MESH MULTIPLICATION PACKAGE INTO CODE_SATURNE AND ACHIEVED RESULTS

A. Ronovský, P. Kabelíková, V. Vondrák, C. Moulinec

Paris - Chatou, France

9.4.2013

Code_Saturne user meeting 2013

Contents

- Motivation
- Preprocessing
- Mesh Multiplication
- Results
- Perspectives

How to achieve exascale

… -> Peta -> Exa

- PRACE, EDF + STFC + IT4I
- Real complex problem
- Fully defined
- Test case: LES in staggered distributed tube bundles
- Architecture
- Solver -> Code_Saturne
- Large mesh (3D) – mesh generators?
- Post-processing
- Visualization

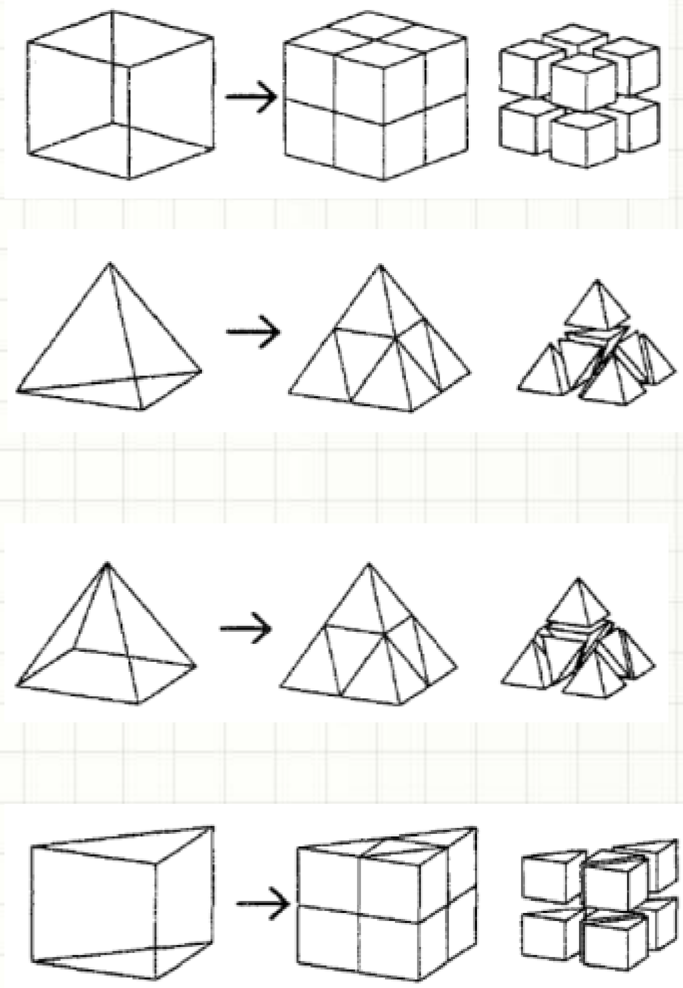
Mesh Multiplication - Overview

- Working with mesh of Billion cells
- Create or load such a mesh is very expensive

- Global refinement
- Existing coarse mesh suitable for CFD simulations, changing size by subdivision of each cell
- Creating very fine mesh, much lesser time of loading and partitioning
- higher accuracy of the solution is attained
- 13 million cell mesh to 6.6 Billion – 10 time steps
- 51 million cell mesh to 26 Billion – 1 time step
- Code_Saturne is able to solve that large problem

Mesh Multiplication - Connectivity

- Several methods of subdivision
- Different behaviour of refinement for hexahedra, tetrahedra, prism or pyramid cells
- Edge midpoints subdivision
- Global connectivity ensured
- Cheap way of indices computation
- No unnecessary core-to-core communication
- Reasonable times of refinement due to the time of whole simulation
- Lot of computational time saved = lot of resources saved for solver



MM and cs_solver.c

- Initialization (global structures)
- Define mesh to read
- Define joining and periodicity
- Set partitioning options
- Read preprocessor output
- Mesh Multiplication
- Mesh joining
- Initialize extended connectivity, ghost cells, halo
- Other mesh modifications (geometry, smoothing)
- Save mesh and discard all temporary structures
- Renumbering of a mesh, group classes, quantities, …
- Main computation
- …

Mesh Multiplication - Algorithm

- Input: coarse mesh
- Pre-processing:
  - Create edge local/global numbering,
  - Create faces to edge connectivity,
  - Define cells.
- Refinement:
  - Create new vertices on edges, on border and interior rectangular faces,
  - Refine all faces that inherit family and group from parent.
- Cell refinement:

  Preparation:
  - Create new vertex in the centre of gravity of the hexahedral cell,
  - Order faces of the cell to ensure positiveness of normal vectors,
  - Prepare indices of vertices.

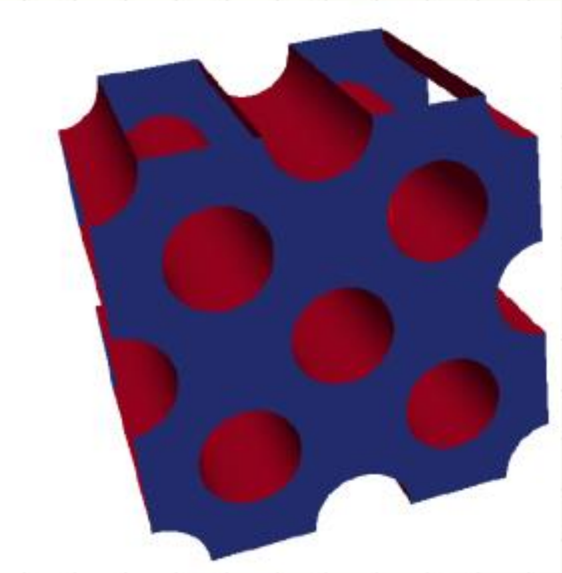
  Cell subdivision:
  - Refine the cell,
  - Create new interior faces,
  - Assign proper face to cell connectivity to each new face and cell.
- Output: refined mesh.

Mesh Multiplication - Indexation

- Vertices
  - From coarse mesh keep indices
  - Edge vertex: n_vertices + edge_idx
  - Rectangular face vertex: n_vertices + n_edges + face_idx
  - Hexa cell vertex: n_vertices + n_edges + n_faces + cell_idx
- Faces
  - Every face refined into 4
  - Refined face: 4*(face_idx - 1) + 1:4
  - New face (cell subdivision): 4*n_faces + T*(cell_idx-1) + 1:T
  - T – depends on mesh (12 for hexa, tetra, 10 for prism,…)
- Cells
  - New cell: T*(cell_idx-1) + 1:T
  - T – depends on mesh (8 for hexa, tetra, prism, …)

Results

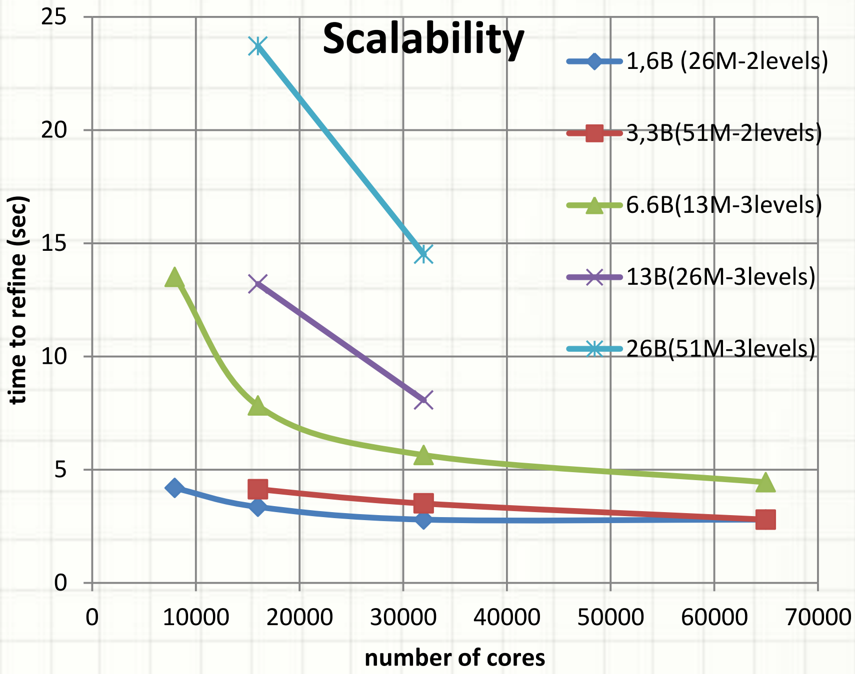
- Different architectures
- Different cases
- Mesh of 51 million cells
- Refined to 26 Billion on 65k cores
- 1 time step C_S – 12288 MPI + 8 OpenMP = ~500s



		Level of MM:	0	1	2	3
Parameters of given mesh	no. of	cells	51M	409M	3.3B	26B
		border faces	1.7M	6.7M	27M	108M
		interior faces	153M	1.2B	9.8B	78B
		vertices	52M	413M	3.3B	26B
Number of cores	16k cores	time [s]	-	2.0(4k)	4.13	23.7
		cells per core	3k	100k	200k	1.6M
	32k cores	time [s]	-	1.89(8k)	3.5	14.5
		cells per core	1.5k	50k	100k	800k
	65k cores	time [s]	-	1.2(16k)	2.79	-
		cells per core	800	25k	50k	400k

Scalability

- Good scalability up to 65k cores
- MM takes just a fraction of time due to whole computation
- MM of coarser mesh is much cheaper then creating and loading fine mesh



Perspectives

- cs_user_mesh
  - Pyramids and prisms – hybrid meshes
  - Option of mesh multiplication for every C_S user (0-default)
- Adaptive refinement
  - Global refinement adaptive to geometry
  - Local refinement based on a priori (geometry,…) and a posteriori (gradient, error, …) estimates
  - Remeshing, demeshing
  - Floating parts of a mesh, changing size, shape
- Polyhedral meshes
  - Global/ adaptive refinement of general polyhedral mesh

Thank you