Elmer

Parallel Computing

ElmerTeam

CSC – IT Center for Science Ltd.

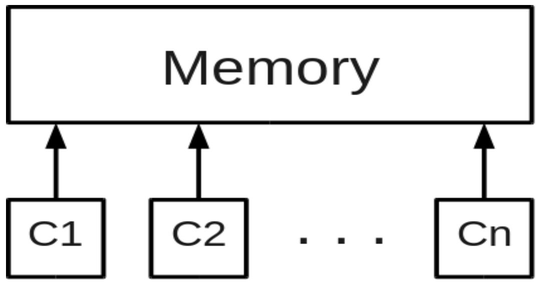
CSC, April 2013

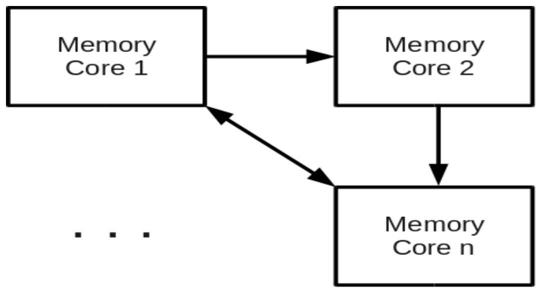
Parallel computing concepts

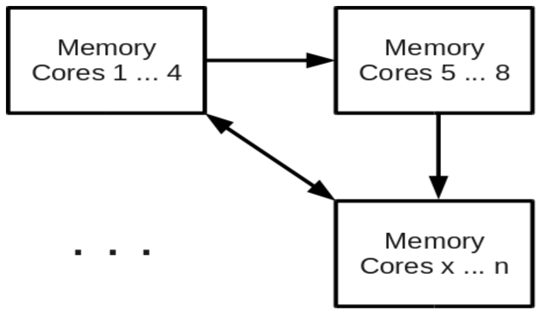
- Parallel computation means executing tasks concurrently
  - A task encapsulates a sequential program and local data, and its interface to its environment
  - Data of those other tasks is remote
- Data dependency means that the computation of one task requires data from an another task in order to proceed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

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





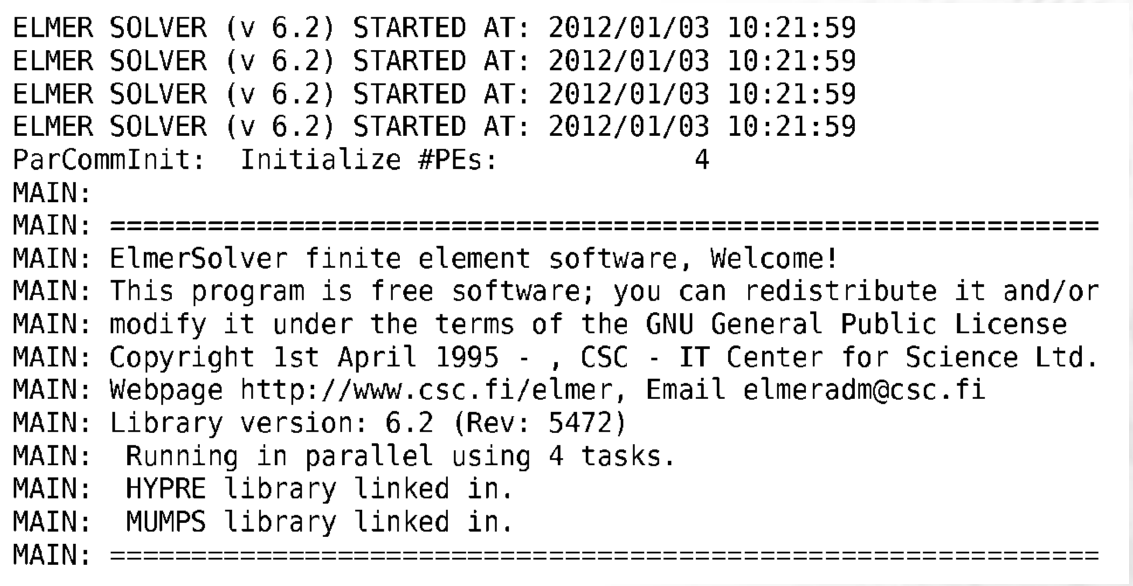


Parallel programming models

- Message passing (OpenMPI)
  - 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

Execution model

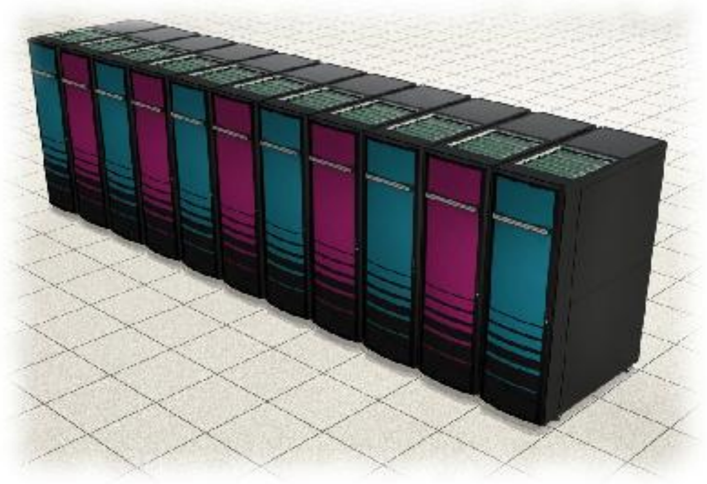
- Parallel program is launched as a set of independent, identical processes
  – The same program code and instructions
  – Can reside in different computation nodes
  – Or even in different computers



```
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 parallel scaling

Weak Scaling

- Increasing the size of the problem
- Ideally, execution time remains constant, when number of cores increases in proportion to the problem size
- Weak scaling is usually limited by the algorithmic scalability

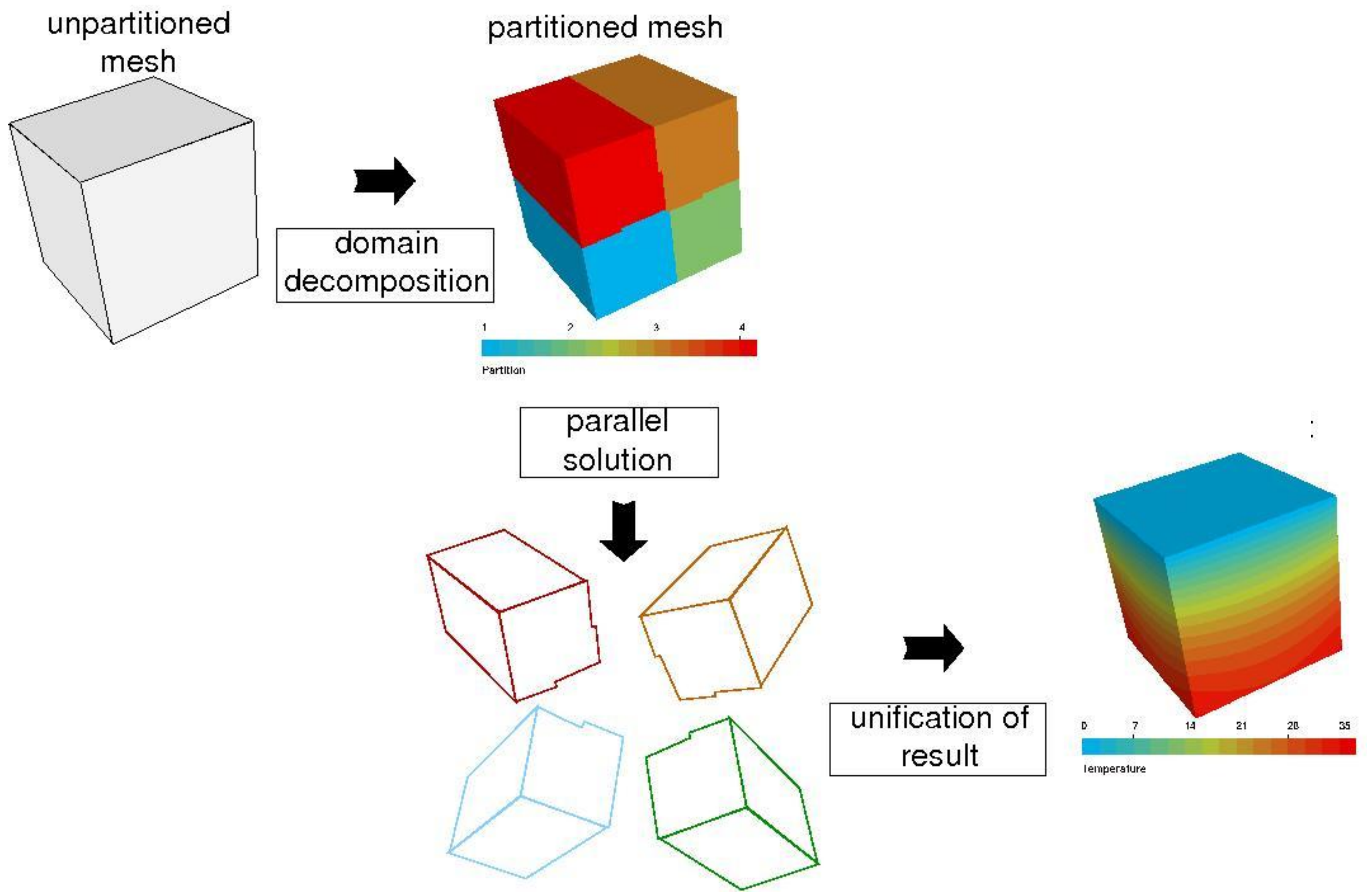
Strong Scaling

- The size of the problem remains constant
- Ideally, execution time decreases in proportion to the increase in the number of cores
- Strong scaling is a better indication of the parallel communication bottle-necks

Parallel computing with Elmer

- Preprocessing
  - Additional pre-processing step for mesh partitioning using ElmerGrid
- Solution
  - Every domain is running its own ElmerSolver_mpi
    - Communication between processes
- Postprocessing
  - Recombination of results to ElmerPost output or
  - Postprocessing with Paraview

Parallel workflow, example



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]

Mesh partitioning with ElmerGrid

- ElmerGrid may start from any serial mesh format that it supports
  - Serial mesh → ElmerGrid → parallel mesh
- Syntax with existing Elmer mesh
  - ElmerGrid 2 2 serialmesh [partoption]
- Syntax with Gmsh mesh
  - ElmerGrid 9 2 serialmesh.msh [partoption]

```
****************** 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

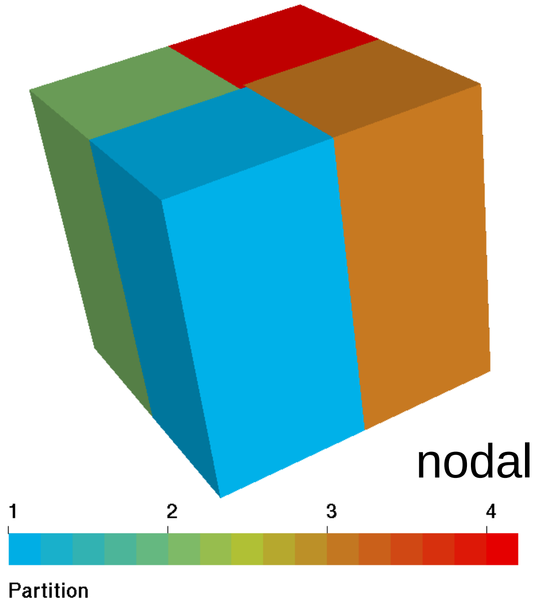
```
The following keywords are related only to the parallel Elmer computations.
-partition int[4]     : the mesh will be partitioned in main directions
-partorder real[3]    : in the above method, the direction of the ordering
-metis int[2]         : the mesh will be partitioned with Metis
-halo                 : 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 partitioning with ElmerGrid

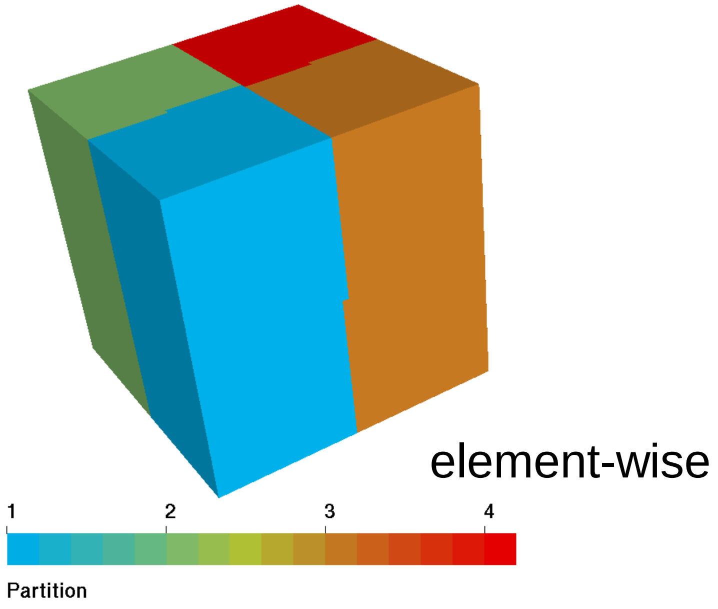
- Two strategies for mesh partitioning
- Recursive division by cartesian directions: -partition
  - Simple shapes (ideal for quads and hexas)
  - Choise between partitioning of nodes or elements first
- Metis graph partitioning library: -metis
  - Generic strategy
  - Includes five different graph partitioning routines from Metis

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`



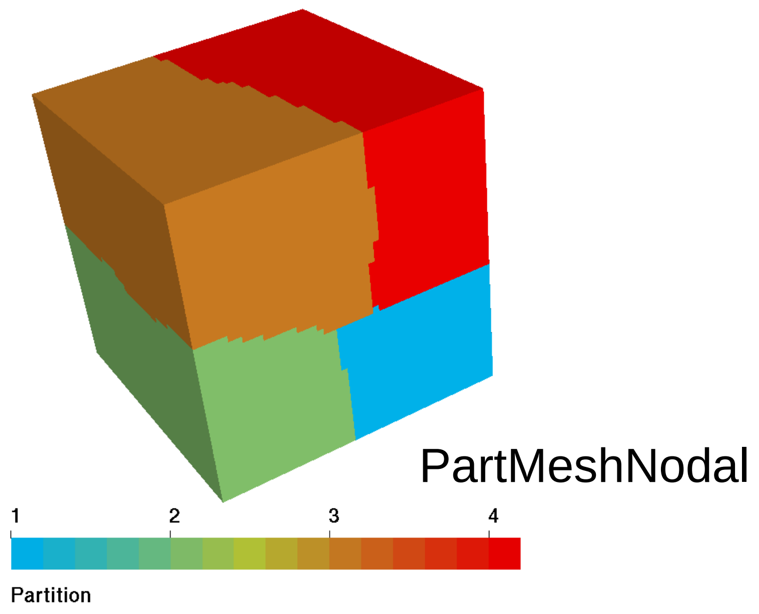
`-partition 2 2 1 0`

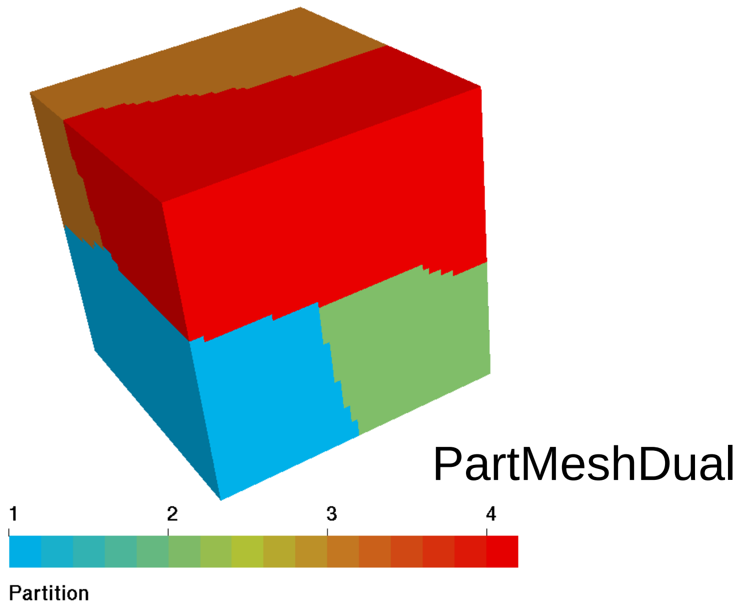


`-partition 2 2 1 1`

ElmerGrid partitioning by Metis

- Using Metis library
  - `ElmerGrid 1 2 meshdir -metis Np Nm`



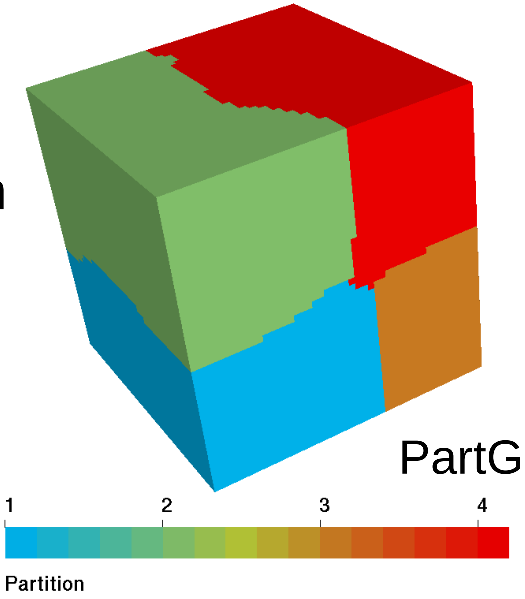


`-metis 4 0`

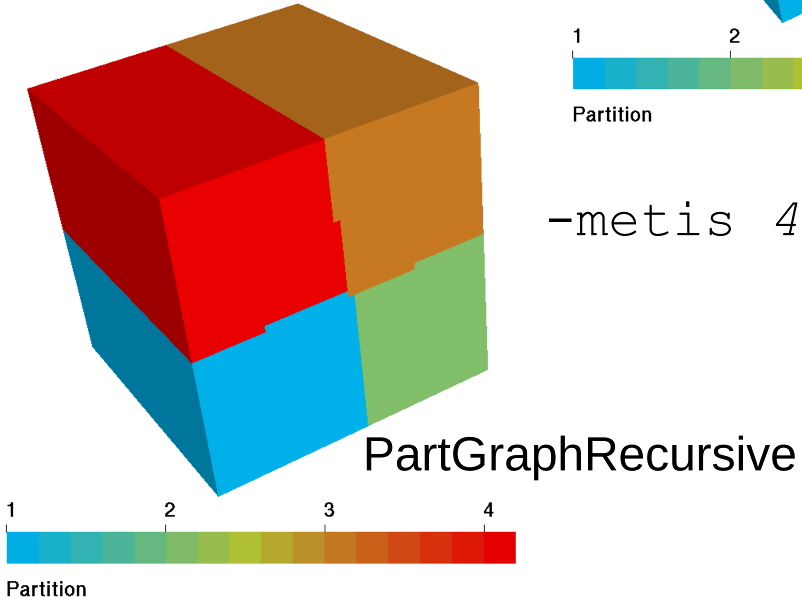
`-metis 4 1`

ElmerGrid partitioning by Metis, continued

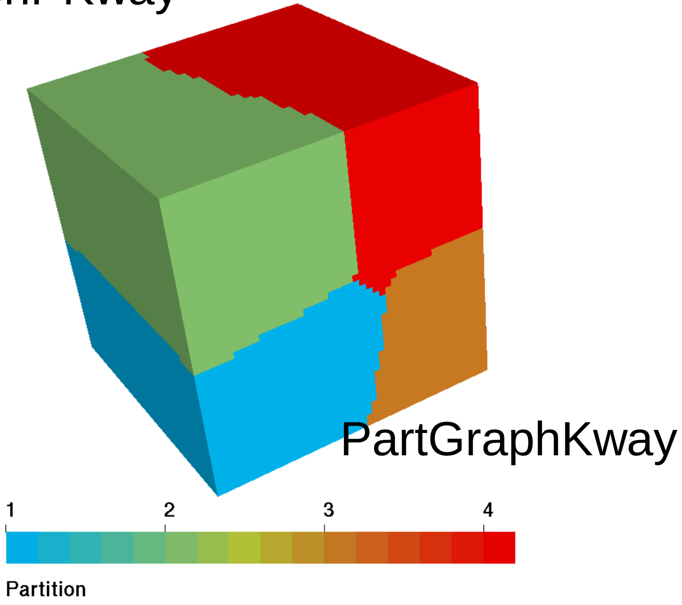
Enforce dual graph with these algrorithms with `-partdual`



`-metis 4 4`



`-metis 4 2`



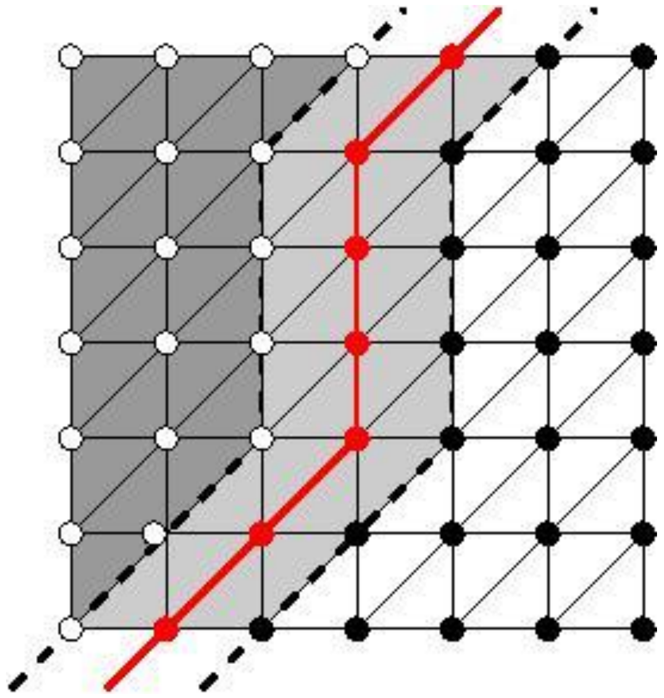
`-metis 4 3`

Accounting for halo elements

- Required when information on neighbouring elements in needed
  - Puts “ghost cell” on each side of the partition boundary.
  - e.g. Disconstinuous Galerkin

- Syntax:

```
ElmerGrid 2 2 meshdir
-metis Np Nm -halo
```



Enforcing periodicity

- Periodic nodes must be in the same partition as they introduce new complex connections
- ElmerGrid can ensure that nodes are on the same partition for simple conforming meshes
- Periodicity is given by 0/1 flag in each direction
- Example:

```
ElmerGrid 2 2 meshdir -metis 4 -periodic 1 0 0
```

Parallellism in Elmer 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
  - FETI: still under development
  - Preconditioners
    - ILUn performed block-wise
    - Diagonal and Vanka exactly the same in parallel
    - GMG also as a preconditioner

Parallel external libraries for Elmer

- MUMPS
  - Direct solver that may work when averything else fails
- Hypre
  - Large selection of methods
  - Algebraic multigrid: Boomer MG
  - Parallel ILU preconditioning
  - Approximate inverse preconditioning: Parasails
- Trilinos
  - Interface to ML multigrid solver
    implemented by Jonas Thies, Univ. of Uppsala
  - ML often provides the fastest linear solver strategy!

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

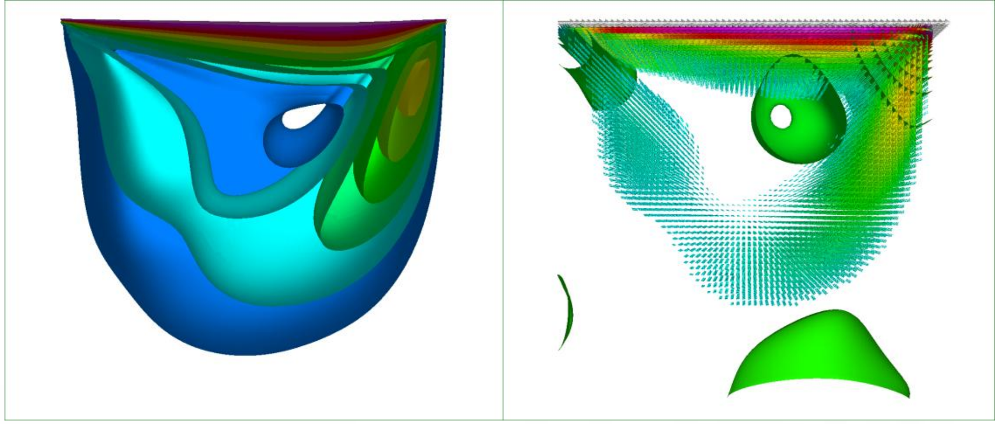
- Partitioned mesh files
- ELMERSOLVER_STARTINFO is always needed to define the command file (.sif)
- Execution with `mpirun -np N ElmerSolver_mpi`
- Calling convention is platform dependent
- Writes results to *N* files

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

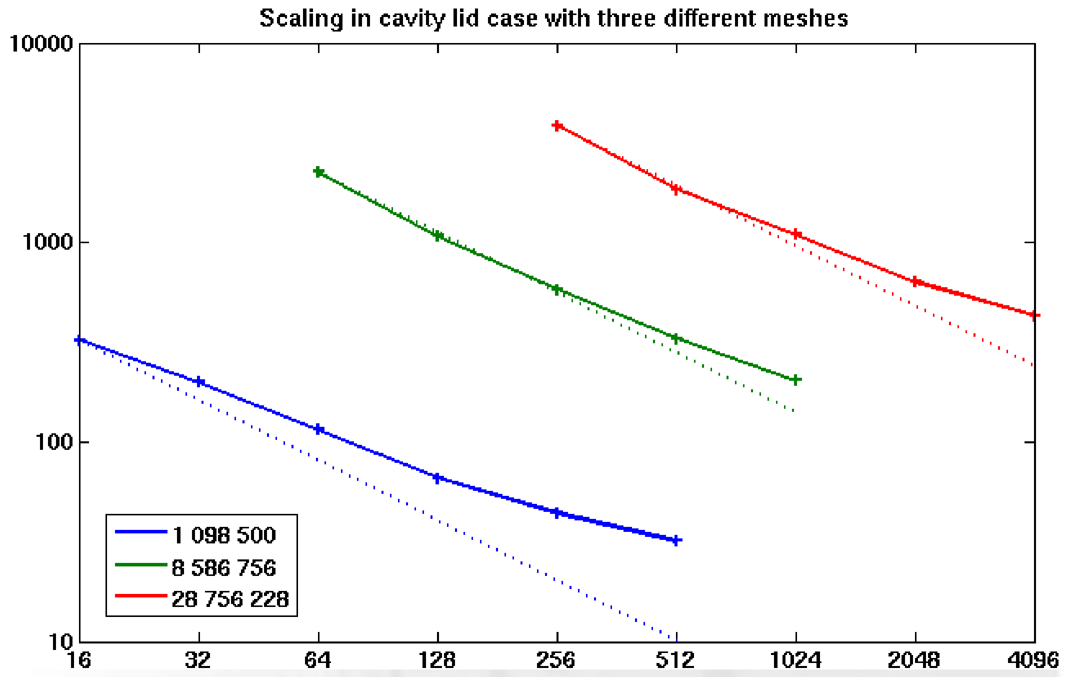
Parallel performance

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



Simulation Juha Ruokolainen CSC, visualization Matti Gröhn, CSC .

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



Parallel computation, example

- Poisson equation with 100 M dofs
- Mesh created from coarse mesh using mesh multiplication
- Solution with geometric multigrid (GMG) utilizing the different mesh levels
- Very good speedup up to ~1000 partitions

#procs	T(P) / s	T(P)/T(2P)
272	279	-
544	152	1.84
1088	76	2.00
2176	50	1.52

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

Failure of standard methods, example

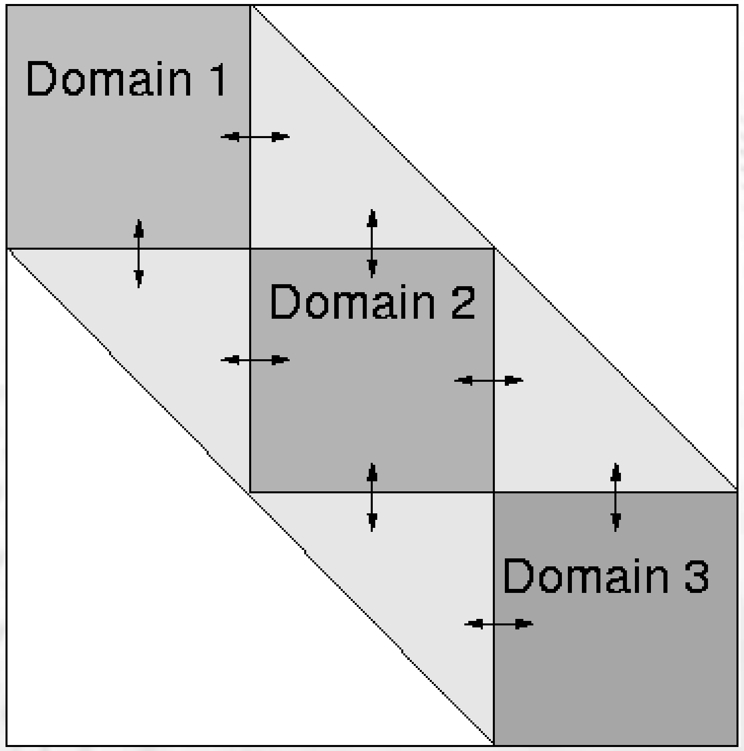
- Already linear elasticity equation may pose problems for parallel solution
- Case of linear elasticity with 500,000 unknowns
- Many standard methods fail to converge, or scale sub-optimally
  - Need for methods with better robustness & scalability

Method \ T (s)	1	2	4	16	32
Umfpack	53533				
Pardiso	290	175	105	80	65
Mumps		285	190	135	86
BiCG+diag	1270	750	450	225	180
BiCG+ILU(1)	1690	1450	X	580	X
Hypre-BiCG+Parasails		505	295	145	110
Hypre-BiCG+ILU(0)		X	X	506	X

Note: Calculations performed on vuori.csc.fi cluster in 2010.
The solvers may have improved in performance since

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



Hybridization in Elmer

- Preliminary work on Open MP + MPI hybridization
- Open MP pragmas have been added for
  - Matrix assembly
  - Sparse matrix-vector multiplication
- The current implementation is efficient for
  - iterative Krylov methods with
  - diagonal or Vanka as preconditioner
- Scaling within one CPU is excellent
- Hybridization will become increasingly important when the number of cores increase

#threds	T(s)
1	341
2	130
4	69
8	47
16	38
32	27

Table: Navier-Stokes equation solved with BiCGStabl(4) and Vanka preconditioning on a HP ProLiant DL580 G7 with quad-core Intel Xeon processors.

Parallel postprocessing with ElmerPost

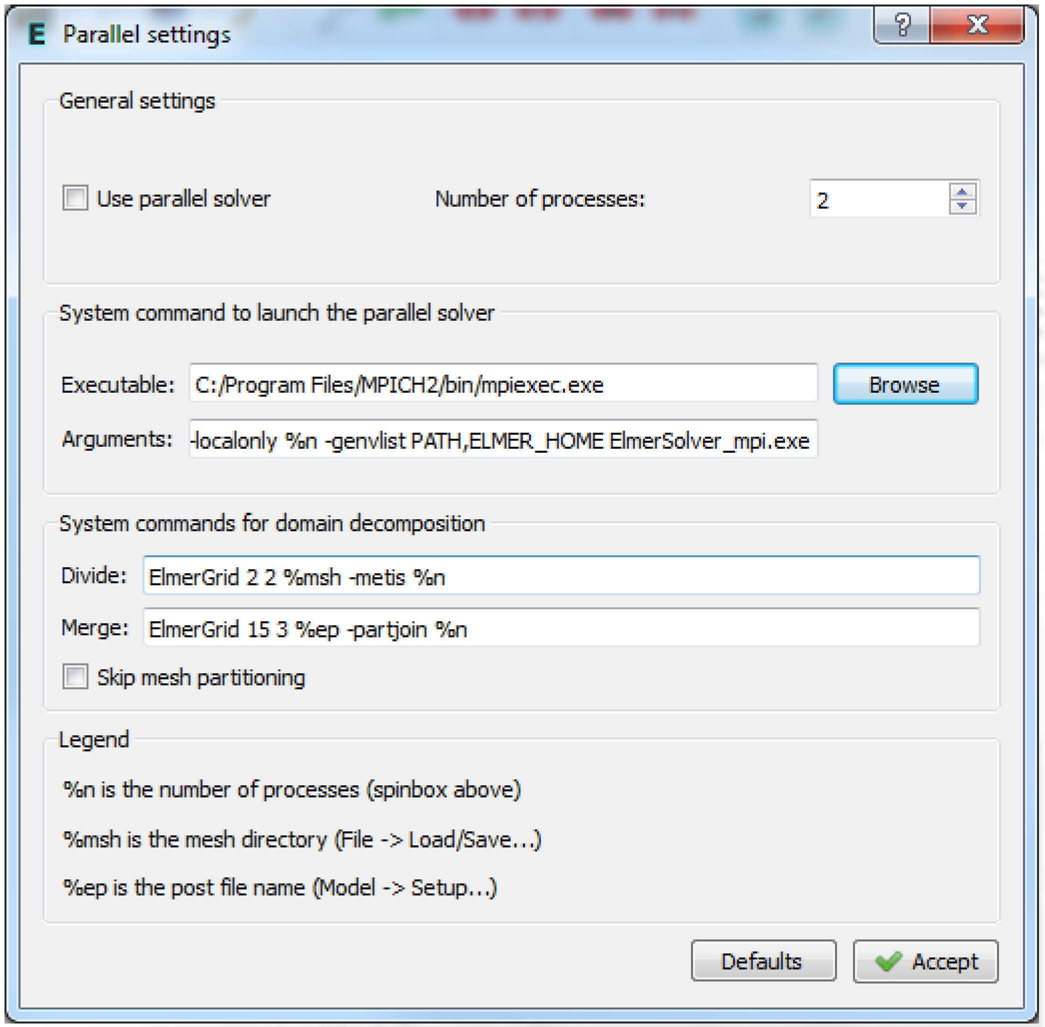
- ElmerSolver writes results in partitionwise.
  - `name.0.ep`
  - `name.1.ep`
  - `...`
  - `name.(N-1).ep`
- ElmerGrid is used to fuse files together into a single file called name.ep
  - `ElmerGrid 15 3 dirname`
- Option for choosing timestep
  - `-saveinterval Nstart Nfin Nstep`

Parallel postprocessing using Paraview

- Use ResultOutputSolver to save data to `.vtu` files
- The operation is exactly the same for parallel 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



General settings

[ ] Use parallel solver   Number of processes: 2

System command to launch the parallel solver

Executable: C:/Program Files/MPICH2/bin/mpiexec.exe

Arguments: -localonly %n -genvlist PATH,ELMER_HOME ElmerSolver_mpi.exe

System commands for domain decomposition

Divide: ElmerGrid 2 2 %msh -metis %n

Merge: ElmerGrid 15 3 %ep -partjoin %n

[ ] Skip mesh partitioning

Legend

%n is the number of processes (spinbox above)

%msh is the mesh directory (File -> Load/Save...)

%ep is the post file name (Model -> Setup...)