Elmer

Parallel Pre- and Postprocessing Strategies to avoid bottle-necks

ElmerTeam

CSC – IT Center for Science Ltd.

PATC Elmer Course
CSC, August 2012

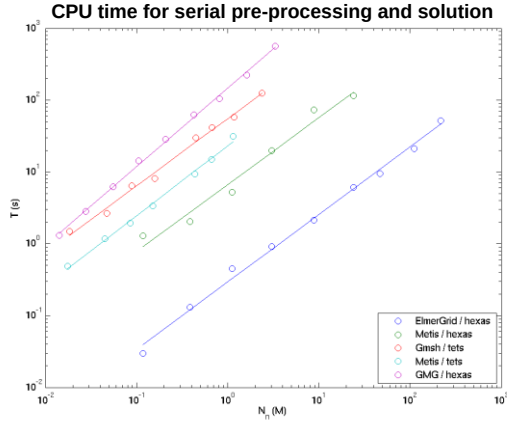
Requirements of parallel scalability

- All steps in the workflow must be considered
- Preprocessing
  - Lack of parallel tools
    -> bottle-necks in {memory, time, I/O}
  - Some possible remedies
- Computation
  - Algorithmic scalability with the problem size (weak scaling)
  - Effective parallel implementation (strong scaling)
- Postprocessing
  - Excellent parallel tools
  - Some possible ways to reduce the data

Analysis of serial workflow

- A simple Poisson problem was solved with one core
- Time consumed for each step was analyzed as a function of problem size *N*
- Observations were fitted to the model: $T=aN^b$
- solution time (GMG)
  > unstructured meshing time (gmsh)
  > partitioning time (Metis)
  > structured meshing time (ElmerGrid)
- Scalability of each step almost linear (b ≈ 1-1.1)

CPU time for serial pre-processing and solution



Scalability model

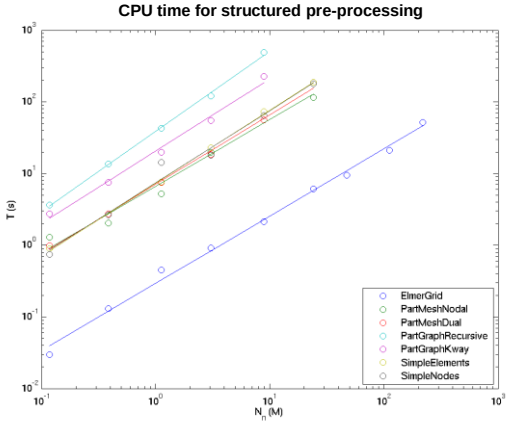
Table 9.1: Serial performance of different tools and algorithms in terms of CPU time and memory consumption

software	algorithm	mesh	α_T(s/M)	β_T	$\alpha_M(b)$
ElmerGrid	meshing	hexas	0.295	0.939	73.8
Metis	PartMeshNodal	hexas	6.67	0.932	377.0
Gmsh	Delaunay	tets	55.2	0.93	1481
Gmsh	Advancing Front	tets	155.1	1.00	643
Metis	PartMeshDual	tets	23.1	0.97	513.4
BiCGStab	CMG + SGS	hexas	134.9	1.100	1595
BiCGStab	ILU0	hexas	198.53	1.544	1717

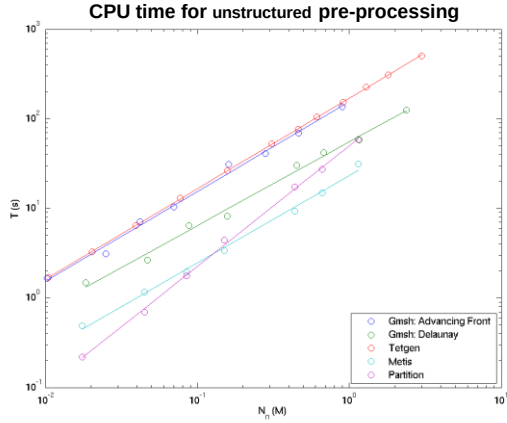
Overcoming bottle-necks in preprocessing

- Meshing is often the most difficult bottle-neck
  - Serial tools used to create up to ~1-10 M nodes
- Options for larger problems
  - Parallel mesh generation
  - Finalizing the mesh in parallel level
- Mesh partitioning is almost always less laborious than meshing
  - Serial partitioning is seldom a problem
  - There are parallel versions of partitioning tools: ParMetis

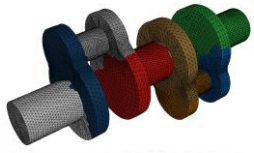
CPU time for structured pre-processing



CPU time for unstructured pre-processing

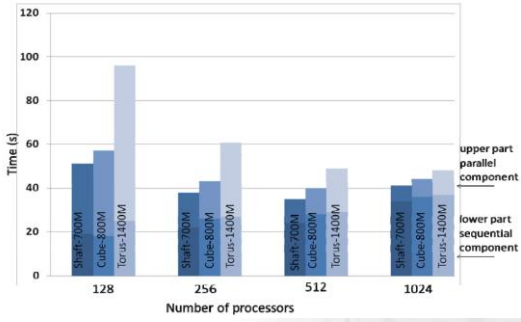


Parallel mesh generation



- Parallel mesh generation is still in its infancy
- No freely available established tools (?)
- Preliminary work for Elmer performed within PRACE in Bogazigi Univ., Istanbul
  - Y. Yılmaz, C. Özturan*, O. Tosun, A. H. Özer, S. Soner *"Parallel Mesh Generation, Migration and Partitioning for the Elmer Application"*
  - Based on netgen serial mesh generation
  - Generate coarse mesh -> partition -> mesh refinement
  - "mesh with size 1.4 billion could be generated in under a minute"
  - Still experimental, writes mesh into disk for Elmer to read -> Introduces a possible I/O bottle-neck
- Ultimately parallel mesh generation should be integrated with an API rather than disk I/O

Parallel mesh generation: performance



Y. Yılmaz et. al.
"Parallel Mesh Generation, Migration and Partitioning for the Elmer Application"

Finalizing the mesh in parallel level

- First make a coarse mesh and partition it
- Division of existing elements (*2^DIM^n* -fold problem-size)
  - Known as "Mesh Multiplication"
  - In Simulation block set "`Mesh Levels = N`"
  - There is a geometric multigrid that utilizes the mesh hierarchy
  - Simple inheritance of mesh grading
- Increase of element order (p-elements)
  - There is also a p-multigrid in Elmer
- Extrusion of 2D layer into 3D for special cases
  - Example: Greenland Ice-sheet
- For complex geometries this is often not an option
  - Optimal mesh grading difficult to maintain
  - Geometric accuracy cannot be increased

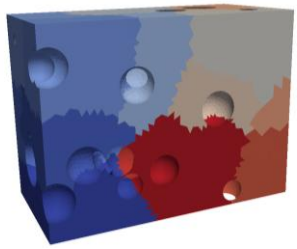
Mesh Multiplication, example

- Mesh multiplication was applied to two meshes
  - Mesh A: structured, 62500 hexahedrons
  - Mesh B: unstructured, 65689 tetrahedrons
- The CPU time used is negligible

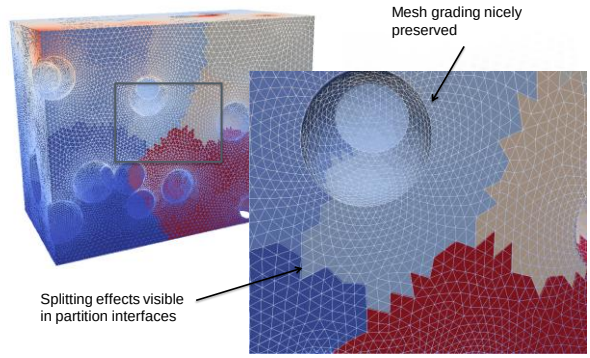
Mesh	#splits	#elems	#procs	T_center (s)	T_graded (s)
A	2	4 M	12	0.469	0.769
	2	4 M	128	0.039	0.069
	3	32 M	128	0.310	0.549
B	2	4.20 M	12	0.369	
	2	4.20 M	128	0.019	
	3	33.63 M	128	0.201	

Example, Mesh multiplication of Swiss Cheese

- Mesh multiplication on parallel level was applied to the swiss cheese case after partitioning with Metis
- Mesh grading is nicely maintained
- Presentation of spherical holes is not improved



Mesh multiplication, close-up



Mesh Multiplication, Use in Elmer

```
Simulation
  Max Output Level = 10
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Steady state
  Steady State Max Iterations = 1
  Output Intervals = 1
  Post File = case.ep

  Mesh Levels = 2
  Mesh Keep = 1
  Mesh Grading Power = 3
  Mesh Keep Grading = True
End
```

Overcoming bottle-necks in postprocessing

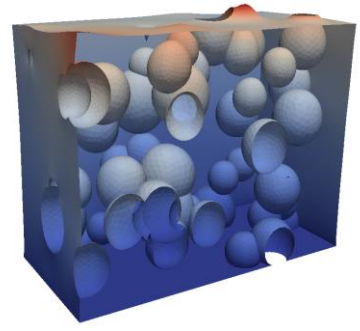
- Visualization
  - Paraview and Visit excellent tools for parallel visualization
  - Still the sheer amount of data may be overwhelming and access to all data is often an overkill
- Reducing data
  - Saving only boundaries
  - Uniform point clouds
  - A priori defined isosurfaces
  - Using coarser meshes for output when hierarchy of meshes exist
- Extracting data
  - Dimensional reduction (3D -> 2D)
  - Averaging over time
  - Integrals over BCs & bodies
- More robust I/O
  - Not all cores should write to disk in massively parallel simulations
  - HDF5+XDML output available for Elmer, mixed experiences

Example, File size in Swiss Cheese

- Memory consumption of vtu-files (for Paraview) was studied in the ”swiss cheese” case
- The ResultOutputSolver with different flags was used to write output in parallel
- Saving just boundaries in single precision binary format may save over 90% in files size compared to full data in ascii
- With larger problem sizes the benefits are amplified

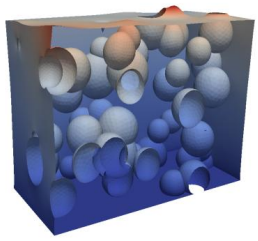
Binary output	Single Prec.	Only bound.	Bytes/node
-	X	-	376.0
X	-	-	236.5
X	X	-	184.5
X	-	X	67.2
X	X	X	38.5

Example, Saving only boundaries



Example, saving boundaries in .sif file

```
Solver 2
  Exec Solver = Always
  Equation = "result output"
  Procedure = "ResultOutputSolve" "ResultOutputSolver"
  Output File Name = case
  Vtu Format = Logical True
  Save Boundaries Only = Logical True
End
```



Relative importance of bottle-necks

- Serial runs
  - Solution is typically the bottle-neck
  - Algorithmic scalability not a major issue
- Small parallel runs (P~=10)
  - Balance between pre- prosessing rather good
  - Algorithmic scalabity already a concern
- Large parallel runs (P~=100)
  - Preprocessing often a bottle-neck but just managable
  - Postprocessing may be heavy and requires consideration
- Massively parallel runs (P~>1000)
  - All phases require special attension
  - Preprocessing either cheap and simple, or complex and parallel
  - Postprocessing often requires parallel strategies
  - Extra care must be be put to the finest details
    - Just taking an FE norm may introduce a bottle-neck

Recipes for optimal scalability in Elmer

- Finalize mesh on a parallel level (no I/O)
  - Mesh multiplication or parallel mesh generation
- Use algorithms that scale well
  - E.g. multigrid methods for elliptic problems
- If the initial problem is difficult to solve effectively divide it into simpler sub-problems
  - One component at a time -> block preconditioners
    - GCR + Block Gauss-Seidel + AMG + SGS
  - One domain at a time -> FETI
  - Splitting schemes (e.g. Pressure correction in CFD)
- Analyze results on-the-fly or reduce the amount of data for visualization