Parallel Mesh Multiplication: Towards Unstructured Grids Peta(Exa?)scale Simulations

Renato N. Elias
Jose J. Camata
Igor T. Ghisi
Alvaro L. G. A. Coutinho

High Performance Computing Center (NACAD)
Federal University of Rio de Janeiro (UFRJ)

Rio de Janeiro, Brazil

Outline I

Outline II

- Generating hexahedral meshes
- Surface Detection
- Parallel Scalability
- Hexahedra to tetrahedra: building conforming mehes

6 Conclusions and Discussion

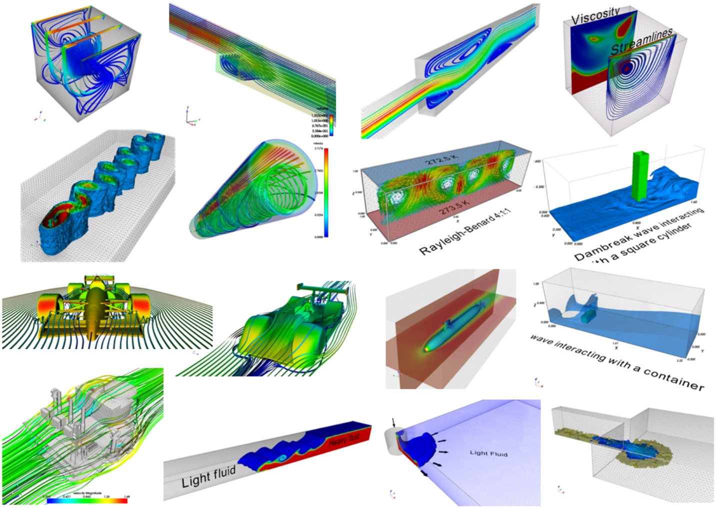
Motivations

- Petaflop computing poses new unconventional challenges;
- Understand some hardware and software issues driven by ultra-large scale simulations;
- Get used with such huge problems.
- Compromise with continuous performance improvements in our software;
- Reach high fidelity simulations in spatial resolution;
- Need for multiphysics and highly integrated solutions;

Our software playground

- **EdgeCFD**: *A parallel and general purpose CFD solver*
  - Edge-based data structure;
  - Hybrid parallel (MPI, OpenMP or both);
  - Finite element method;
  - SUPG/PSPG formulation for incompressible flow;
  - RB-VMS turbulence treatment;
  - $\mathbf{u}$-p fully coupled flow solver;
  - SUPG/YZβ formulation for advection diffusion;
  - Free-surface flows (VoF and Level Sets);
  - Adaptive time step control;
  - Inexact-Newton solver;
  - Dynamic deactivation;
  - Mesh entities ordered according to computer architecture.
  - FOI (Fluid-Object-Interaction) formulation;
  - **MM (Mesh Multiplication)**
- Upcomings:
  - SUPG/YZβ formulation for compressible flow;
  - Geometric Multigrid and Multilevel Preconditioners;

EdgeCFD in action



Large Scale Simulations

First of all: What's large?

1. **Large in size** ⇒ Unstructured grids with billions of elements;
2. **Large in coupling** ⇒ multiphysics/multiscale interactions;
3. **Large in physical parameters** ⇒ different viscosity, density, velocity, pressure, ...
4. **Large in control parameters** ⇒ tolerances, solver options, etc...
5. **Large in complexity** ⇒ several softwares and human intervention, reproducibility and information tracking.

Large Scale Simulations

First of all: What's large?

1. **Large in size** ⇒ Unstructured grids with billions of elements;
2. **Large in coupling** ⇒ multiphysics/multiscale interactions;
3. **Large in physical parameters** ⇒ different viscosity, density, velocity, pressure, ...
4. **Large in control parameters** ⇒ tolerances, solver options, etc...
5. **Large in complexity** ⇒ several softwares and human intervention, reproducibility and information tracking.

Teams overlapping and collaboration

- **Item 5** ⇒ see Marta Mattoso and Eduardo Ogasawara's talks in Provenance and Parallel Workflow management (group 2);
- **Items 3 and 4** ⇒ We have people working in Uncertaint Quantification using item's 1 and 5 results...

Towards "Large/huge in Size" Simulations

Concerns in the "large/huge in size" topic:

1. Visualization;
2. (Parallel) Mesh generation;
3. Efficient solvers;
4. Data storage and management;
5. Minimize data translation (avoid external libraries...)

What we've been doing...

Large scale visualization using ParaView:

- Compressed and distributed data formats (Xdmf/HDF5)
- Remote parallel offscreen rendering
- Coprocessing
- Tiled wall display

Parallel Meshes:

- Uniform mesh refinement ("Mesh Multiplication");
- Octree parallel mesh generation (automatic mesh generation).

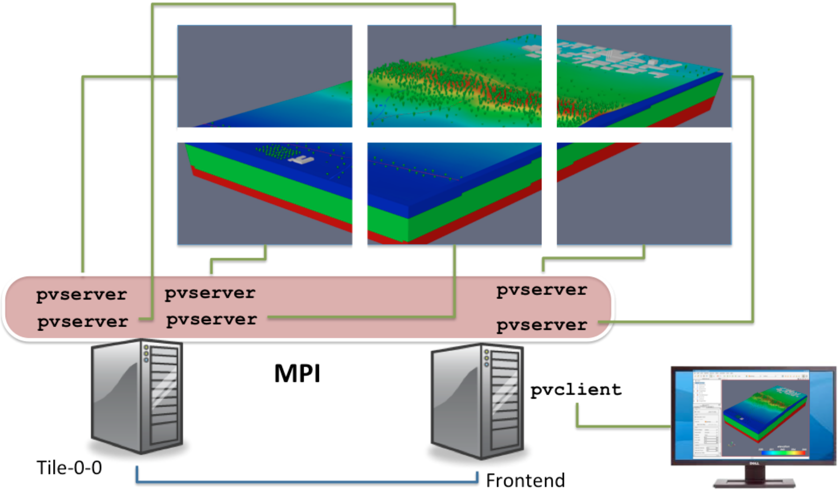
Apparatus for high resolution visualization

Tiled wall:

- Three DELL Workstations
  - 24 cores
  - Intel Xeon E5506 @ 2.13GHz
  - 36 GB RAM
  - 5 NVIDIA Quadro FX 6000 (dual head boards)
  - 1 TB storage
- 10 DELL Monitor
  - 31 with maximum resolution of 2560x1600
  - Tiled wall with 9 Monitors
  - 1 monitor for administration

Tiled wall: current setup

Tiled wall and ParaView: Basic Setup



Mesh Multiplication: Definition

What's this?

Just an algorithm to increase mesh size by uniformly refining elements

What it's not?

A mesh generator neither a mesh dynamic adaptivity method.

- **Mesh generator** $\Rightarrow$ Creates elements from computational models (from scratch);
- **Mesh adaptivity method** $\Rightarrow$ Takes some metric to guide mesh refinement;

Mesh Multiplication: Definition

What's this?

Just an algorithm to increase mesh size by uniformly refining elements

What it's not?

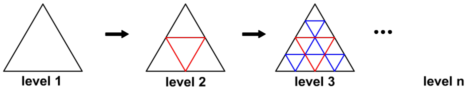
A mesh generator neither a mesh dynamic adaptivity method.

- **Mesh generator** ⇒ Creates elements from computational models (from scratch);
- **Mesh adaptivity method** ⇒ Takes some metric to guide mesh refinement;

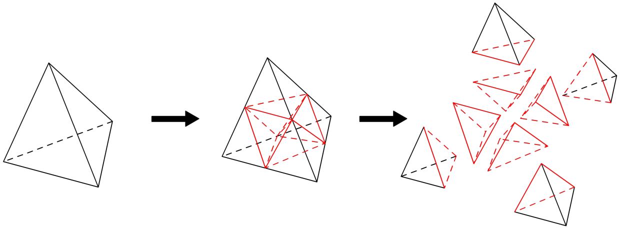
Mesh multiplication ⇒ Takes an **existing mesh** and apply the same refinement scheme to the whole mesh.

Mesh Multiplication: Basic examples

- One picture is better than a thousand words...



- Tetrahedron is a little bit more complicated...



- Mesh growth ratio is: $8^{(lvl-1)} \times nel$

Mesh Multiplication: Implementation

Basic idea:

- Just split existing tetrahedra elements into new ones recursively;

Is that all?

"*Nothing is so easy that couldn't get more complicated...*"

- Basic idea is perfect for sequentially generated meshes (*...but nothing interesting can be done in serial* ☹);
- Refinement of distributed meshes poses a new problem:
  - we must take care of the communication interface among processors.

Mesh Multiplication: Properties (1/2)

Why is it good?

1. Easy to implement (but not too much... ☺). See item 1 in the next slide;
2. *Base* mesh can be built in cheap workstations (even notebooks);
3. Minimal communication cost is required during refinement process (the major pain in parallel mesh generation);
4. MM can be easily applied to any existing mesh;
5. Parallel load balance is inherited from base mesh partitions;
6. Refinement levels can be naturally used in geometrical multigrid.
7. Well, finer meshes $\Rightarrow$ better results...

Mesh Multiplication: Properties (2/2)

Why isn't it so good?

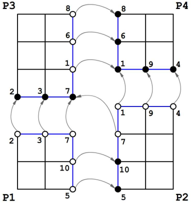
1. Not so easy to implement in parallel (no pain, no gain ☺);
2. Destroys ordering schemes (but it could be easily recovered, of course);
3. (At first) does not improve boundary model representation.

Mesh Multiplication: Properties (2/2)

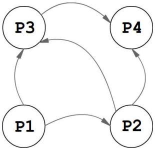
Why isn't it so good?

1. Not so easy to implement in parallel (no pain, no gain ☺);
2. Destroys ordering schemes (but it could be easily recovered, of course);
3. (At first) does not improve boundary model representation.

Communication map/graph and why it's an issue...

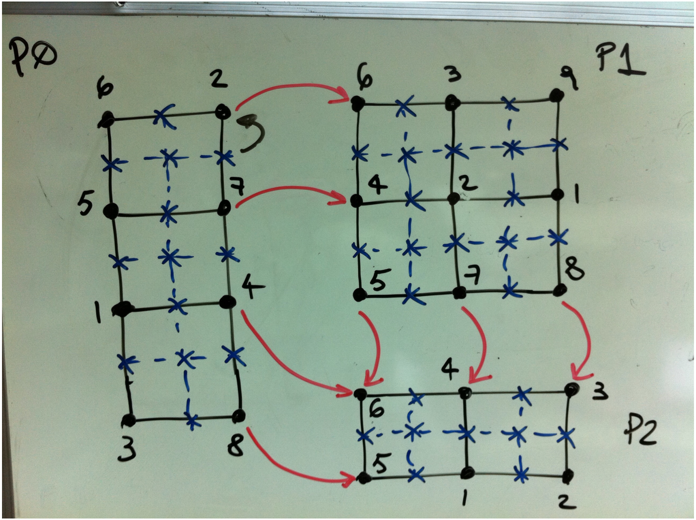


(a) 4 mesh partitions



(b) Master-slave communication map

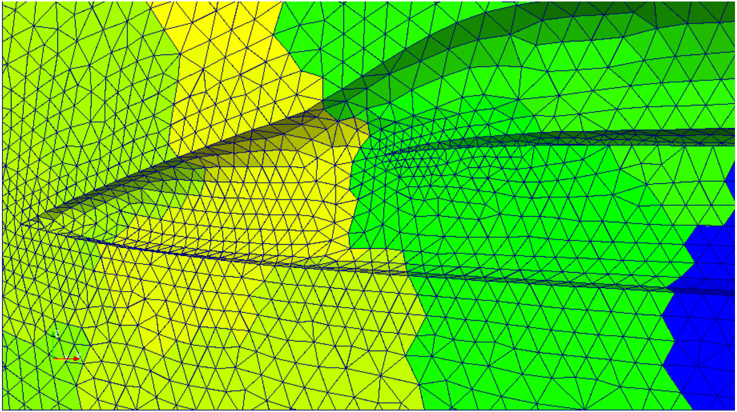
Communication map in MM:



2D MM communication map treatment

Mesh Multiplication in Action 1/4

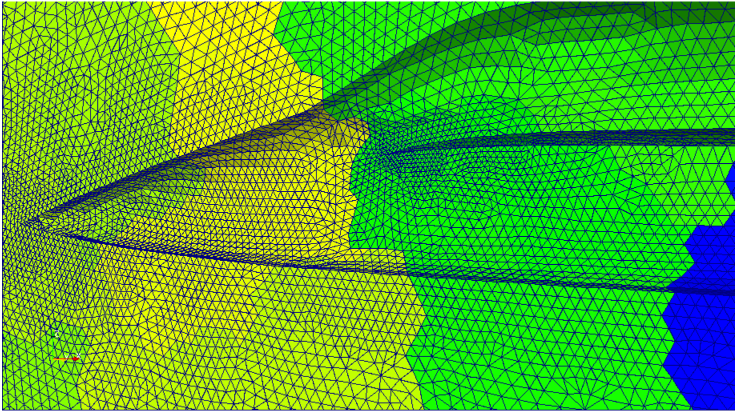
Model YF17, 120 cores, SGI Altix ICE 8400



Level 1 (528,915 tets)

Mesh Multiplication in Action 2/4

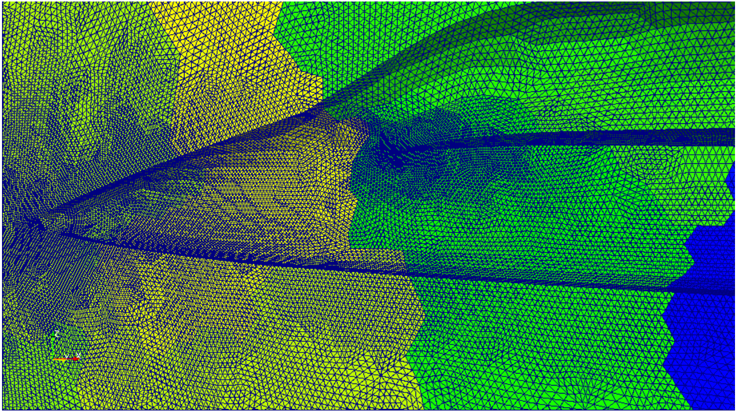
Model YF17, 120 cores



Level 2 (4,231,320 tets)

Mesh Multiplication in Action 3/4

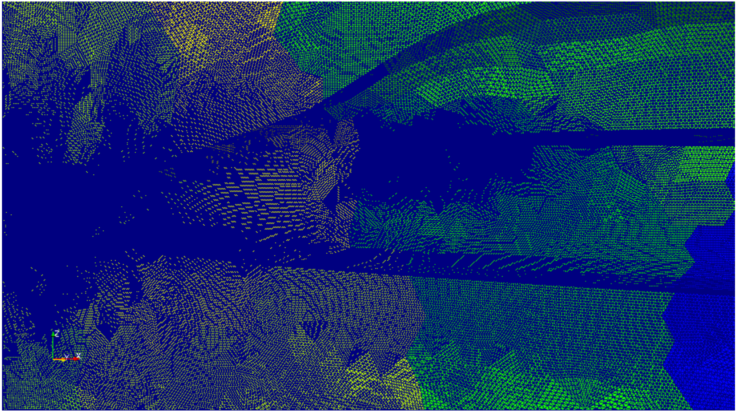
Model YF17, 120 cores



Level 3 (33,850,560 tets)

Mesh Multiplication in Action 4/4

Model YF17, 120 cores



Level 4 (270,804,480 tets)

Test case: Cavity flow (3D unit cubic domain)

Test performed in 120 cores (SGI Altix ICE 8400)

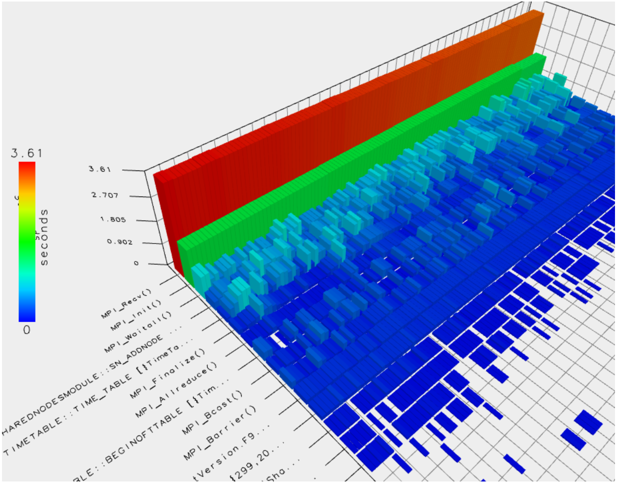
Mesh levels × time

LEVEL	TETS	NODES	EDGES	TIME (sec)
1	108,840	929,400	159,360	0.00
2	870,720	1,088,760	1,141,800	0.20
3	6,965,760	2,230,560	8,623,440	0.45
4	55,726,080	10,854,000	66,986,400	1.98
5	**445,808,640**	77,840,400	527,972,160	**18.10**

NOTE: Mesh multiplication costs are not affected by geometry complexity.

Parallel Profiling 2/3

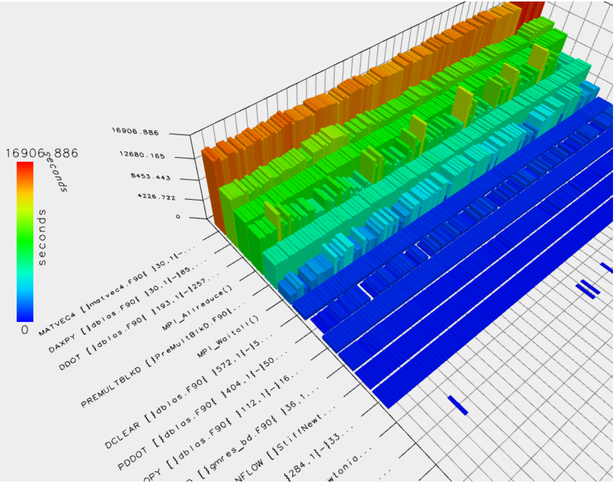
3D Cavity Flow, 4 MM levels, 55M of tets, 120 cores



Mesh Multiplication Only

Parallel Profiling 3/3

3D Cavity Flow, 4 MM levels, 55M of tets, 120 cores



100 time steps simulation

Storage demands

- 3D Cavity Flow in 120 Cores:
  - 55.7M of tetrahedra;
  - 10.8M of nodes;
  - 66.9M of edges;
  - Each processor requires **6.2MB** per solution file (Xdmf/compressed HDF5);
  - 6.2MB $\times$ 120 processes = 744MB per record;
  - 100 time steps[1] stored would require **74.4GB**;
  - **and it's a very modest simulation...**

[1] 10 simulation seconds in 14 wall time hours

Mesh Multiplication: Next steps

- Improve visualization and storage methods for large scale meshes
  - ParaView coprocessing;
  - Offscreen remote and parallel rendering (using the tiled wall);
  - Suitable file formats (compressed and distributed Xdmf/HDF5)
- Extend MM to other EdgeCFD solvers (should be easy...)
- Recover cache friendly data order for all mesh entities;
- Improve geometry representation;
- Develop multigrid method and/or multilevel preconditioners.

Octree parallel mesh generation

Why automatic mesh generation?

- Usual methodology for finite elements simulations is:
  - Apply a mesh partitioning scheme to an existing mesh and
  - Map the mesh parts into the target parallel system.
- Weakness
  - mesh size limited by hardware;
  - associated partitioning problem is NP-complete;

Octree parallel mesh generation

Why automatic mesh generation?

- Usual methodology for finite elements simulations is:
  - Apply a mesh partitioning scheme to an existing mesh and
  - Map the mesh parts into the target parallel system.
- Weakness
  - mesh size limited by hardware;
  - associated partitioning problem is NP-complete;

Main issue: Generate meshes with billions of nodes and elements and deliver such meshes to processors of large-scale systems can be **unpractical**

Octree parallel mesh generation

- Recent progress in, scalable, automatic mesh gereration based on octrees
  - Dendro library: executions on 4000 cores[2]
  - **p4est**: executions on 220.320 cores [3]
  - Octree meshes using GPGPU (Park and Shin (2012))

Our Objectives are...

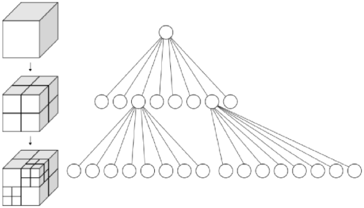
- present a scalable parallel octree generator able to representing arbitrary surfaces, and
- Extract conforming tetrahedral meshes from the resulting octree

[2] Sundar et al 2007, Low-constant parallel algorithms for finite element simulations using linear octrees. In Proc. SC ?07, New York, NY, USA; 25:1?25:12.

[3] Burstedde et al 2011, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing 33(3), p. 1103-1133

Linear Octrees

- Def.: *An octree is a tree data structure in which internal nodes has exactly eight children*
- Literature shows an extensive use of octrees
  - Many variations of octree structures
  - Different Boundary fit algorithms
  - For mesh generation, different conforming techniques
- Linear octree
  - complete list of leaf nodes
  - octants are encoded by a scalar key: Morton Code
  - hierarchical and regular nature of octree allow an efficient parallel implementation



Why are linear octrees good?

- do not require to store internal nodes
- small overhead associated with pointers usage

Generating hexahedral meshes

- Fist step: Building a parallel octree structure
  - create an initial partitioned octree among processors
  - Refine octants that intecept an immersed surface
    - Inteception test uses boundary box tree
  - Execute a parallel 2:1 balancing constraint algorithm
    - no leaf octants sharing at level l shares an edge or face with another leaf at level greater than $l + 1$
  - Execute a new octree partitiong
- Second step: Remove octants inside or outside the immersed surface
- Third step: Generate a non-conforming hexahedra mesh
  - decode Morton code from linear octree
  - get octants nodes, hanging nodes and ghost nodes
  - get elements

Is our surface detection algorithm able to capture any arbitrary surface?

STL Surface	h-level: 8	h-level: 9
Volume: 2.3797×10^{5}	Error*: 2.0×10^{-4}	Error: 2.0×10^{-4}
Volume: 4.7828×10^{-4}	Error: 1.24×10^{-2}	Error: 1.2×10^{-3}

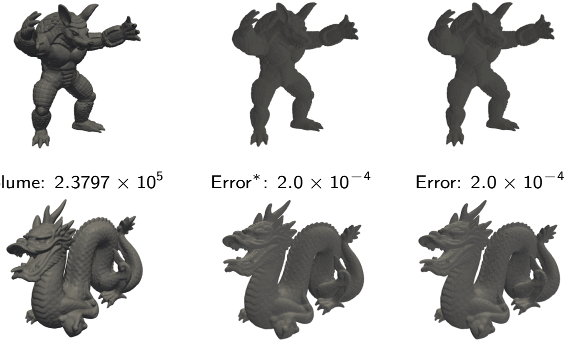
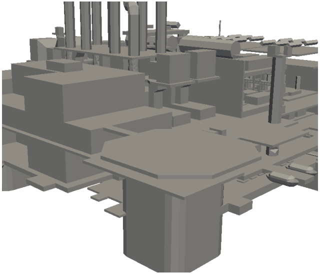


Figure : Armadillo and Dragon Surfaces and their octree representation with 8 and 9 refinement levels

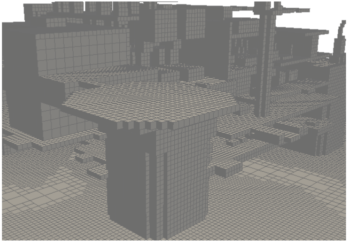
*: Relative volume error

How much fast can we extract hexahedral mesh structures from balanced linear octrees?

Case Study: Generate a finite element mesh from a complex surface of a real life offshore platform



(a)



(b)

Figure : Offshore platform: (a) heliport view and (b) finite element mesh detail around heliport

How much fast can we extract hexahedral mesh structures from balanced linear octrees?

An isogranular analysis is performed by tracking the execution time while increasing the problem size and number of processors

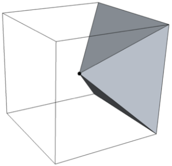
Offshore platform: Mesh sizes and CPU time					
CPUs	Levels	Elements	Anchor Nodes	Hanging nodes	Total Time
32	11	53,498,110	35,931,290	15,482,795	106.405
128	12	142,495,654	143,116,154	61,562,108	135.249
512	13	569,885,503	572,099,070	246,023,557	169.129
2048	14	**1,131,815,284**	1,136,125,418	488,655,212	**194.051**

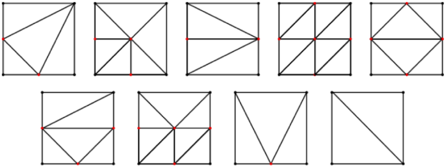
Our scheme was able to generate 3.4 billion octants in less than 10 seconds per 1.6 million octants per core

Note: Experiments were conducted on the Ranger cluster at The Texas Advanced Computer Center (TACC)

Hexahedra to tetrahedra: building conforming mehes

- Conforming techniques: FREY e GEORGE (2000) / BERG et. al (1998)
  - Decompose octants in 6 pyramidal elements by inserting a central node
  - Define 9 templates for face triangulation
  - Connect all face nodes with the central node
  - Does not require modifications in the octree construction
  - Templates for all possible hanging nodes configuration

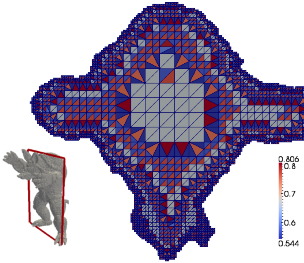




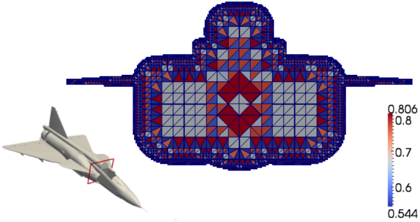
Mesh Quality

- BRANETS and CAREY (2005):

$$Q_0 = \frac{72\sqrt{3}V}{(\sum_{i=1}^{6} l_i^2)^{3/2} } \tag{1}$$



Armadillo Model

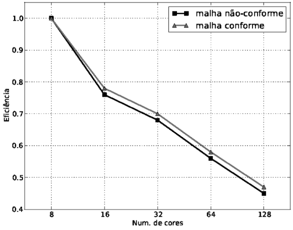


Aircraft Model

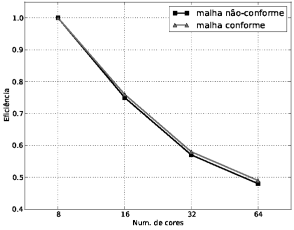
- Mesh quality is predictable due to templates
- Reference in future works (element smoothing)

Strong Scalability

Comparison between non-conforming and conforming mesh generators



Armadillo - 11 Ref. levels



Aircraft - 12 ref. levels

Note: Tests were carried out an SGI Altix-ICE 8400 (NACAD/COPPE/UFRJ)

- Low intrusion on octree construction
- Low impact on parallel execution

Future Work:

- Element smoothing and boundary fitting
- Support to another geometry formats
- Attach a finite element solver over the octree

Concluding Remarks and Future Directions

- MM is an useful tool to quickly generate meshes for high fidelity simulations
- During MM, body geometry has also to be improved
- Reodering mesh entities is easy for final level, but what about the others, if we think about MG?
- Visualization has to be improved, consequently parallel I/O
- Octree mesh generation scheme can be used to set up base mesh for MM
- High fidelity simulations on huge meshes generate lots of data; need to avoid data movement
- How to manage those complex computational infrastructure to get insight from the simulations?
- Parameter sweep and UQ adds another complexity layer, meaning hundreds of simulations, in thousands of cores, how to do it? That's big data!

Want to see more?

- Guerra, G., Rochinha, F. A., Elias, R., Oliveira, D., Ogasawara, E., Dias, J., Mattoso, M. L. Q., Coutinho, A. L. G. A., Uncertainty Quantification in Computational Predictive Models for Fluid Dynamics Using Workflow Management Engine. International Journal for Uncertainty Quantification, v. 2(1), p. 53-71, 2012.
- Camata, J. J., Coutinho, A. L. G. A.., Parallel implementation and performance analysis of a linear octree finite element mesh generation scheme, Concurrency and Computation, 2012, DOI: 10.1002/cpe.2869
- Elias, R. N., Camata, J. J., Aveleda, A.A. ; Coutinho, A. L. G. A., Evaluation of Message Passing Communication Patterns in Finite Element Solution of Coupled Problems, Lecture Notes in Computer Science, 2011. v. 6449. p. 306-313.
- Lins, E. F., Elias, R. N., Rochinha, F. A., Coutinho, Alvaro L. G. A., Residual-based variational multiscale simulation of free surface flows. Computational Mechanics, v. 46, p. 545-557, 2010.

Acknowledgements

- Supported by CNPq, Petrobras, ESSS, FAPERJ and ANP.
- Computer time on the Ranger is provided by the Texas Advanced Computer Center (TACC) at University of Texas at Austin. Part of this work has been developed under the Joint Cooperation Agreement between The Institute for Computational Sciences and Engineering (ICES), at the University of Texas at Austin and COPPE/Federal University of Rio de Janeiro Brazil.
- We are also indebted to Mr. H. Araujo from Petrobras Research Center for providing us the offshore platform STL file.

Thanks for your attention ;o)