# Parallel Computing

with

# Elmer

ElmerTeam

CSC – IT Center for Science, Finland

Elmer FEM webinar

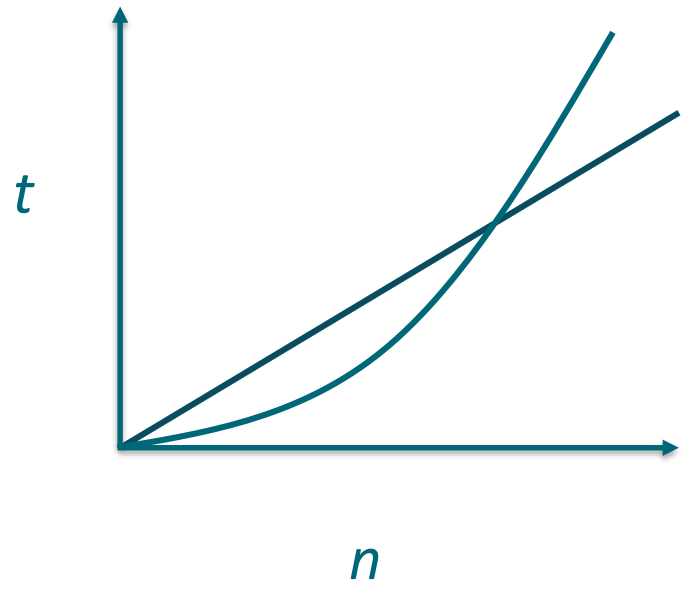
2021

# Outline for today

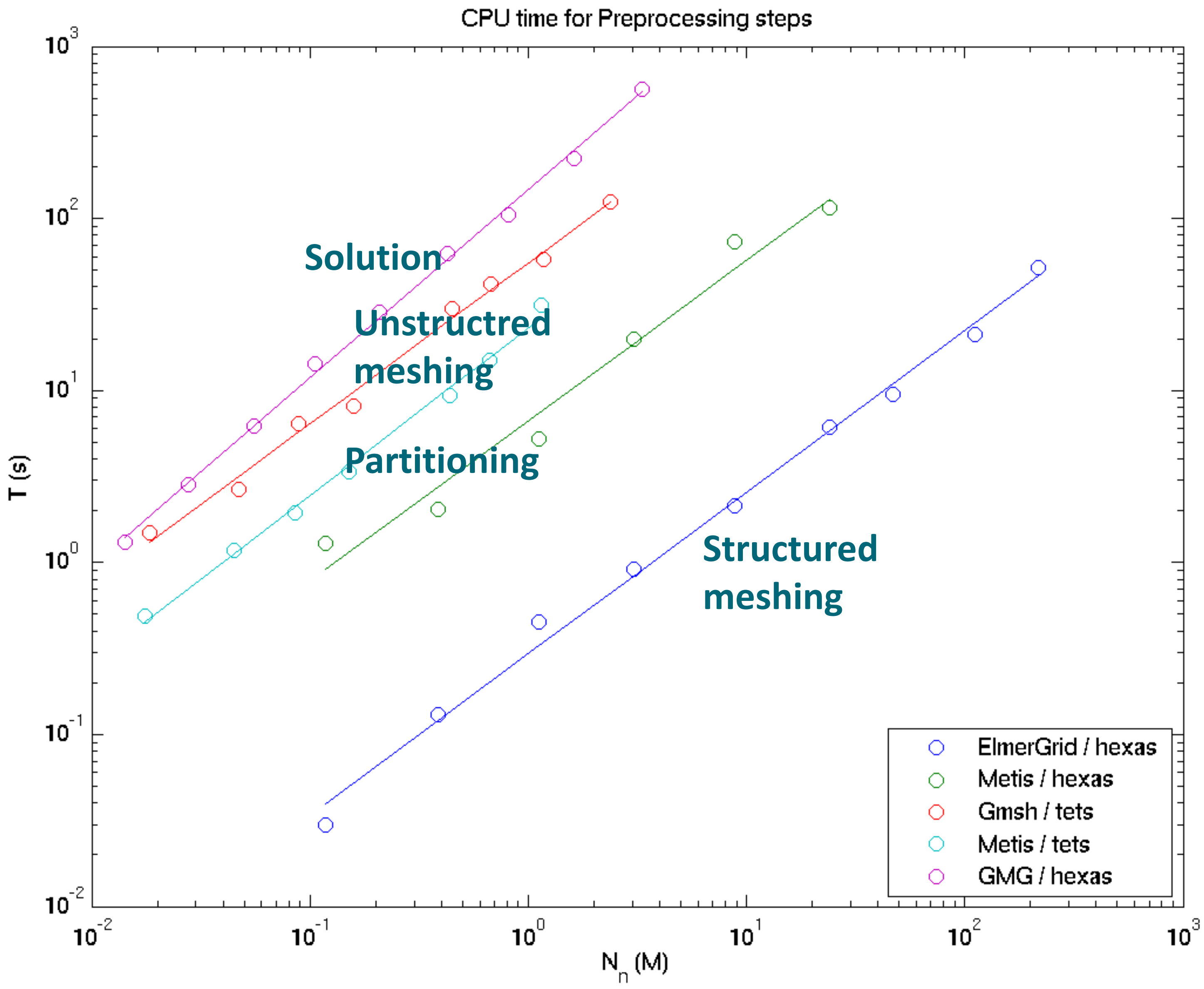
- Algorithmic scalability
- Linear solvers & preconditioners
- Parallel computing principles
- Strong and weak scaling
- Partitioning of meshes
- Parallel vs. serial solution
- Examples on HPC platforms
- Architecture developments

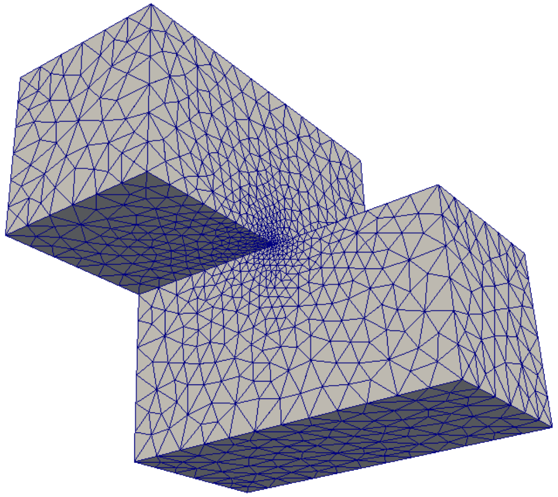
# Algorithmic scalability

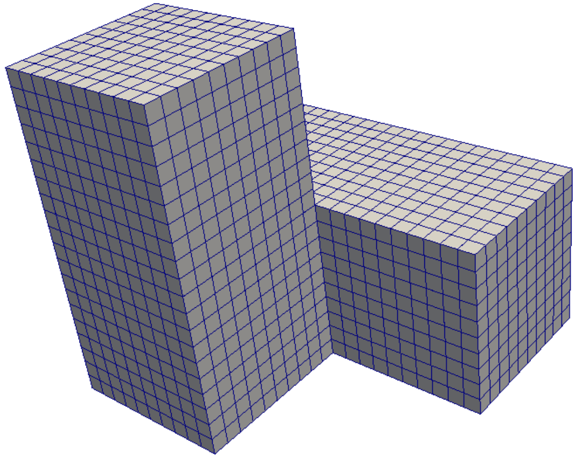
- Before going into parallel computation let's study where the bottle-necks will appear in the serial system
- Each algorithm/procedure has a characteristic scaling law that sets the lower limit to how the solution time ***t*** increases with problem size ***n***
  - The parallel implementation cannot hope to beat this limit systematically
- Targeting very large problems the starting point should be nearly optimal (=linear) algorithm!



# CPU time for serial pre-processing and solution

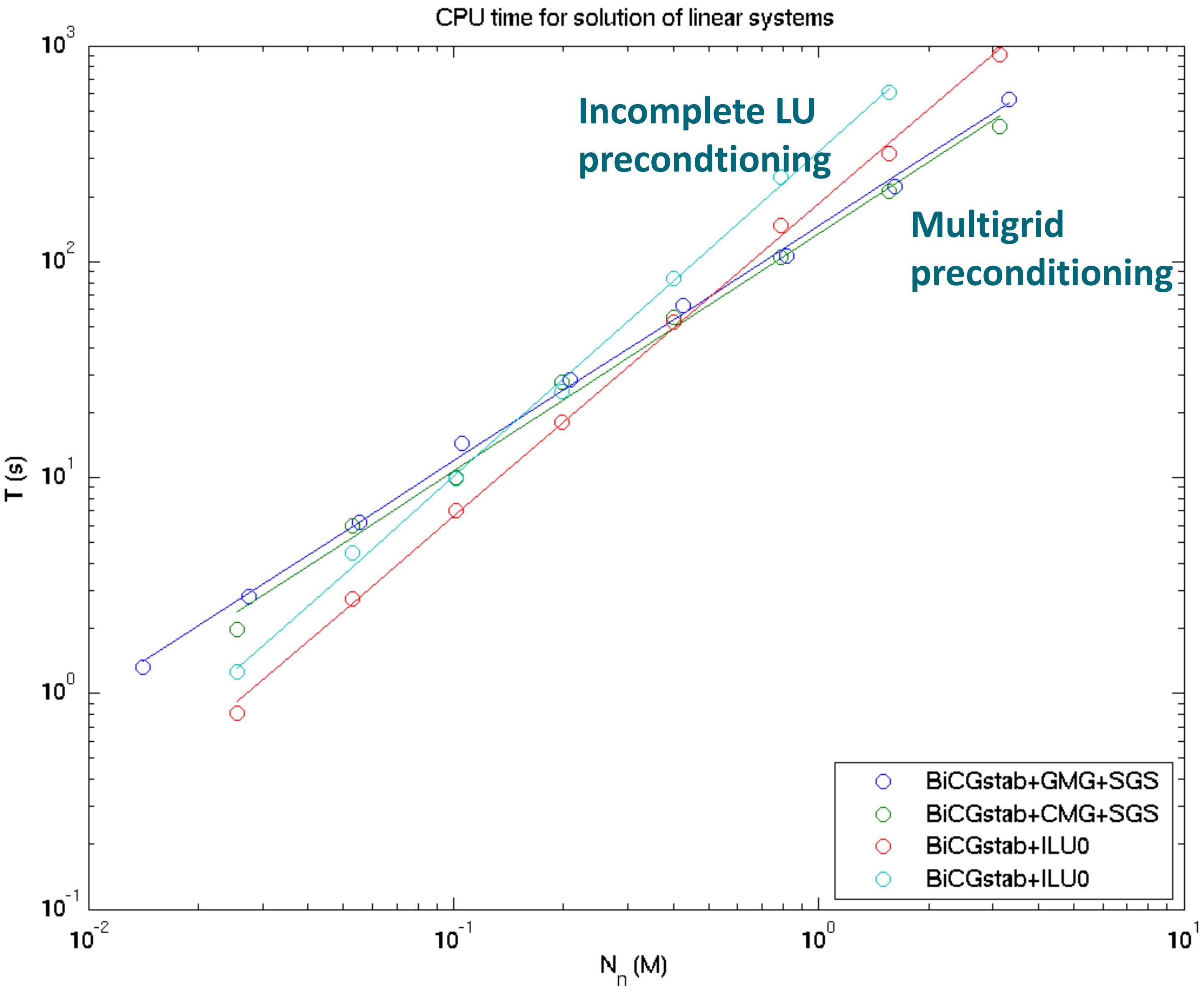


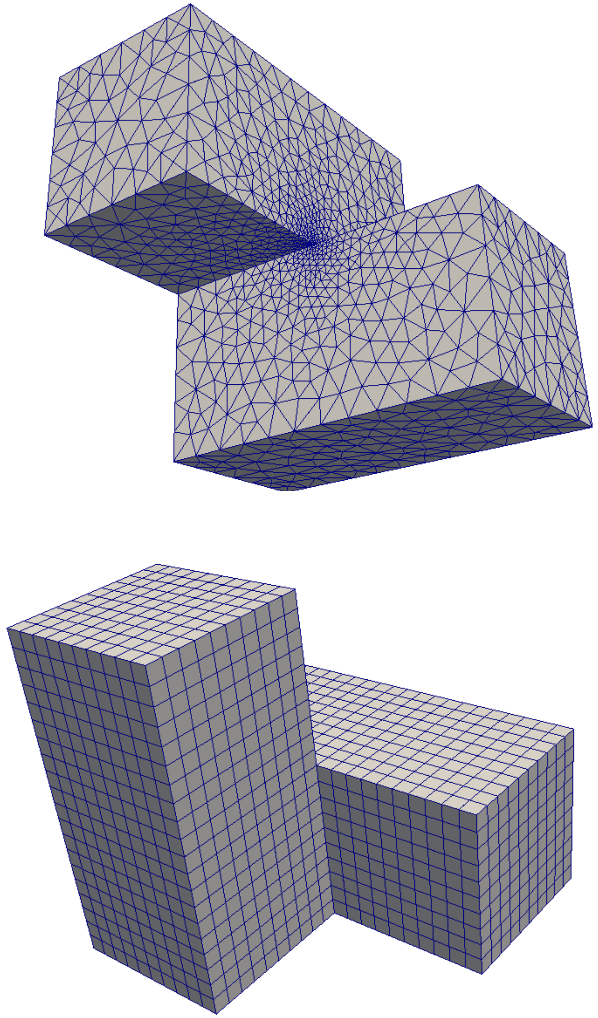





”winkel”

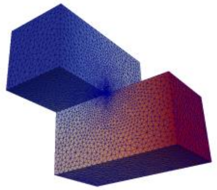
# CPU time for serial solution – one level vs. multilevel



”winkel”

# Serial Poisson equation at "Winkel"



$$t = \alpha n^{\beta}$$

- Success of various iterative methods determined mainly by preconditioning strategy
- Best preconditioner is clustering multigrid method (CMG)
- For simple Poisson almost all preconditioners work reasonable well
- Direct solvers differ significantly in scaling
- For vector valued problems number of possible strategies increases due to various splitting techniques
  - Monolithic vs. segregated methods

| Linear solver | alpha | beta |
|---|---|---|
| BiCGStab+CMG0(SGS1) | 178.30 | 1.09 |
| GCR+CMG0(SGS2) | 180.22 | 1.10 |
| Idrs+CMG0(SGS1) | 175.20 | 1.10 |
| … | | |
| BiCgStab + ILU0 | 192.50 | 1.13 |
| … | | |
| CG + vanka | 282.07 | 1.16 |
| Idrs(4) + vanka | 295.18 | 1.16 |
| … | | |
| CG + diag | 257.98 | 1.17 |
| BiCgStab(4) + diag | 290.11 | 1.19 |
| … | | |
| MUMPS(PosDef) | 4753.99 | 1.77 |
| MUMPS | 12088.74 | 1.93 |
| umfpack | 74098.48 | 2.29 |

# Linear solvers – example in Elmer sif file

```
Linear System Solver = Iterative
Linear System Iterative Method = "GCR" ! BiCGStab, BiCGStabl, GMRes, Idrs, …
Linear System Max Iterations = 500
Linear System Convergence Tolerance = 1.0E-08
Linear System Abort Not Converged = False
Linear System Preconditioning = "ILU0"  ! ILU0, ILU1, ILU2, ILUT
!Linear System ILUt Tolerance = 1.0e-3
Linear System Residual Output = 10
!Idrs Parameter = 4
!BiCGStabl Polynomial Degree = 6

!Linear System Residual Mode = Logical True
!Linear System Robust = Logical True ! Works with GCR and BiCGStabl

! Direct alternative
!Linear System Solver = Direct
Linear System Direct Method = MUMPS ! umfpack
```

# Linear solvers used with Elmer

## We must solve large sparse linear systems: $\boldsymbol{Ax = b}$

### Iterative methods

- Internal Krylov methods (serial & **parallel**)
  - HUTIter library: CG, BiCGStab, BiCGStabl, GMRes, TMQMR, QMR
  - Recent additions: GCR, Idrs, BiCGStabl
- Internal Algebraic multigrid (serial)
  - AMG and CMG methods (not very robust)
- Hypre (**parallel**)
  - Linear solvers
  - Both Krylov methods & BoomerAMG
- Trilinos (**parallel**)
- AmgX (**parallel**)

### Direct methods

- Banded (serial only)
- Umfpack (serial only)
- MUMPS (serial and **parallel**)
- MKL Pardiso (**parallel**, not free)
- ...

# Preconditioning of linear systems

- Instead of solving the original linear system, one may solve the (left) preconditioned system:

$$PAx = Pb$$

  where P is an approximation of the inverse if A
  - ILUn, Incomplete LU depomposition with fill pattern defined by $A^n$
  - Diagonal precondtioner, ***P=1/diag(A)***
  - No strict guidelines on construction, experimental numerics
- ***P*** may also be considered to an operator
  - Multigrid as precondioner
- The goal of this preconditioned system is to reduce the condition number
  - Results to more robust and faster convergence of linear system
- Typically iterative solution: Krylov method + preconditioner
- Parallellization adds the challenge of preconditioning
  - Many preconditioners are not exactly the same in parallel

- Preconditioners in HUTiter
  - ILUn, n=0,1,2,3,…
  - ILUt, specific tolerance
  - Diagonal
  - Vanka
  - AMG and AMG
- Preconditioners in Hypre
  - Parasails
  - BoomerAMG
  - ILUn
  - …

# Parallel computing concepts

## Computer architectures

- 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 (within nodes) approaches

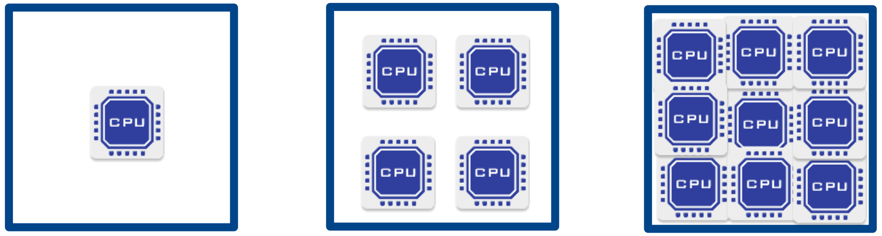

## Programming models

- Threads (pthreads, OpenMP)
  - Can be used only in shared memory computer
  - Limited parallel scalability
  - Simpler or less explicit programming
- Message passing (MPI)
  - Can be used both in distributed and shared memory computers
  - Programming model allows good parallel scalability
  - Programming is quite explicit
- Massively parallel FEM codes use typically MPI as the main parallelization strategy

# Weak vs. strong parallel scaling

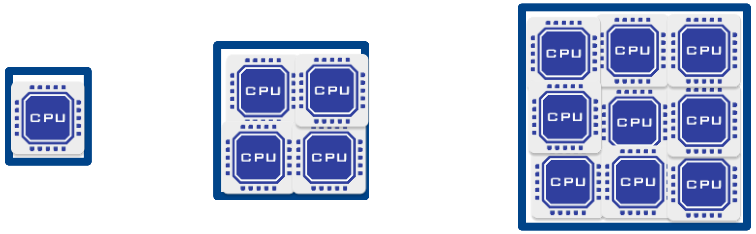
## Strong scaling

- How the solution time ***T*** varies with the number of processors ***P*** for a fixed total problem size.
- Optimal case: ***P x T = const.***
- A bad algorithm may have excellent strong scaling
- Typically $10^4$-$10^5$ dofs needed in FEM for good strong scaling



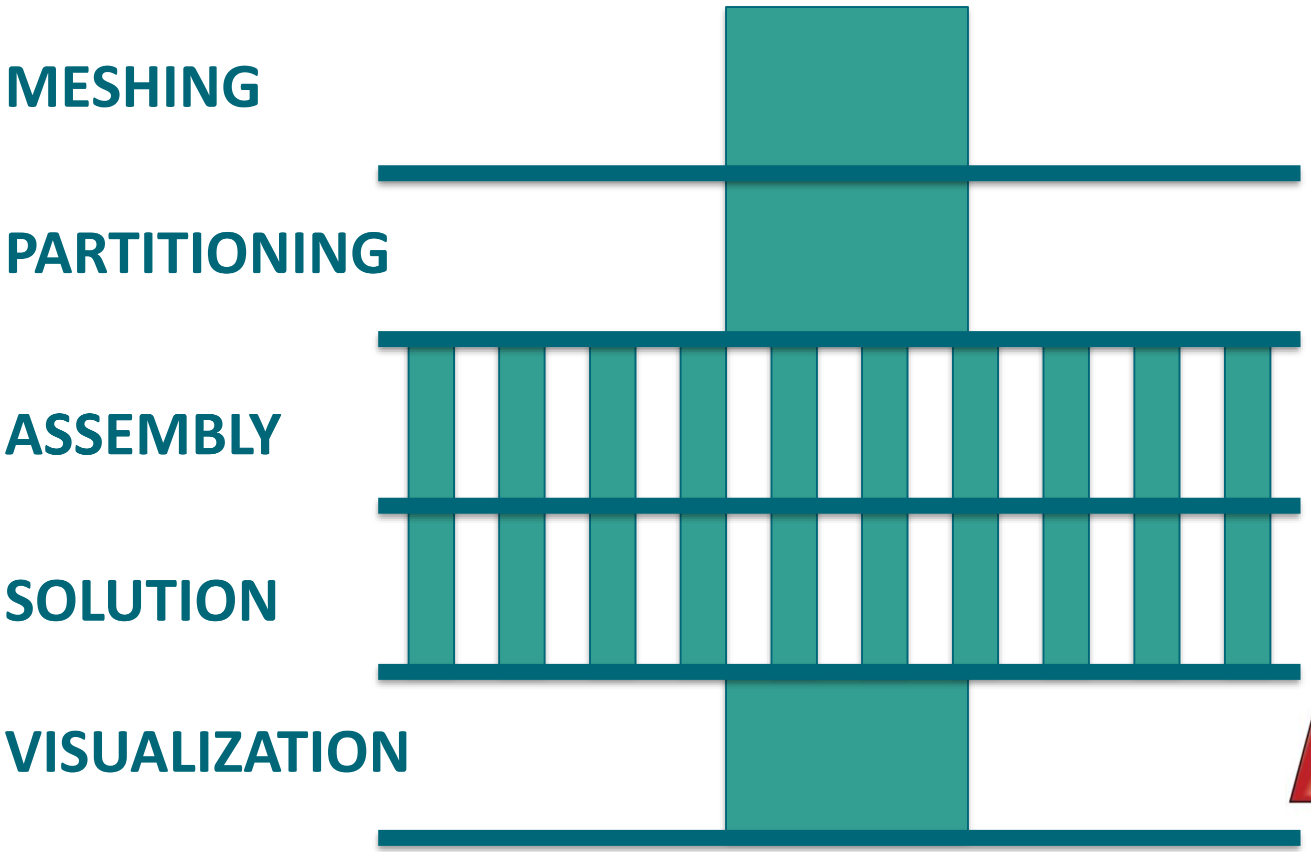
## Weak scaling

- How the solution time ***T*** varies with the number of processors ***P*** for a fixed problem size per processor.
- Optimal case: ***T=const.***
- Weak scaling is limited by algorithmic scaling!



# Basic Parallel workflow (of Elmer)

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




Gmsh
NETGEN
ElmerGrid
METIS
hypre high performance preconditioners
Elmer
MUMPS
ParaView

# Mesh partitioning with ElmerGrid

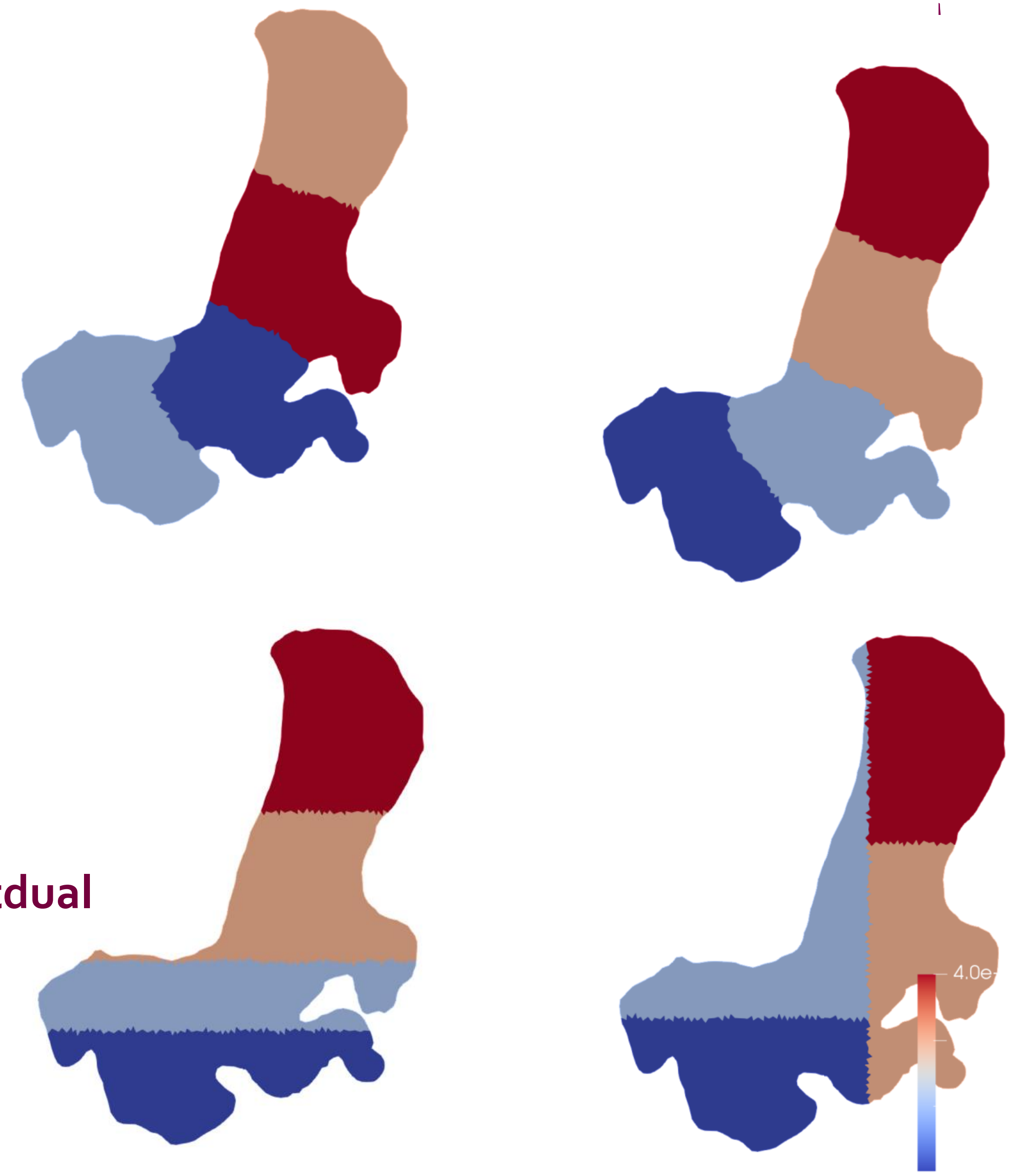
- Two main strategies for mesh partitioning

- **Metis** graph partitioning library:
  **-metiskway #np**
  **-metisrec #np**
  - Generic strategy
  - Call different graph partitioning routines from Metis

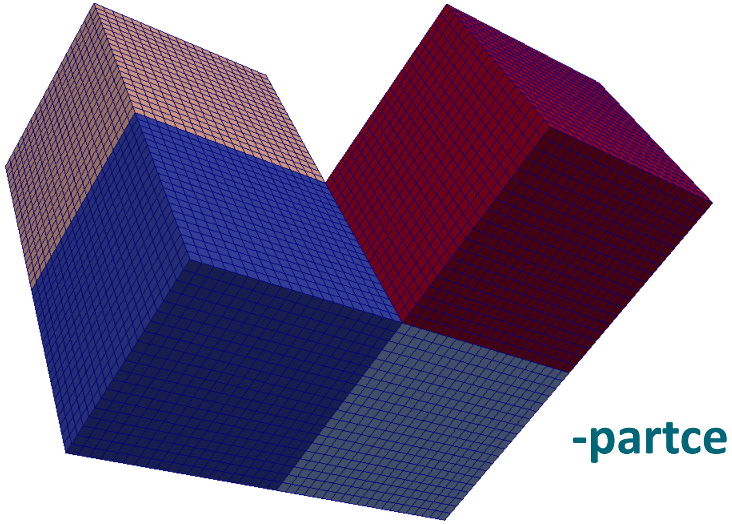
- Division by cartesian directions:
  **-partition nx ny nz**
  **-partcell nx ny nz**
  - Simple shapes (ideal for quads and hexas)

- Partitioning may be done in nodal or dual graph: **-partdual**

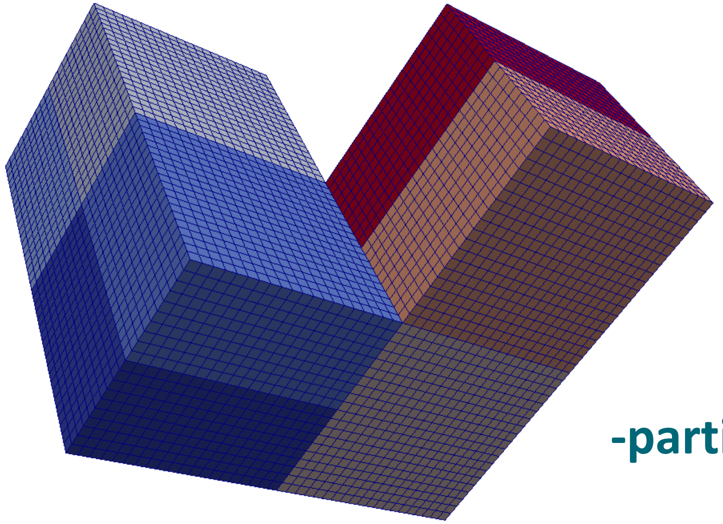
- Optiomal method for partitioning is highly case-dependent
  - Objective is to minimize communication



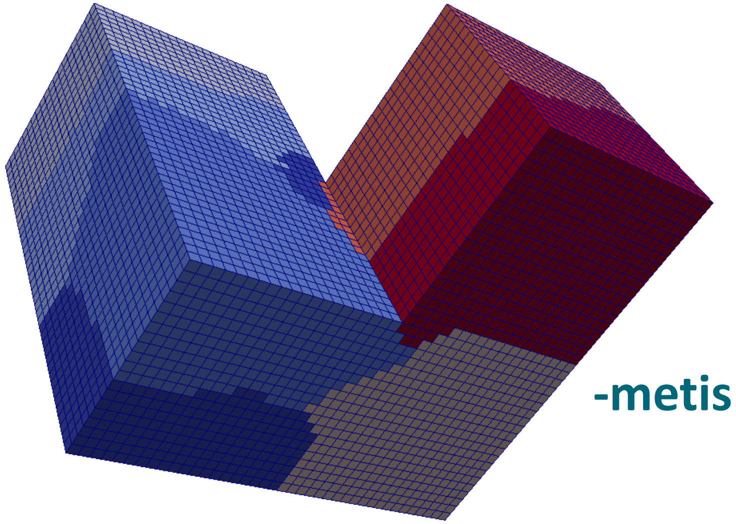
# Mesh partitioning with ElmerGrid – structured mesh



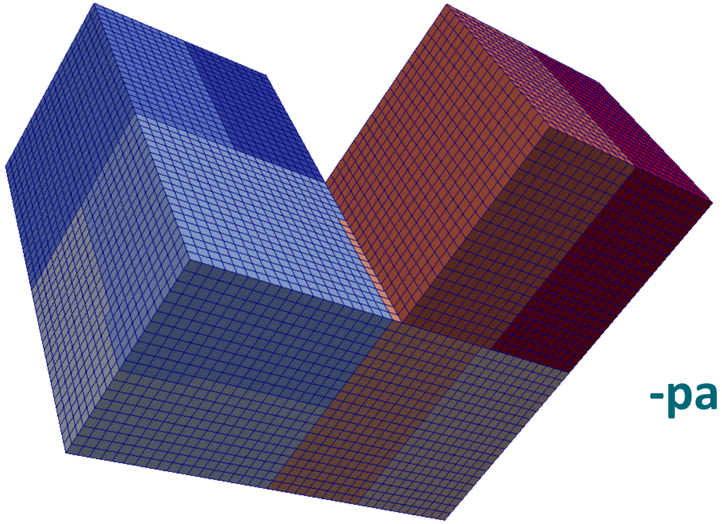
-partcell 2 2 2



-partition 2 2 2

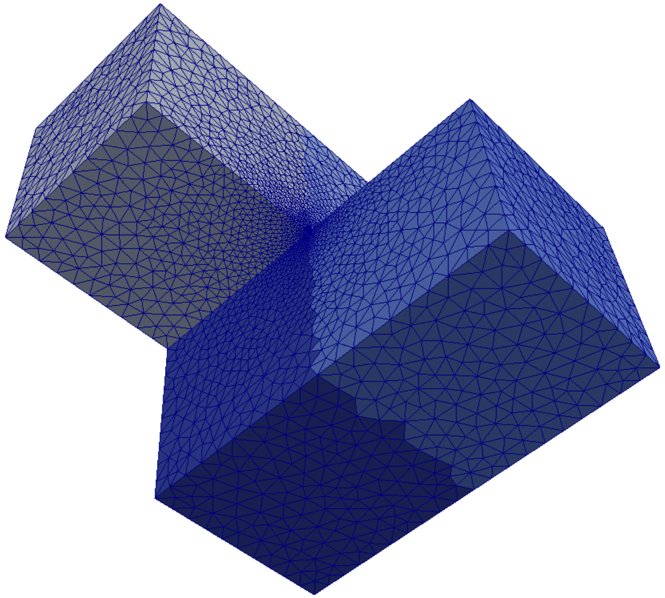


-metis 8

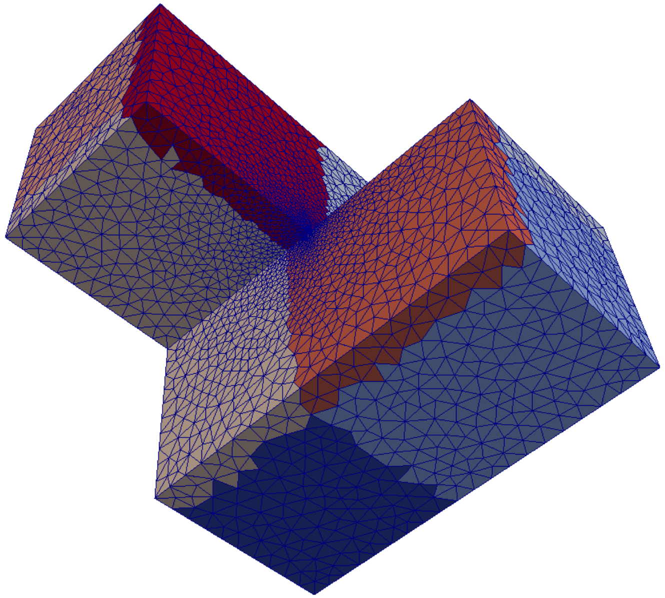


-partdual -metisrec 8

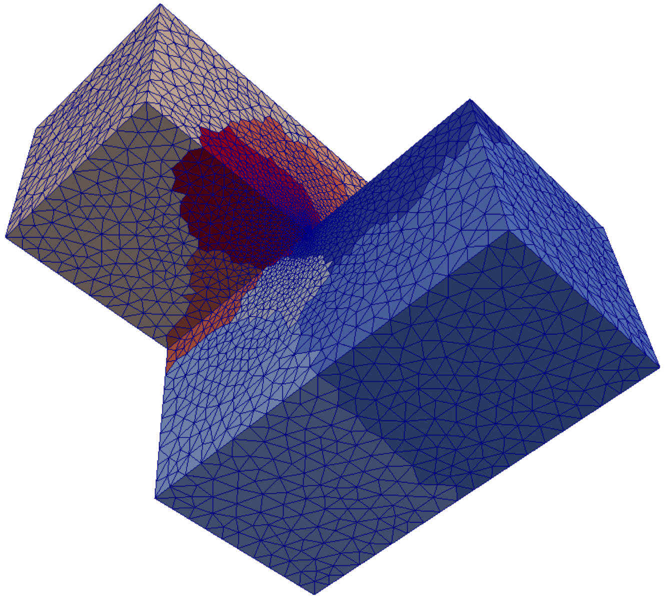
# Mesh partitioning with ElmerGrid – unstructured mesh



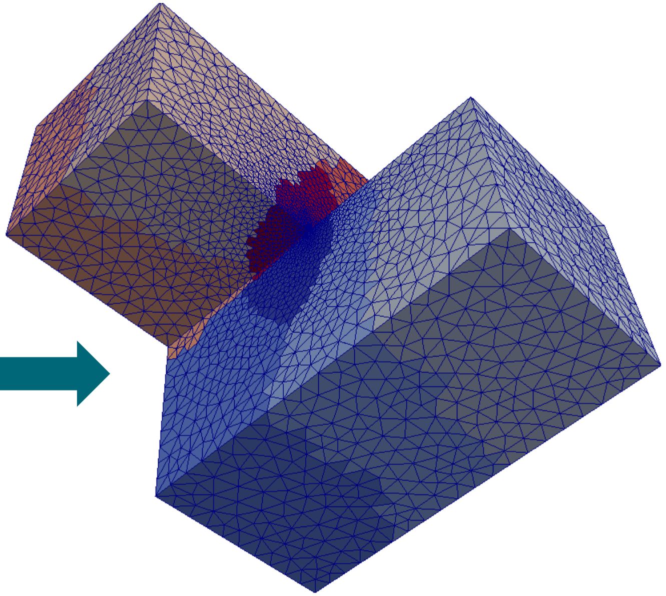
-partcell 2 2 2



-partition 2 2 2



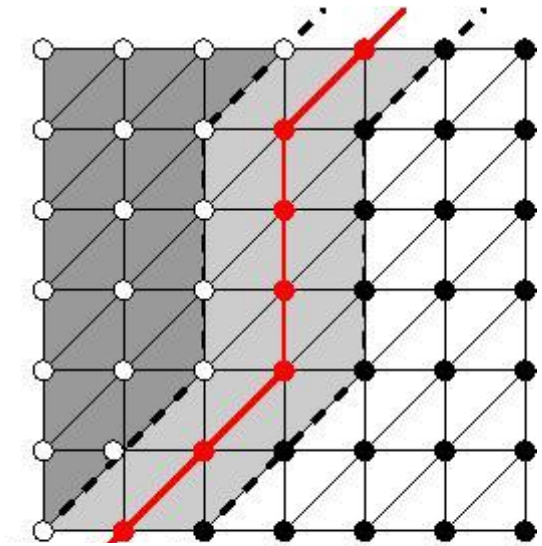
-metisrec 8

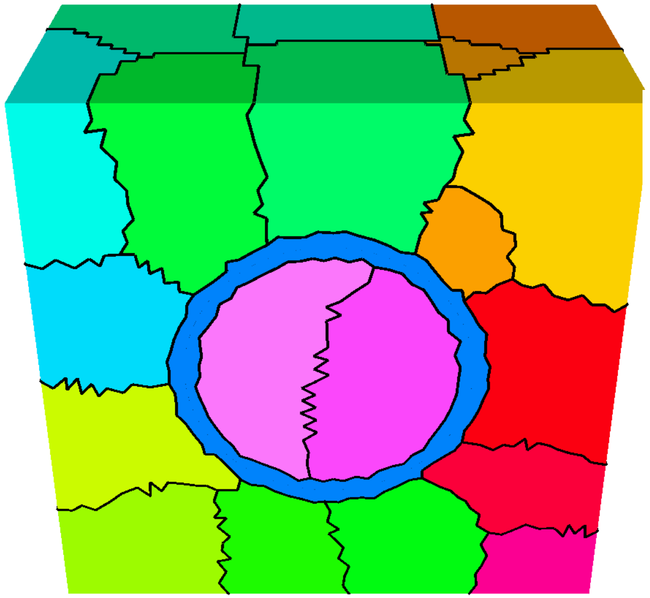


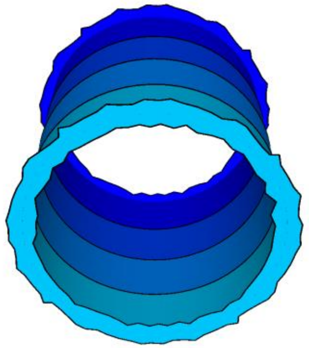
-partdual -metisrec 8

# Accounting for halo elements

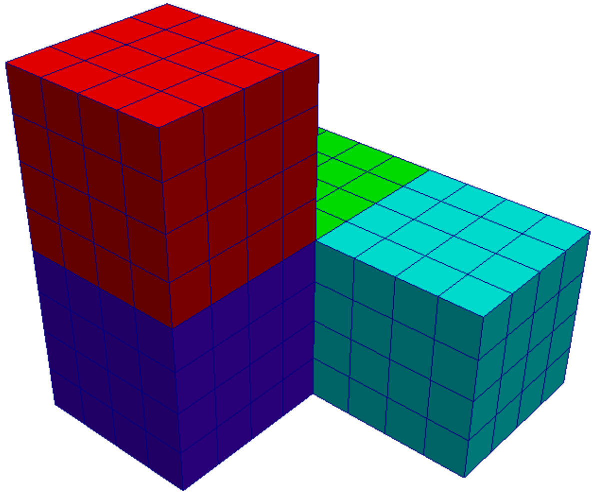
- It is often desirable to share part of the mesh to neighbouring partitions to eliminate communication
  - These are called "halo elements"
- Standard halo provides information on neighbouring element layer: **-halo**
  - Puts "ghost cell" on each side of the partition boundary.
  - e.g. Disconstinuous Galerkin
- Special halos for boundary conditions that are connected in some way
  - e.g. Rotating interfaces

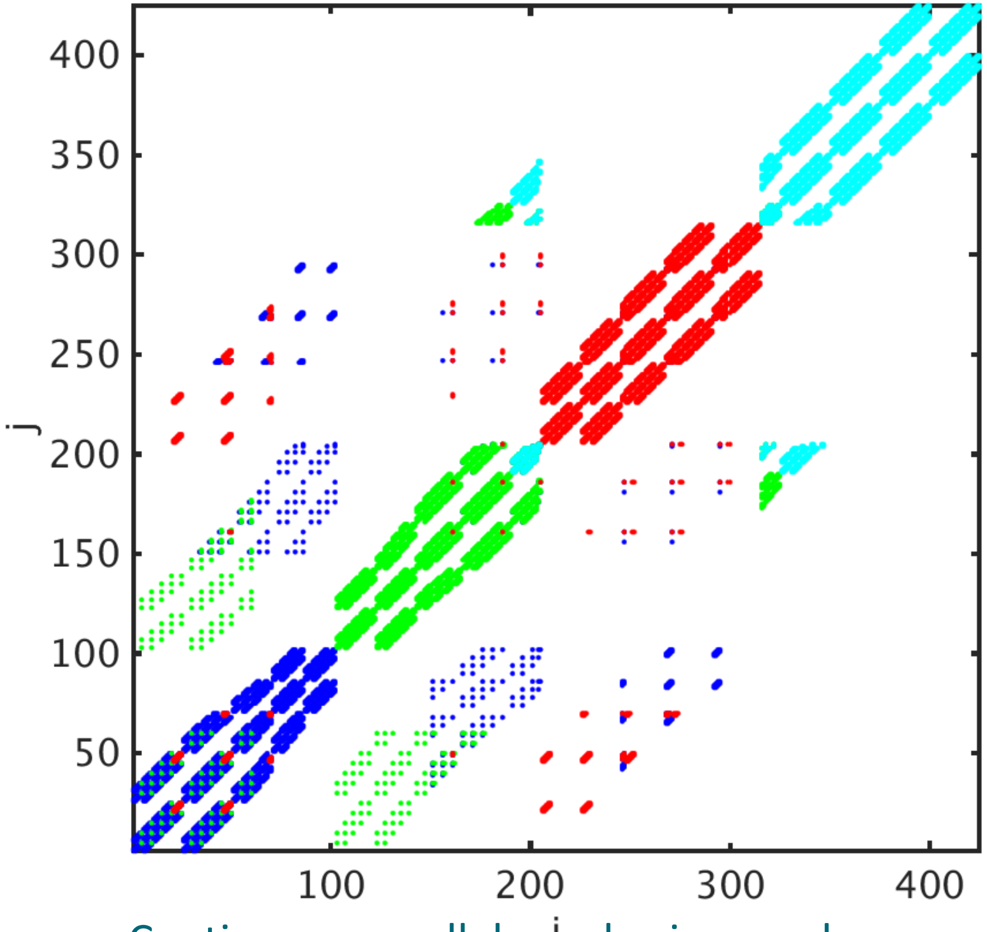






# Partitioning and matrix structure

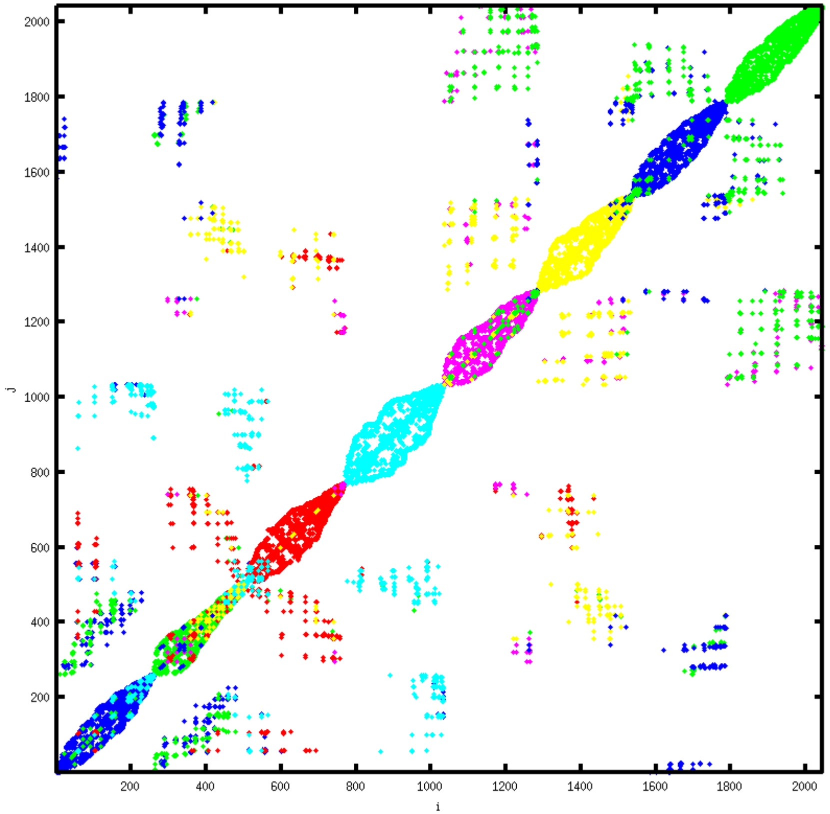


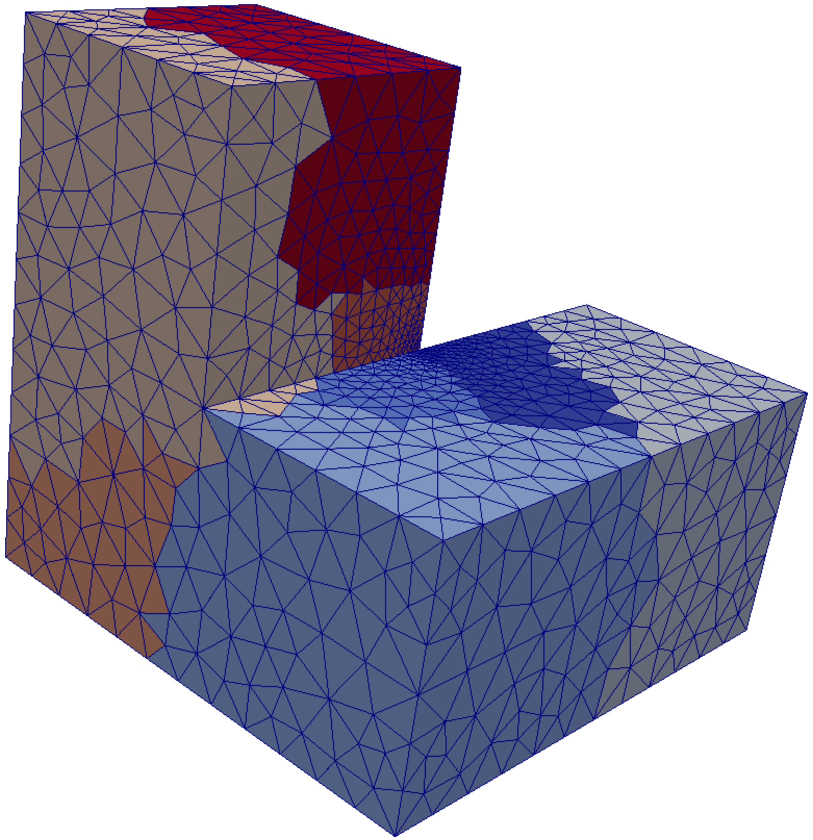


Contiguous parallel numbering used

- Shared nodes result to need for communication.
  - Each dof has just one owner partiotion and we know the neighbours for
  - Owner partition usually handles the full row
  - Results to **point-to-point communication** in MPI
- Matrix structure sets challenges to efficient preconditioners in parallel
  - It is more difficult to implement algorithms  that are sequential in nature, e.g. ILU
  - Krylov methods require just matrix vector product, easy!
- Communication cannot be eliminated. It reflects the local interactions of the underlying PDE

# Partitioning and matrix structure – unstructured mesh



Metis partitioning into 8

- Partitioning should try to minimize communication
- Relative fraction of shared nodes goes as N^(-1/DIM)
- For vector valued and high order problems more communication with same dof count

# 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]

# Serial vs. parallel solution

## Serial

- Serial mesh files
- Execution with
  `ElmerSolver case.sif`
- Writes results to one file: `vtu` files

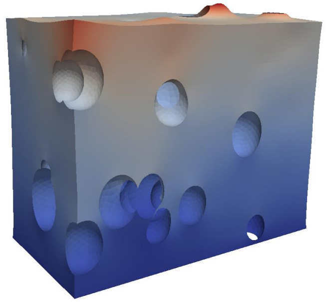
## Parallel

- Partitioned mesh files
- Execution with
  `mpirun -np N ElmerSolver_mpi case.sif`
- Calling convention is platform dependent
- Writes results to *N* `vtu` files + one `pvtu` file

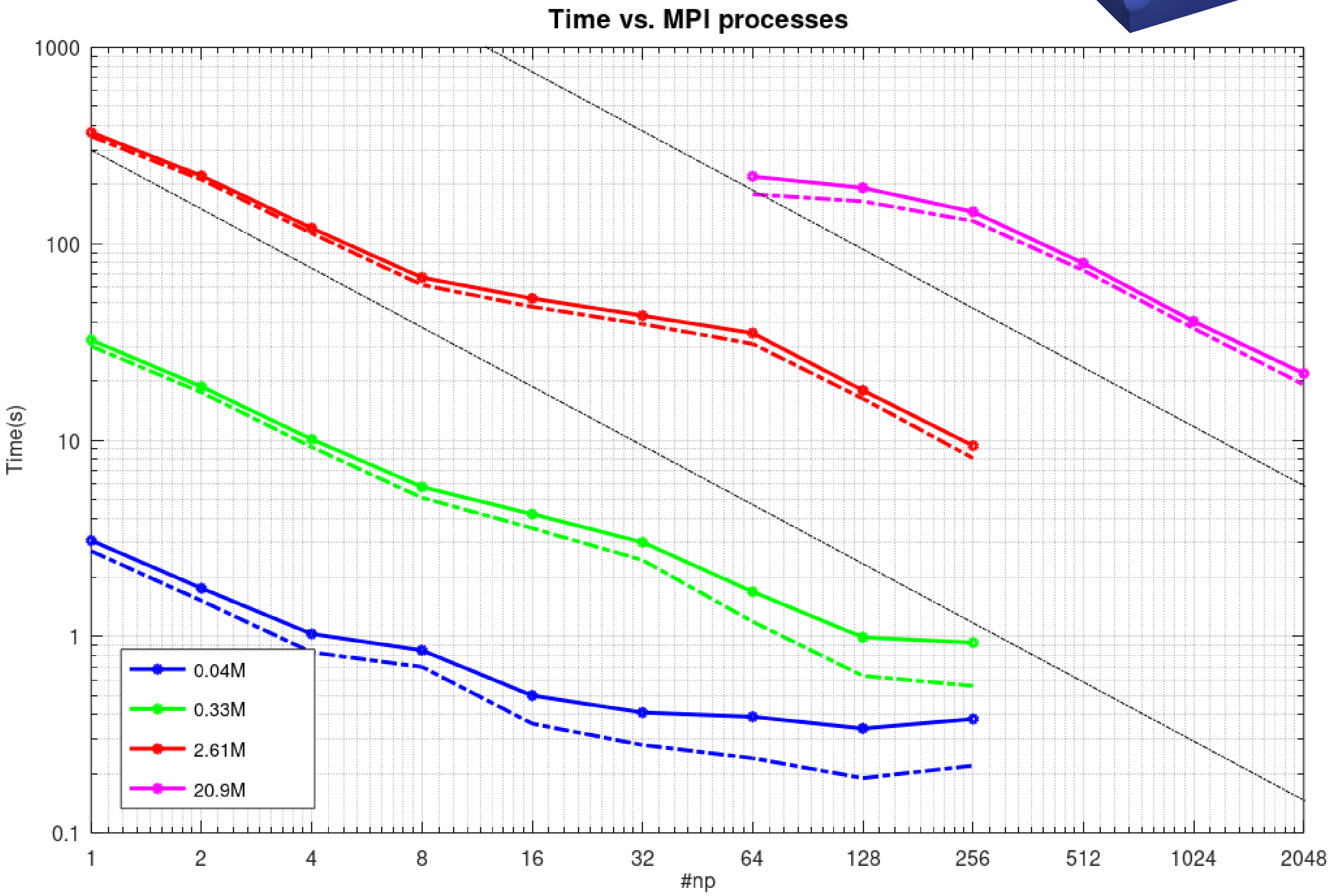
# Parallel postprocessing using Paraview

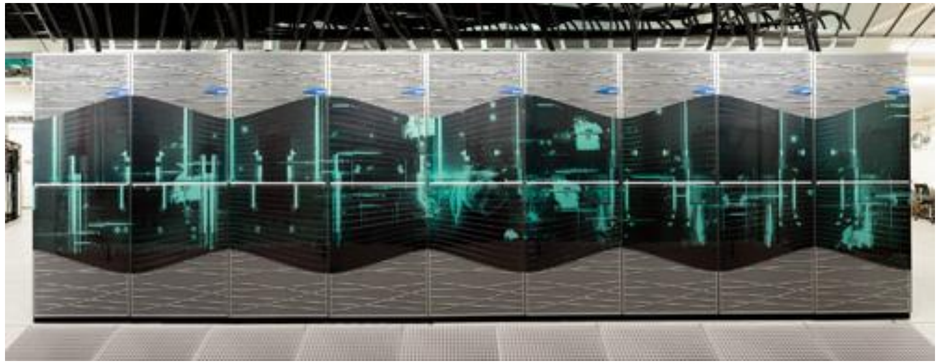
- Use ResultOutputSolver to save data to `.vtu` files
- The operation is almost the same for parallel data as for serial data
- There is a extra file `.pvtu` that holds is a wrapper for the parallel `.vtu` data of each partition

# Scalability of Navier problem on "Cheese"



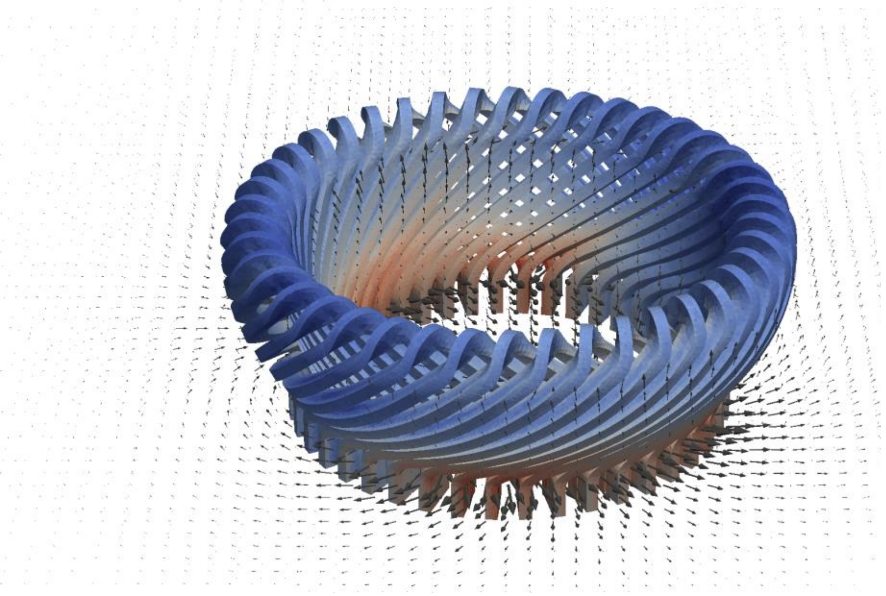
- Good strong scalability is obtained when there are 10's thousands of dofs for each partition
  - Optimal strong scaling depicted by black lines
- Relative fraction of communication is reduced
- Weak scalability may still be bad
- Complicated memory hierarchy and I/O makes it difficult to estimate the speed



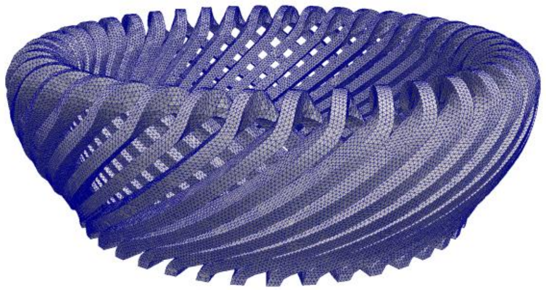



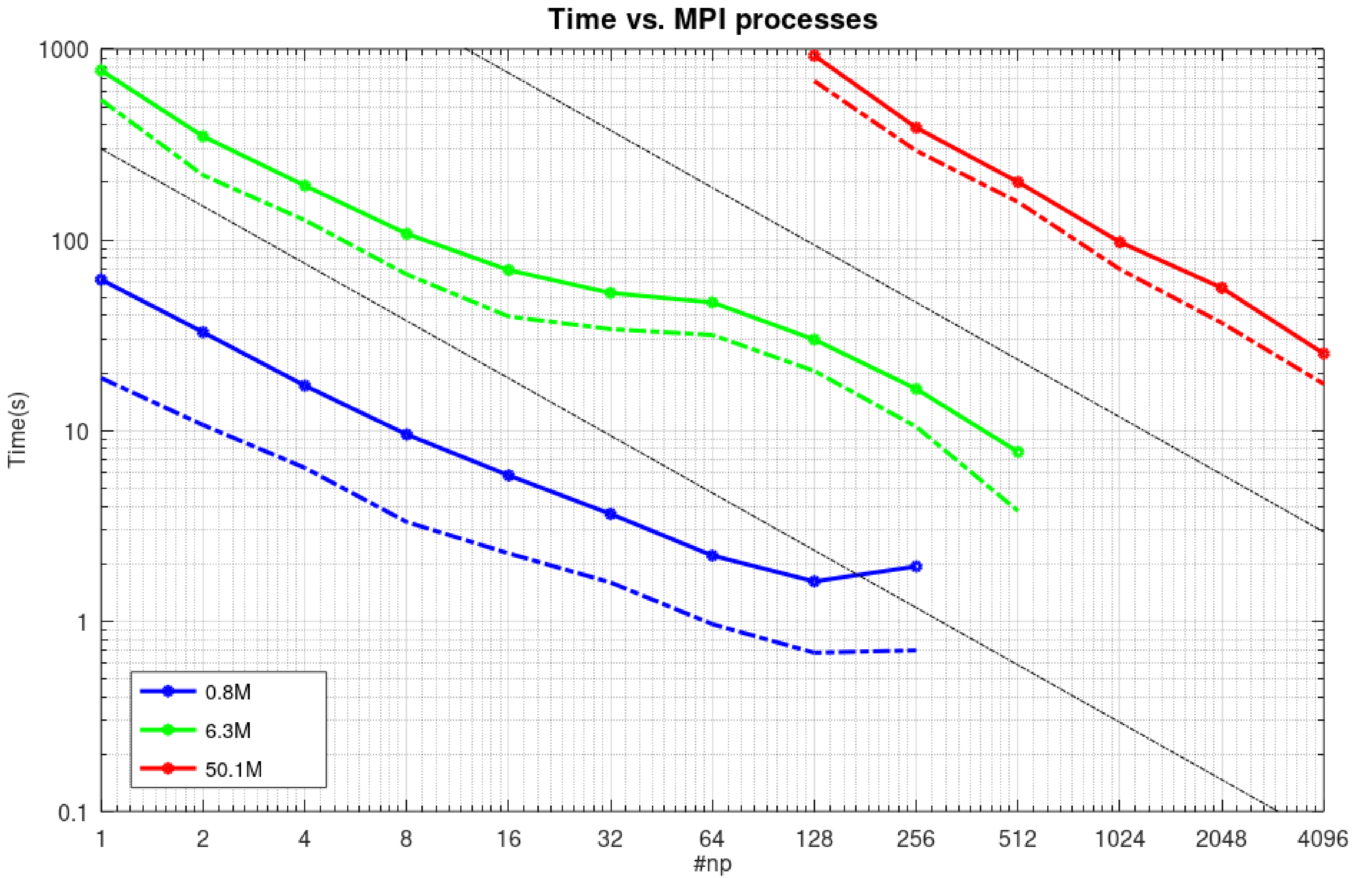
Test case for nodal Navier equation using CG+ILU0 on mahti.csc.fi

# Scalability of edge element AV solver for end-windings



*Magnetic field strength (up) and computational mesh (down) of an electrical engine end-windings. Meshing M. Lyly. Simulation on Mahti, J. Ruokolainen, CSC, 2021.*

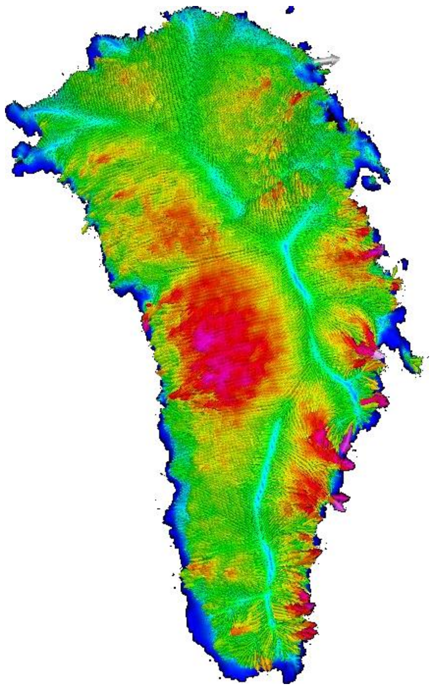


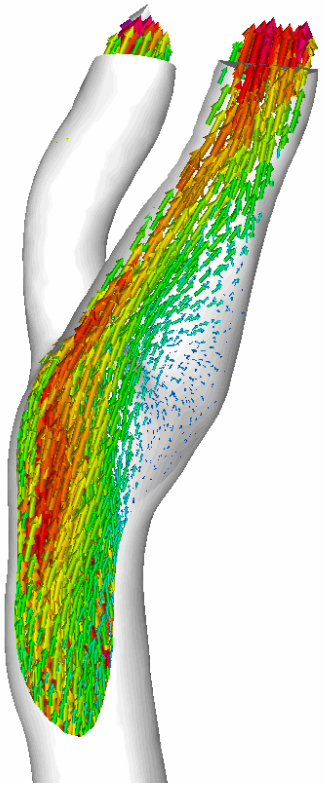


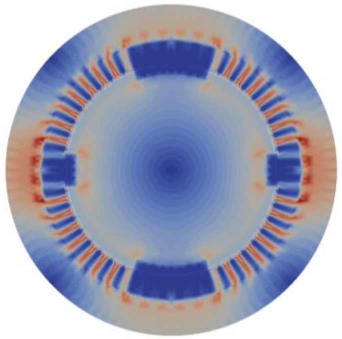

Test case for magnetostatics using BiCGStabl+none

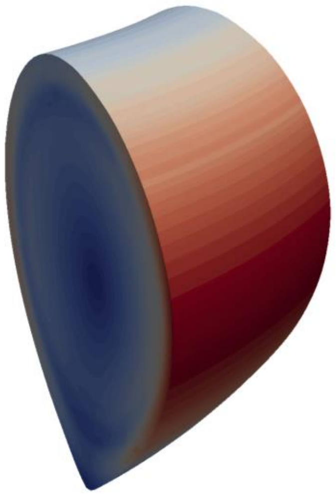
# Challenge of real-world problems

- Linear solver libraries work great for many standard problems
  - Scalability demonstrated up to 1000's of cores
- Unfortunately many of the real world cases are
  - Unsymmetric
  - Constrained
  - Compromized in mesh quality (aspect ratio)
  - Etc.
- Often the target number of cores is often rather modest
  - 100's of cores
  - But direct solvers are still too slow or memory intensive
- We look on strategies that split the complex problems into more simple ones where standard libraries excel
  => block preconditioning









# Block preconditioning

- 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, acoustics,…
  - Strongly coupled multiphysics problems
- Preconditioner need not to be just a matrix, it can be a procedure!
- **Idea**: Use as preconditioner a procedure where the components are solved one-by-one 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
  - Multilevel methods

# Block precontioning

- Given a block system

$$\begin{bmatrix} \mathbf{K}_{11} & \cdots & \mathbf{K}_{1N} \\ & \cdots & \\ \mathbf{K}_{N1} & \cdots & \mathbf{K}_{NN} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}$$

- Block Gauss-Seidel

$$\mathrm{P} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{K}_{21} & \mathbf{K}_{22} & \mathbf{0} & \cdots \\ \cdots & & & \end{bmatrix}$$

Block Jacobi

$$\mathrm{P} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{K}_{22} & \mathbf{0} & \cdots \\ \cdots & & & \end{bmatrix}$$

- Preconditioner is the operator which produces the new search direction $s^{(k)}$
- Use GCR to minimize the residual $||\mathbf{b} - \mathbf{K}\mathbf{x}^{(k)}||$ over the space $\mathcal{V}_k = \mathbf{x}^{(0)} + \mathrm{span}\{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(k)}\}$

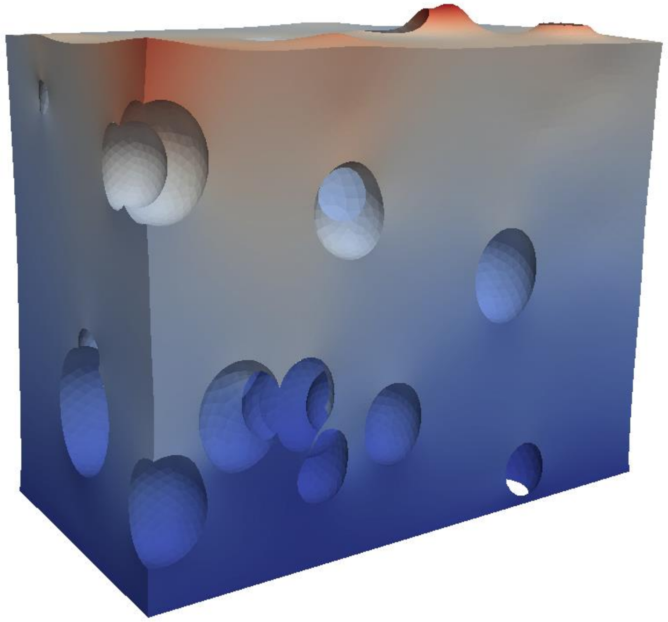
# Motivation for using block preconditioner

- Comparison of algorithm **scaling** in linear elasticity between different preconditioners
  - ILU1 vs. block preconditioning (Gauss-Seidel) with agglomeration multigrid for each component
- 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 |

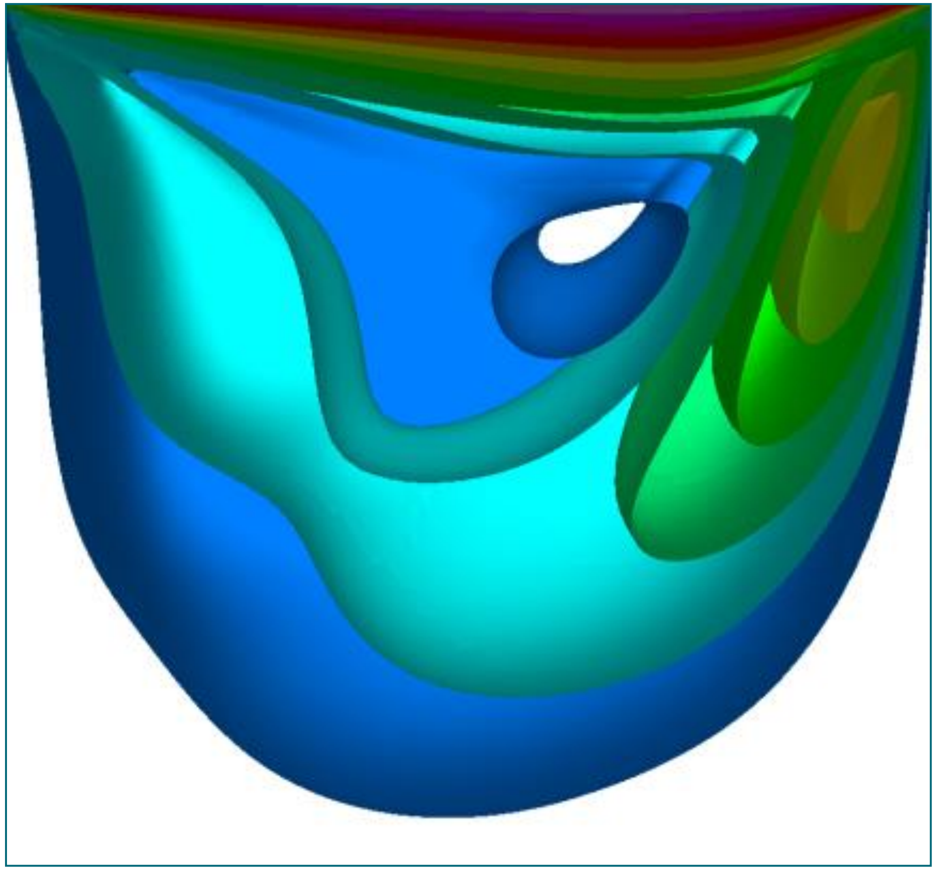
***Simulation Peter Råback, CSC.***

* No convergence was obtained



# Block preconditioner: 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 |

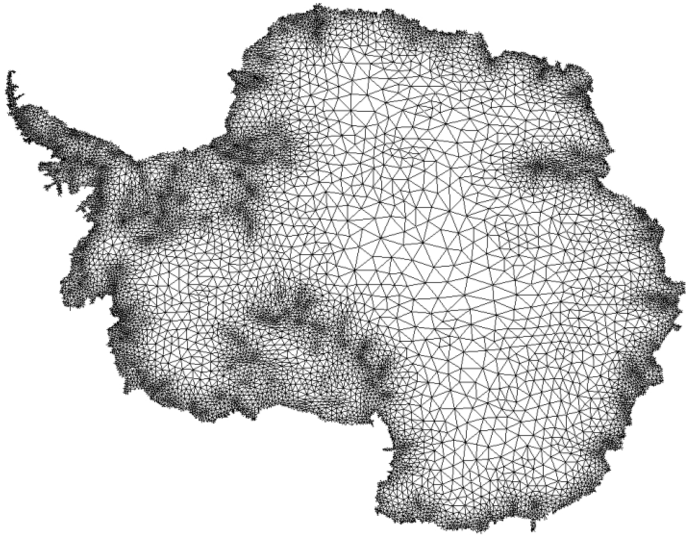


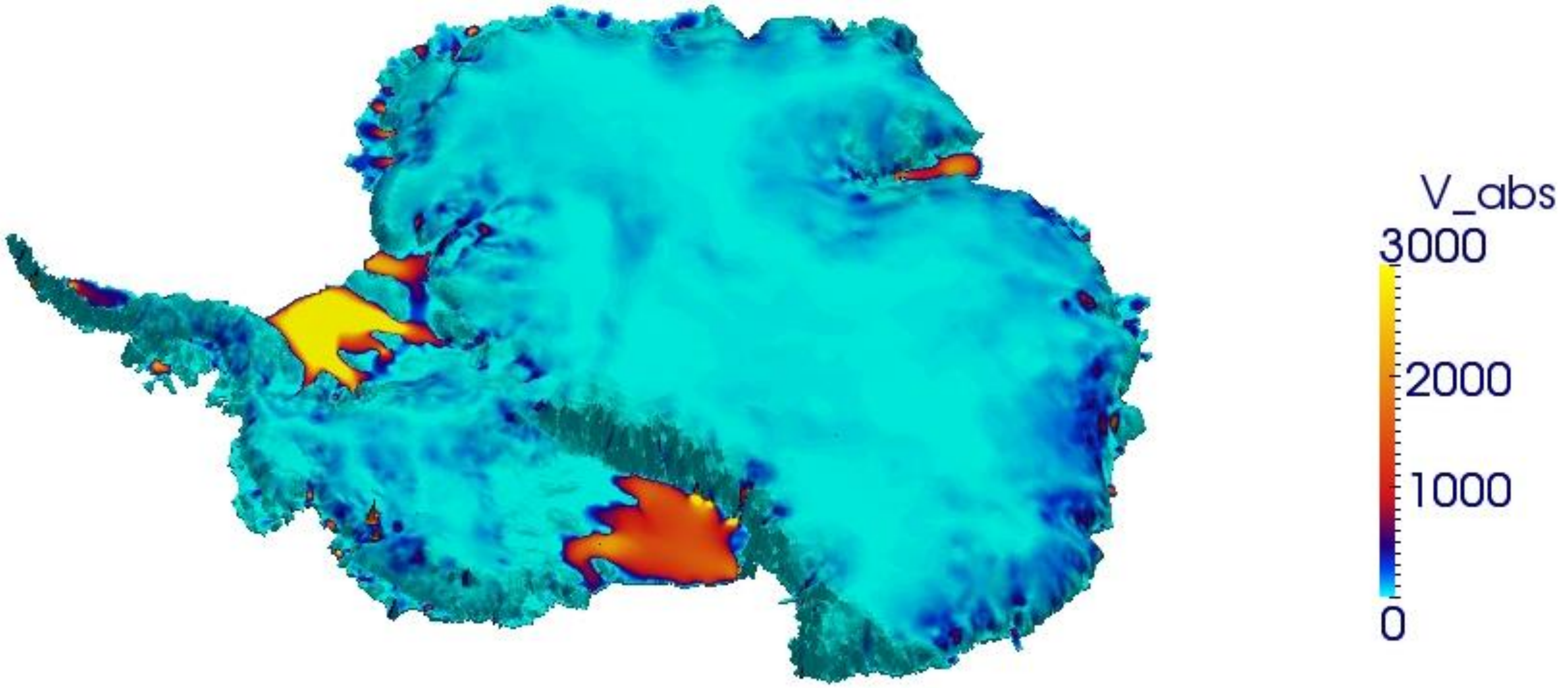
*Velocity solves with Hypre: CG + BoomerAMG preconditioner for the 3D driven-cavity case (Re=100) on Cray XC (Sisu).*
*Simulation Mika Malinen, CSC, 2013.*

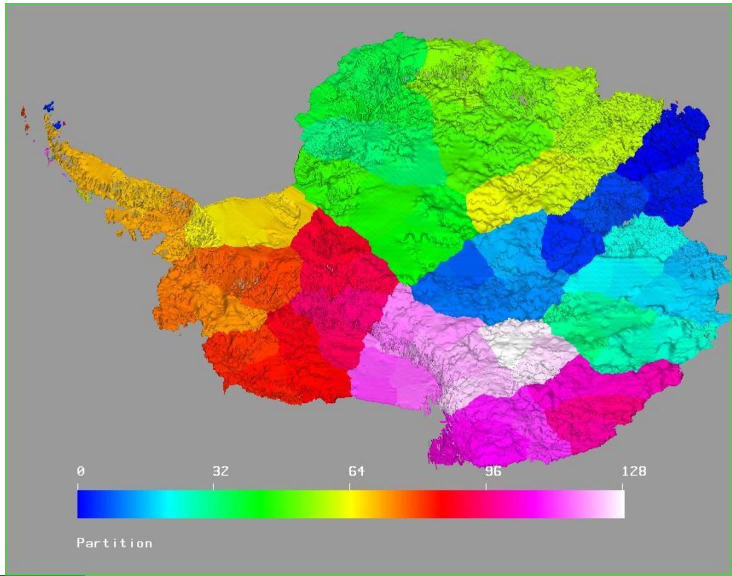
***O(~1.14)***

# Block preconditioner for computational glaciology

- New generation **IncompressibleNSVec** uses many best practices
- Also block preconditioner applied for robust and speedy execution

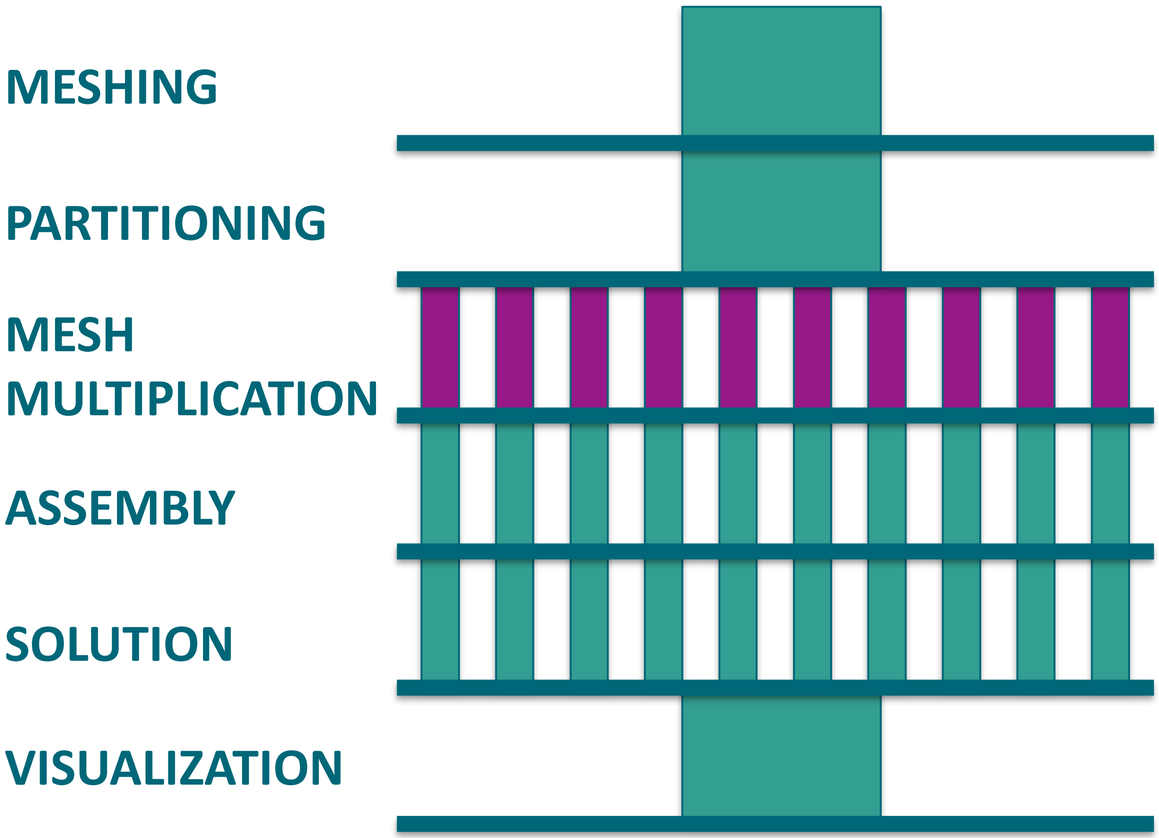




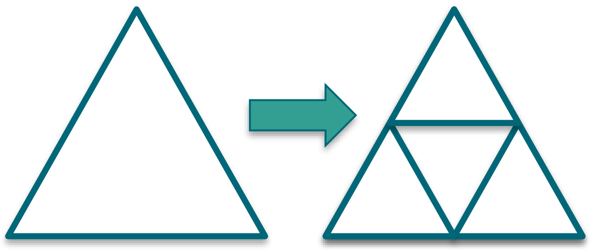
# Parallel workflow in Elmer II

- Large meshes may be finilized at the parallel level

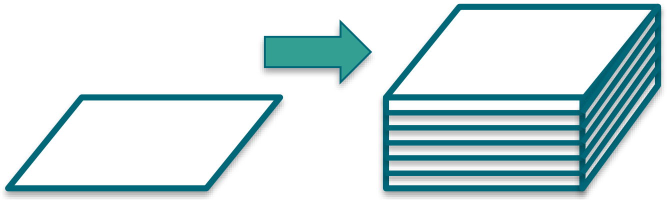


# Finalizing the mesh in parallel level

- 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



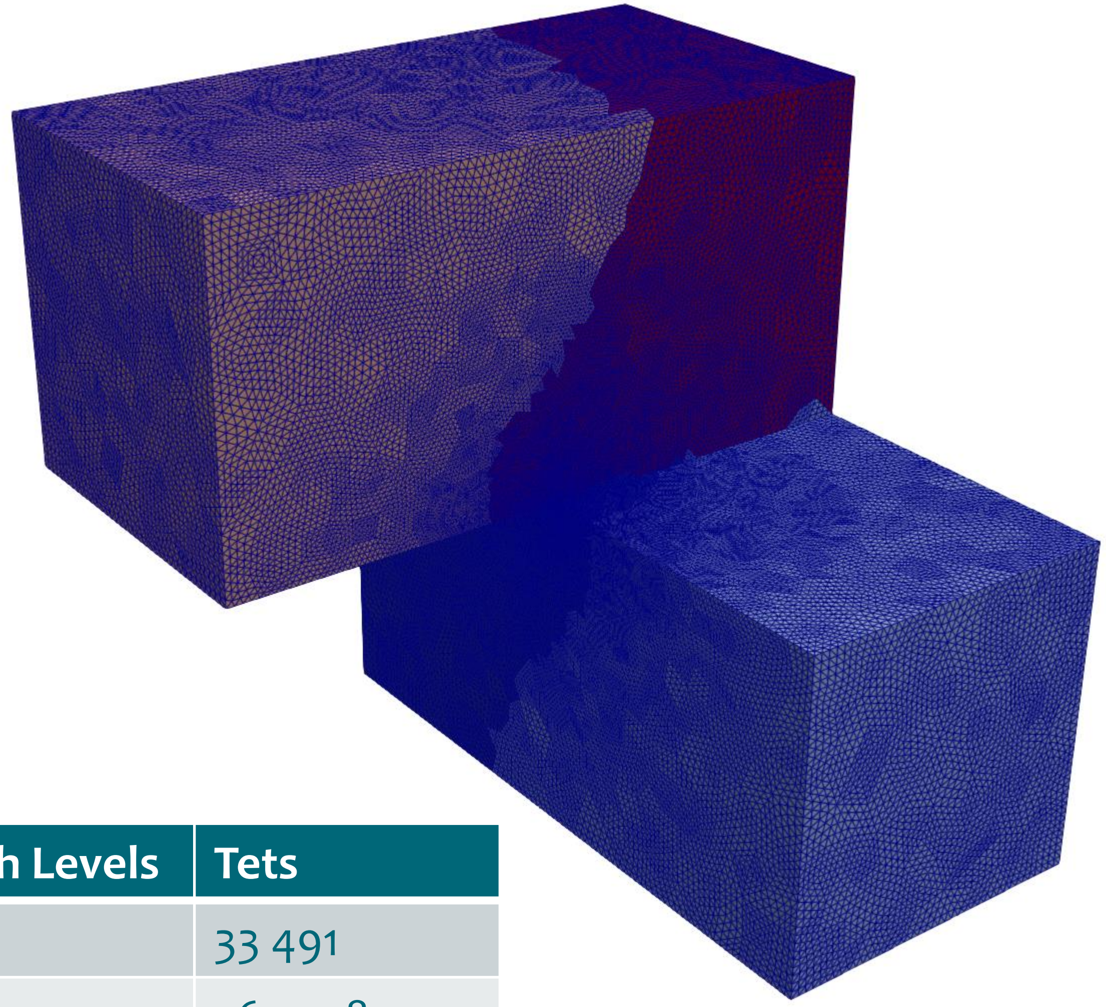
- Increase of element order (p-elements)
  - There is also a p-multigrid in Elmer



- Extrusion of 2D layer into 3D for special cases
  - Example: Large Ice-sheets
- For complex geometries this is often not an option
  - Optimal mesh grading difficult to maintain
  - Geometric accuracy cannot be increased

# Internal Mesh Multiplication

- Each edge is split into half
  - 8-fold number of elements in 3D
  - 4-fold number of elements in 2D
- Eliminates I/O bottle-neck
- Extremely fast compared to other steps
- Works on serial and parallel level

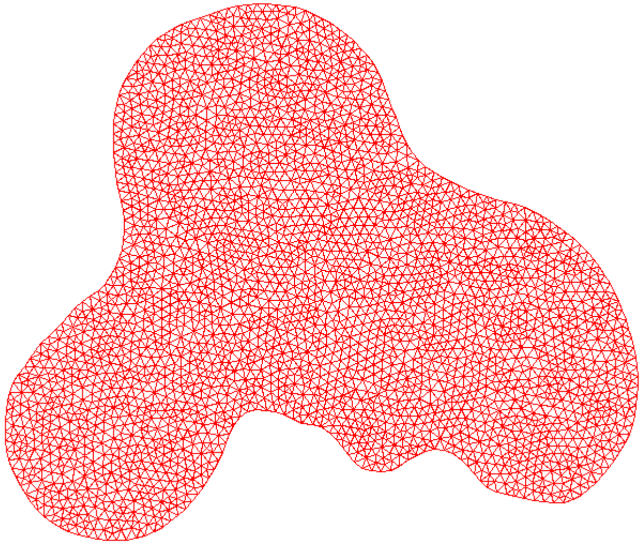


```
Simulation
  …
  Mesh Levels = 2
  Mesh Keep = 1
!  Mesh Grading Power = 3
!  Mesh Keep Grading = True
End
```

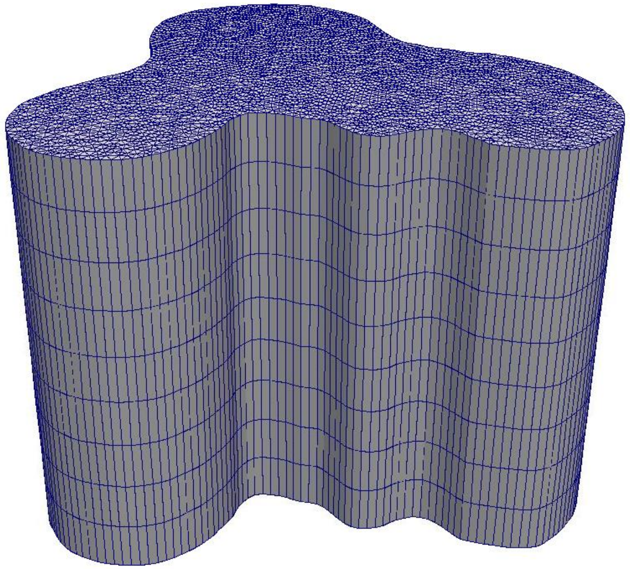
| Mesh Levels | Tets |
|---|---|
| 1 | 33 491 |
| 2 | 267 928 |
| 3 | 2 143 424 |

# Internal mesh extrusion

- Start from 2D and exrude to 3rd dimension
- Extremely fast, serial & parallel



**2D mesh by Gmsh**



**3D internally extruded mesh**

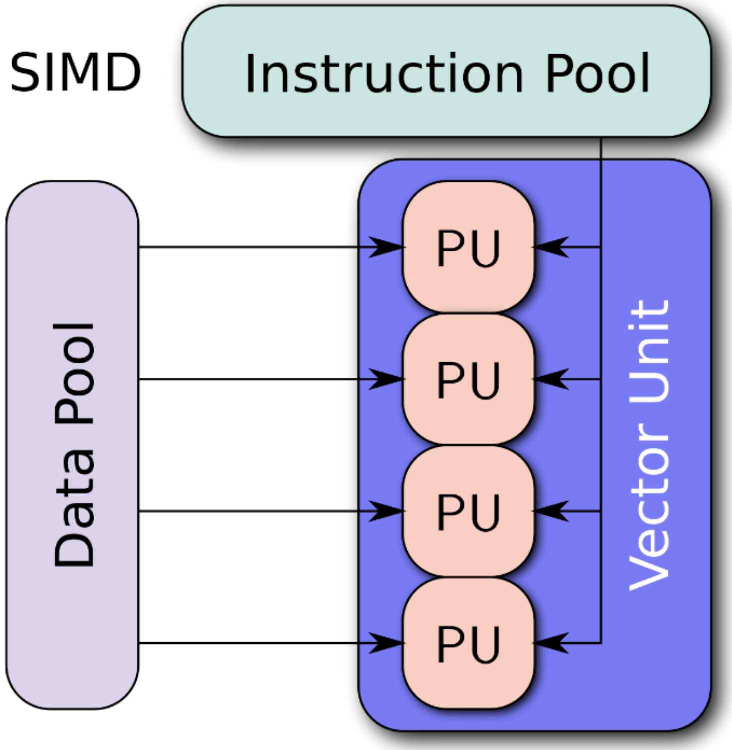
**Design Alvar Aalto, 1936**

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

# Elmer and vectorization

- New computer architectures use **SIMD** (=vector) units to do fast computations
- If you (on an Intel chip) don't utilize this, you a priori loose ¾ of your performance
- FEM: **assembly** = creating the matrix

  solution = solving it
- Until recently, assembly procedures in Elmer did not utilize SIMD
- Some new solvers do:
  - NSIncompressibleVec
  - StatCurrentSolveVec
- Gains depend on the number of integration points



# Elmer and threading

- The number of threads in CPUs keep increasing but the clock speed has stagnated
- Significant effort has been invested for the hybrization of Elmer
  - In practice means **OpenMP** pragmas in the code
  - Assembly process has been multithreaded and vectorized
  - "Coloring" of element to avoid race conditions
- Speed-up of assembly for typical elements varies between 2 to 8.
- As an accompanion the multitreaded assembly requires multithreaded linear solvers.
- New generation solvers ideally both vectorized and threaded!

| Multicore speedup, P=2<br>128 threads on KNL, 24 threads on HSW | | | | |
|---|---|---|---|---|
| Element (#ndofs, #quadrature points) | Speedup | | Optimized local matrix evaluations / s | |
| | KNL | HSW | KNL | HSW |
| Line (3, 4) | 0.7 | 2.0 | 4.2 M | 14.5 M |
| Triangle (6, 16) | 2.5 | 3.9 | 2.6 M | 6.5 M |
| Quadrilateral (8, 16) | 2.8 | 4.0 | 2.6 M | 6.6 M |
| Tetrahedron (10, 64) | 7.9 | 6.3 | 1.0 M | 1.5 M |
| Prism (15, 64) | 8.3 | 5.8 | 0.8 M | 0.9 M |
| Hexahedron (20, 64) | 7.2 | 5.8 | 0.6 M | 0.9 M |

**Speed-up assembly process for poisson equation using 2nd order p-elements. Juhani Kataja, CSC, IXPUG Annual Spring Conference 2017.**

# Elmer on GPUs

- It is diffucult to move a large legacy code onto GPUs
- Easiest to start from critical sections that are offloaded to the GPU
  - Solution of linear systems
- Interface to AMGX added recently
  - Serial interface
  - Multi tasking (MPI) and multi-GPU interface
  - Found well working setup for elasticity (Navier) problem (CG+AMG)
- Ideal for problems where matrix stays the same between calls and r.h.s. changes

- Performance sweet-spot: 4GPU+20 MPI ranks
- Speed-up vs only using the node's CPUs: 16/12 = 1.3X

| MPI/GPU | 5 | 10 | 20 | 40 |
|---|---|---|---|---|
| 1 | Out of memory | 11/25s | 27/35s | AMGX exception |
| 2 | 8/30s | 13/25s | 14/22s | 46/50s |
| 4 | 4/26s | 5/16s | 6s/12s | 15/20s |

T. Zwinger, J. Ruokolainen & G. Gadeschi, 2020

# Recipes for resolving scalability bottle-necks

- Use algorithms that scale well when possible
  - E.g. Multigrid methods
  - Tailored preconditioners
- If the initial problem is difficult to solve effectively divide it into simpler sub-problems
  - One component at a time -> block preconditioners
  - One domain at a time -> FETI
  - Splitting schemes (e.g. Pressure correction in CFD)
- Finalize mesh on a parallel level (minimize I/O)
  - Mesh multiplication or parallel mesh generation
- Analyze results on-the-fly and reduce the amount of data for visualization
- Take use on new developments in architecture

# Most important Elmer resources

- http://www.csc.fi/elmer
  - Official Homepage of Elmer at CSC
- http://www.elmerfem.org
  - Discussion forum, wiki, elmerice community
- https://github.com/elmercsc/elmerfem
  - GIT version control
- http://youtube.com/elmerfem
  - Youtube channel for Elmer animations
- http://www.nic.funet.fi/pub/sci/physics/elmer/
  - Download repository
- Further information: elmeradm@csc.fi

Thank you for your attention!