Parallel computing with Elmer

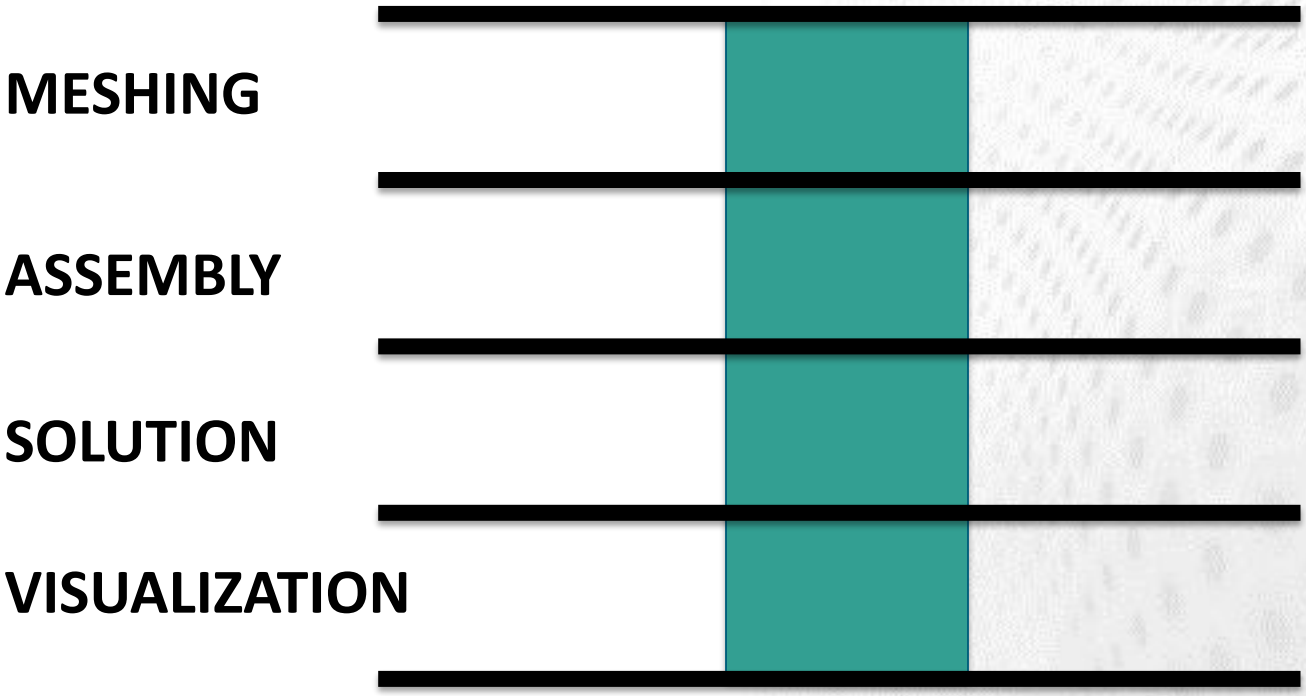
Peter Råback

ElmerTeam

CSC – IT Center for Science

Serial workflow of Elmer

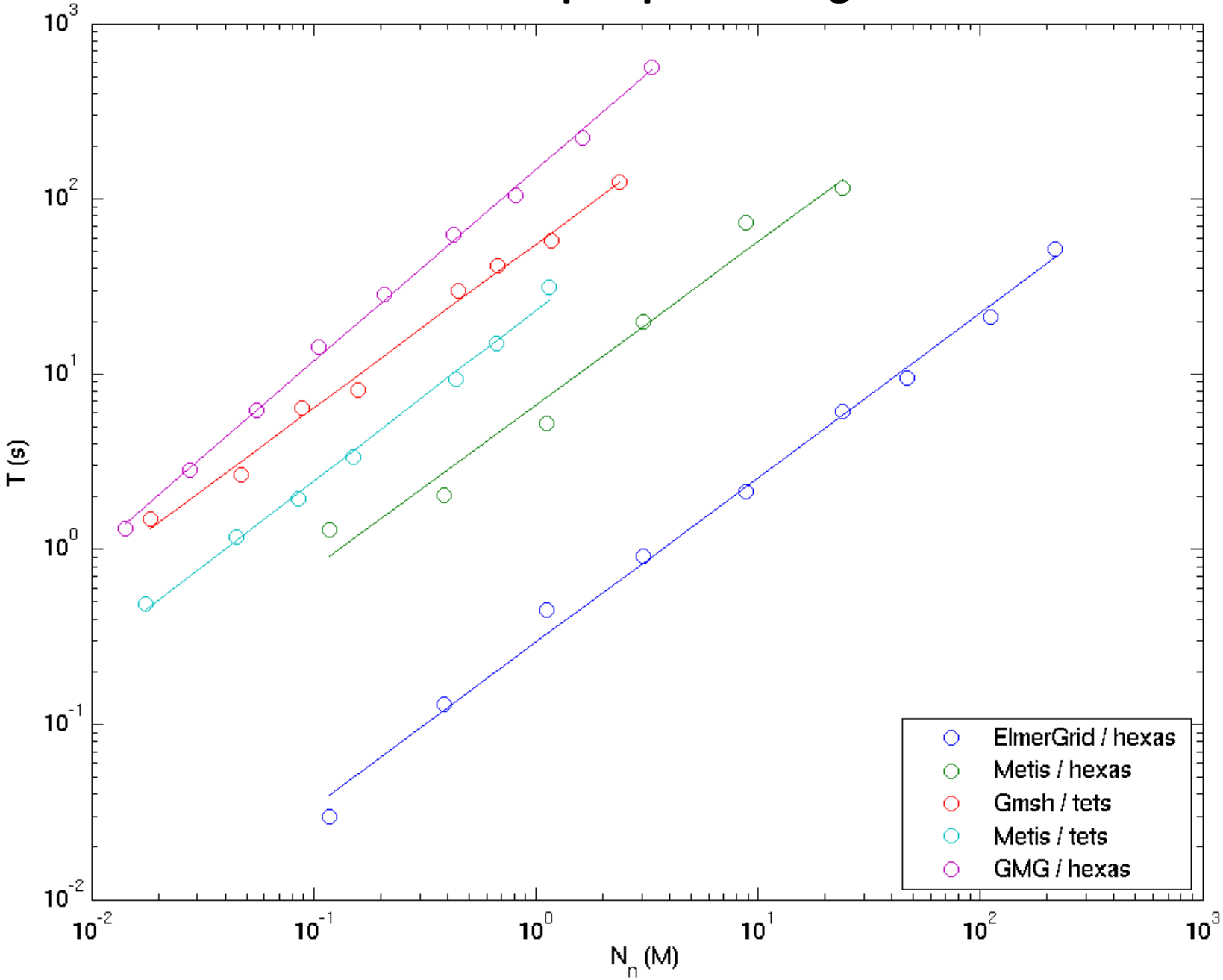
- All steps in the workflow are serial
- Typically solution of the linear system is the main bottle-neck
- For larger problems bottle-necks starts to appear in all phases of the serial workflow



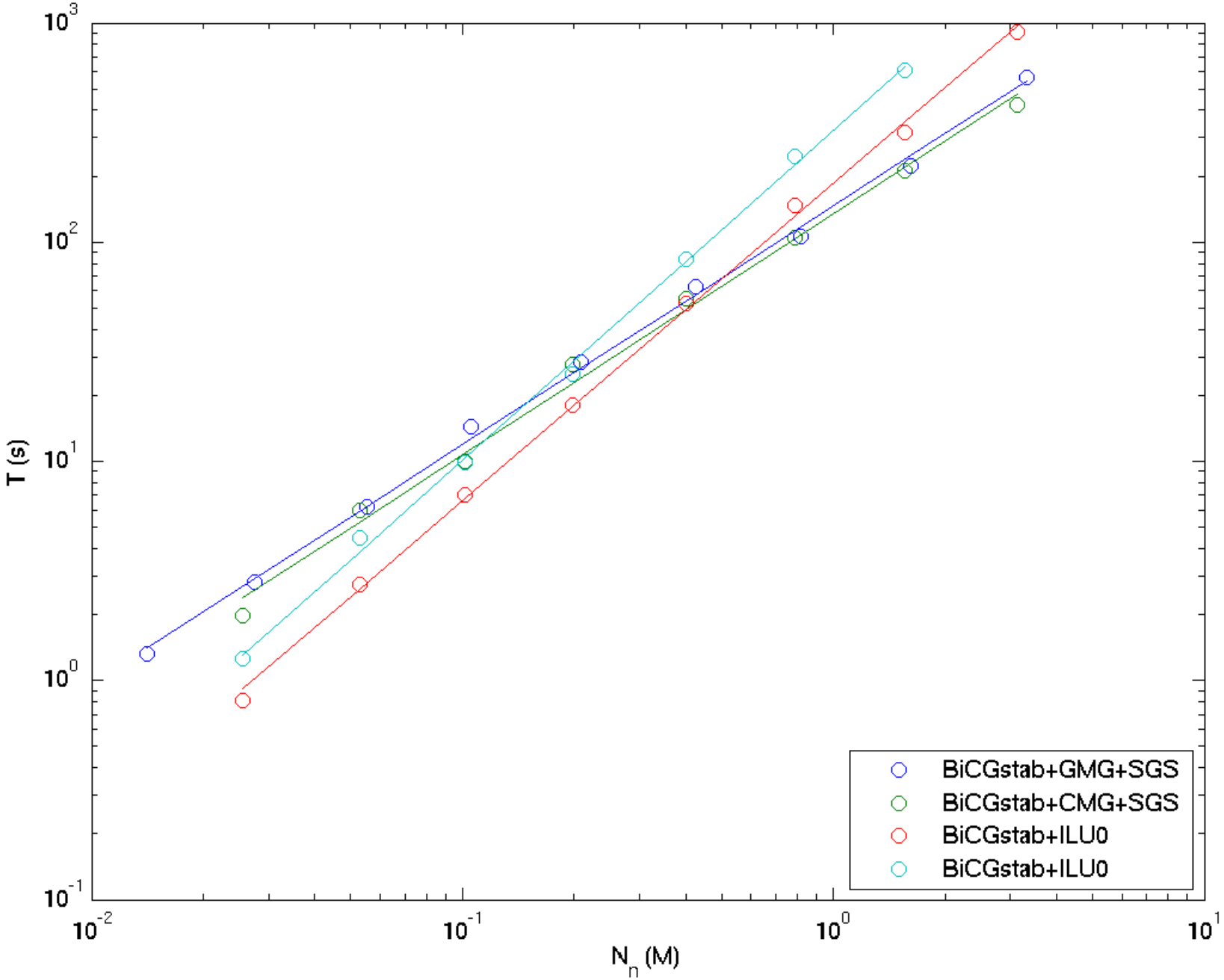
Algorithmic scalability

- Each algorithm has a characteristic scaling law that sets the lower limit to how the solution time increases with problem size
- Typical scaling for linear solvers
  - Multigrid: *O(~n log(n))*
  - Iterative Krylov methods: *O(~n^1.5)*
- The parallel implementation cannot hope to beat this limit
  - Targeting large problems the starting point should be nearly optimal algorithm!

CPU time for serial pre-processing and solution



CPU time for solution – one level vs. multilevel



Example: Scalability model

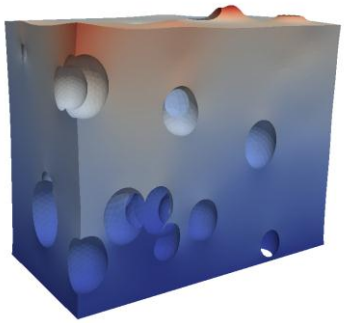
Table 9.1: Serial performance of different tools and algorithms in terms of CPU time and memory consumption

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

T(solution) > T(tet meshing) > T(partitioning) > T(hex meshing)

The solution is the first bottleneck even for simple equations, for complex equations and transient problems even more so!

Motivation for using optimal linear solvers



- Comparison of scaling in linear elasticity between different preconditioners: ILU1 vs. block preconditioning with multigrid
- At smallest system performance about the same
- Increasing size with 8^3=512 gives the block solver scalability of ***O(~1.03)*** while ILU1 fails to converge

	BiCGstab(4)+ILU1		GCR+BP(AMG)	
#dofs	T(s)	#iters	T(s)	#iters
7,662	1.12	36	1.19	34
40,890	11.77	76	6.90	45
300,129	168.72	215	70.68	82
2,303,472	>21,244*	>5000*	756.45	116

* No convergence was obtained

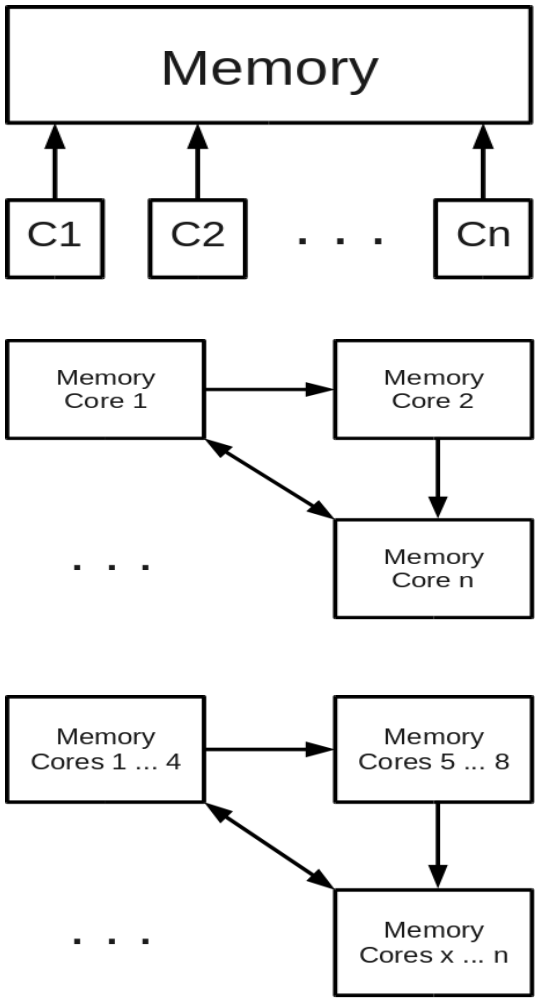
Simulation Peter Råback, CSC, 2012.

Parallel computing concepts

- Parallel computation means executing several tasks concurrently
  - A task encapsulates a sequential program and local data, and its interface to its environment
  - Data of those other tasks is remote
- Data dependency means that the computation of one task requires data from an another task in order to proceed
  - FEM is inherently data dependent reflecting the interactions of the physical reality

Parallel computers

- Shared memory
  - All cores can access the whole memory
- Distributed memory
  - All cores have their own memory
  - Communication between cores is needed in order to access the memory of other cores
- Current supercomputers combine the distributed and shared memory approaches



Parallel programming models

- Message passing (MPI)
  - Can be used both in distributed and shared memory computers
  - Programming model allows good parallel scalability
  - Programming is quite explicit
- Threads (pthreads, OpenMP)
  - Can be used only in shared memory computer
  - Limited parallel scalability
  - Simpler or less explicit programming
- Elmer historically uses MPI
  - Recent developments towards multithreading using OpenMP

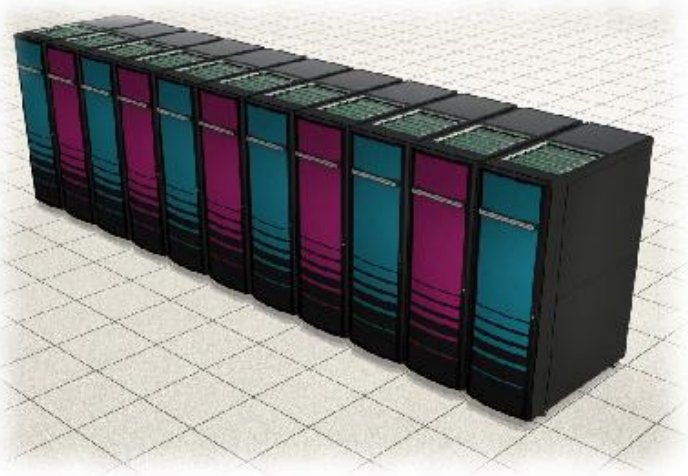
Execution model

- Parallel program is launched as a set of independent, identical processes
  - The same program code and instructions
  - Can reside in different computation nodes
  - Or even in different computers

```
ELMER SOLVER (v 6.2) STARTED AT: 2012/01/03 10:21:59
ELMER SOLVER (v 6.2) STARTED AT: 2012/01/03 10:21:59
ELMER SOLVER (v 6.2) STARTED AT: 2012/01/03 10:21:59
ELMER SOLVER (v 6.2) STARTED AT: 2012/01/03 10:21:59
ParCommInit:  Initialize #PEs:           4
MAIN:
MAIN: =============================================================
MAIN: ElmerSolver finite element software, Welcome!
MAIN: This program is free software; you can redistribute it and/or
MAIN: modify it under the terms of the GNU General Public License
MAIN: Copyright 1st April 1995 - , CSC - IT Center for Science Ltd.
MAIN: Webpage http://www.csc.fi/elmer, Email elmeradm@csc.fi
MAIN: Library version: 6.2 (Rev: 5472)
MAIN:  Running in parallel using 4 tasks.
MAIN:  HYPRE library linked in.
MAIN:  MUMPS library linked in.
MAIN: =============================================================
```

General remarks about parallel computing

- Current CPU's in your workstations
  - Six cores (e.g. AMD Opteron Shanghai)
- Multi-threading
  - e.g. OpenMP
- High performance Computing (HPC)
  - Message passing, e.g. OpenMPI

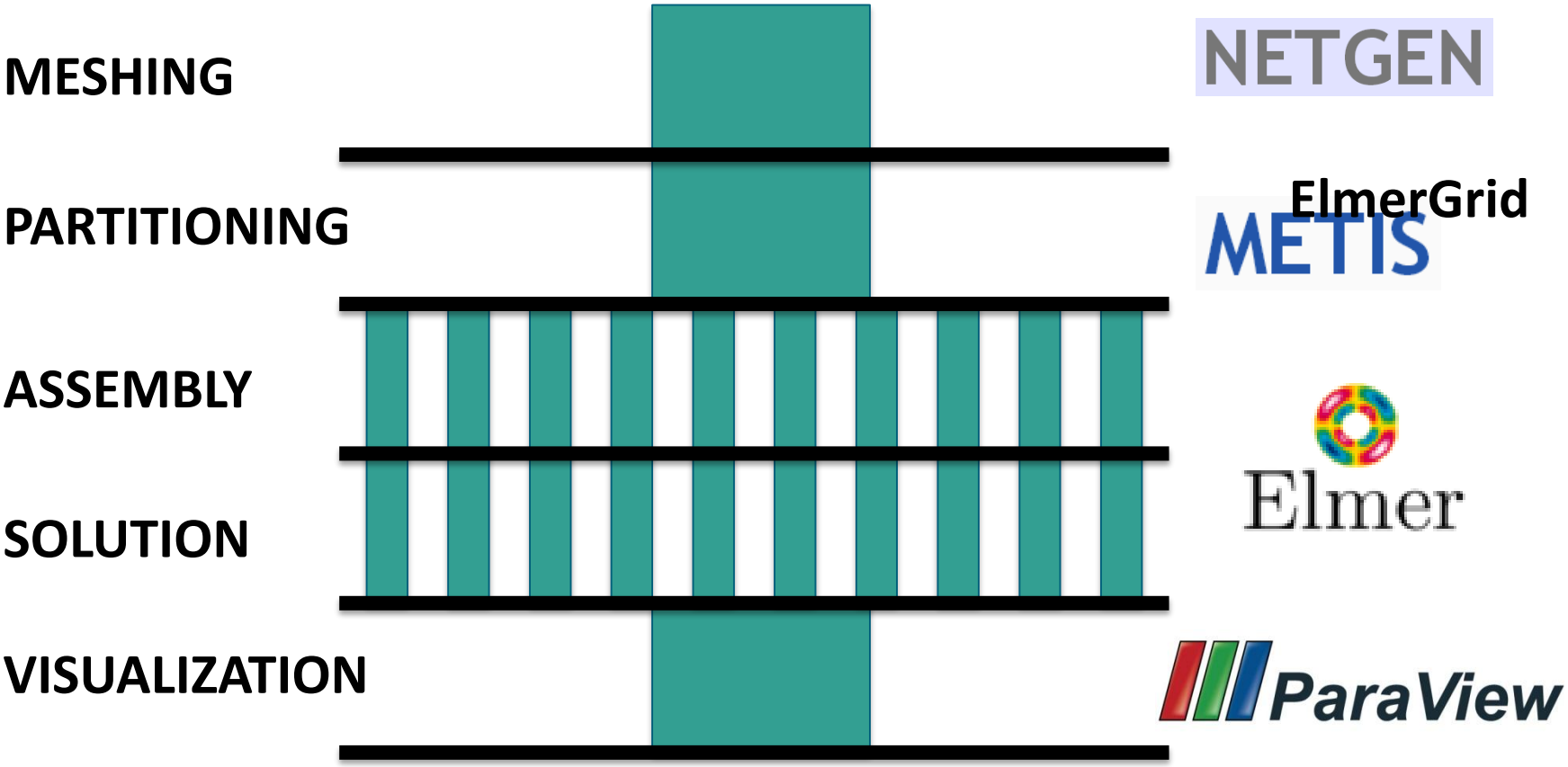


Weak vs. strong scaling

- In parallel computing there are two common notions
- **strong scaling**
  - How the solution time varies with the number of processors for a *fixed total problem size*.
  - Optimal case: $P \times T = const$.
  - A bad algorithm may have excellent strong scaling
  - Typically 1e4-1e5 dofs needed in FEM for good scaling
- **weak scaling**
  - How the solution time varies with the number of processors for *a fixed problem size per processor*.
  - Optimal case: *T=const.*
  - Weak scaling is limited by algorithmic scaling

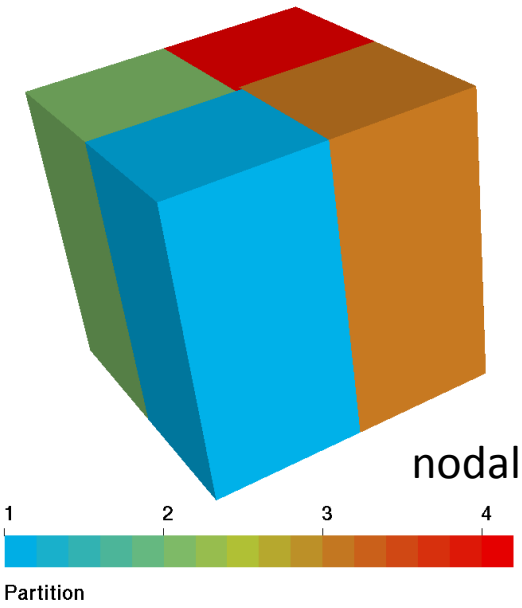
Parallel workflow of Elmer I

- Both assembly and solution is done in parallel using MPI
- Assembly is trivially parallel
- This is the most common parallel workflow

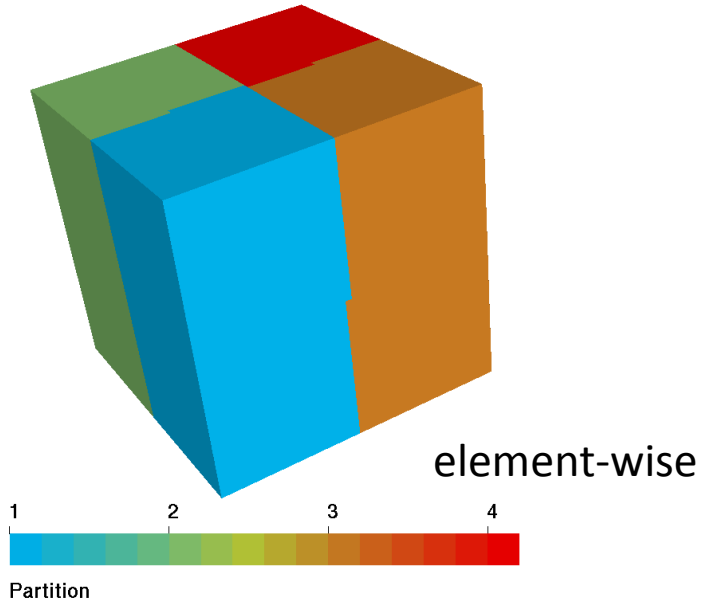


ElmerGrid partitioning by direction

- Directional decomposition (*Np=Nx*Ny*Nz*)
  - `ElmerGrid 2 2 meshdir -partition Nx Ny Nz Nm`
- Optional redefinition of major axis with a given normal vector
  - `-partorder nx ny nz`



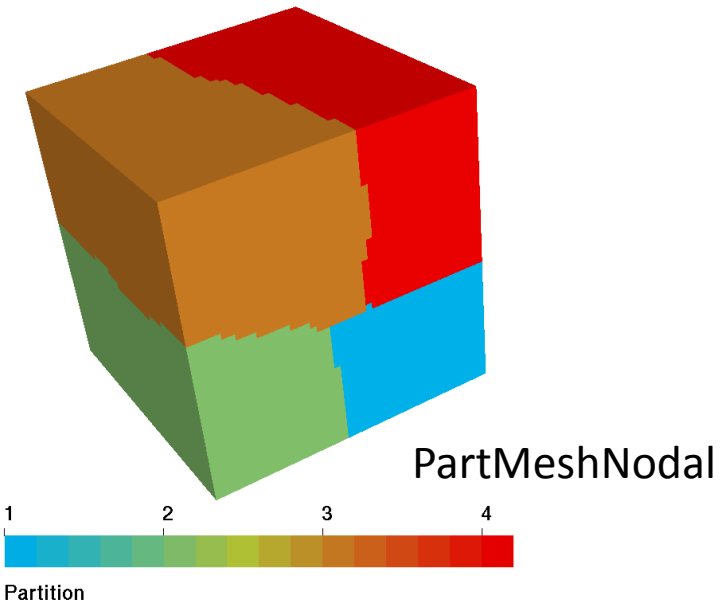
`-partition 2 2 1 0`



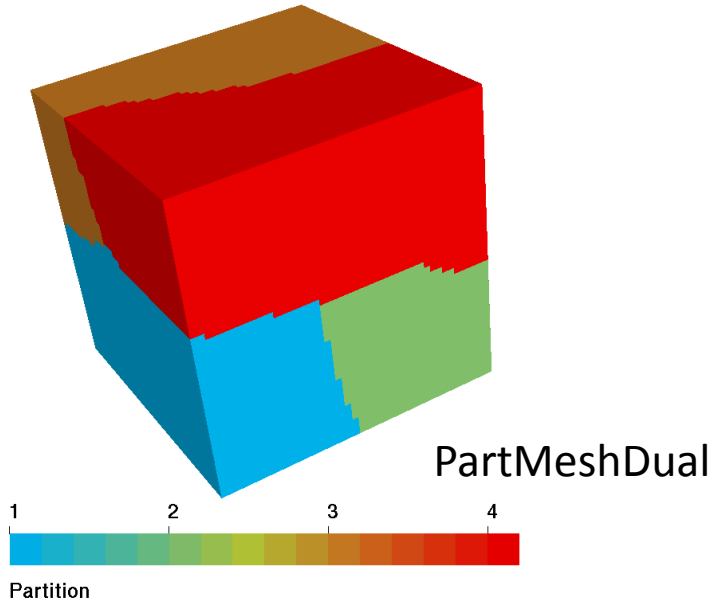
`-partition 2 2 1 1`

ElmerGrid partitioning by Metis

- Using Metis library
  - `ElmerGrid 2 2 meshdir -metis Np Nm`

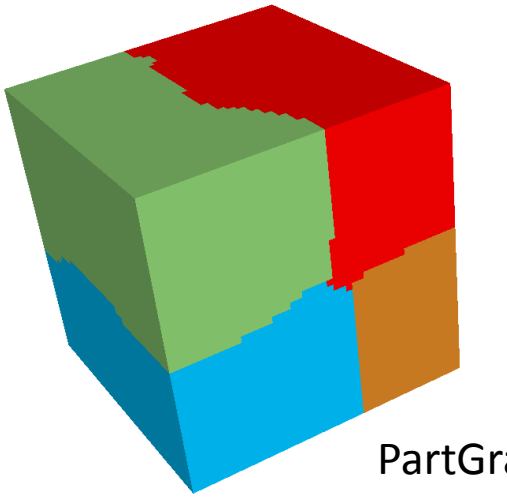


`-metis 4 0`



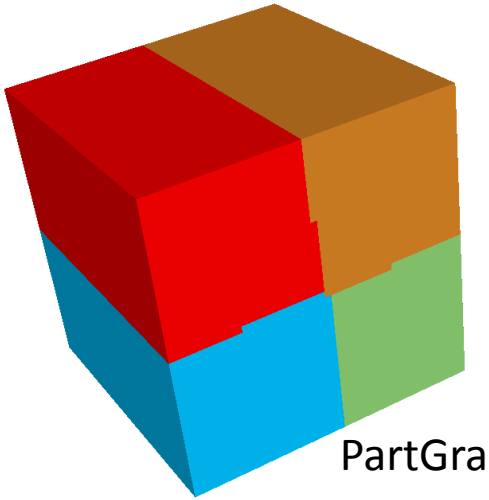
`-metis 4 1`

ElmerGrid partitioning by Metis, continued



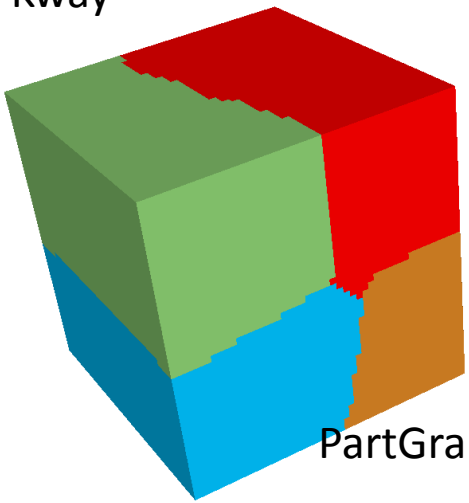
PartGraphPKway

`-metis 4 4`



PartGraphRecursive

`-metis 4 2`



PartGraphKway

`-metis 4 3`

********** Elmergrid **********
This program can create simple 2D structured meshes consisting of linear, quadratic or cubic rectangles or triangles. The meshes may also be extruded and revolved to create 3D forms. In addition many mesh formats may be imported into Elmer software. Some options have not been properly tested. Contact the author if you face problems.

The program has two operation modes
A) Command file mode which has the command file as the only argument
'ElmerGrid commandfile.eg'

B) Inline mode which expects at least three input parameters
'ElmerGrid 1 3 test'

The first parameter defines the input file format:
1) .grd : Elmergrid file format
2) .mesh.* : Elmer input format
3) .ep : Elmer output format
4) .ansys : Ansys input format
5) .inp : Abaqus input format by Ideas
6) .fil : Abaqus output format
7) .FDNEUT : Gambit (Fidap) neutral file
8) .unv : Universal mesh file format
9) .mphtxt : Comsol Multiphysics mesh format
10) .dat : Fieldview format
11) .node,.ele: Triangle 2D mesh format
12) .mesh : Medit mesh format
13) .msh : GID mesh format
14) .msh : Gmsh mesh format
15) .ep.i : Partitioned ElmerPost format

The second parameter defines the output file format:
1) .grd : ElmerGrid file format
2) .mesh.* : ElmerSolver format (also partitioned .part format)
3) .ep : ElmerPost format
4) .msh : Gmsh mesh format

Listing of ”magic numbers” when calling ElmerGrid without parameters

Parallel options of ElmerGrid

```
The following keywords are related only to the parallel Elmer computations.
-partition int[4]     : the mesh will be partitioned in main directions
-partorder real[3]    : in the above method, the direction of the ordering
-metis int[2]         : the mesh will be partitioned with Metis
-halo                 : create halo for the partitioning
-indirect             : create indirect connections in the partitioning
-periodic int[3]      : decleare the periodic coordinate directions for parallel me
-partjoin int         : number of partitions in the data to be joined
-saveinterval int[3] : the first, last and step for fusing parallel data
-partorder real[3]    : in the above method, the direction of the ordering
-partoptim            : apply aggressive optimization to node sharing
-partbw               : minimize the bandwidth of partition-partion couplings
-parthypre            : number the nodes continously partitionwise
```

Mesh structure of Elmer

Serial

`meshdir/`

- `mesh.header`
  size info of the mesh
- `mesh.nodes`
  node coordinates
- `mesh.elements`
  bulk element defs
- `mesh.boundary`
  boundary element defs with reference to parents

Parallel

`meshdir/partitioning.N/`

- `mesh.n.header`
- `mesh.n.nodes`
- `mesh.n.elements`
- `mesh.n.boundary`
- `mesh.n.shared`
  information on shared nodes

for each i in [0,N-1]

Parallellism in ElmerSolver library

- Parallelization mainly with MPI
  - Some work on OpenMP threads
- Assembly
  - Each partition assemblies it's own part, no communication
- Parallel Linear solvers included in Elmer
  - Iterative Krylov methods
    - CG, BiCGstab, BiCGStabl, QCR, GMRes, TFQMR,…
    - Require only matrix-vector product with parallel communication
  - Geometric Multigrid (GMG)
    - Utilizes mesh hierarchies created by mesh multiplication
  - Preconditioners
    - ILUn performed block-wise
    - Diagonal and Vanka exactly the same in parallel
    - GMG also as a preconditioner
  - FETI, Finite element tear and interconnect

Parallel external libraries for Elmer

- MUMPS
  - Direct solver that may work when averything else fails
- Hypre
  - Large selection of methods
  - Algebraic multigrid: Boomer MG (classical multigrid)
  - Parallel ILU preconditioning
  - Approximate inverse preconditioning: Parasails
- Trilinos
  - Interface to ML multigrid solver (agglomeration multigrid)
  - Krylov solvers

Serial vs. parallel solution

Serial

- Serial mesh files
- Command file (.sif) may be given as an inline parameter
- Execution with `ElmerSolver [case.sif]`
- Writes results to one file

Parallel

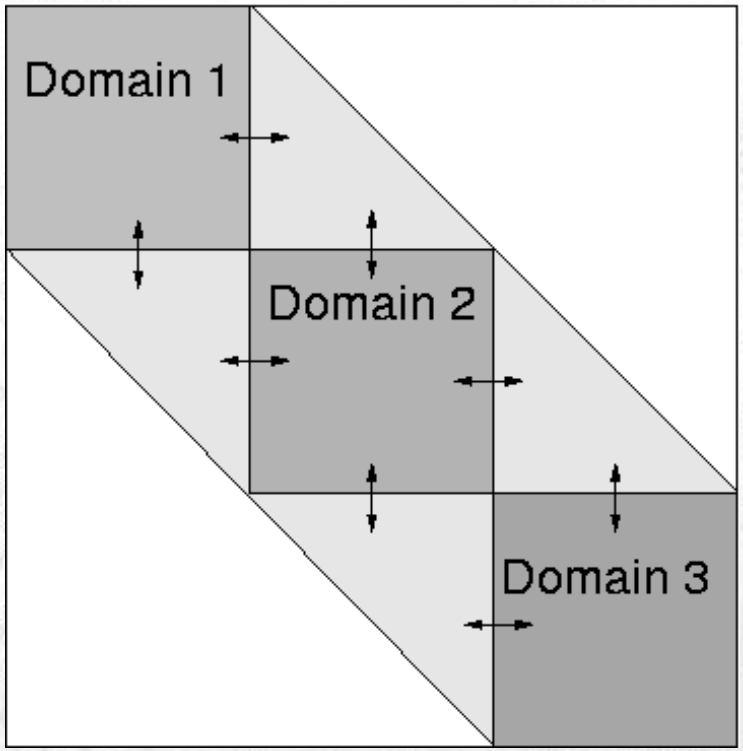
- Partitioned mesh files
- ELMERSOLVER_STARTINFO is always needed to define the command file (.sif)
- Execution with `mpirun -np N ElmerSolver_mpi`
- Calling convention is platform dependent
- Writes results to *N* files

Observations in parallel runs

- Typically good scale-up in parallel runs requires around 1e4 dofs in each partition
  - Otherwise communication of shared node data will start to dominate
- To take use of the local memory hierarchies the local problem should not be too big either
  - Sometimes superlinear speed-up is observed when the local linear problem fits to the cache memory
- Good scaling has been shown up to thousands of cores
- Simulation with over one billion unknowns has been performed
- Preconditioners not always the same in parallel
  - May detoriorate parallel performance

Differences in serial and parallel algorithms

- Some algorithms are slightly different in parallel
- ILU in ElmerSolver library is performed only blockwise which may result to inferior convergence
- Diagonal and vanka preconditions are exactly the same in parallel



Parallel postprocessing with ElmerPost

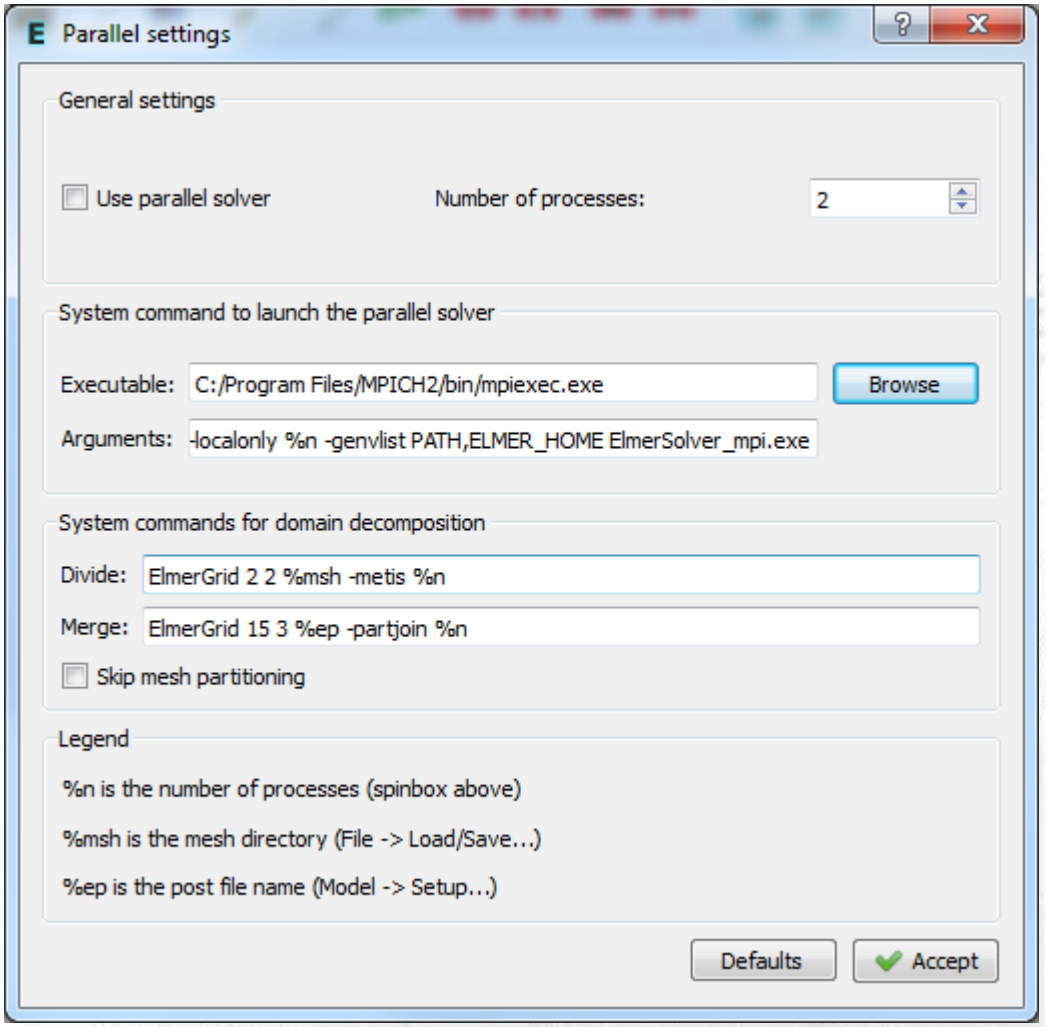
- ElmerSolver writes results in partitionwise.
  - `name.0.ep`
  - `name.1.ep`
  - `...`
  - `name.(N-1).ep`
- ElmerGrid is used to fuse files together into a single file called name.ep
  - `ElmerGrid 15 3 dirname`
- Option for choosing timestep
  - `-saveinterval Nstart Nfin Nstep`

Parallel postprocessing using Paraview

- Use ResultOutputSolver to save data to `.vtu` files
- The operation is almost the same for parallel data as for serial data
- There is a extra file `.pvtu` that holds is a wrapper for the parallel `.vtu` data of each partition

Parallel computation in ElmerGUI

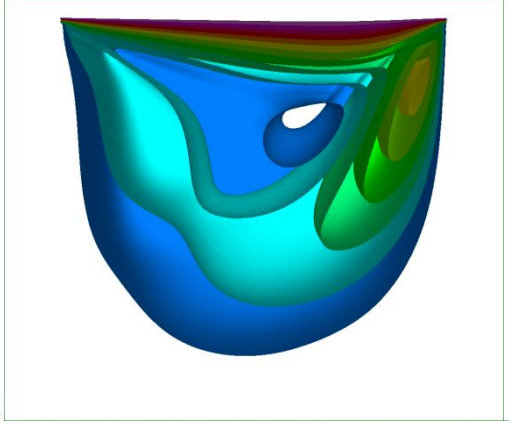
- If you have parallel environment it can also be used interactively via **ElmerGUI**
- Calls **ElmerGrid** automatically for partiotioning and fusing
- Not supported by current Windows version

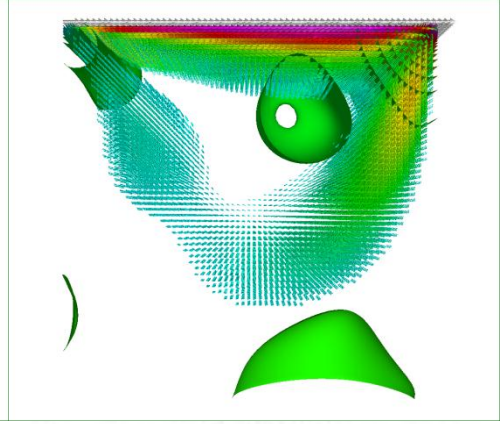


Parallel performance

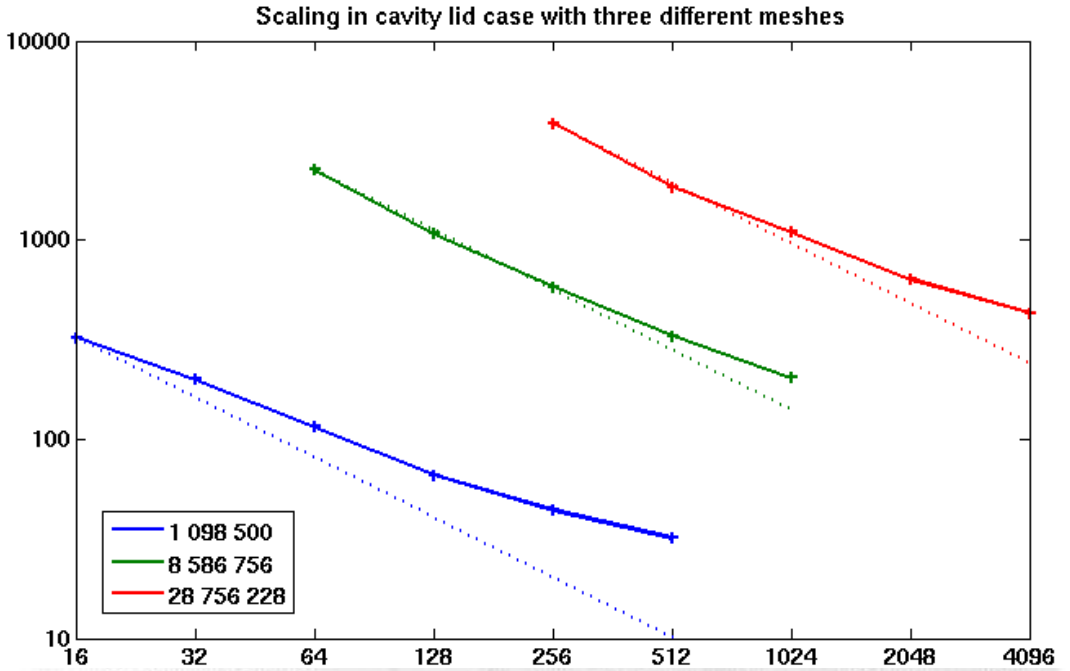
- Cavity lid case solved with the monolithic N-S solver
- Partitioning with Metis
- Solver Gmres with ILU0 preconditioner

Louhi: Cray XT4/XT5 with 2.3 GHz 4-core AMD Opteron. All-in-all 9424 cores and Peak power of 86.7 Tflops.



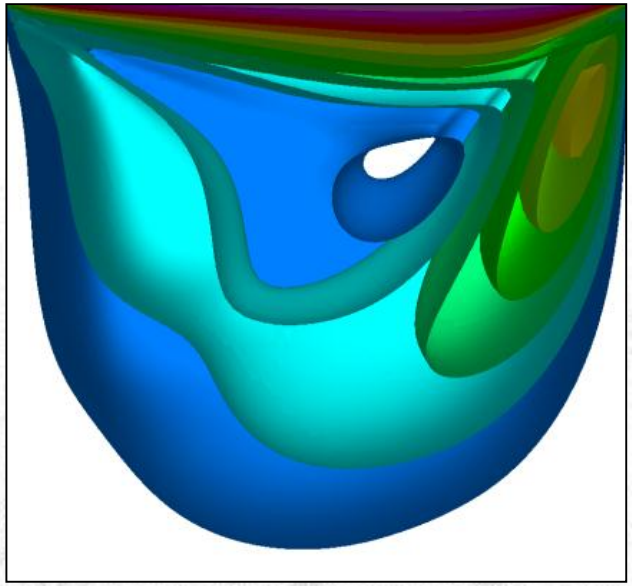


Scaling of wall clock time with dofs in the cavity lid case using GMRES+ILU0. Simulation Juha Ruokolainen, CSC, visualization Matti Gröhn, CSC, 2009.



Block prec.: Weak scaling of 3D driven-cavity

Elems	Dofs	#procs	Time (s)
34^3	171,500	16	44.2
43^3	340,736	32	60.3
54^3	665,500	64	66.7
68^3	1,314,036	128	73.6
86^3	2,634,012	256	83.5
108^3	5,180,116	512	102.0
132^3	9,410,548	1024	106.8



O(~1.14)

Velocity solves with Hypre: CG + BoomerAMG preconditioner for the 3D driven-cavity case (Re=100) on Cray XC (Sisu).
Simulation Mika Malinen, CSC, 2013.

Block preconditioning in Elmer

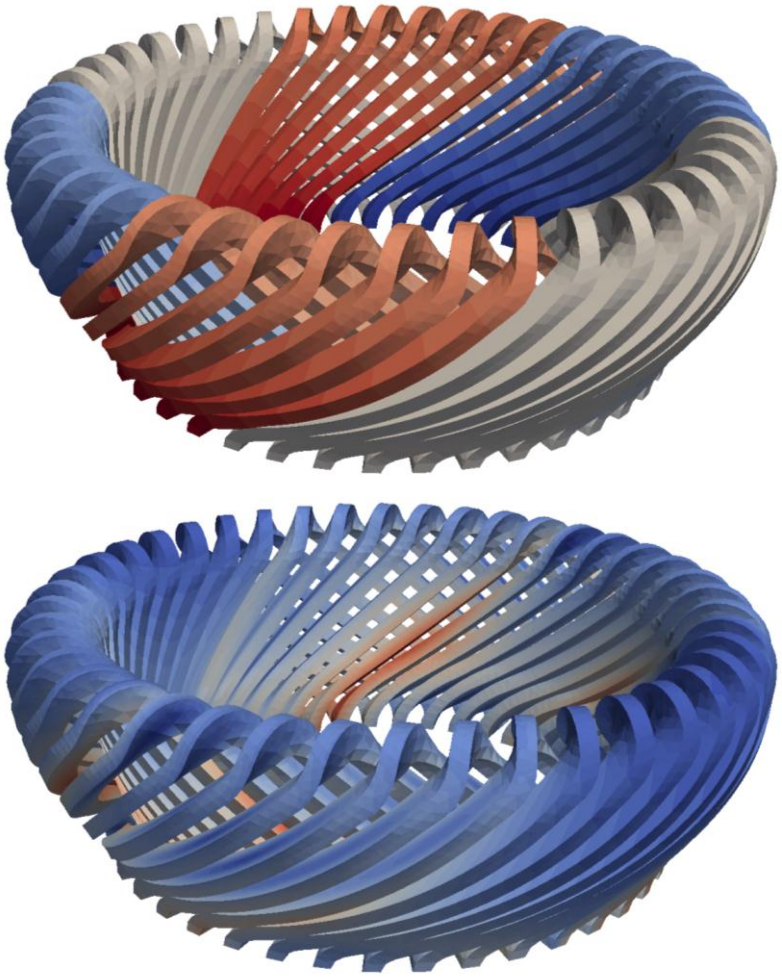
- In Parallel runs a central challenge is to have good **parallel preconditioners**
- This problem is increasingly difficult for PDEs with vector fields
  - Navier-Stokes, elasticity equation,...
- Idea: Use as preconditioner a procedure where the components are solved one-by-one (like in Gauss-Seidel) and the solution is used as a search direction in an outer Krylov method
- Number of outer iterations may be shown to be bounded
- Individual blocks may be solved with optimally scaling methods (AMG)
- PARA2012 Presentation: M. Malinen et al.
  "Parallel Block Preconditioning by Using the Solver of Elmer"

Example: Weak scaling of Elmer (FETI)

#Procs	Dofs	Time (s)	Efficiency
8	0.8	47.80	-
64	6.3M	51.53	0.93
125	12.2M	51.98	0.92
343	33.7M	53.84	0.89
512	50.3M	53.90	0.89
1000	98.3M	54.54	0.88
1331	131M	55.32	0.87
1728	170M	55.87	0.86
2197	216M	56.43	0.85
2744	270M	56.38	0.85
3375	332M	57.24	0.84

Solution of Poisson equation with FETI method where local problem (of size 32^3=32,768 nodes) and coarse problem (distributed to 10 partitions) is solved with MUMPS. Simulation with Cray XC (Sisu) by Juha Ruokolainen, CSC, 2013.

Scalability of edge element AV solver for end-windings

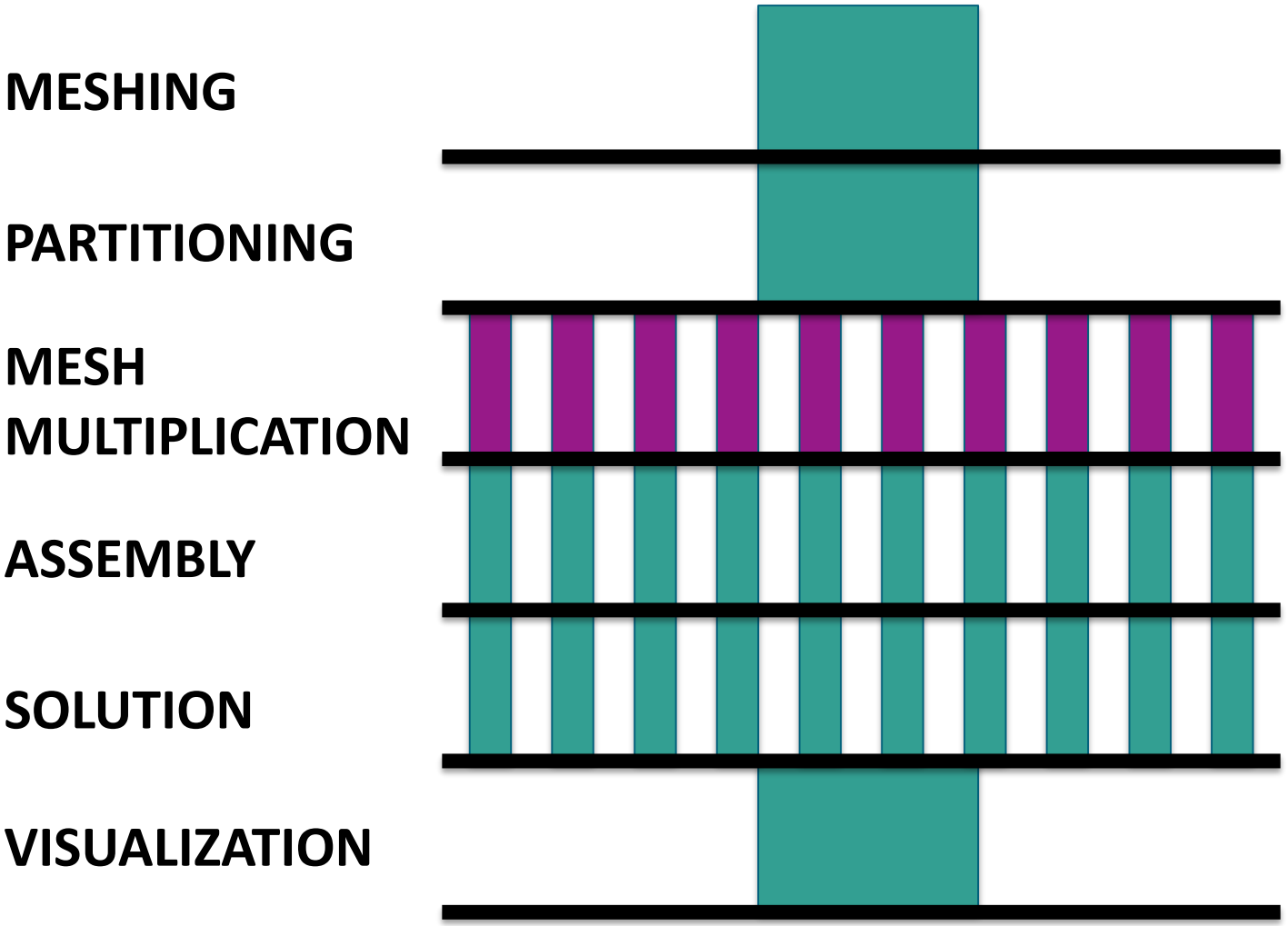


#Procs	Time(s)	T_{2P}/T_P
4	1366	-
8	906	1.5
16	260	3.5
32	122	2.1
64	58.1	2.1
128	38.2	1.8
256	18.1	2.1

Magnetic field strength (left) and electric potential (right) of an electrical engine end-windings. Meshing M. Lyly, ABB. Simulation (Cray XC, Sisu) J. Ruokolainen, CSC, 2013.

Parallel workflow in Elmer II

- Large meshes may be finilized at the parallel level



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

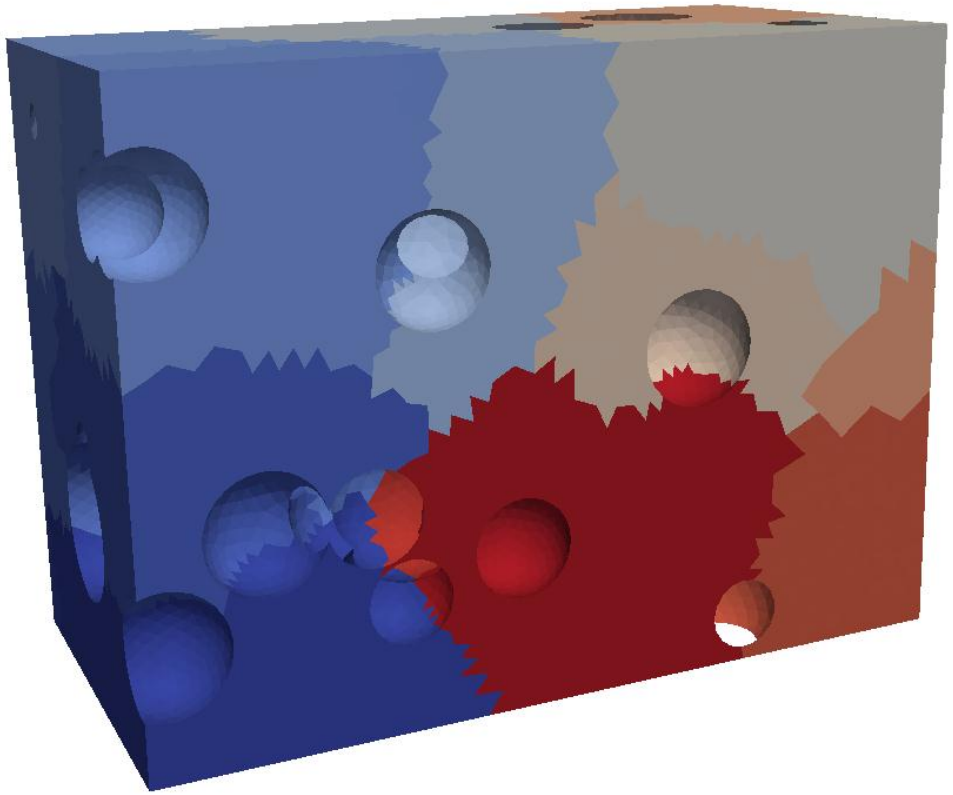
Mesh Multiplication, example

- Implemented in Elmer as internal strategy ~2005
- Mesh multiplication was applied to two meshes
  - Mesh A: structured, 62500 hexahedrons
  - Mesh B: unstructured, 65689 tetrahedrons
- The CPU time used is negligible

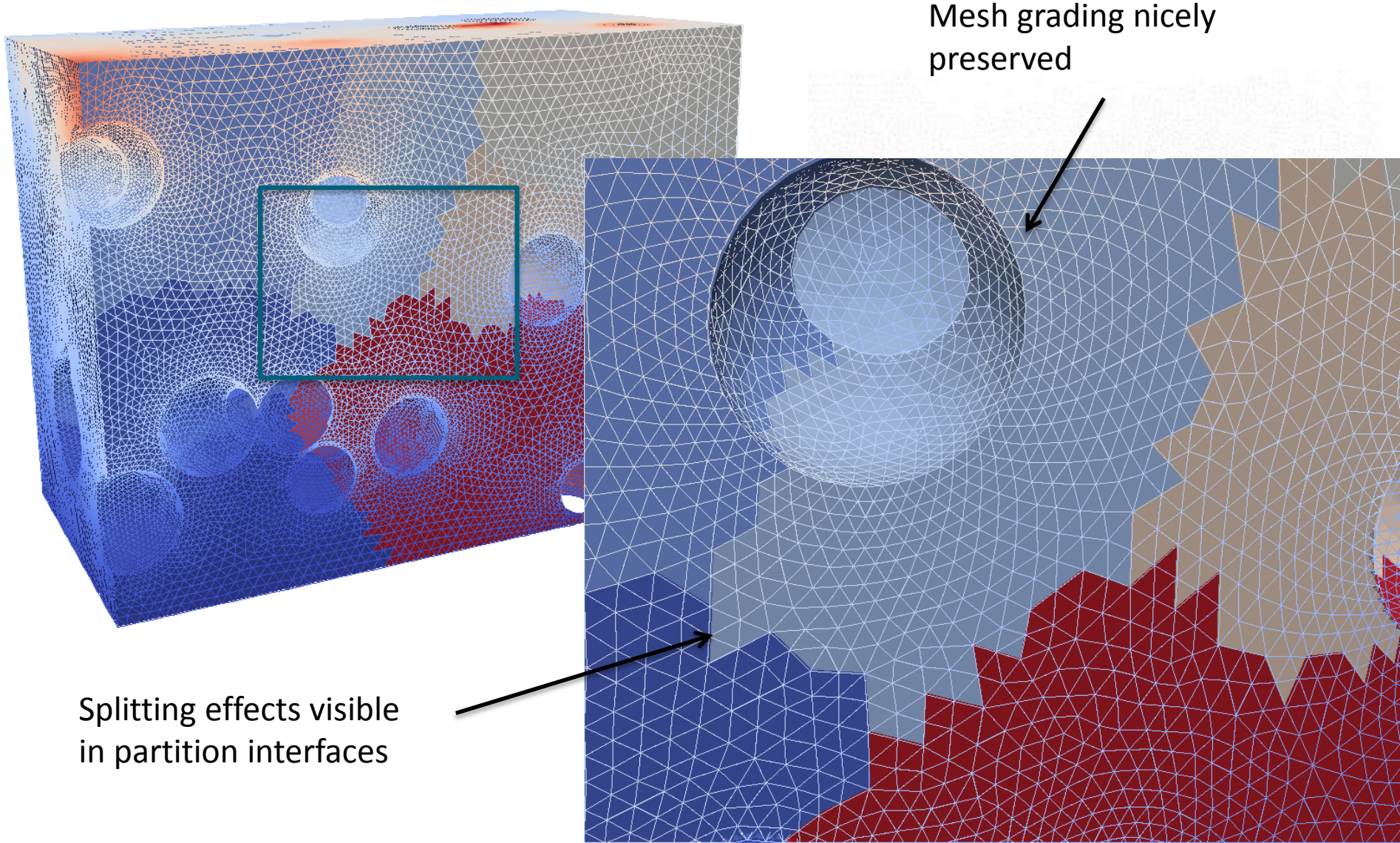
Mesh	#splits	#elems	#procs	T_center (s)	T_graded (s)
A	2	4 M	12	0.469	0.769
	2	4 M	128	0.039	0.069
	3	32 M	128	0.310	0.549
B	2	4.20 M	12	0.369	
	2	4.20 M	128	0.019	
	3	33.63 M	128	0.201	

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 balls is not improved



Mesh multiplication, close-up

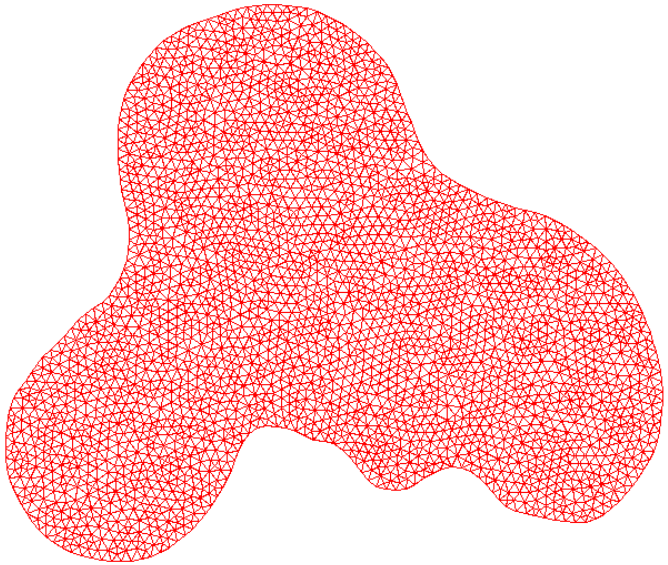


Mesh Multiplication, Use in Elmer

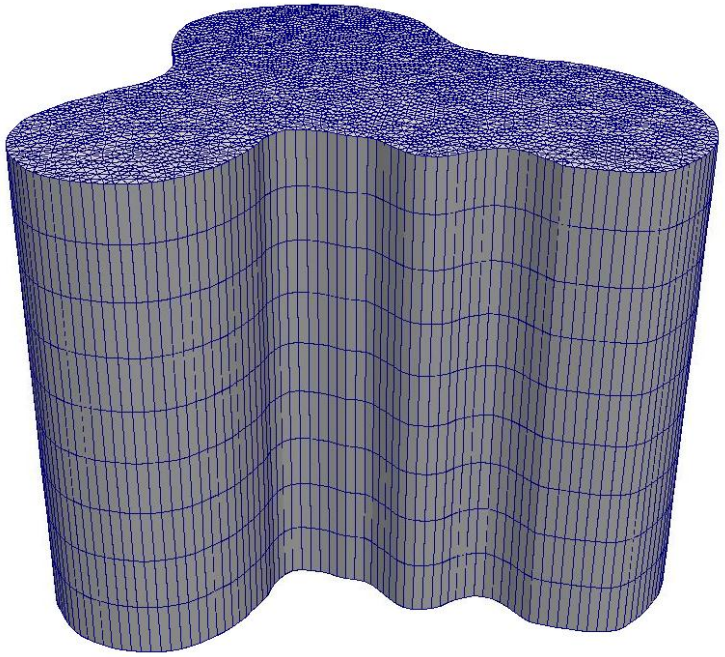
```
Simulation
  Max Output Level = 10
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Steady state
  Steady State Max Iterations = 1
  Output Intervals = 1
  Post File = case.ep

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

Parallel internal extrusion

- First partition a 2D mesh, extrude an the parallel level



2D mesh by Gmsh

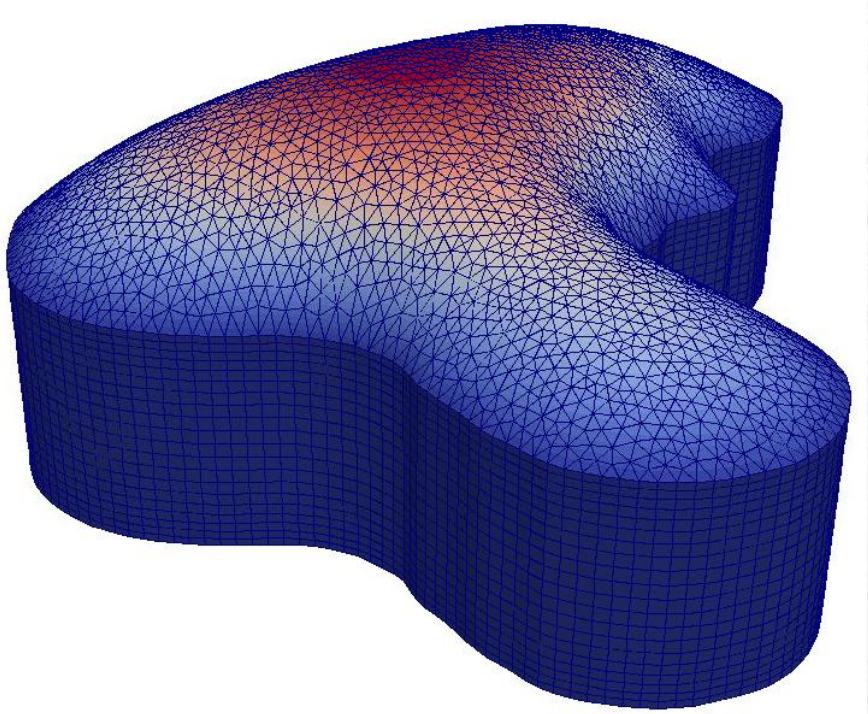


3D internally extruded mesh

```
Simulation
  Extruded Mesh Levels = 10
  Extruded Mesh Density = Variable Coordinate 3
    Real MATC f(tx) ! Any function
  …
```

Using extruded meshes

- Meshes can be cheaply adjusted to the geometrical height models by taking use of the extruded structure
  - Beneficial for the simulation of large continental ice-sheets



Parallel workflow in Elmer III

- Bottle-necks in preprocessing resolved by parallel meshing

COARSE MESHING

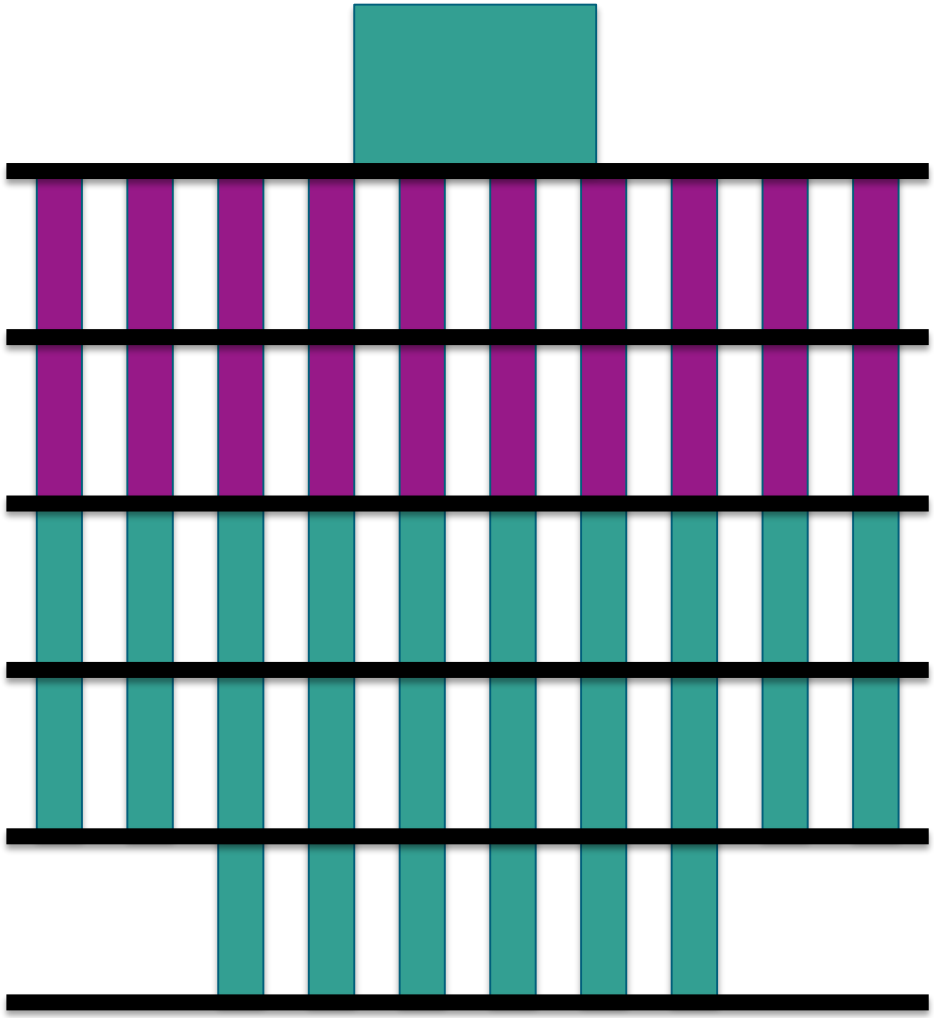
PARTITIONING

FINE MESHING

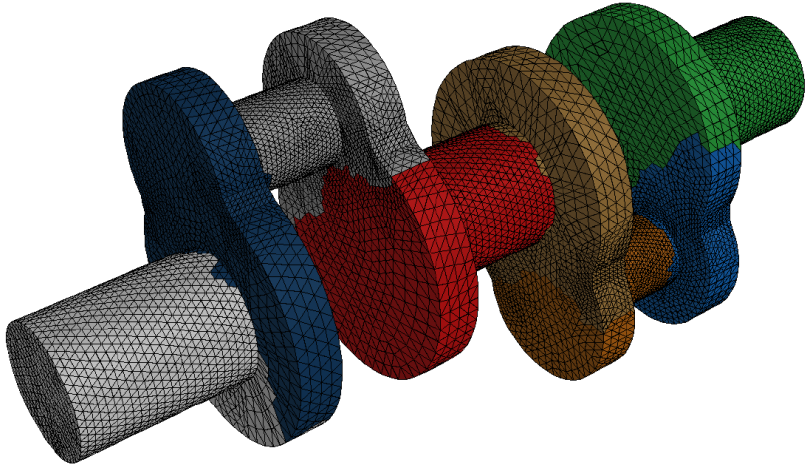
ASSEMBLY

SOLUTION

VISUALIZATION

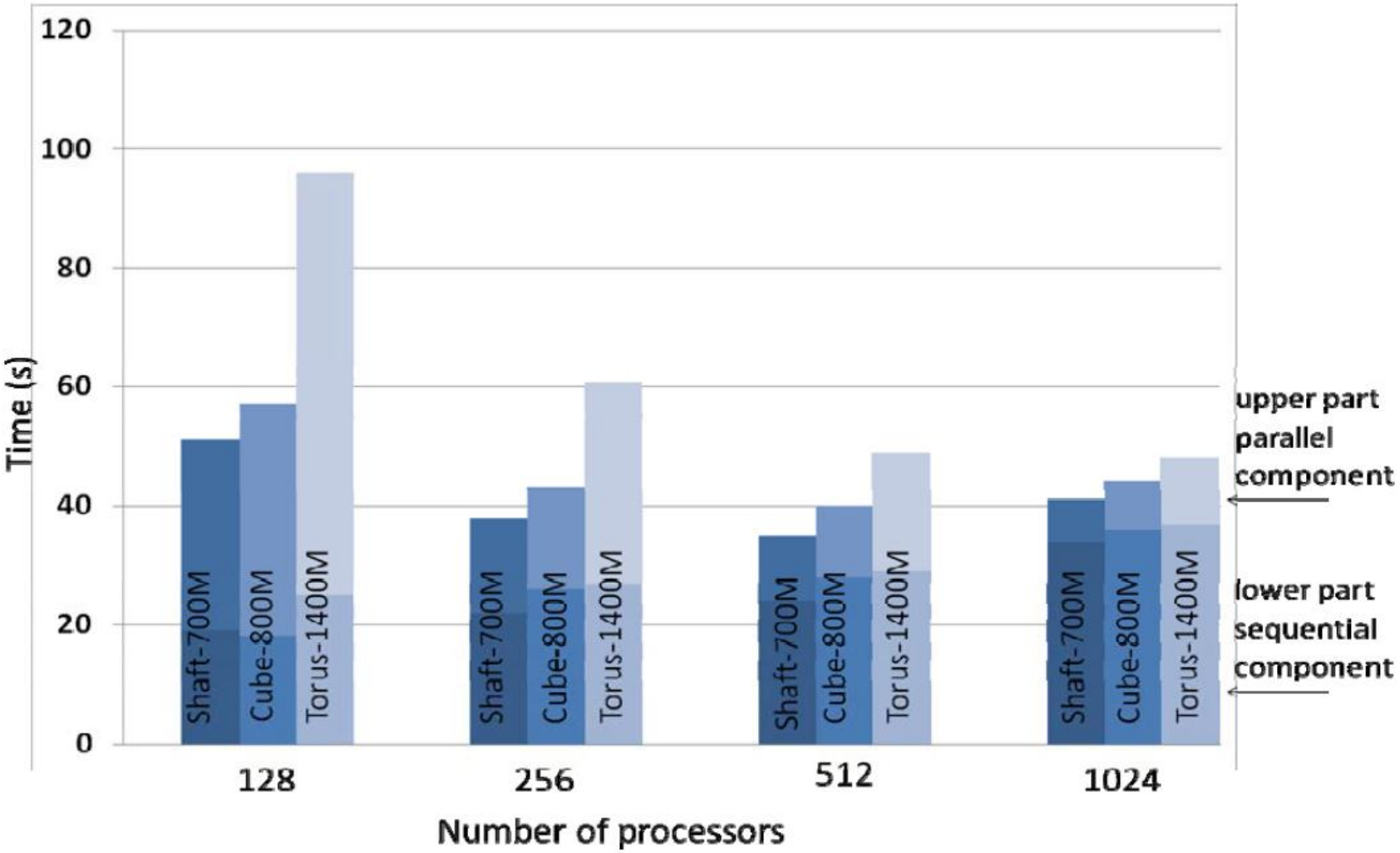


Parallel mesh generation



- 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"*

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

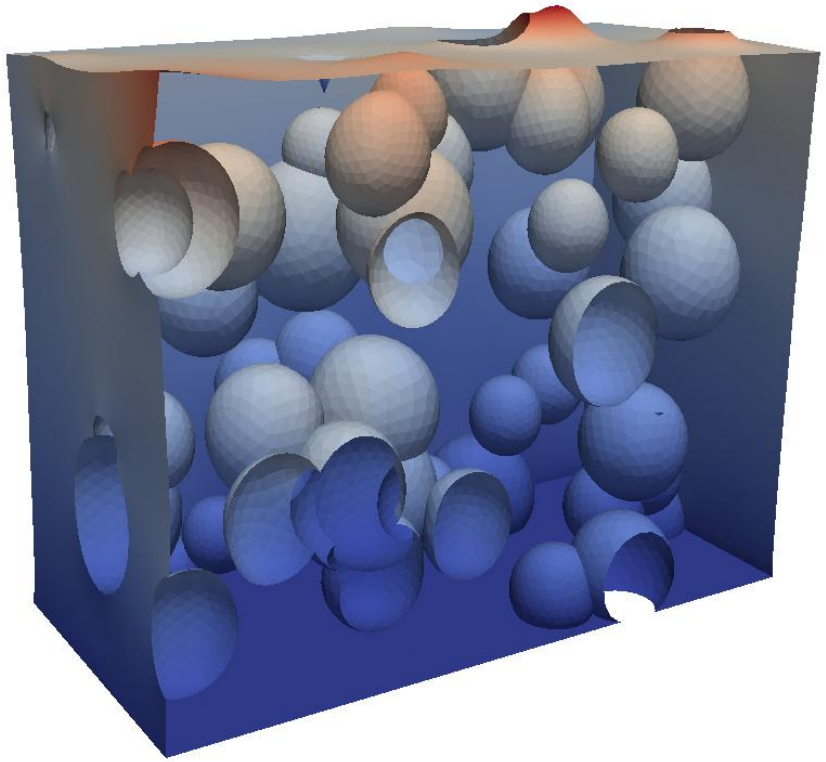
- Memory consumption of vtu-files (for Paraview) was studied in the ”swiss cheese” case
- The ResultOutputSolver with different flags was used to write output in parallel
- Saving just boundaries in single precision binary format may save over 90% in files size compared to full data in ascii
- With larger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

Simulation Peter Råback, CSC, 2012.

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
```



Recipes for resolving scalability bottle-necks

- Finalize mesh on a parallel level (no I/O)
  - Mesh multiplication or parallel mesh generation
- Use algorithms that scale well
  - E.g. Multigrid methods
- 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 and reduce the amount of data for visualization

Future outlook

- Deeper integration of the workflow
  - Heavy pre- and postprocessing internally or via API
- Cheaper flops from new multicore environments
  - Interesting now also for the finite element solvers
  - Usable via reasonable programming effort; attention to algorithms and implementation
- Complex physics introduces always new bottle-necks
  - Rotating boundary conditions in parallel…

Thank you for your attention!