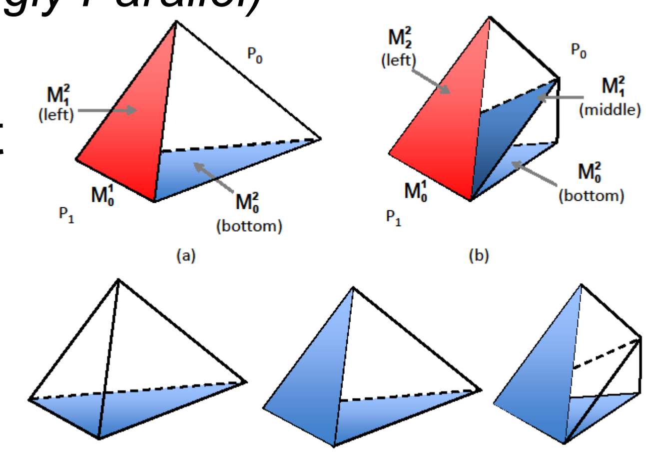
Ex) "*Rgn>Face=Edge>Vtx*" is the user's input

Step 1: improve balance for mesh regions

Step 2.1: improve balance for mesh edges

Step 2.2: improve balance for mesh faces

Step 3: improve balance for mesh vertices

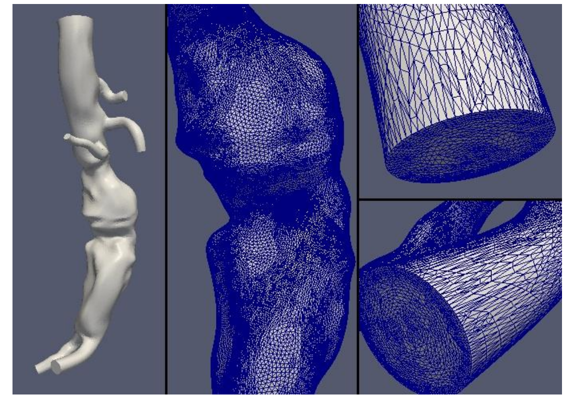


Mesh element selection

ParMA – Multi-Criteria Partition Improvement (Zhou)

Table 1:Tests

Original	Zoltan's Hypergraph
LIIPBMod	partition improvement
ParMA #1	$Vtx > Rgn$
ParMA #2	$Vtx = Edge > Rgn$
ParMA #3	$Edge > Rgn$
ParMA #4	$Edge = Face > Rgn$



133M region mesh on 16k parts

Table 2:Balance of partitions

AAA 133M	MeanRgn	Rgn Imb.	MeanFace	Face Imb.	MeanEdge	Edge Imb.	MeanVtx	Vtx Imb.
Original	8177	4.3%	17,315	5.39%	11,023	9.07%	1886	19.41%
LIIPMod	8177	9.26%	-	-	-	-	1867	4.99%
ParMA#1	8177	4.99%	-	-	-	-	1865	4.99%
ParMA #2	8177	5.99%	-	-	10,973	4.91%	1870	4.99%
ParMA #3	8177	5.98%	-	-	11,013	4.99%	-	-
ParMA #4	8177	5.93%	17,309	4.97%	11,014	4.99%	-	-

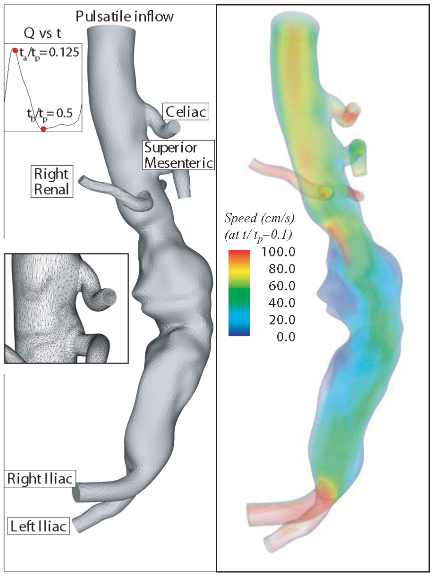
Table 3: Time usage and iterations (tests on Jaguar Cray XT5 system)

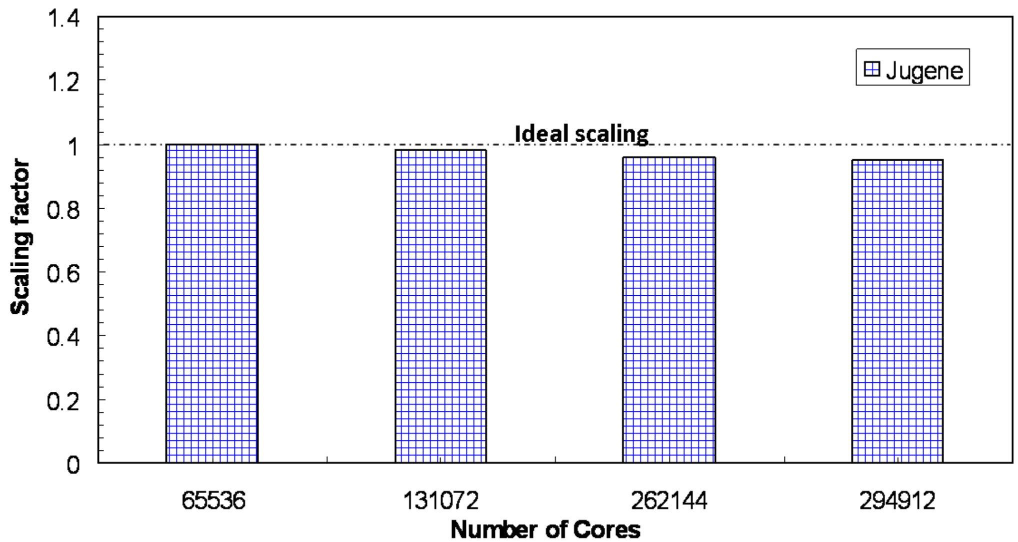
test	time(s)	Iter_Rgn	Iter_Face	Iter_Edge	Iter_Vtx
Original	249	-	-	-	-
LIIPBMod	4.2	-	-	-	3
ParMA #1	6.6	8	-	-	2
ParMA #2	8.8	6	-	1	3
ParMA #3	5.5	1	-	3	-
ParMA #4	5.5	1	1	3	-

SCOREC

PHASTA - Strong Scaling (K. Jansen)

AAA 5B elements: 288k Cores on JUGENE IBM BG/P





5*B* elements mesh (Jugene:IBM BG/P)		Eqn. form.		Eqn. soln.		Total	
num. of core	avg. elem./core	time	s-factor	time	s-factor	time	s-factor
65,536 (base)	76,480	119.64	1	162.59	1	288.23	1
131,072	38,240	59.69	1.00	84.09	0.97	143.78	0.98
262,144	19,120	30.02	1.00	43.24	0.94	73.26	0.96
294,912	16,995	26.71	1.00	39.15	0.92	65.86	0.95

without ParMA strong scaling factor is
0.88 *(time is 70.5 secs),*
for production runs savings can be in ***43 cpu-years***

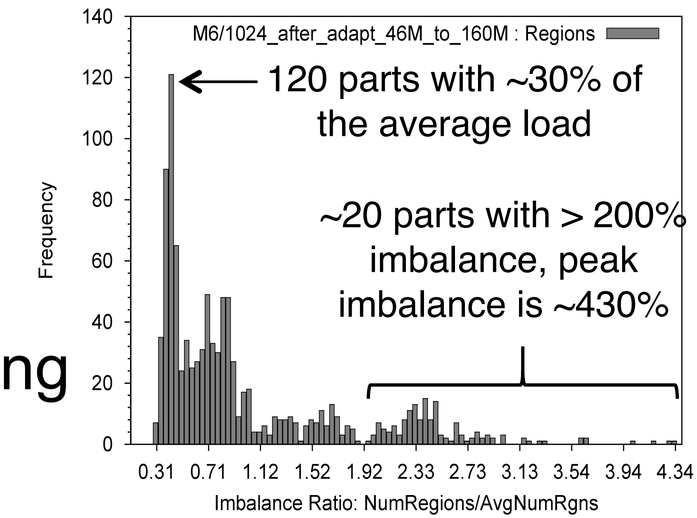
ParMA – Predictive Load Balancing

Parallel unstructured mesh adaptation typically generate parts with 400% or more imbalance on non-trivial geometries due to local coarsening & refinement.

Refining then repartitioning can exceed available memory in some processes, even if the system's total memory is sufficient.

Solution: Redistribute mesh *before* adapting

- Merge parts that will be coarsened to create some empty parts.
- Split parts with substantial refinement into the empty parts to remove imbalance spikes of refined mesh.
- Refine/coarsen the mesh.
- Apply ParMA's diffusive partition improvement



Histogram of element imbalance in 1024 part adapted mesh on Onera M6 wing if no load balancing is applied prior to adaptation.

Closing Remarks

Research Contributions

- Parallel mesh data structure with all needed mesh-level operations for adaptive simulations on a massively parallel computers

Future Directions

- Architecture-awareness: node-socket-core-processing unit
- Identifying optimal granularity and major h/w factors for max. scalability
- Interaction with other threaded/non-threaded parallel library
- Two-level partitioning with ParMA

More Information Online

- PUMI: http://www.scorec.rpi.edu/FMDB/
- ParMA: http://redmine.scorec.rpi.edu/projects/parma
- SCOREC: http://www.scorec.rpi.edu
- FASTMath: http://www.fastmath-scidac.org

Thank You

WOLFHPC 2012

SC12
Salt Lake City, Utah

FASTMATH

SCOREC

Argonne NATIONAL LABORATORY

CCNI COMPUTATIONAL CENTER for NANOTECHNOLOGY INNOVATIONS

KAUST

For More Information Contact:

smithc11@rpi.edu