



Parallel workflows in computational engineering with open source software

D.Sc. Peter Råback
CSC – IT Center for Science

PATC course on parallel workflows
Stockholm, 4-6.12.2013

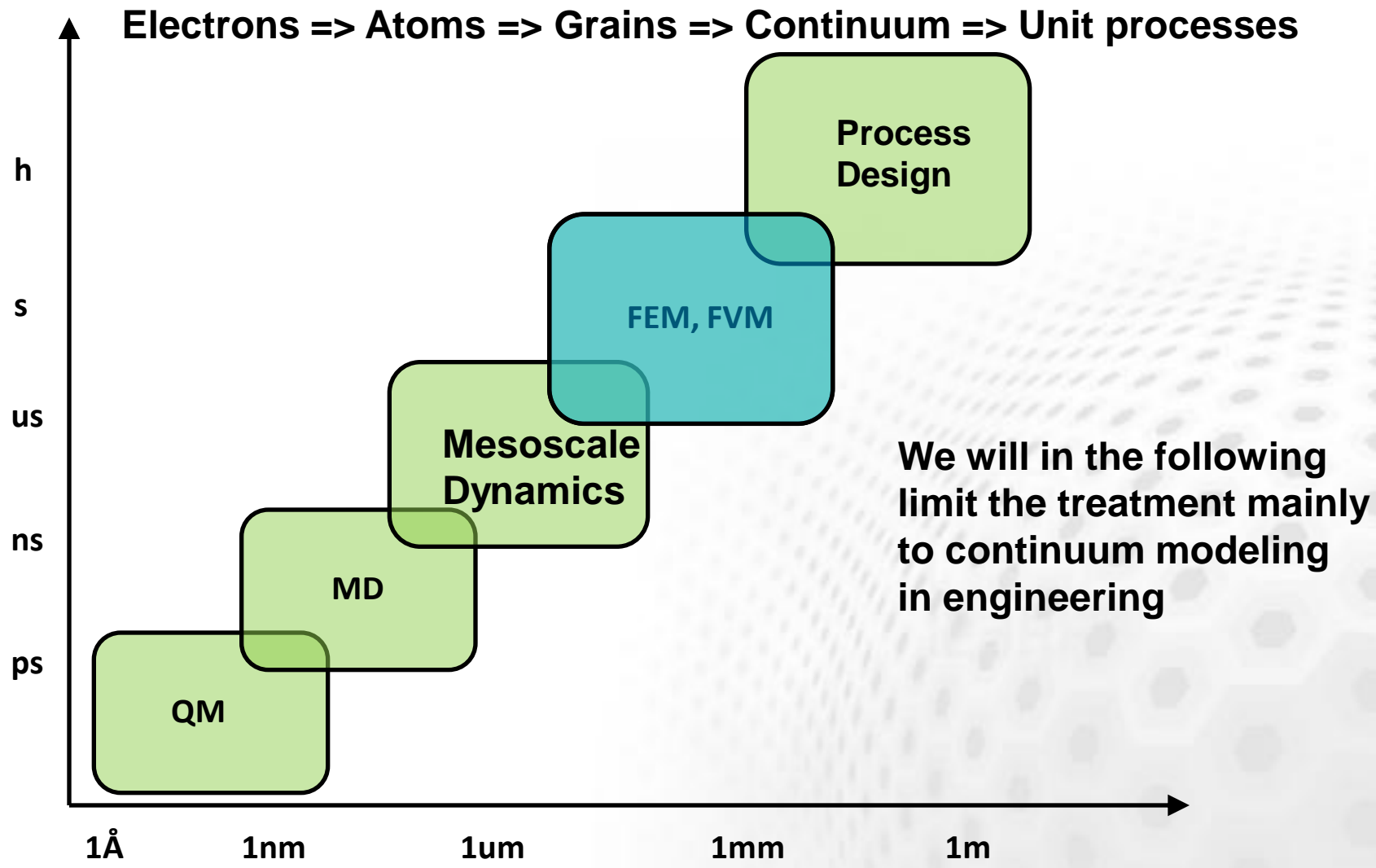
Outline of the presentation



- Computational engineering
- Open Source software
- Parallel workflows



Hierarchy of computational models (in material science)



Computational Engineering

- Mainly based on classical physics
 - Continuum mechanics (fluids & solids)
 - Maxwell's equations for electromagnetic fields
 - Statistical physics and thermodynamics (with chemical reactions)
- These models may be expressed by partial differential equations (PDEs)
- The closure of the equations require material laws
 - conductivities, permeabilities, viscosity, diffusivity,...
 - Free energies, chemical rate constants,...
- Historically the PDEs in the field of CE could only be solved analytically in some simple cases
- The computational approach has given the classical fields a renaissance

Space discretization methods

- **Finite Difference** method (google: 1.25 M)
 - Old timer, still a lot of use in basic physics
- **Finite Volume** method (google: 1.29 M)
 - The prevailing method in computational fluid dynamics
- **Finite element** method (google: 4.10 M)
 - Workhorse of computational engineering
- Other basis: spectral, wavelet
 - some special uses in simple geometries
- Meshless method
 - Pointless method? Still no field where it would rule
- Particle based methods
 - Shows promise in complex CFD

- Note: Usually time discretization is done using finite difference method
 - Explicit and implicit timestepping

Finite Volume vs. Finite element

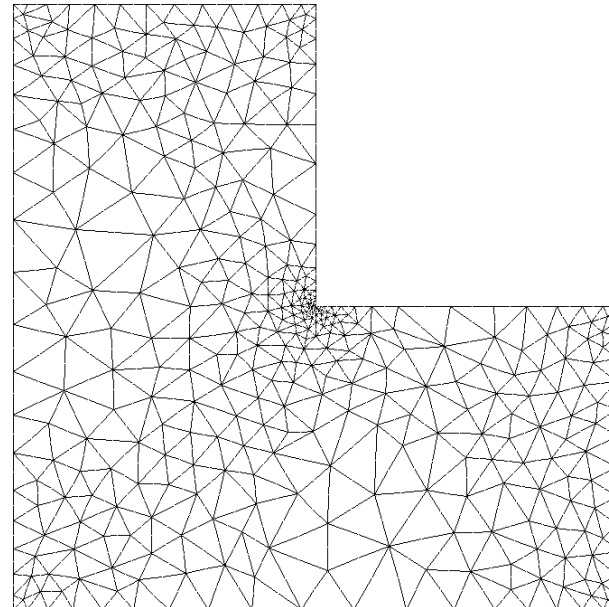
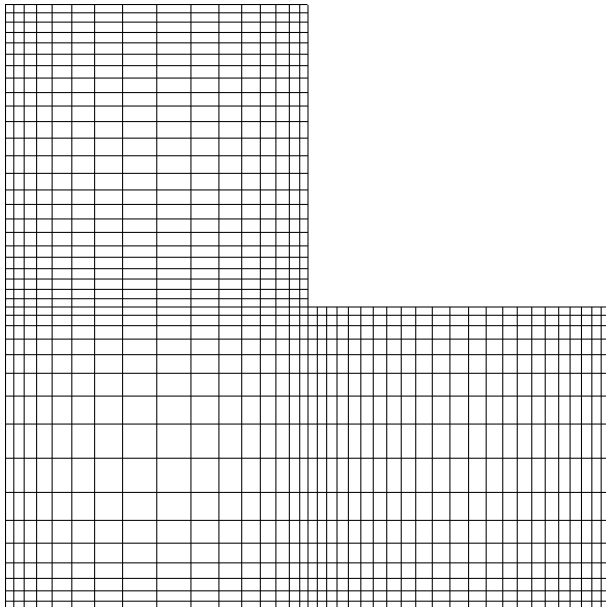


- ➊ In computational engineering the two main methods are FVM and FEM
 - Both can deal with arbitrary shapes
- ➋ Finite element method
 - Naturally suited for elliptic PDEs in weak form
 - Extended to parabolic PDEs by stabilization methods
 - Most generic method: CEM, CSM, CFD,...
- ➌ Finite volume method
 - Naturally suited for parabolic PDEs in conservative form
 - Extended to elliptic equations in the steady state limit
 - Most popular methods for CFD

Mesh types



- Computational meshes in FEM and FVM can be either structured or unstructured
- In a structured mesh each (inner) node has the same topology (number of neighbouring nodes)
- Multiblock structured meshes may in principle utilize more efficient data structures
- In practice, **unstructured data formats** are used CE



Unstructured meshes and matrix structure



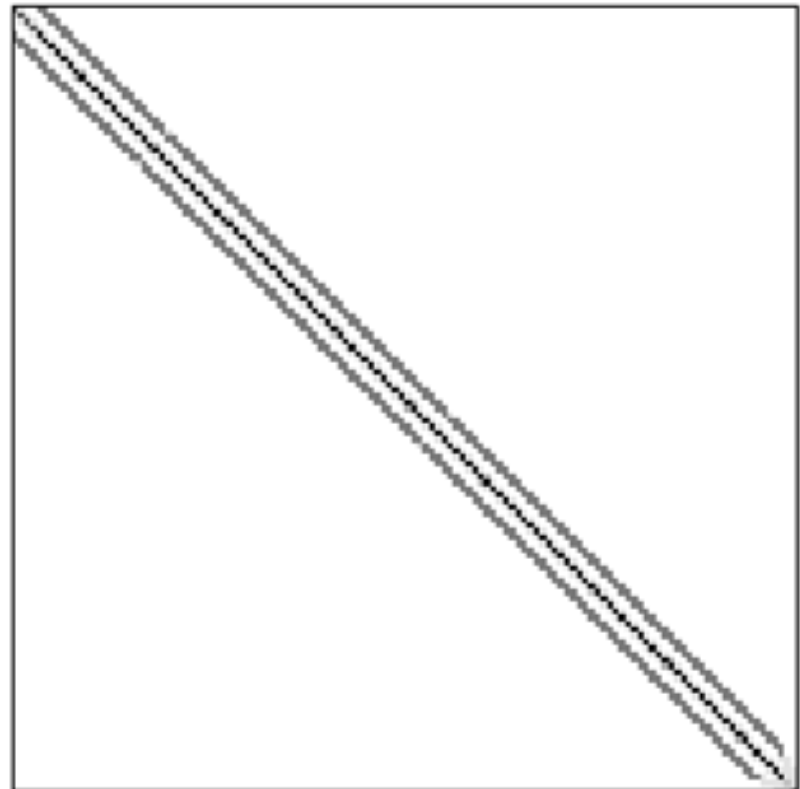
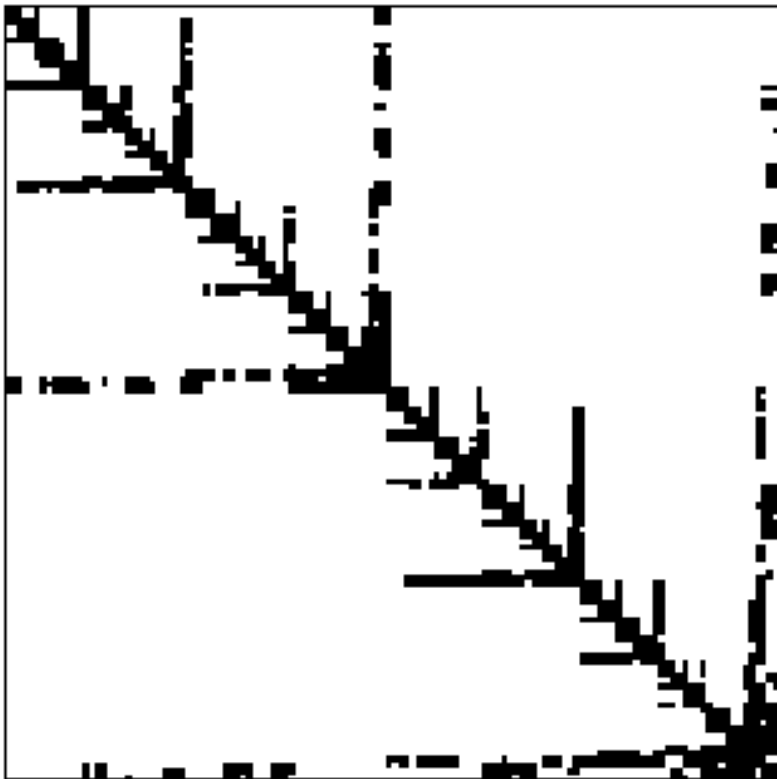
- ➊ PDEs on unstructured mesh result to linear systems, $Ax=b$, with sparse matrix structure
 - “Sparse linear systems”
 - Sparsity reflects the locality of PDEs & local support of basis
 - E.g. for nodal elements all nodes i,j within element result to entry (i,j) in the stiffness matrix
- ➋ Standard sparse matrix formats results to indirect memory addressing
 - Fetching the data from memory becomes the bottle-neck
 - Challenges for vectorization & multithreading
 - Poorly suited for GPU architectures
- ➌ Usually unstructured linear problems are solved in parallel with MPI

Example: Sparse matrices

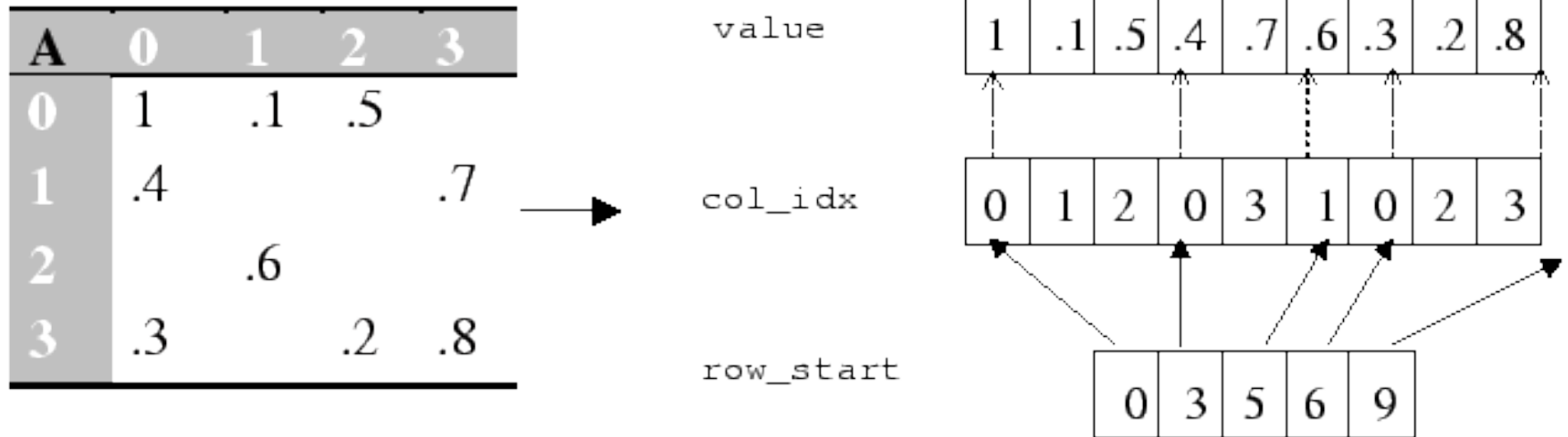


University of Florida sparse matrix collection

<http://www.cise.ufl.edu/research/sparse/matrices/>



Compressed sparse row format



Standard SMVM algorithm for computing $y = Ax$

```

for( r = 0; r < m; r++ ) { // for each row
    y[r] = 0;
    for (i = row_start[r]; i < row_start[r+1]; i++) {
        y[r] += a[i] * x[col_idx[i]];
    }
}

```



```
!-----
!> Matrix vector product (v = Au) for a matrix given in CRS format.
!-----
SUBROUTINE CRS_MatrixVectorMultiply( A,u,v )
!-----
REAL(KIND=dp), DIMENSION(*), INTENT(IN) :: u !< Vector to be multiplied
REAL(KIND=dp), DIMENSION(*), INTENT(OUT) :: v !< Result vector
TYPE(Matrix_t), INTENT(IN) :: A !< Structure holding matrix
!-----
INTEGER, POINTER, CONTIGUOUS :: Cols(:),Rows(:)
REAL(KIND=dp), POINTER, CONTIGUOUS :: Values(:)

INTEGER :: i,j,n
REAL(KIND=dp) :: rsum
!-----

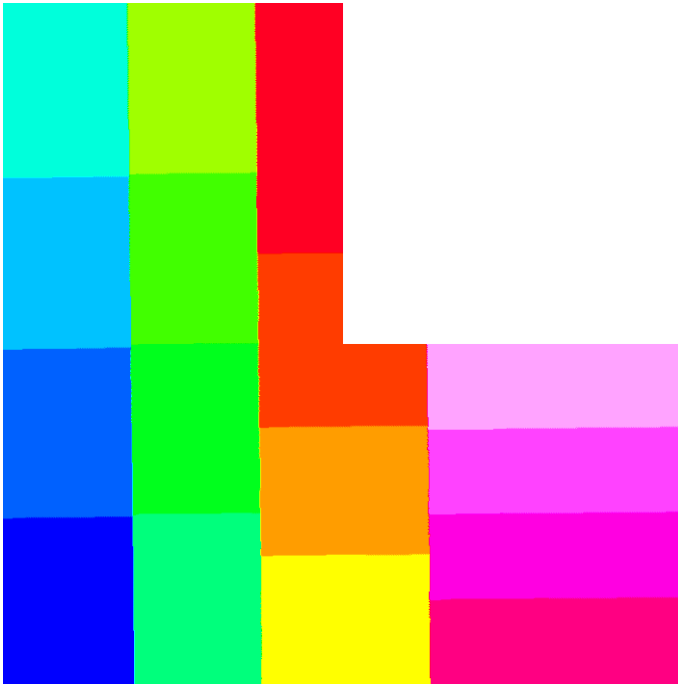
n = A % NumberOfRows
Rows => A % Rows
Cols => A % Cols
Values => A % Values

!$omp parallel do private(j,rsum)
DO i=1,n
rsum = 0.0d0
DO j=Rows(i),Rows(i+1)-1
rsum = rsum + u(Cols(j)) * Values(j)
END DO
v(i) = rsum
END DO
!$omp end parallel do
!-----
END SUBROUTINE CRS_MatrixVectorMultiply
!-----
```

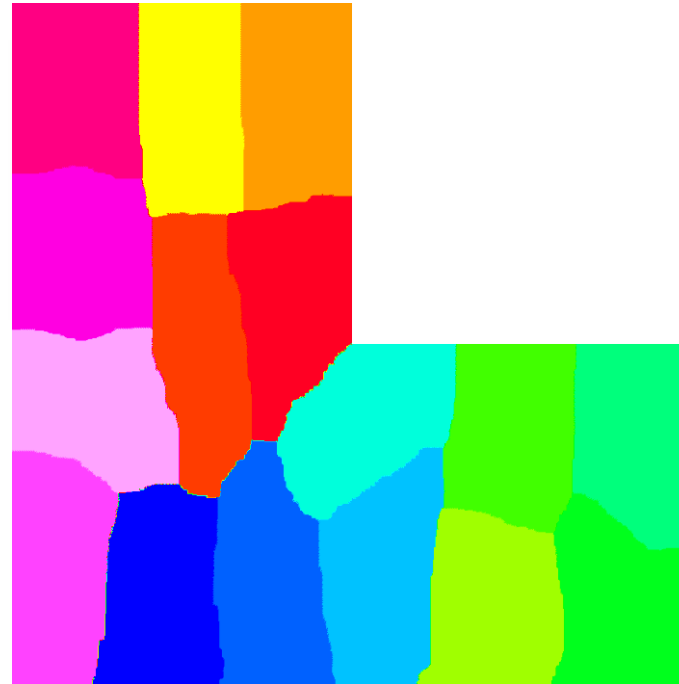
Unstructured meshes and parallelization

- ➊ It is natural to divide the computational mesh into subdomains
 - “Mesh partitioning”
 - Heuristic methods that try to minimize communication
- ➋ Communication required mainly at the interfaces where shared nodes are located
 - Fraction of shared nodes in 3D scales as $\sim(P/N)^{(1/3)}$
 - Relative importance of communication increases with number of partitions and decreases with size of problem (typically $1e4-1e5$ dofs for partition)
- ➌ Problems in computational engineering require fast connections between processors
 - Suitable applications for true supercomputers



Partitioning in 2D



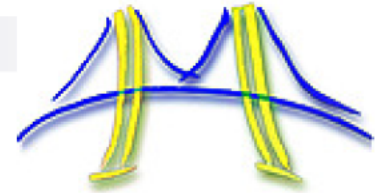
Partition by hierarchical 4 times 4 divisions in x- and y-directions



Partition to 16 domains by Metis algorithm
www-users.cs.umn.edu/~karypis/metis/



Phillip Colella's "Seven dwarfs"



High-end simulation in the physical sciences = 7 numerical methods:

1. Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
 2. Unstructured Grids
 3. Fast Fourier Transform
 4. Dense Linear Algebra
 5. Sparse Linear Algebra
 6. Particles
 7. Monte Carlo
- A dwarf is a pattern of computation and communication
 - Dwarfs are well-defined targets from algorithmic, software, and architecture standpoints



Open Source software solutions

Free / Open Source software



- Definition of **free** software
 - Software can be *used*, *studied*, and *modified* without restrictions
 - Software can be *copied* and *redistributed* in modified or unmodified form either without restriction, or with minimal restrictions only to ensure that further recipients have the same possibility.
- In English language the word free has two meanings
 - Free as in "free beer" (suom. Ilmainen)
 - Free as in "free speech" (suom. vapaa)
 - Free software movement was ideologically rooted whereas current concept of **Open Source** software is more pragmatic

Main categories of licences



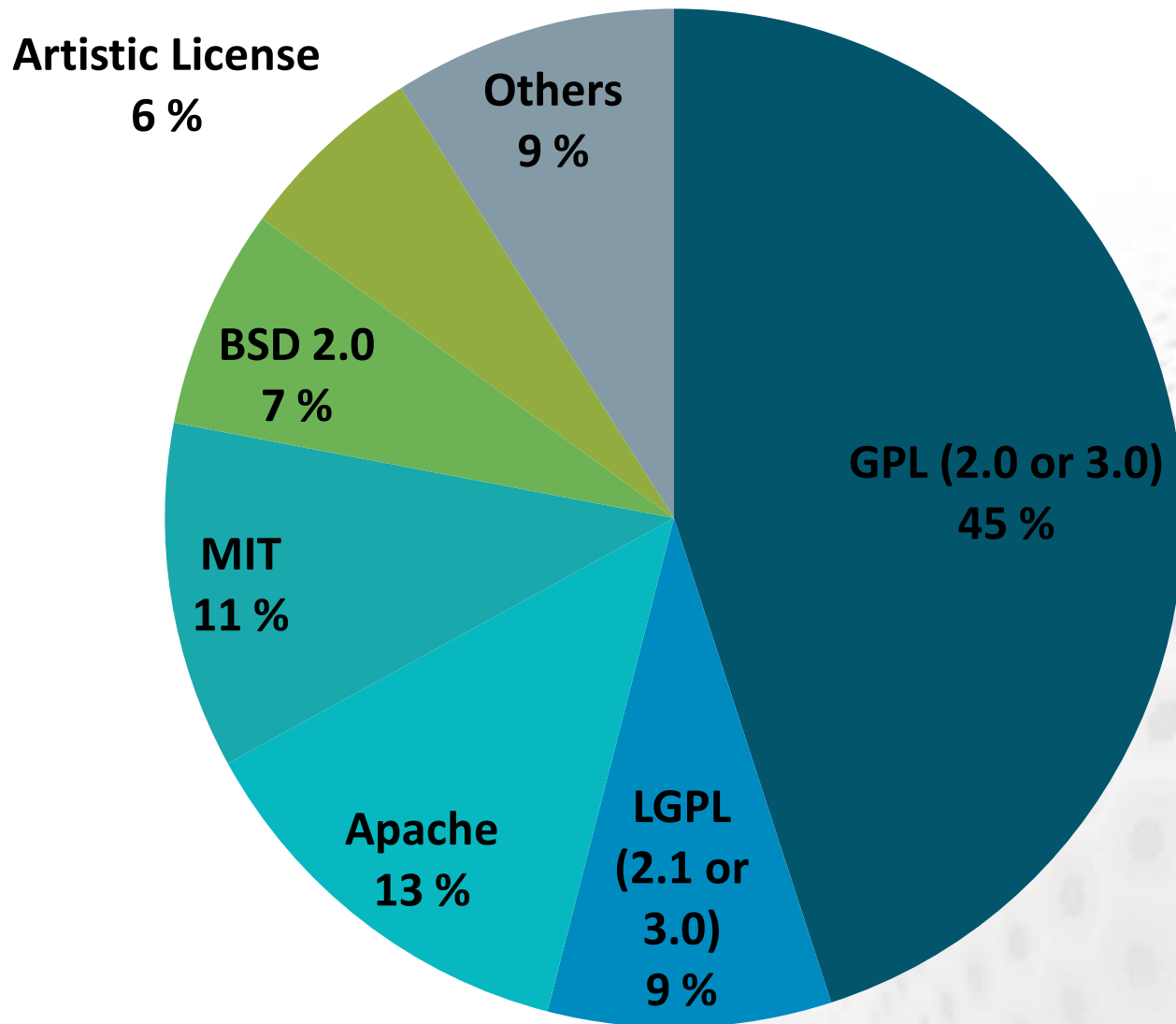
Restrictive licences

- **GNU, LGPL**
- Derived work must carry the same license – if published
("viral effect")
- Also known as *"copyleft"* licenses

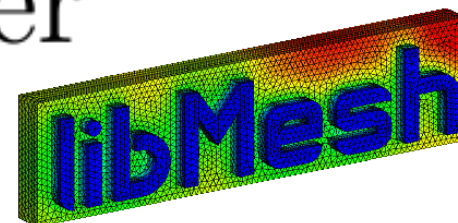
Permissive licences

- **BSD, MIT, Apache**
- Minimal requirements on how software may be redistributed
- Some differences among patent rights and author integrity between the three

License share



Open Source software for Computational Engineering

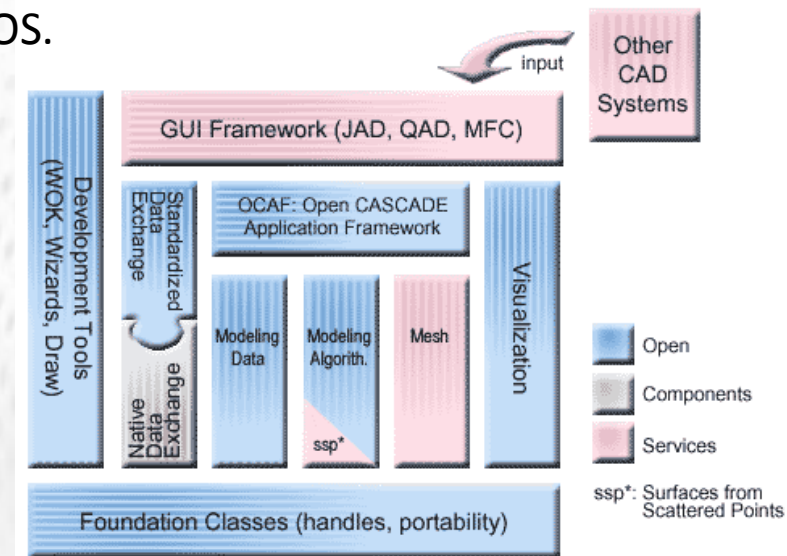


CAD – OpenCASCADE

<http://www.opencascade.com/>

<http://www.opencascade.org/>

- What is it?
 - Open CASCADE is a powerful CAD/CAM/CAE kernel and development platform for 3D modeling applications.
 - It consists of reusable C++ object libraries and a set of development tools available under OS.
 - Modular structure (see diagram)
- Development history
 - EUCLID-IS CAD/CAM system 1987
 - Published under Open Source in 1999 as OpenCASCADE
 - Customers CEA, BMW, SAMTECH, EADS, RINA, Alcatel,...
- The only proper CAD library under Open Source?

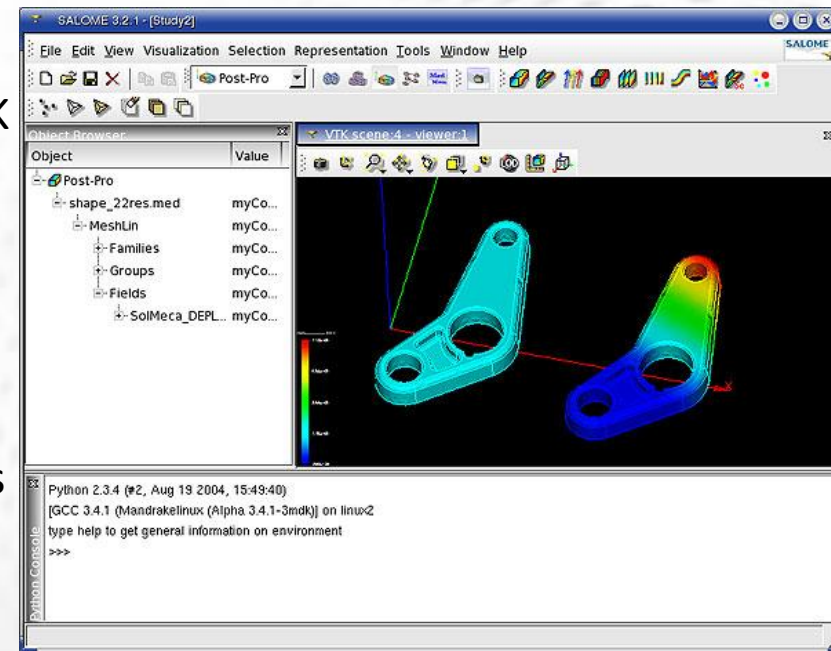


CAD – SALOME



<http://www.salome-platform.org/>

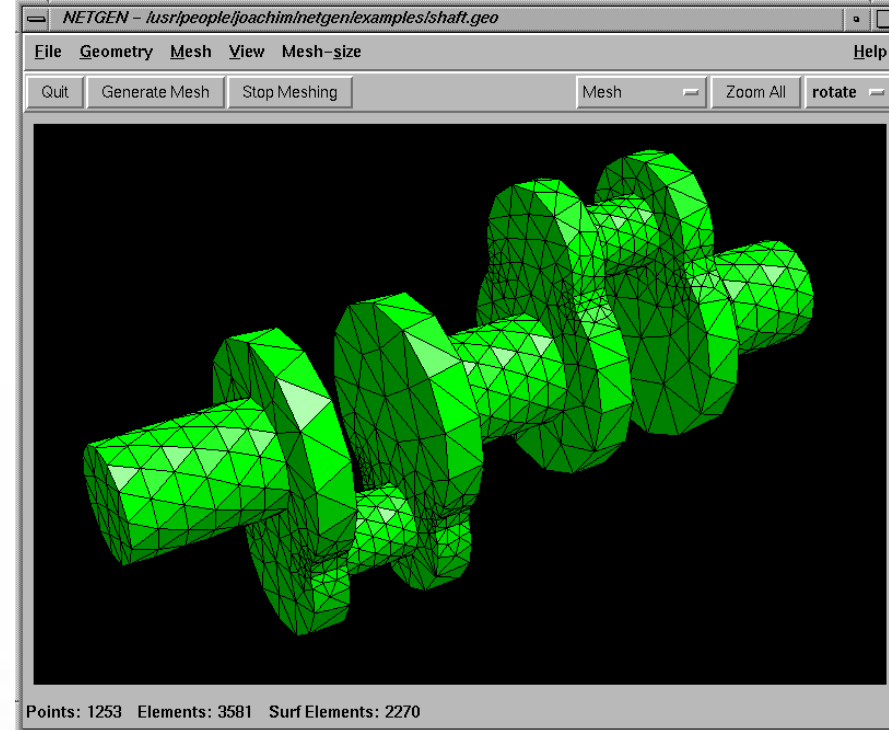
- What is it?
 - Free software that provides a generic platform for Pre and Post-Processing for numerical simulation.
- Based on a number of free software libraries
 - Qt, OpenCASCADE, Doxygen, Python, VTK
- Main functions
 - Create/modify, import/export (IGES, STEP), repair/clean CAD models
 - Mesh CAD elements, check mesh quality, import/export mesh (MED, UNV, ASCII)
 - Handle physical properties and quantities attached to geometrical items
 - Perform computation using one or more external solvers (coupling)
 - Display computation results
 - Manage studies (creation, save, reload)



Meshing - Netgen

<http://www.hpfem.jku.at/netgen/>

- What is it?
 - An automatic 2D/3D tetrahedral mesh generator
 - Developed mainly by Joachim Schöberl
- Key features
 - Accepts input from constructive solid geometry (CSG) or boundary representation (BRep) from STL file format
 - Connection to OpenCASCADE deals with IGES and STEP files
 - Contains modules for mesh optimization and hierarchical mesh refinement
 - LGPL library
- Netgen library is utilized by a large number of GUI projects



CFD - OpenFOAM



<http://www.opencfd.co.uk/openfoam/>

- No 1 CFD software under open source
- Features
 - Based on C++ modules which are used to build number of solvers
 - Uses finite volume numerics to solve systems of partial differential equations ascribed on any 3D unstructured mesh of polyhedral cells.
 - Comes with models for fluid flows involving chemical reactions, turbulence and heat transfer
 - Includes some rude utilities for pre- and post-processing
 - Fully parallelizable with iterative solvers
 - License under GPL
- OpenFOAM may be the best example of OS service in CE
 - Started as a PhD project, now owned by ESI Group
 - Many small consultancy companies and major R&D departments base their operation on OpenFOAM

FEM – freefem++

<http://www.freefem.org/ff++>

➤ What is it?

- One of the 1st free libraries (traces back to MacFEM, 1985)
 - Developed by O. Pironneau, F. Hecht et al.
- A language dedicated to the finite element method that enables easy solution of Partial Differential Equations (PDE)
- Idea has been copied and refined (Comsol multiphysics, FEnics etc.)
- Mainly educational use nowadays

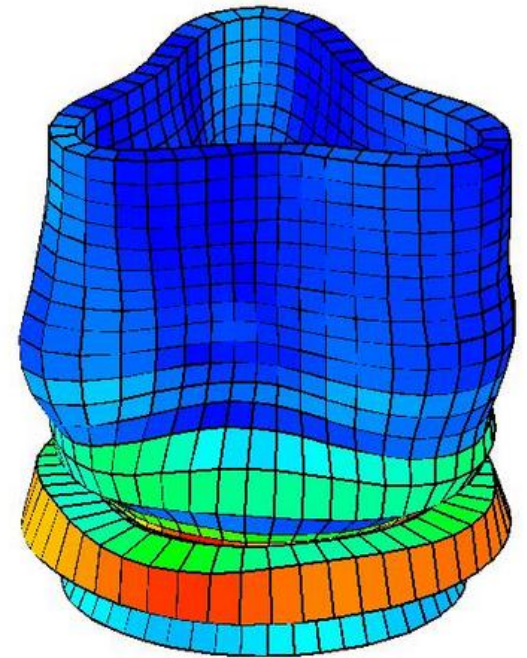


FEM library – deal.II

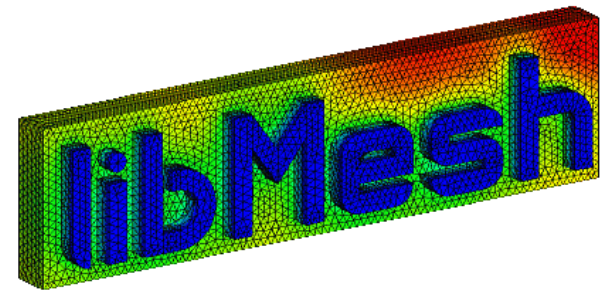


➤ What is it?

- A Finite Element **D**ifferential **E**quations **A**nalysis **L**ibrary
- A program library rather than end-user program
- Computational solution of partial differential equations using adaptive finite elements
- Uses state-of-the-art programming techniques to offer you a modern interface to the complex data structures and algorithms
- main aim is to enable rapid development of modern finite element codes
- Good demonstration of a modern approach taking use of the best available tools



FEM library - libMesh



➤ What is it

- Library for the numerical simulation of partial differential equations using arbitrary unstructured discretizations on serial and parallel platforms
- Provides adaptive mesh refinement computations in parallel
- libMesh currently supports 1D, 2D, and 3D steady and transient finite element simulations.
- Makes use of high-quality whenever possible: PETSc, LASPack, SLEPc, Metis, Triangle, Tetgen
- Active development:
Univ. of Texas at Austin, Technische Universität Hamburg, Sandia National Laboratories, NASA Lyndon B. Johnson Space Center

FEM - Elmer



<http://www.csc.fi/elmer>

➤ What is it

- Multiphysical finite element software under open source
- Primarily targeted for end-users, but also a library
- Development started 1995, GPL 2005, LGPL 2012

➤ Features

- GUI, Solver & Postprocessor
- All basic element types (1D, 2D, 3D, nodal, edge, face, p, DG)
- Large number of different physical equations

➤ Uses many open source libraries

- CAD: OpenCASCADE
- Meshing: Netgen, Tetgen
- Lin.Alg: Umfpack, MUMPS, Hypre, Lapack, Parpack
- Visualization: VTK

Numerics



- This area is inherently part of academic developments
 - Many of the best products are published under Open Source
- Linear algebra for dense matrices
 - Lapack
- Direct sparse solvers
 - Umfpack, Mumps, Spools, ...
- Eigenvalue solvers
 - Arpack, Parpack
- Iterative solvers, preconditioners
 - Hypre
- Graph partitioning
 - Metis, Scotch, ParMetis, PT Scotch
- Collections of different tools for parallel computing
 - PETSc, Trilinos

Visualization - VTK



<http://www.vtk.org/>

➤ What Is it?

- Software system for 3D computer graphics, image processing, and visualization

➤ Features

- Consists of a C++ class library and several interpreted interface layers including Tcl/Tk, Java, and Python.
- VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods
- Supports parallel processing

➤ Professional support provided by Kitware Inc.

- Proper documentation not free
- Supported by a number of large institutions: Los Alamos and Sandia nat.lab.

Visualization - Paraview

<http://www.paraview.org/>

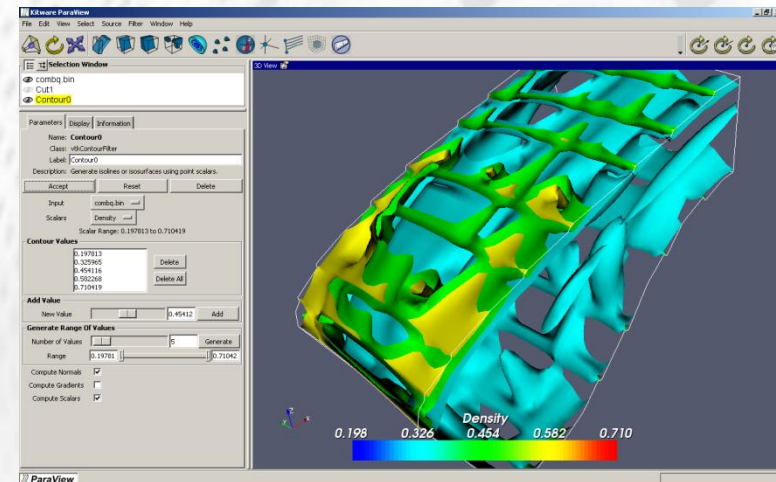


➤ What Is it?

- An open-source, multi-platform data analysis and visualization application
- Developed to analyze extremely large datasets using parallel computing

➤ Features

- Data exploration may be done interactive or using batch processing
- Can be run on laptops and supercomputers
- Based on VTK library

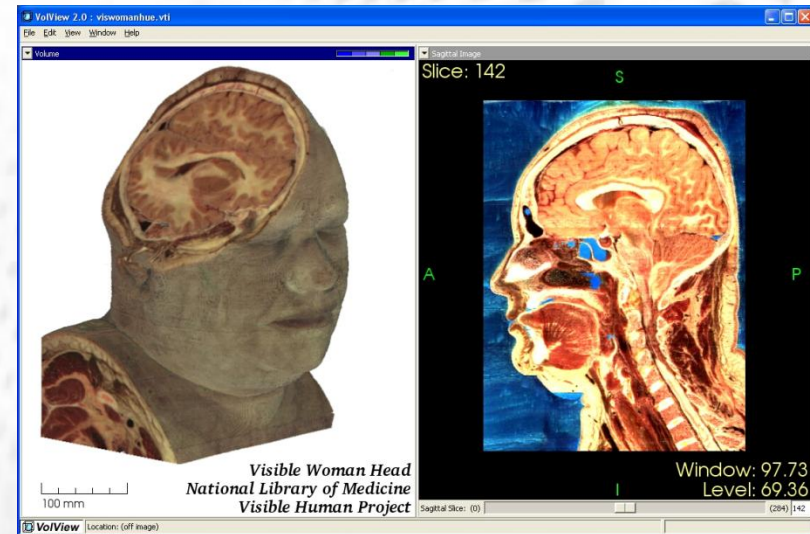


Visualization - VisIT



<http://wci.llnl.gov/visit/>

- What is it?
 - Interactive parallel visualization and graphical analysis tool for viewing scientific data on Unix and PC platforms
 - Developed by Department of Energy (DOE)
 - Rather similar in features as Paraview



Qt



<http://qt.digia.com>

- Qt is a cross-platform complete development framework written in C++
 - High level of abstraction makes coding process very efficient



**Code less.
Create more.
Deploy everywhere.**

- Initially developed by Trolltech -> Nokia -> Digia
- Used by number of software tools in CE
 - SALOME, Paraview, ElmerGUI,...

Python



- Python is a programming language that allows for quick testing and prototyping
- Python bindings available in many libraries: Qt, SALOME, VTK, Paraview, PetSc, Trilinos,...



Open source software in CE

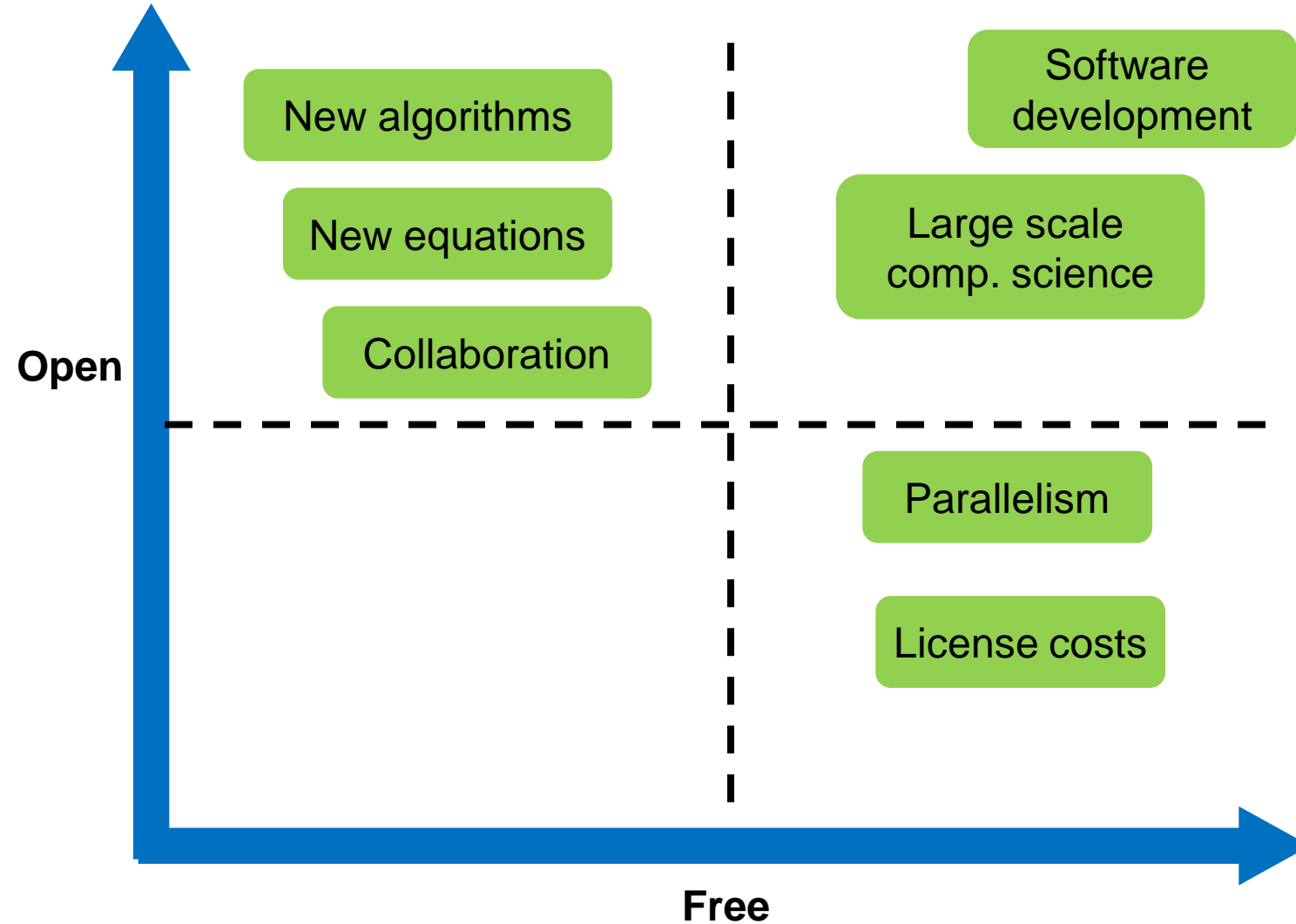


- Academically rooted stuff is top notch
 - Linear algebra, solver libraries
 - Petsc, Trilinos, OpenFOAM, LibMesh++, ...
- CAD and mesh generation not that competitive
 - OpenCASCADE legacy software
 - Mesh generators netgen, tetgen, Gmsh are somewhat limited
 - Also for OpenFOAM there is development of commercial preprocessing tools
- Users may need to build their own workflows from the most suitable tools
 - Also in combination with commercial software
 - Excellent libraries for software development (Qt, python,...)

Reasons to use open source software in CE

free as in "beer" vs. free as in "speech"

CSC



Benefits of the openness of the code



- ➊ In collaboration all parties have access to the software
 - Companies, universities, consultants,...
- ➋ Open source software has more different roles
 - May be used to attract a wider spectrum of actors
- ➌ Also fundamental ideas may be tested with the software
 - Algorithms, models,...
 - Compatible with scientific method: falsification
- ➍ More possibilities to built tailored solutions
 - OS codes have usually good extendability & customizability
- ➎ At least some control over the intellectual property
 - Own model development does not become a hostage to **vendor lock in**
 - Sometimes rules GPL licence out of question

What kind of industry might utilize OS codes?



- Small (consultancy) company for which commercial prices may be unreasonable
- Company with strong academic collaboration involving new computational methods
- Company doing in-house simulator development for their technology
- Company that needs to use HPC for their simulation needs



Weaknesses of OS software in CE



➤ CAD & Meshing

- There is no process that would bring the best software under open source

➤ Lack of standardization

- Bottom-up type (Bazaar) of open source projects seem fundamentally incompatible with ISO 9001 standard
- One should perhaps not design buildings using OS software for the computation...

➤ Big business

- There are no global service organization for OS software (except maybe for OpenFOAM)
- The information management of CAD and simulation data is becoming an integral part of the work flow in large businesses and currently OS does not have solutions for that (?)

How the software for the course was chosen



- There is no generic solution for parallel mesh generation
- There are many excellent numerical libraries
 - Not directly usable for end-users
- There are numerous scalable FEM and FVM software
 - Fenics & Elmer are both popular FEM packages with somewhat different approach
 - Nek5000 presents extreme scalability
- Two excellent parallel visualization software under open source
 - Paraview & Visit
- For all the software presentations will be given by dedicated experts & developers of the software

Workflows for Computational Engineering

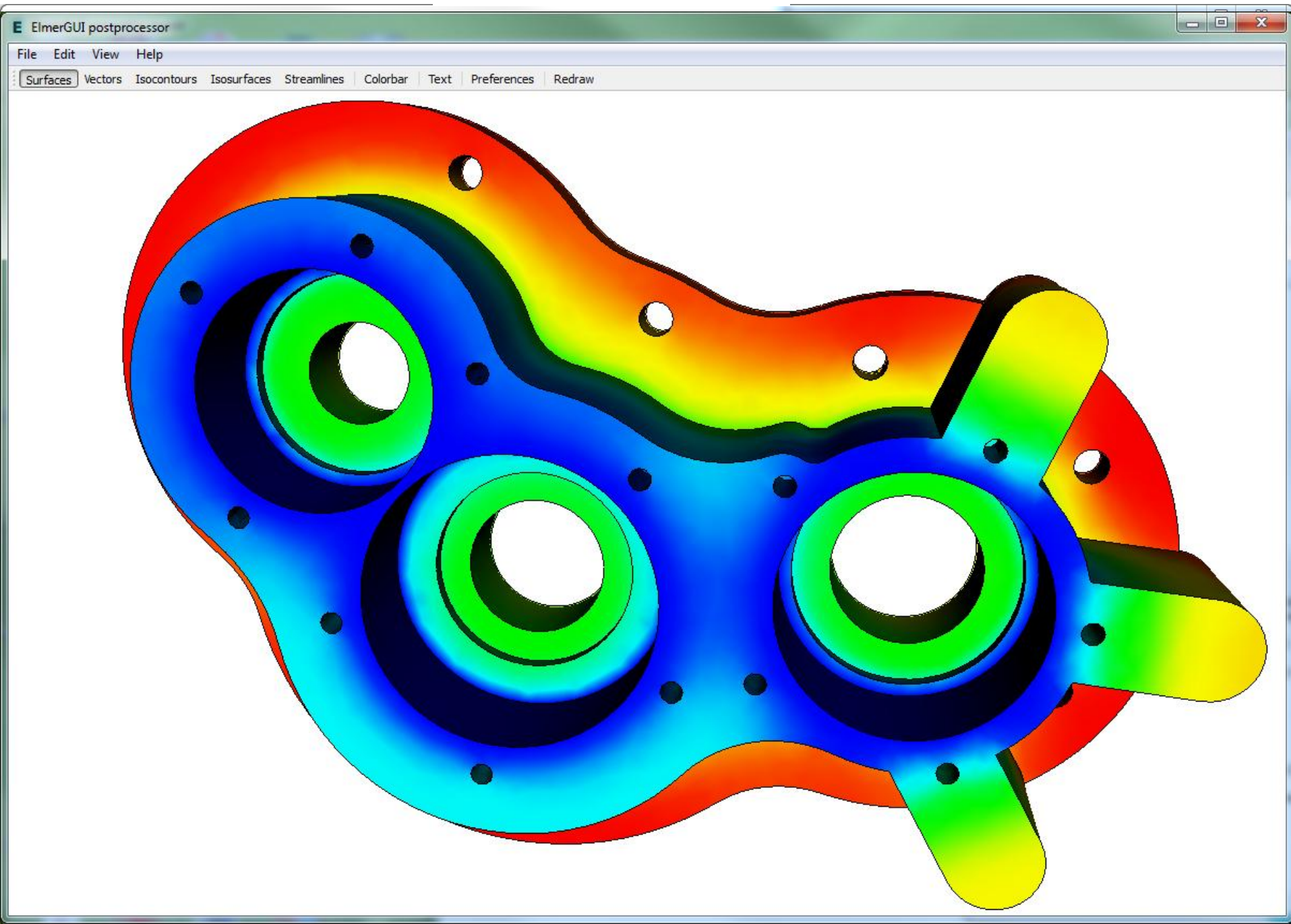
Basic workflow in computational engineering



- Preprocessing
 - Geometry definition
 - Mesh generation
 - Case definition
- Solution
 - Assembly of equations
 - Solution of the linear systems (implicit methods)
- Postprocessing
 - Visualization
 - Extracting information

SERIAL WORKFLOW:

VISUALIZATION



Serial workflow



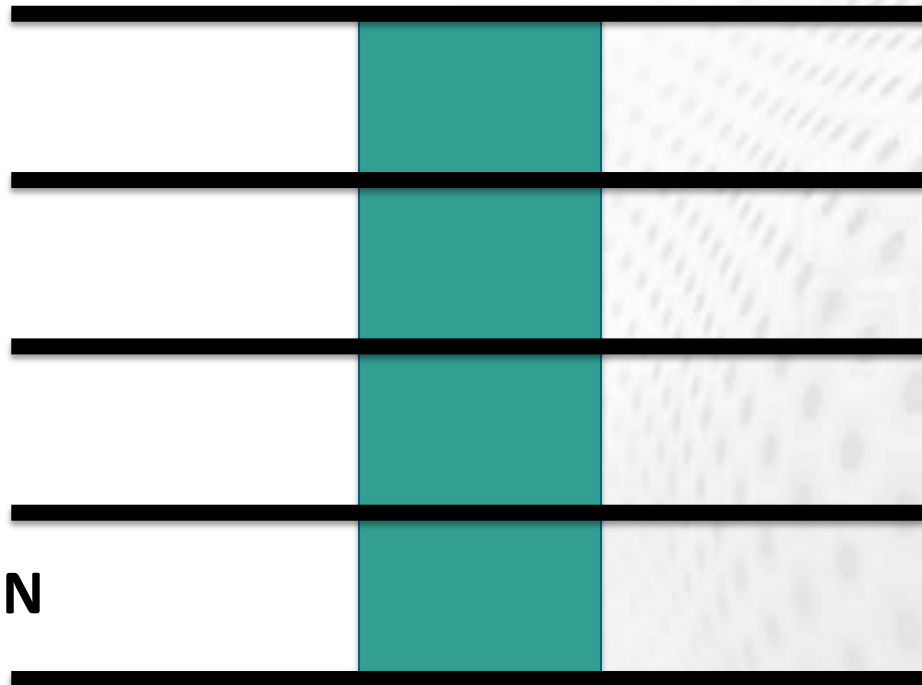
- All steps in the workflow are serial
- Typically solution of the linear system is the main bottle-neck

MESHING

ASSEMBLY

SOLUTION

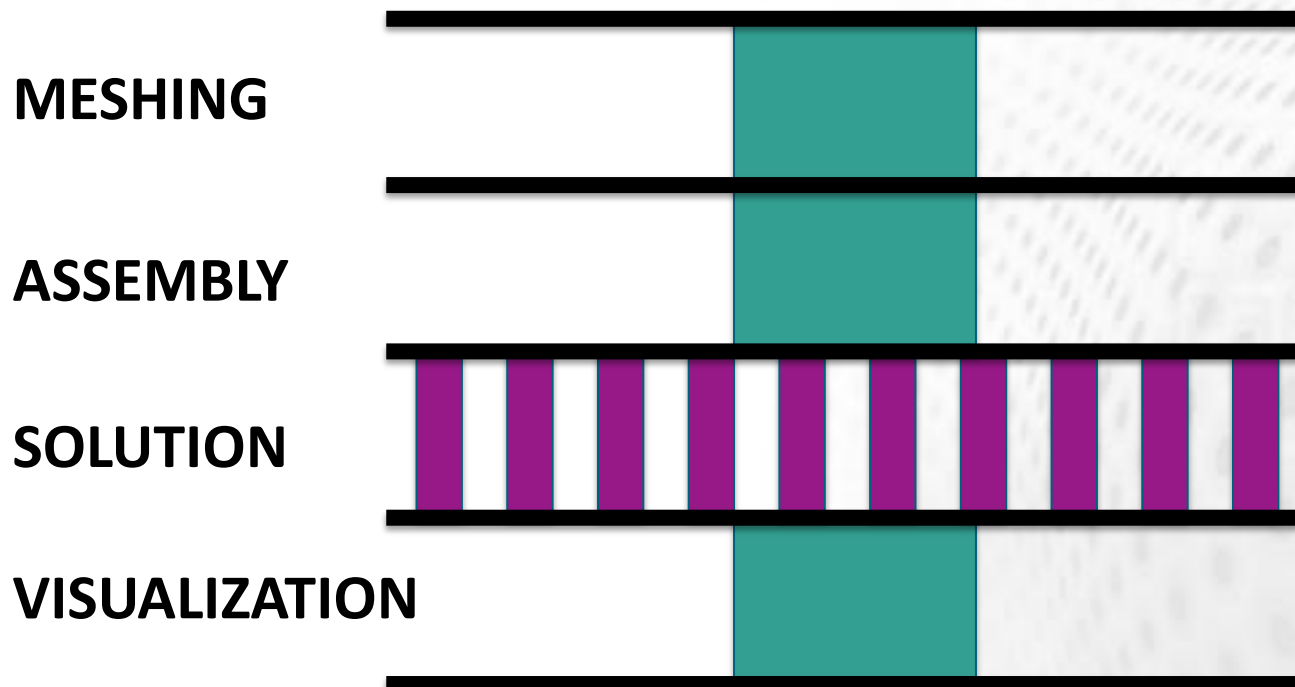
VISUALIZATION



Parallel workflow I



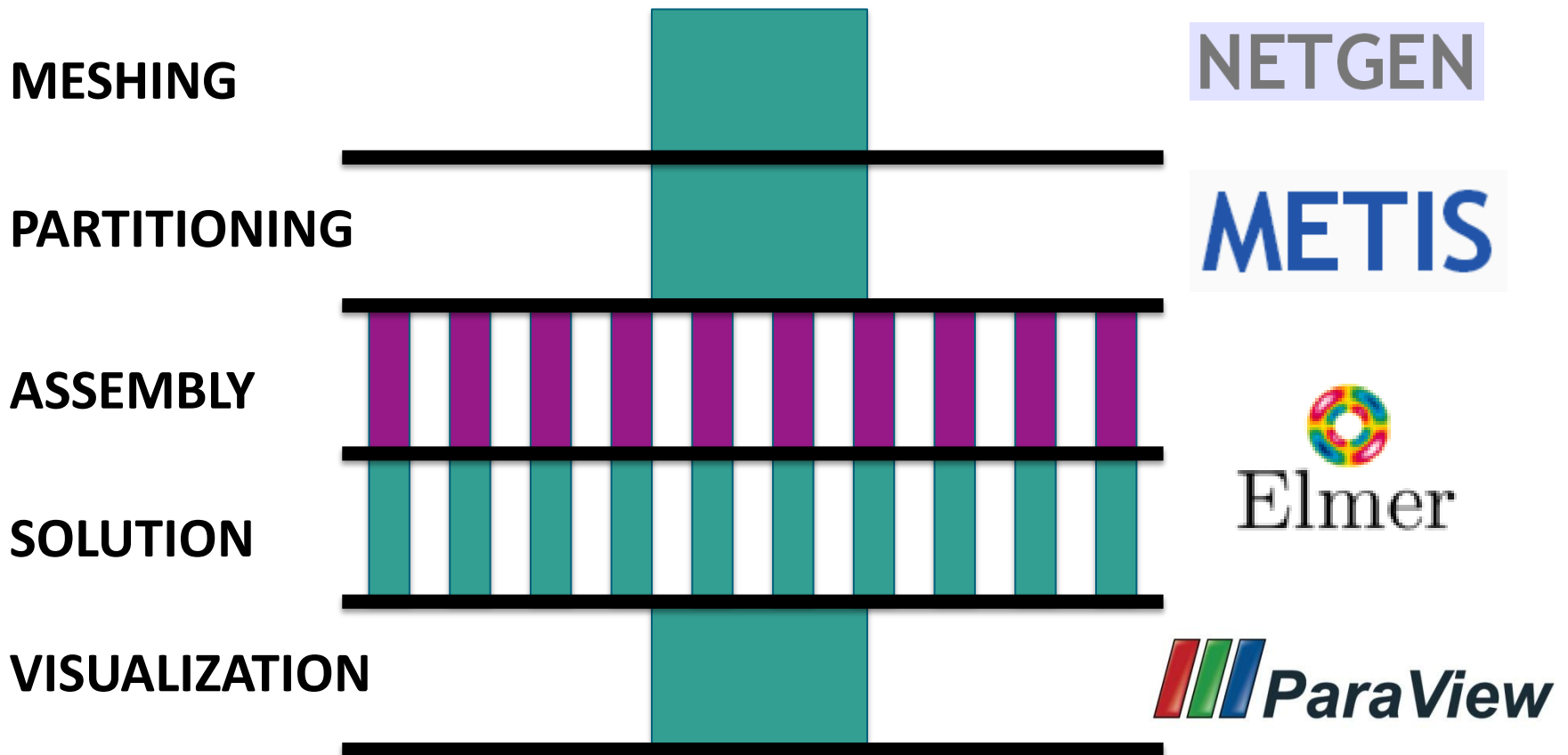
- Solution is boosted by parallel solution only
 - Easy to take into use by using suitable multithreaded libraries
- Finite element assembly typically uses 5-30%
 - Only moderate speed-up to be gained



Parallel workflow II

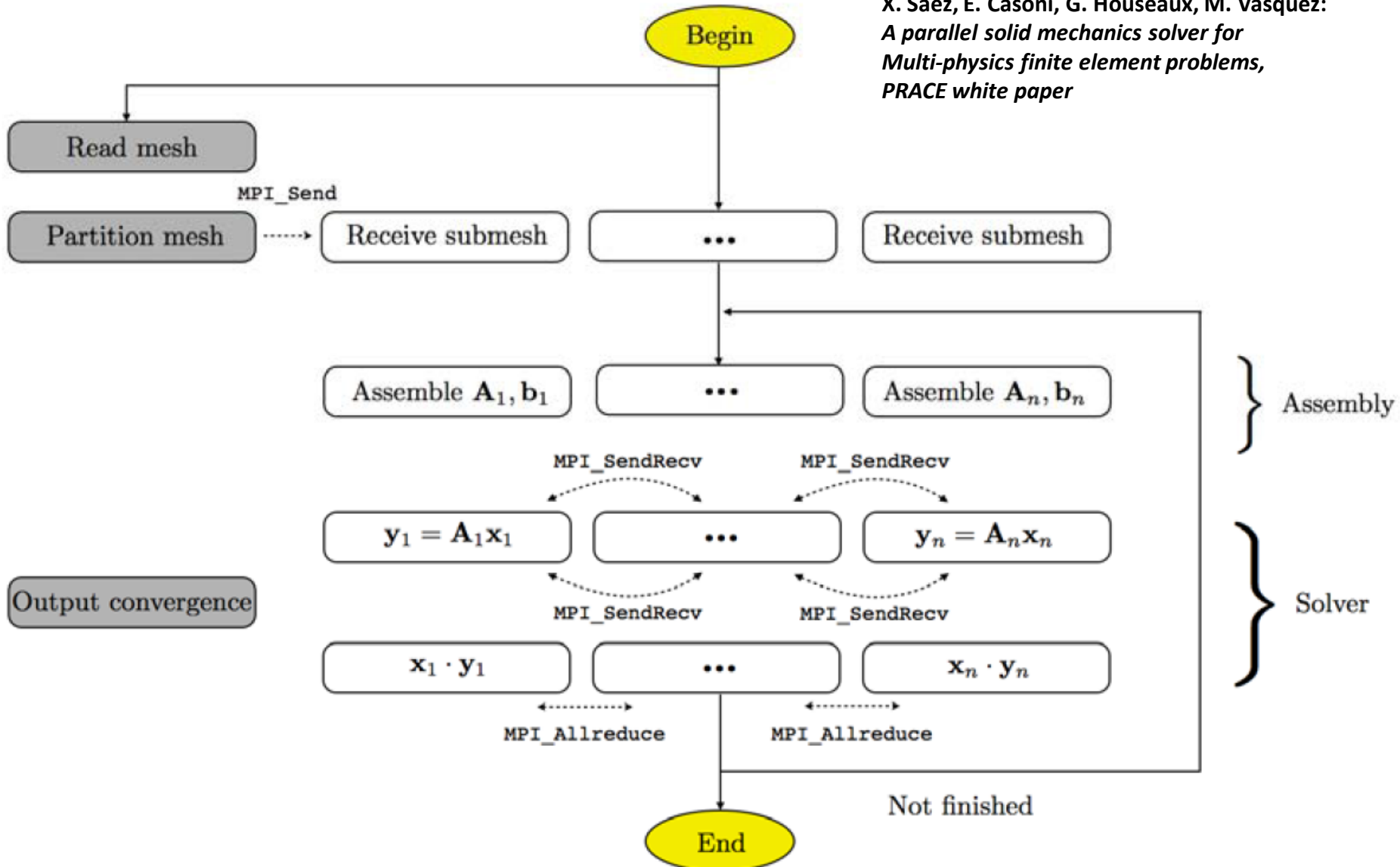


- Both assembly and solution is done in parallel using MPI
- Assembly is trivially parallel
- This is the most common parallel workflow



Example: Parallel workflow of Alya FEM code

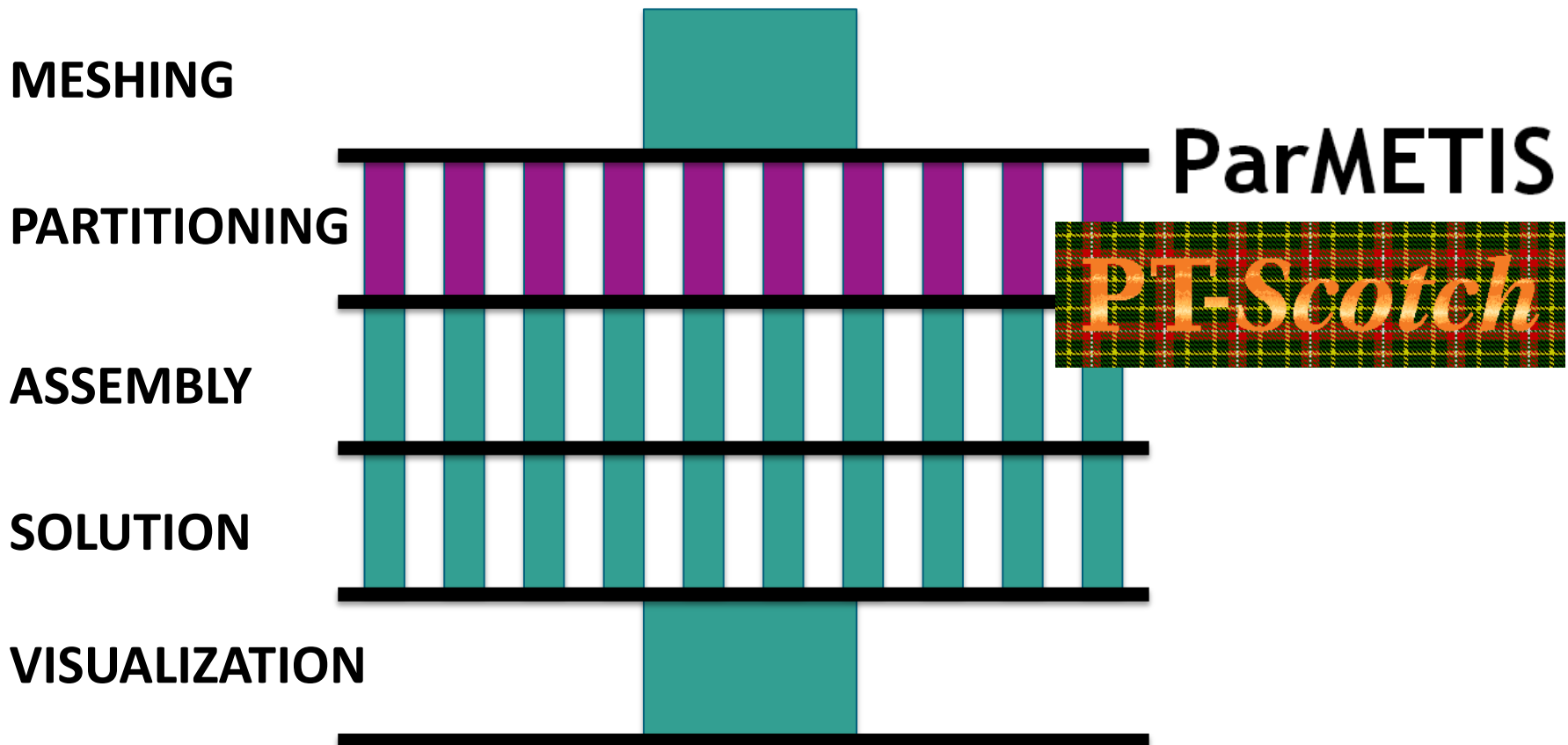
X. Saez, E. Casoni, G. Houseaux, M. Vasquez:
*A parallel solid mechanics solver for
Multi-physics finite element problems,
PRACE white paper*



Parallel workflow III



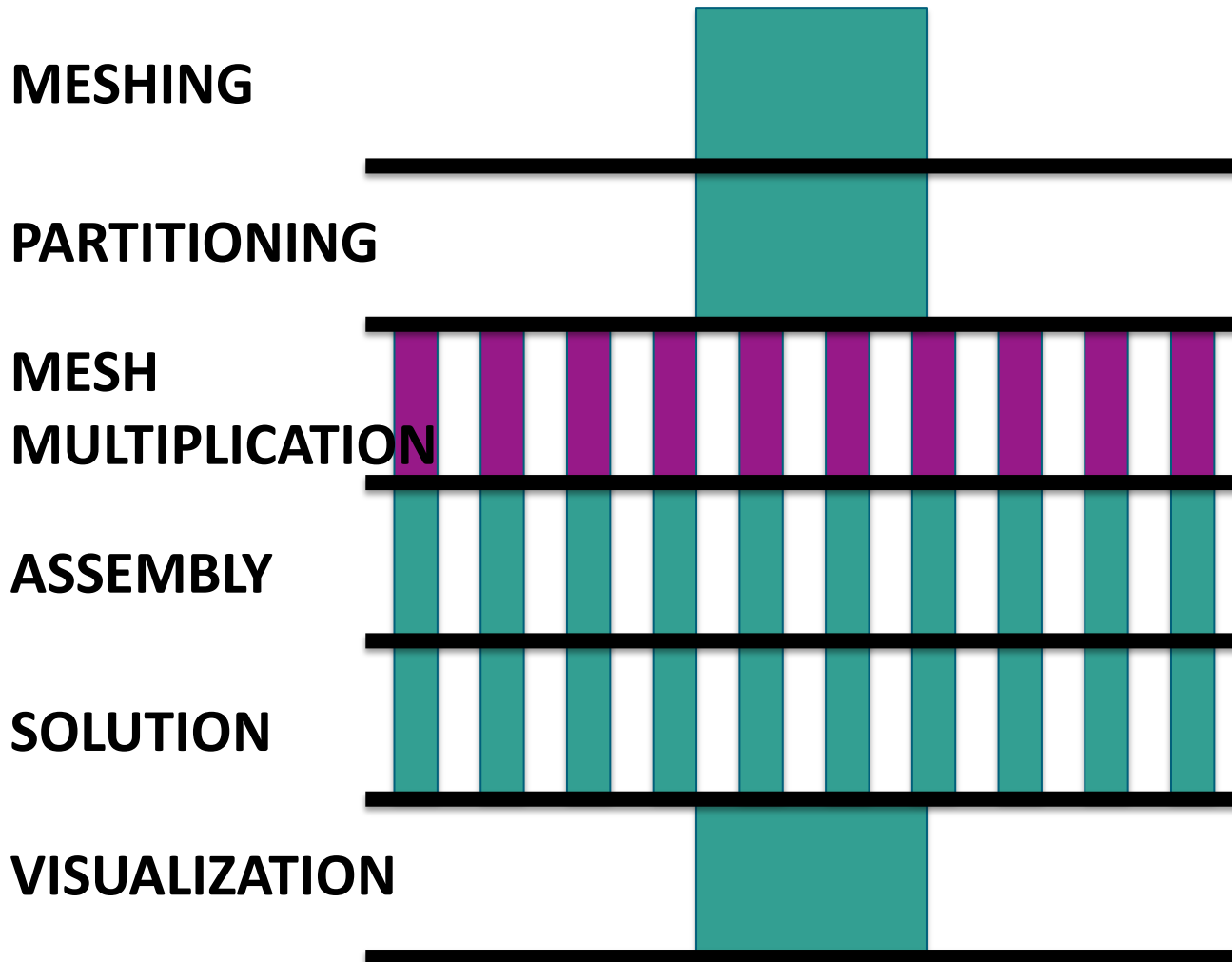
- Partitioning may also be done in parallel
- Partitioning is usually not the most severe bottle-neck



Parallel workflow IV



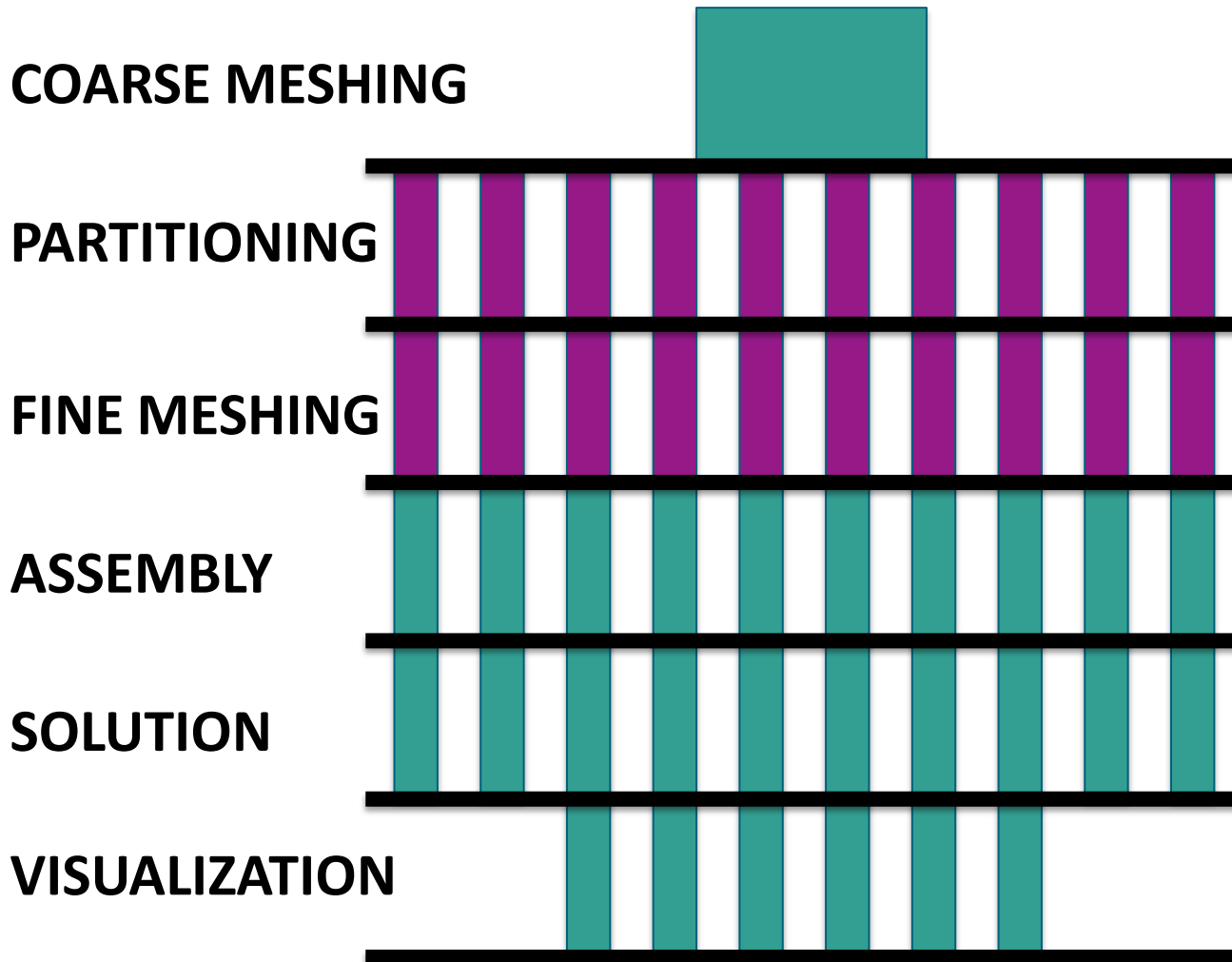
- Large meshes may be finalized at the parallel level



Parallel workflow V



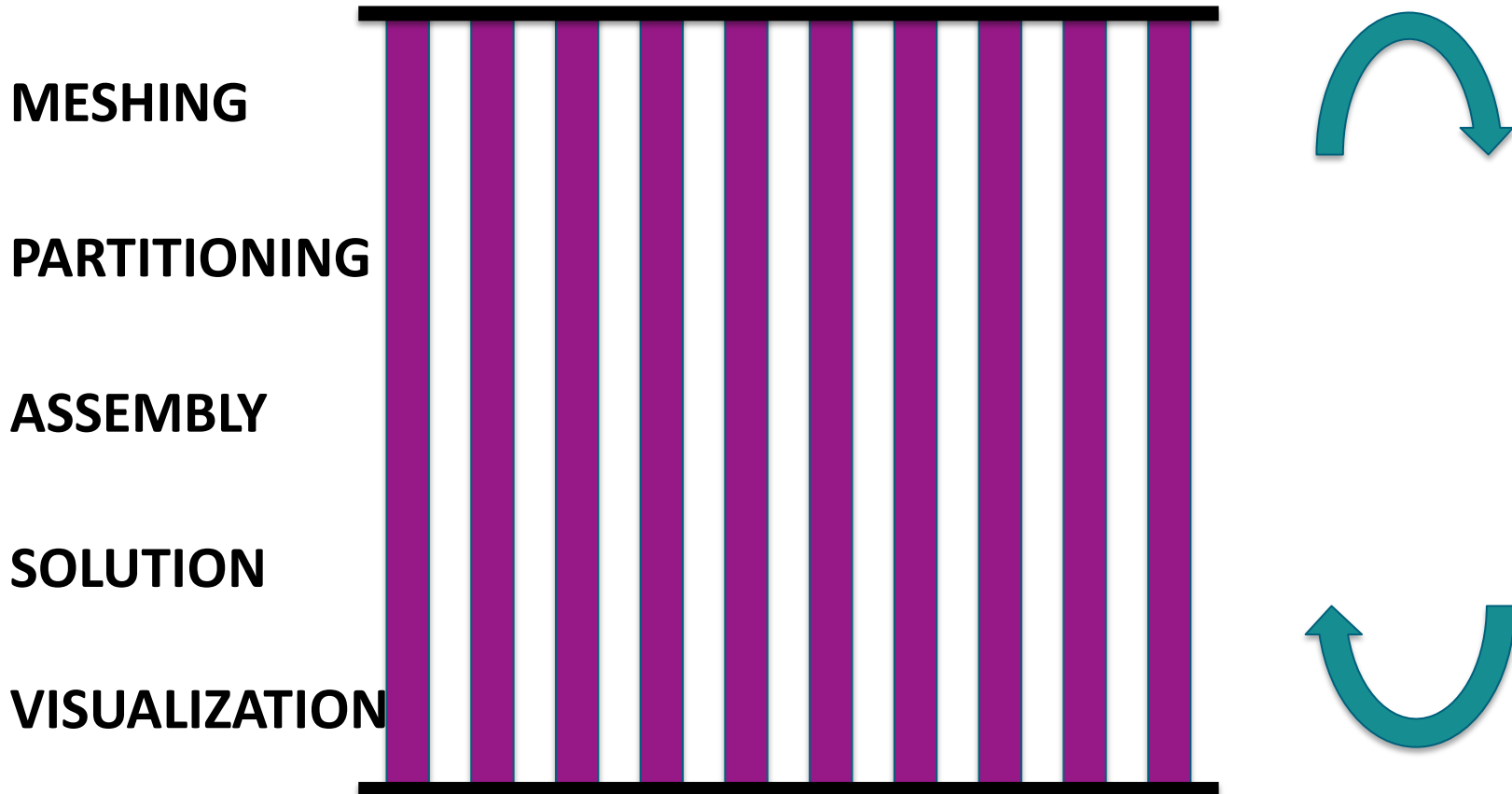
- Bottle-necks in preprocessing resolved by parallel meshing



Parallel workflow VI



- The ultimate workflow could include integrated geometry-accurate adaptive re-meshing and re-partitioning with parallel on-the-fly visualization

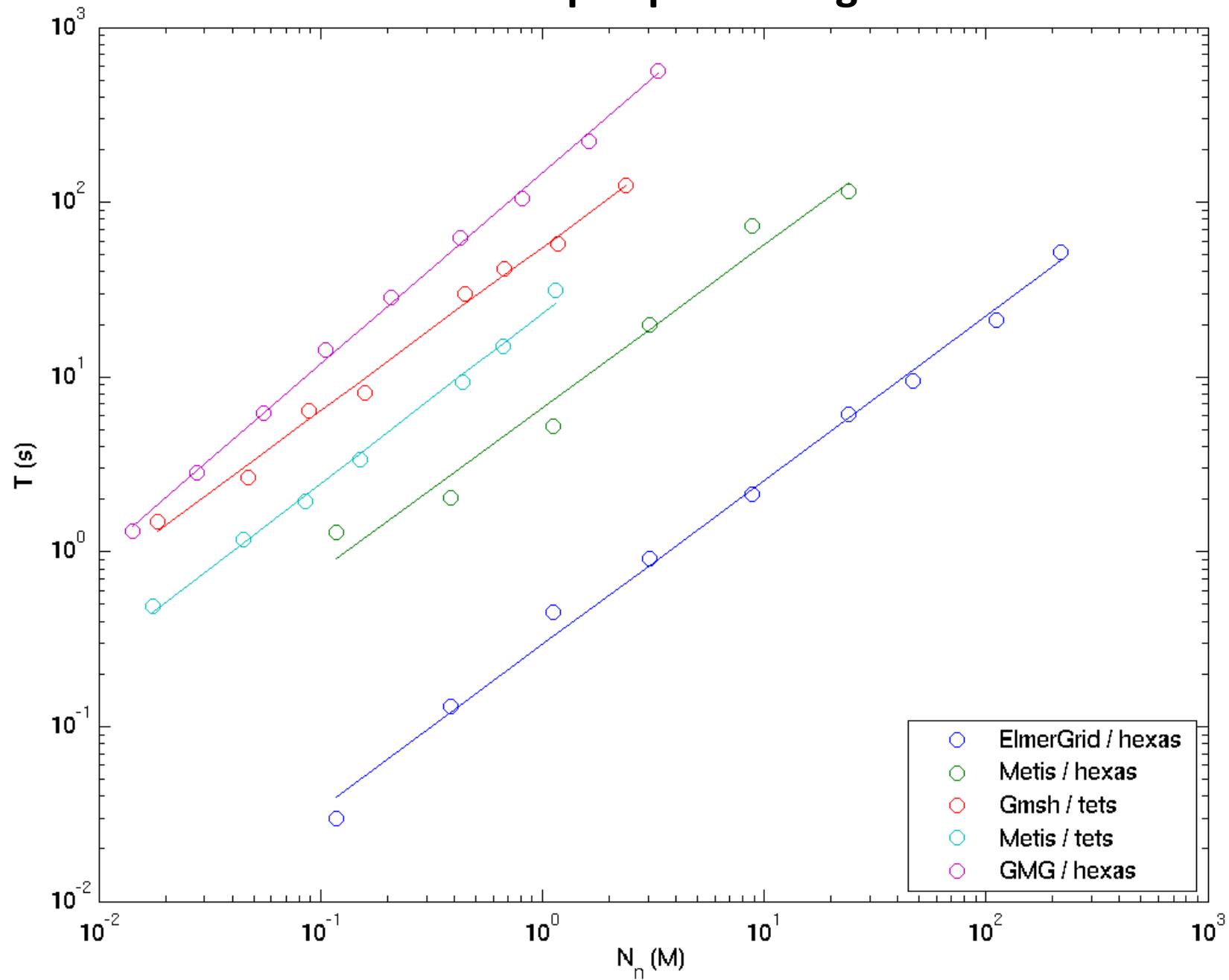


Algorithmic scalability

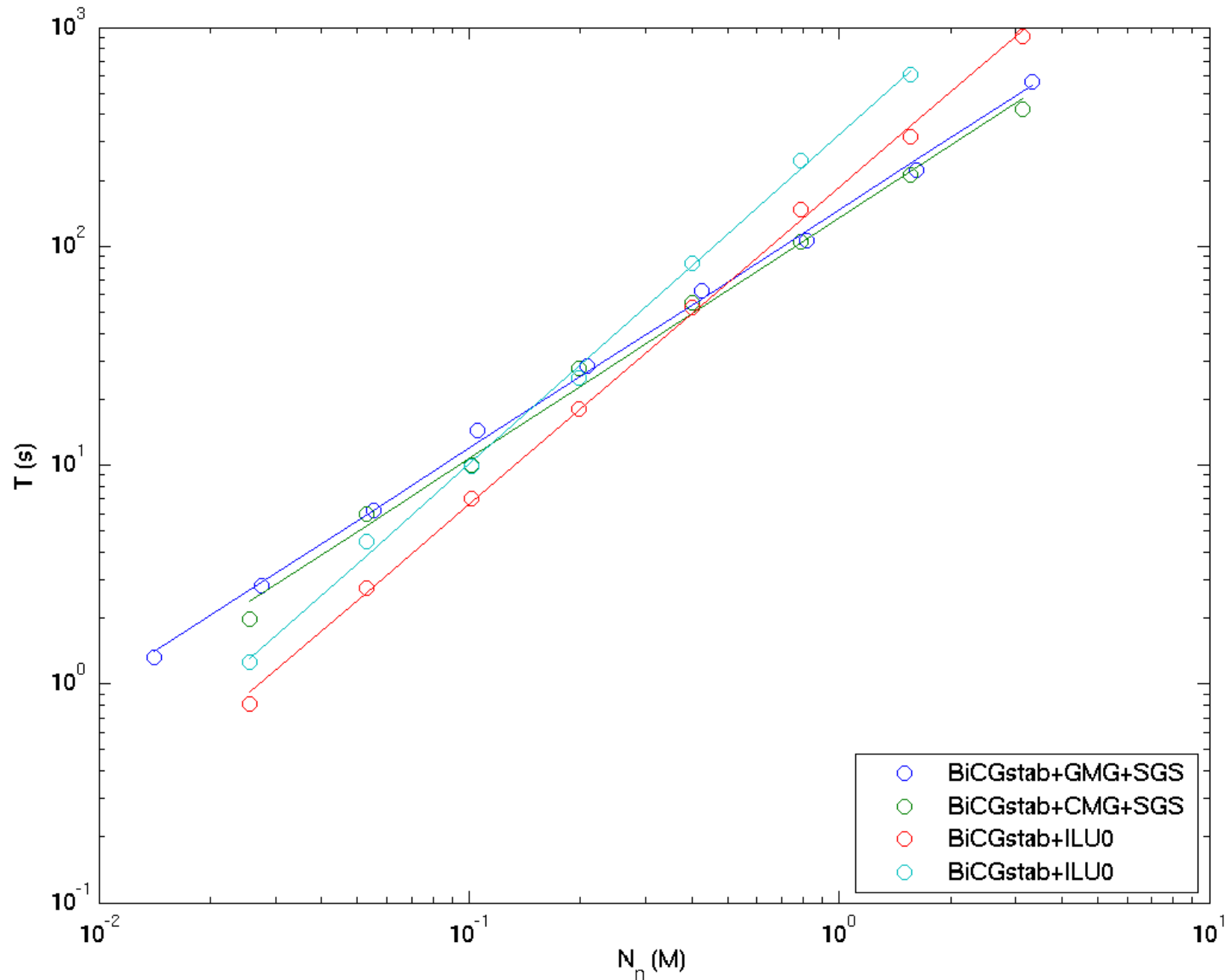


- Each algorithm has a characteristic scaling law that sets the lower limit to how the solution time increases with time
 - E.g. average scaling for sorting:
 - Quicksort $O(n \log(n))$
 - Insertion sort: $O(n^2)$
- The parallel implementation cannot hope to beat this limit
 - Targeting large problems 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

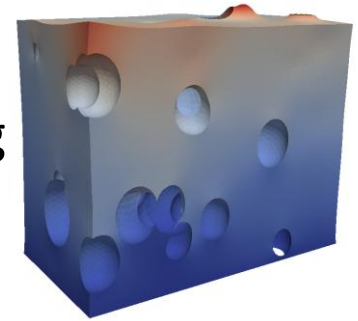
software	algorithm	mesh	α_T (s/M)	β_T	$\alpha_M(b)$
ElmerGrid	meshing	hexas	0.295	0.939	73.8
Metis	PartMeshNodal	hexas	6.67	0.932	377.0
Gmsh	Delaunay	tets	55.2	0.93	1481
Gmsh	Advancing Front	tets	155.1	1.00	643
Metis	PartMeshDual	tets	23.1	0.97	513.4
BiCGStab	CMG + SGS	hexas	134.9	1.100	1595
BiCGStab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

$T(\text{solution}) > T(\text{tet meshing}) > T(\text{partitioning}) > T(\text{hex meshing})$

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 $O(\sim 1.03)$ 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

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: $P \times T = \text{const.}$
 - A bad algorithm may have excellent strong scaling
 - Typically $1e4$ - $1e5$ dofs needed in FEM/FVM for good scaling
- **weak scaling**
 - How the solution time varies with the number of processors for a *fixed problem size per processor*.
 - Optimal case: $T = \text{const.}$
 - Weak scaling is limited by algorithmic scaling

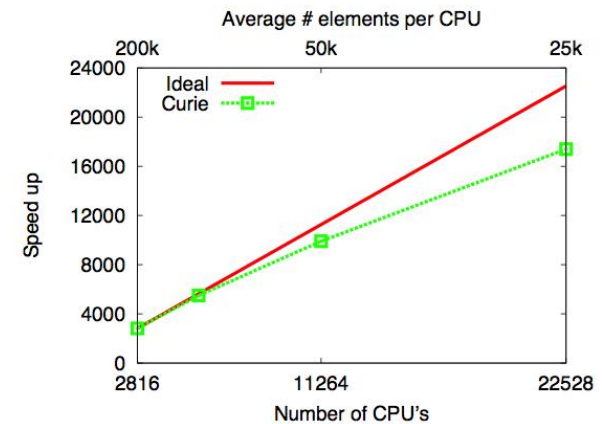
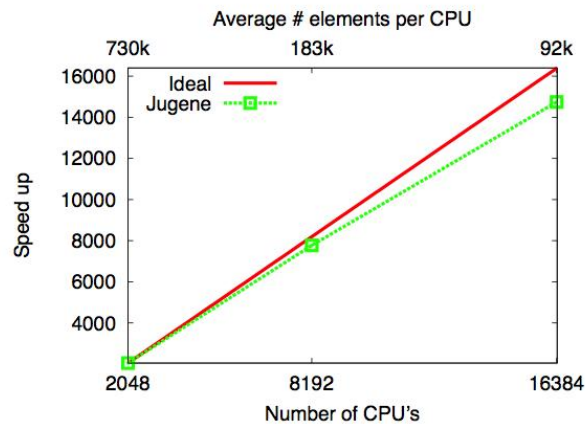
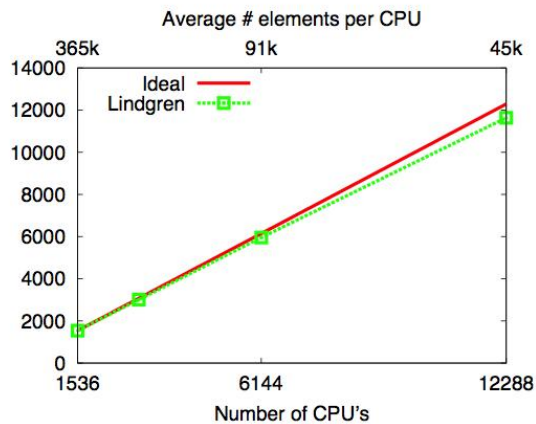
Example: Strong scaling of Alya code

Speedup for implicit Navier-Stokes solver, 550M element mesh
Available on-line: <https://wikiar2012.bsc.es>

Lindgren - Cray XE6 - Sweden

Jugene - Blue Gene/P - Germany

Curie - BullX - France



Example: Strong scaling of Code_Saturne

Optimization of Code_Saturne for Petascale simulations

C. Moulinec et al. , PRACE white paper, 2012

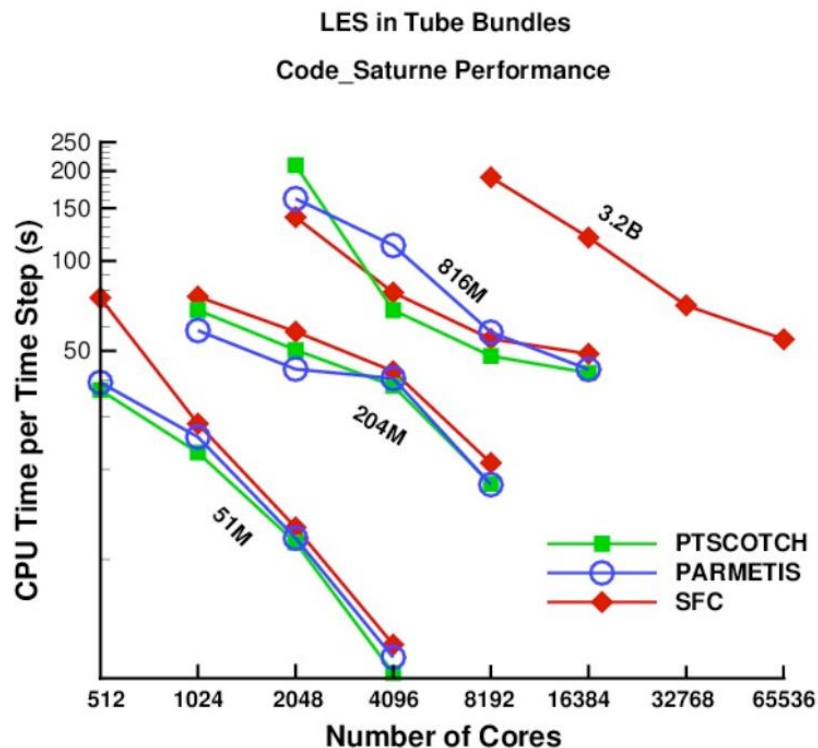
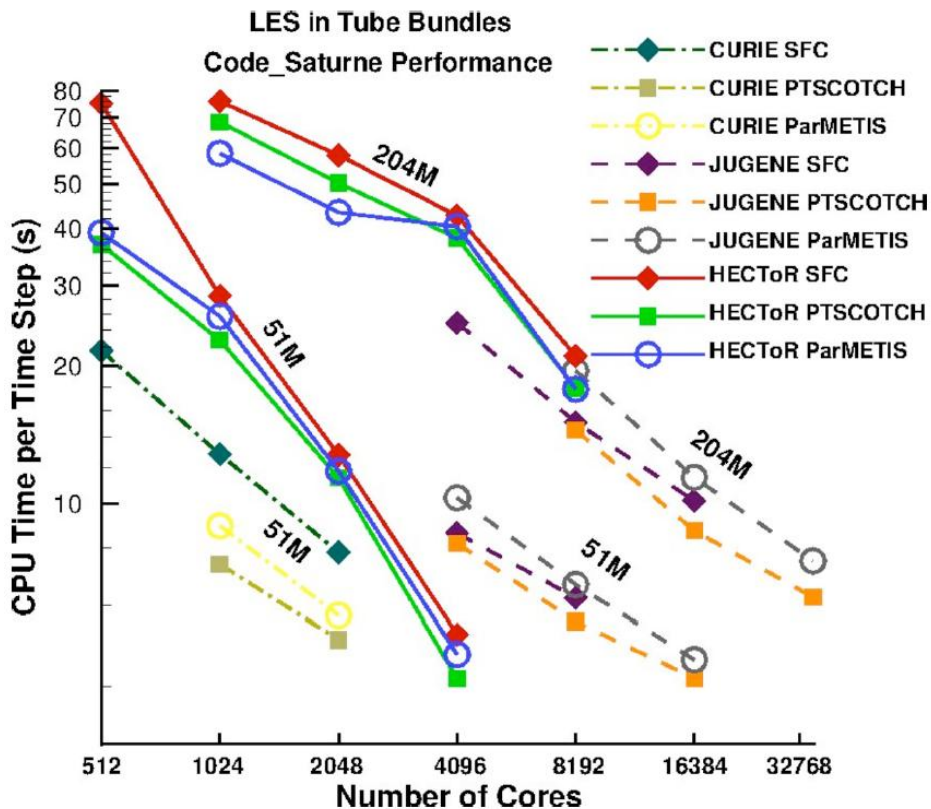


Fig 2 (a): Comparative performance of Code_Saturne on JUGENE, CURIE and HECToR (Cray XE6) for the 51M case and the 204M case for JUGENE and HECToR. (b): CPU time per time step in seconds required by Code_Saturne for the 51M, 204M, 816M and 3.2B cases, when PT-SCOTCH, ParMETIS and SFC are used as partitioners.

Example: Strong scaling of OpenFOAM



Current bottlenecks in the scalability of OpenFOAM
on massively parallel clusters , M. Culpo, PRACE white paper.

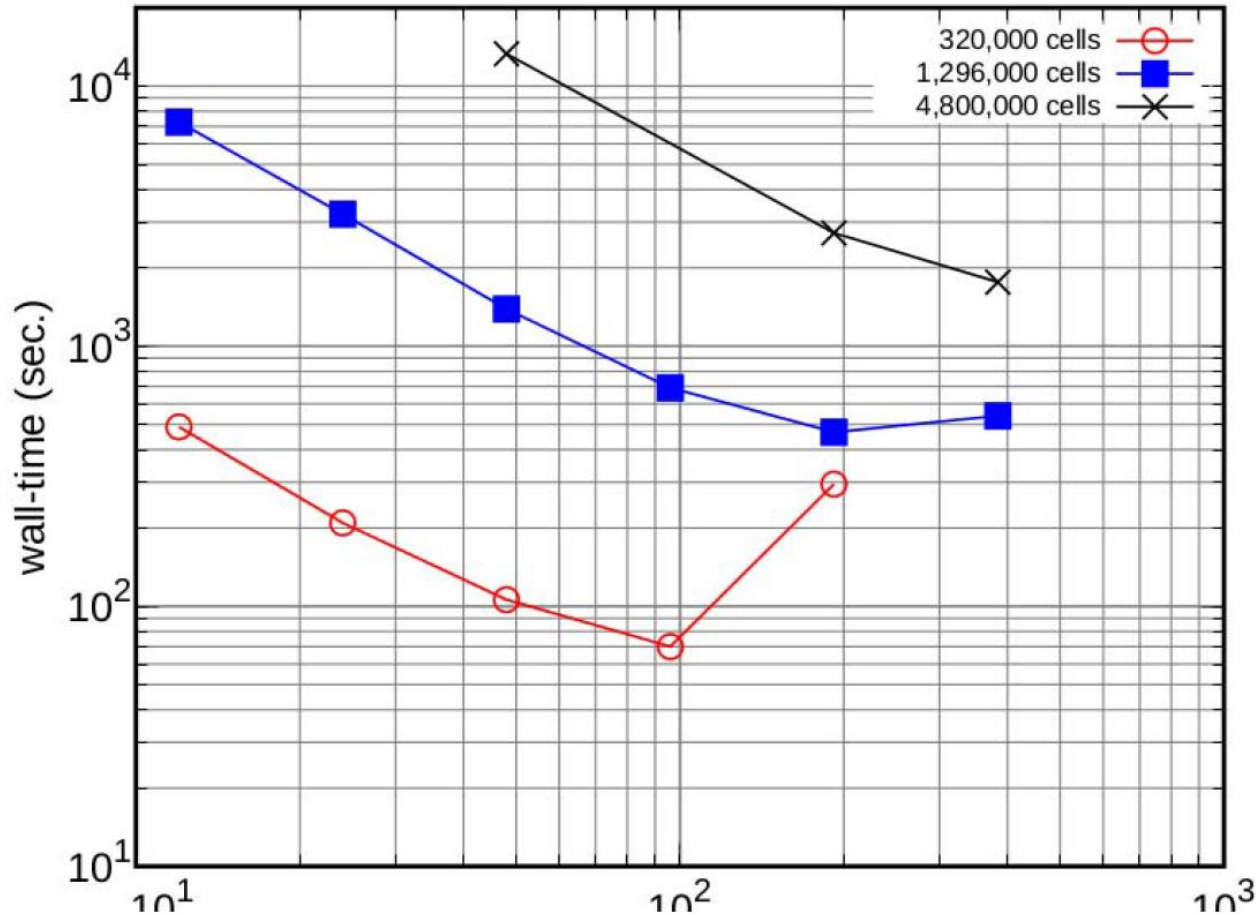


Figure 2 : : Strong scaling results for the droplet splash test case. Three different meshes of increasing size have been used. The scalability behavior resembles that of the lid driven cavity flow benchmark

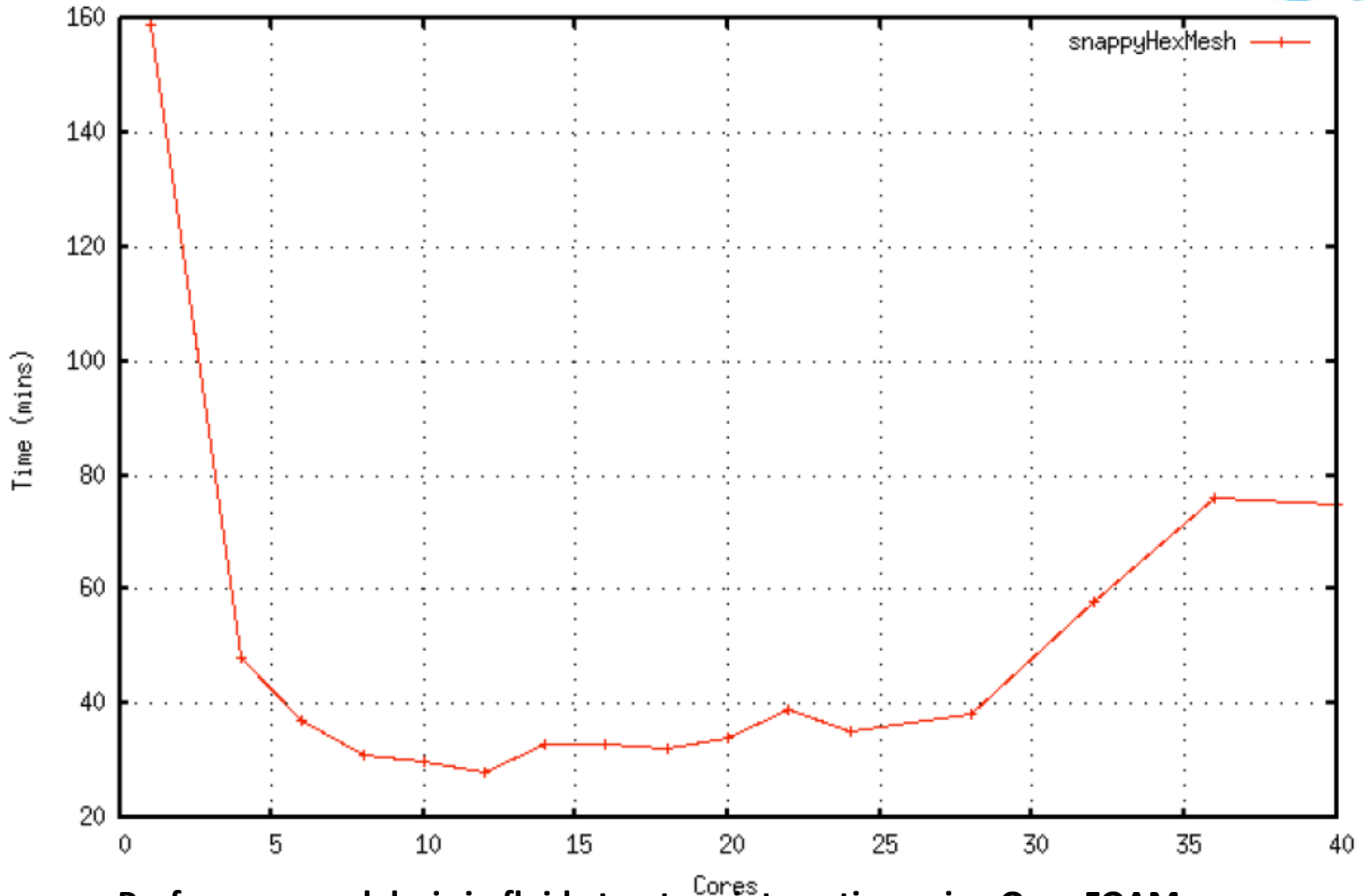
Example: Weak scaling of Elmer (FETI)



#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.

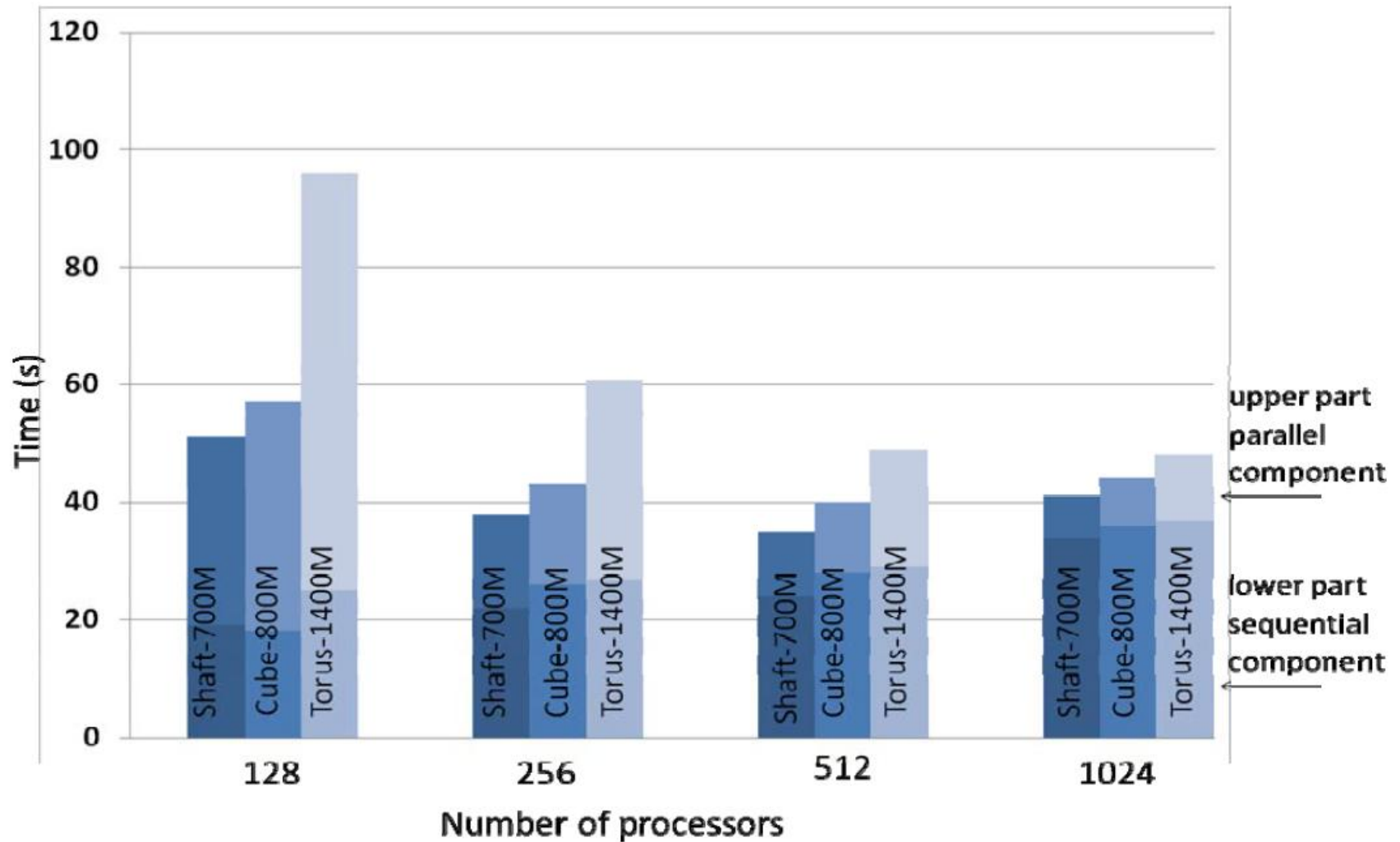
Example: Strong scaling of SnappyHexMesh



Performance analysis in fluid-structure interaction using OpenFOAM

M. Moylesa et al., PRACE white paper, 2012

Parallel mesh generation: performance



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

Conclusions



- ➊ **FEM** and **FVM** are the dominant methods in computational engineering
- ➋ The **unstructured meshing** & local PDEs result to **sparse linear systems** that determine many aspects in the solution process
 - Two of the “seven dwarfs of HPC”
- ➌ Many capable parallel software under Open Source
 - Users may still need to build their own workflows
- ➍ The state-of-art of parallelization varies between the steps
 - Solution of linear systems has a great number of good solutions
 - Preprocessing steps usually done at least partly in serial
 - Excellent software for parallel visualization (next presentation)
- ➎ One should pay careful attention to the algorithmic and parallel scalability of the software
 - Multilevel algorithms