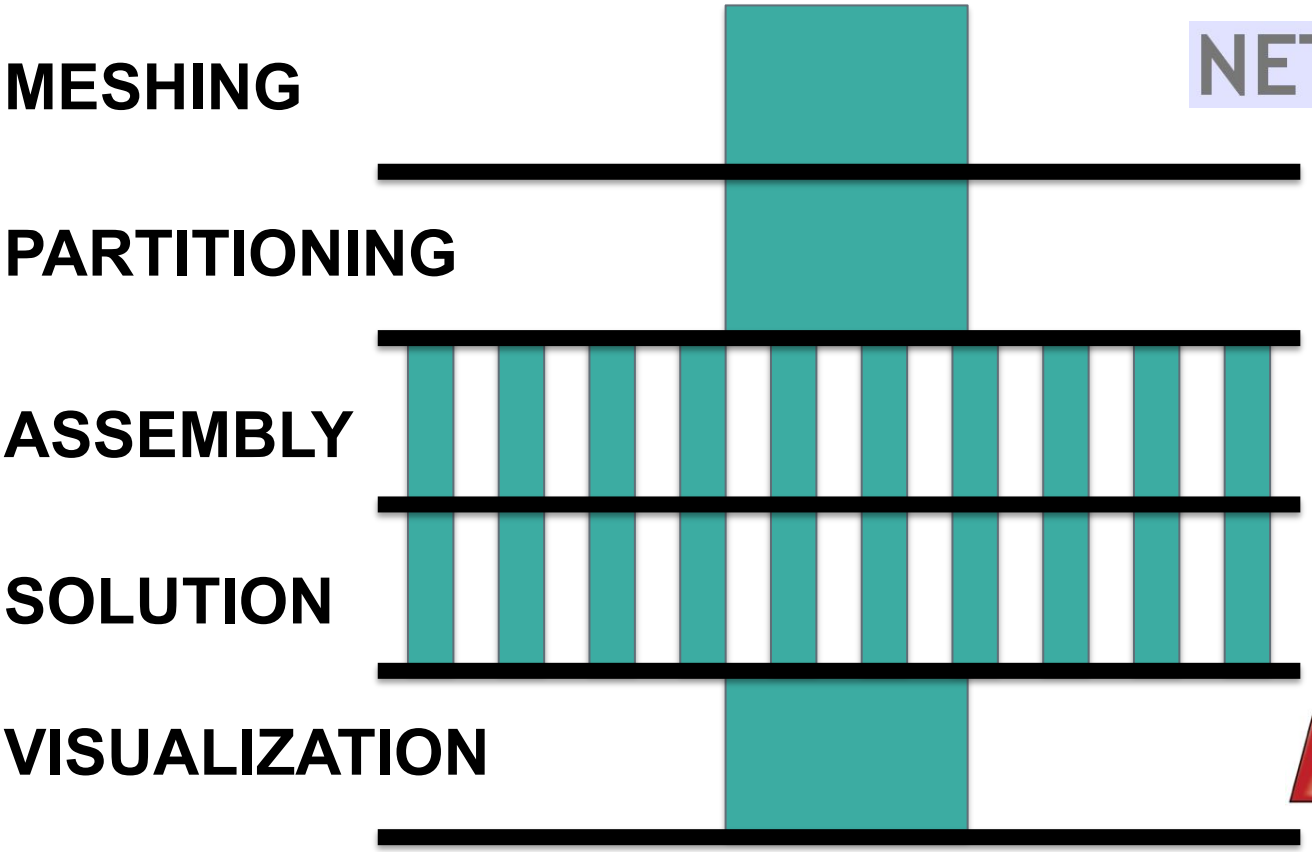
EGU Elmer/Ice Splinter Meeting 2014

Elmer/Ice new development

Elmer-team

Parallel concept of Elmer



NETGEN

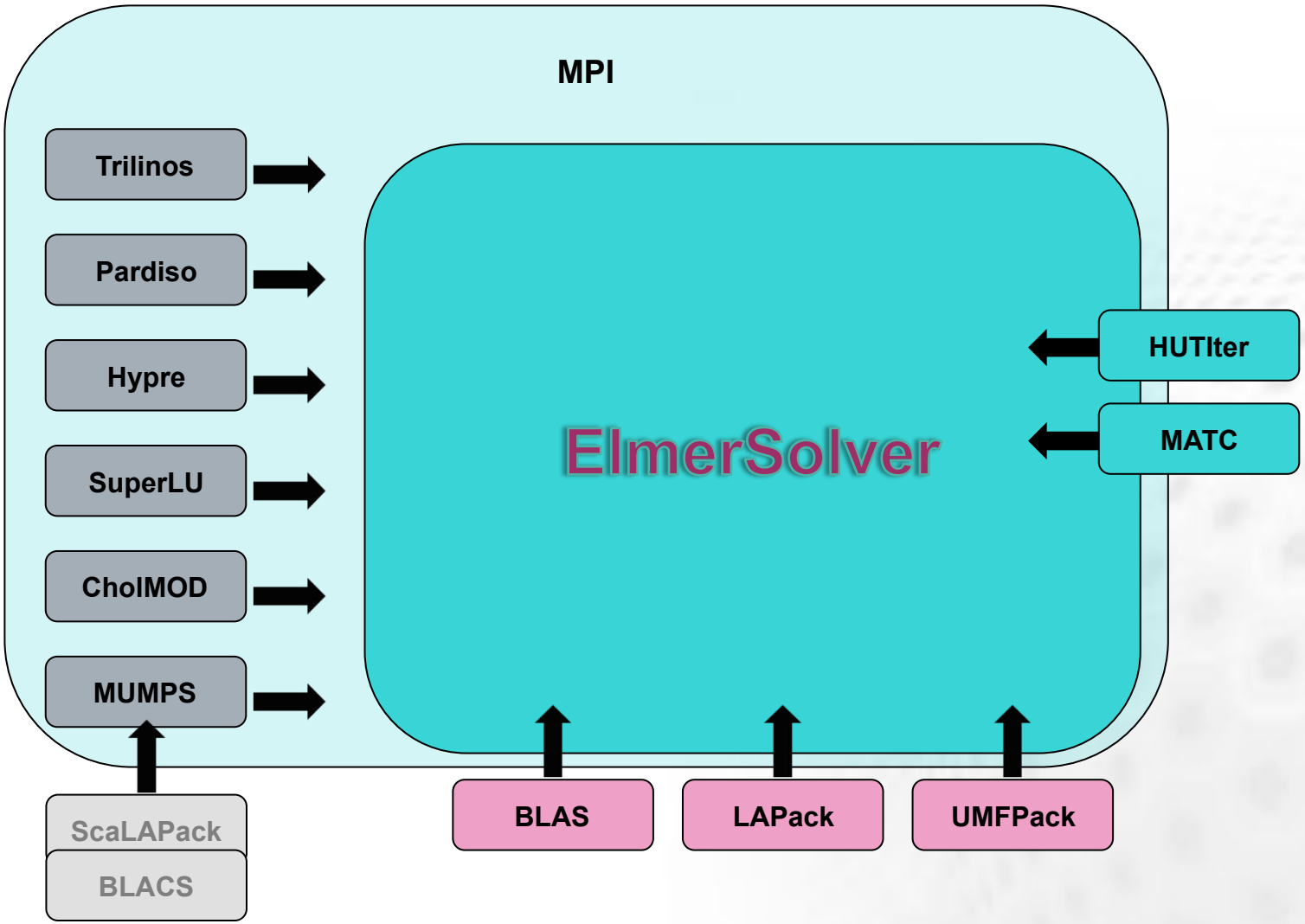
GMSH

METIS

Elmer

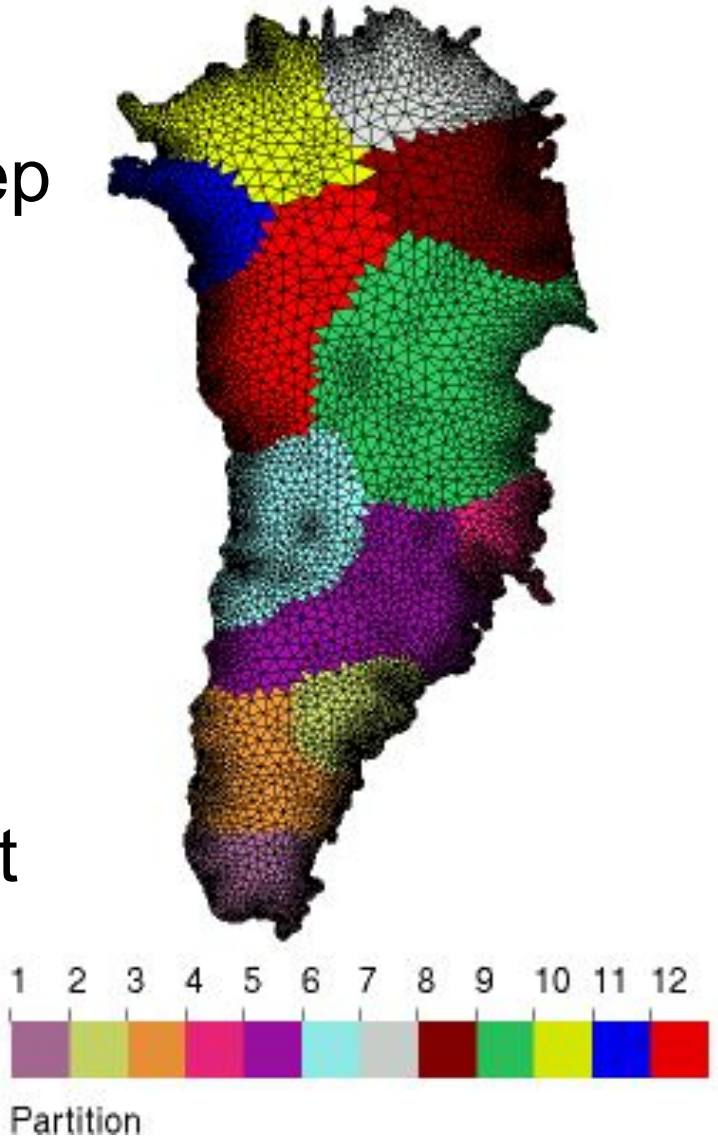
ParaView

Parallel concept of Elmer

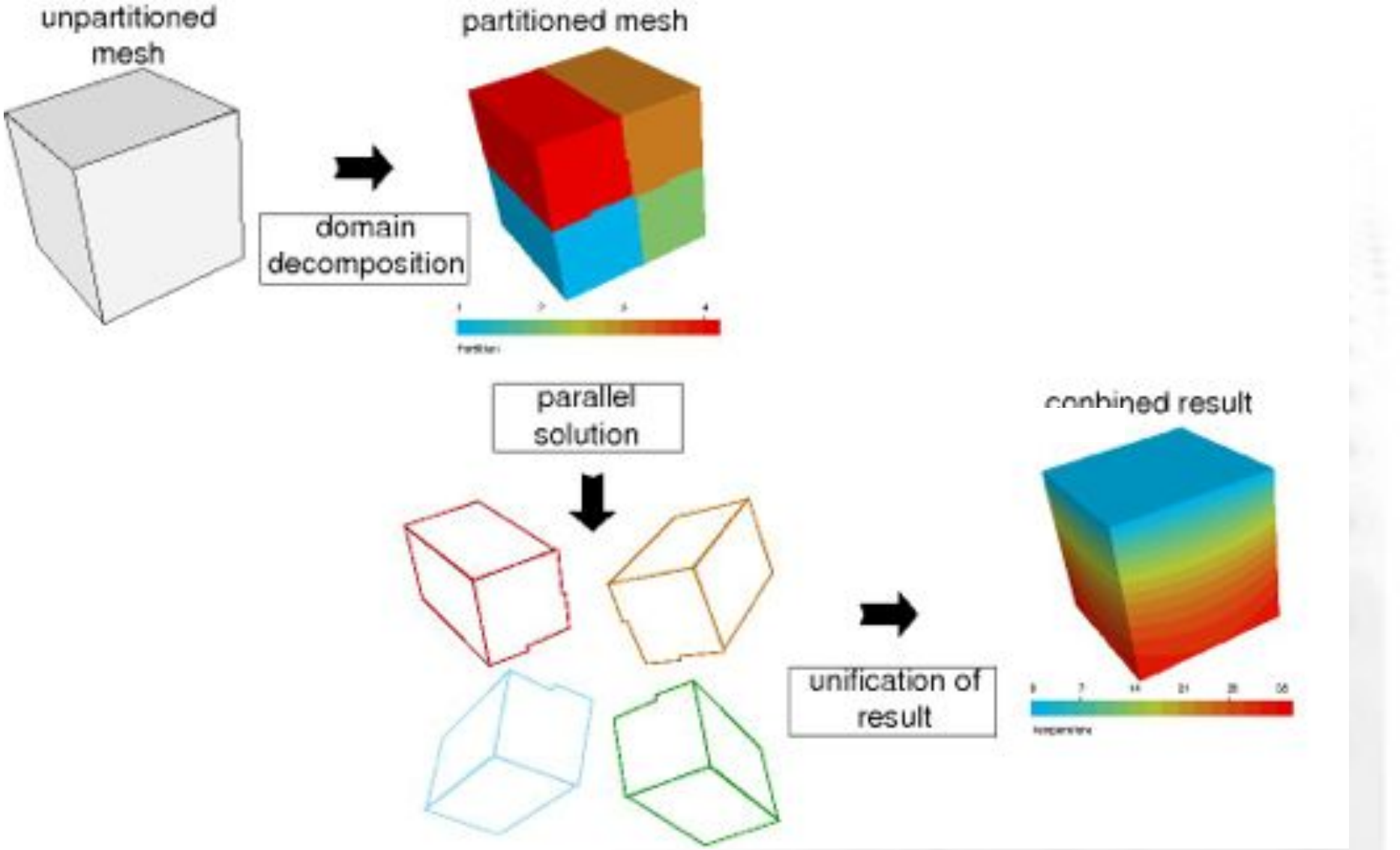


Parallel Concept of Elmer

- Domain decomposition
- Additional pre-processing step (splitting)
- Every domain is running its ”own” ElmerSolver
- Parallel process communication: Message Passing Interface (MPI)
- Re-combination of ElmerPost output



Parallel Concept of Elmer



Elmer parallel mesh

- Best way to partition:

  Serial mesh → `ElmerGrid` → parallel mesh

- General syntax:

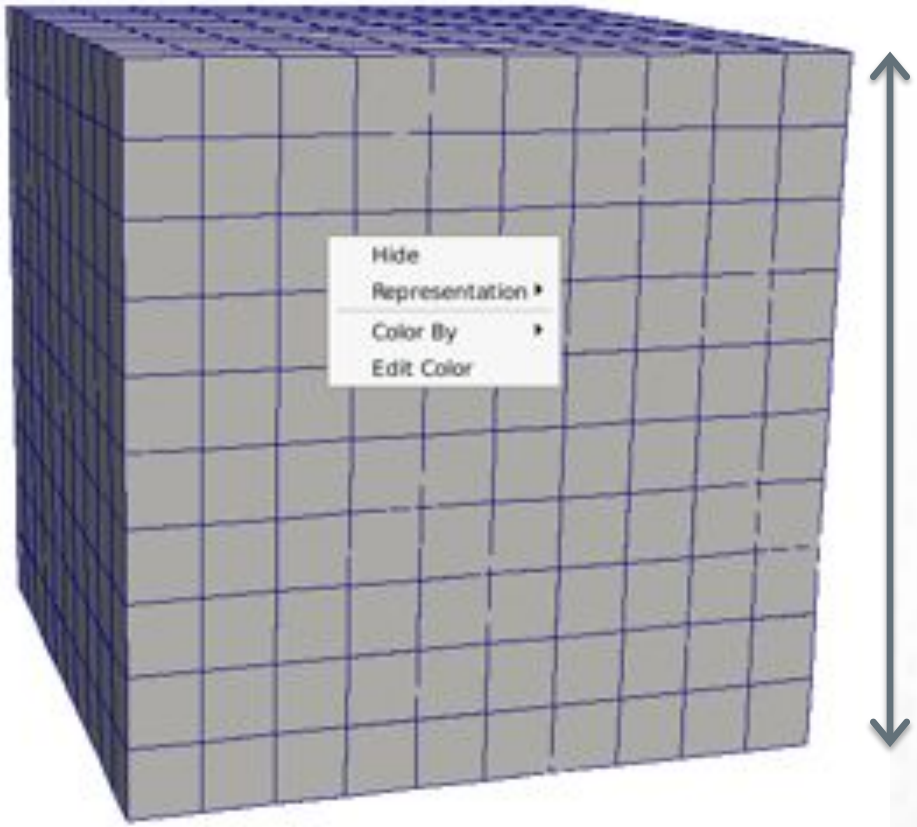
  `ElmerGrid 2 2 existing [`*`partoption`*`]`

- Principle 2 partitioning techniques:
  1. Along Cartesian axis (simple geometries/topologies)
  2. Using METIS library

Internal Extrusion

- Implemented as an internal strategy in Elmer (2013)
  - Juha, Peter & Rupert
- First partition a 2D mesh, then extrude into 3D
- Implemented also for partitioned meshes
  - Extruded lines belong to the same partition by construction!
- Deterministic, i.e. element and node numbering determined by the 2D mesh
  - Complexity: O(N)
- There are many problems of practical problems where the mesh extrusion of a initial 2D mesh provides a good solution
  - One such field is glaciology where glaciers are thin, yet the 2D approach is not always sufficient in accurary

Internal extrusion



`Extruded Mesh Levels = 11`

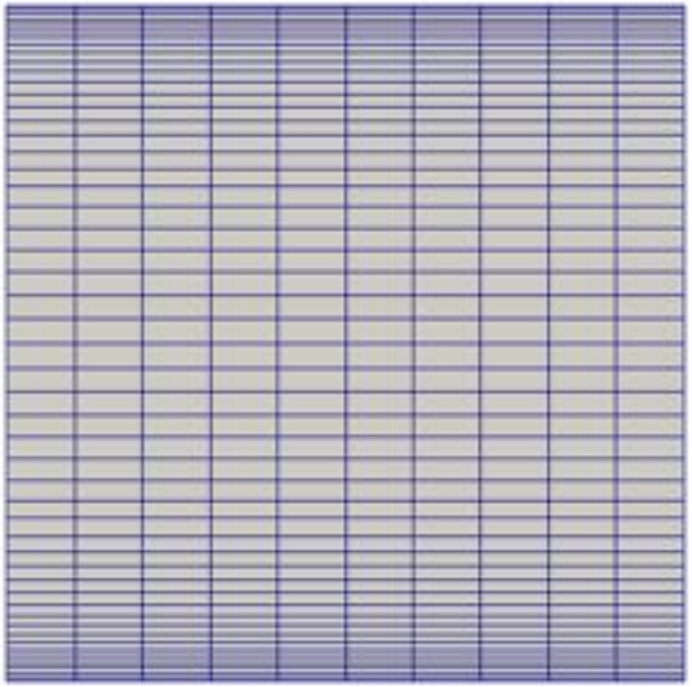
By default z in [0,1]

Internal extrusion

```
Extruded Mesh Levels = 11
Extruded Mesh Density = Variable Coordinate 1
  Real MATC "0.2+sin(pi*tx)"
```

Any functional dependence is ok as long as it is positive!

The optimal division is found iteratively using Gauss-Seidel type of iteration and large variations make the iterations converge slowly.



Internal Extrusion

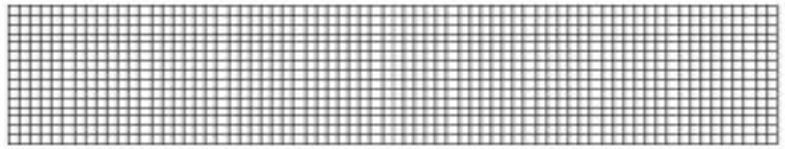
- **StructuredMesh Mapper** to impose geometry to a topological prism (3D)
- Define functions at bottom:

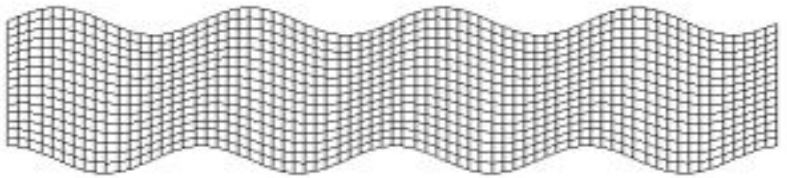
```
Bottom Surface = Variable
   "Coordinate 1"
    Real MATC "0.1*cos(5*tx)"
```

- And surface:

```
Bottom Surface = Variable
   "Coordinate 1"
    Real MATC "0.1*cos(5*tx)"
```

```
Solver 2
  Equation = "MapCoordinate"
  Procedure = "StructuredMeshMapper"
"StructuredMeshMapper"
  Active Coordinate = Integer 2
End
```





ElmerSolver parallel

- Different executable: `ElmerSolver_mpi`
- Depending on platform/MPI: `mpirun -np N`

```
> mpirun -np 6 ElmerSolver_mpi
```

- Needs information for different processes, which SIF to load: `ELMERSOLVER_STARTINFO`
- User defined functions/routines usually do not need special rewriting for MPI

ElmerSolver parallel

- Alternative pre-condtioner in Hypre:
  - ParaSails (sparse approximate inverse preconditioner):

```
Linear System Preconditioning = String
  "ParaSails"
```

  - BoomerAMG(Algebraic Multigrid):

```
Linear System Preconditioning = String
  "BoomerAMG"
```

ElmerSolver parallel

- Alternative Solver:
  - BoomerAMG(Algebraic Multigrid):

```
Linear System Solver = "Iterative"
Linear System Iterative Method =
   "BoomerAMG"
```

  - MUMPS (Multifrontal parallel direct solver):

```
Linear System Solver = Direct
Linear System Direct Method = Mumps
```

ElmerSolver parallel

- Already linear elasticity equation may pose problems for parallel solution
- Case of linear elasticity with 500,000 unknowns

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

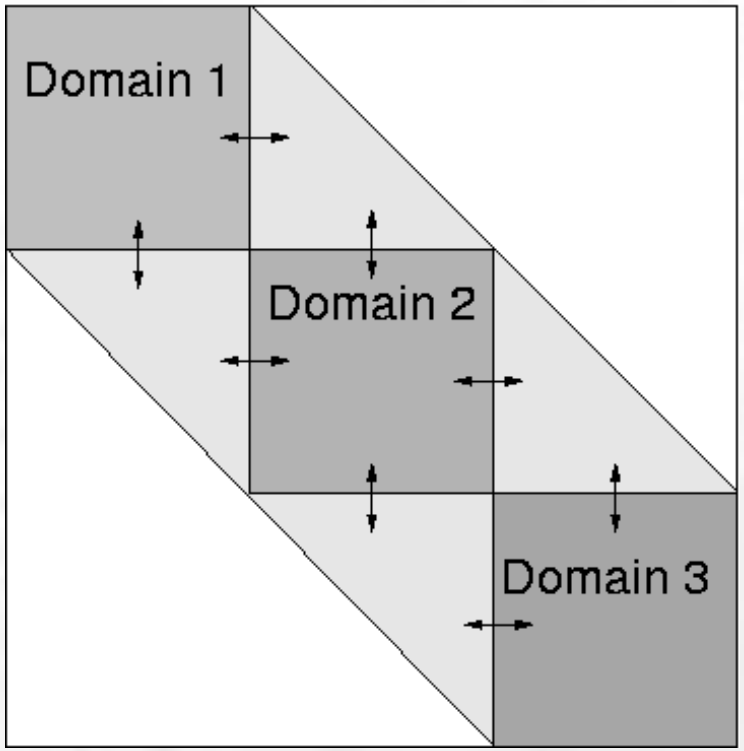
Note: Calculations performed on vuori.csc.fi cluster in 2010.
The solvers may have improved in performance since

ElmerSolver parallel

- Different behaviour of ILU preconditioner

  - Not available parts at partition boundaries
  - Sometimes work
  - If not, use Hypre:

```
Linear System Use Hypre
  = Logical True
```



Parallel postprocessing

- Elmer writes results in parallel

  `name.0.ep, name.1.ep, ...`

  `... , name.`*`(N-1)`*`.ep`

- ElmerPost: fusing into one file

  `ElmerGrid 15 3 name`

  fuses all timesteps (also non-existing) into a single file called `name.ep` (existing will be overwritten!)

  – Special option for only partial fuse:

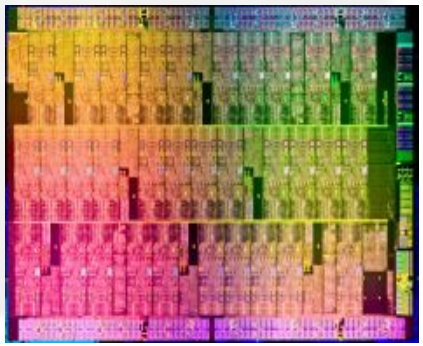
  `-saveinterval` *`start end step`*

Porting Elmer to MIC

- Porting work started Q2/12
- Focus to build ElmerSolver on a MIC
- Build process not entirely trivial
  - Tricks to fool automake
  - Manual editing of some resulting config-files
- ElmerSolver consistency tests
  - Initially 152 of 215 tests passed successfully
  - After a few hours of work 198 of 215 tests passed successfully

Porting Elmer to MIC

- MIC = Many Integrated Core
  - x86 –architecture
  - Up to 60 cores with 4-way HT
  - Single MIC core not as powerful as Xeon core
- ElmerSolver porting on MIC started on 2Q/ 2012
  - Sparse matrix vector products vectorized
  - Support for MKL Pardiso and SpDGEMV added
  - Some solvers modified to support OpenMP
  - Code is thread safe
    - all consistency tests passed



Elmer OpenMP status

- Internally OpenMP threading supported by
  - Solver API routines related to element assembly
  - Time integration routines
  - Sparse matrix vector products
  - Element assembly loop of some solvers (MagnetoDynamics2D, ShallowWaterNS, StatElecSolve, ThermoElectricSolver)
- Library support for OpenMP exists in
  - External BLAS routines
  - External LAPACK routines
  - Direct solvers such as Cholmod, SPQR and Pardiso

MIC: Assembly

Poisson model problem, 1M Hexahedral elements

2xXeon E5 2670, time(s)/nthr

Gauss points	1	16	32
Standard implementation			
8	52.01	5.90	3.02
64	378.46	23.84	21.26
Vectorized implementation			
8	11.40	1.40	0.76
64	42.53	2.76	2.73

Xeon Phi 7110, time(s)/nthr

Gauss points	1	60	240
Standard implementation			
8	698.42	11.88	7.04
64	5034.97	94.09	47.57
Vectorized implementation			
8	167.10	3.04	2.37
64	566.46	9.19	6.36

MIC: CG Performance

- Implemented standard conjugate gradient (CG)in a NUMA aware way
  - Distribute matrix row wise to different threads
  - Construct the whole iteration loop as single parallel region (avoid fork/join overhead)
  - Use local temporal vectors
  - Align accesses to global work vectors on 64-byte boundaries
- Compiler: use **`ASSUME_ALIGNED`** and **`SIMD`** pragmas
- Reduction inside parallel region: **`ATOMIC`** and **`BARRIER`**

MIC: CG Performance

2xXeon E5 2670, time(s)/nthr

Problem	1	16	32
Standard implementation			
3Delas_small	1,13	0,25	0,33
3Delas_med	8,59	2,51	2,67
3Delas_large	36,34	10,39	10,72
NUMA aware implementation			
3Delas_small	1,15	0,20	0,07
3Delas_med	8,83	2,27	1,28
3Delas_large	36,49	10,76	5,69

Xeon Phi 7110, time(s)/nthr

Problem	1	60	240
Standard implementation			
3Delas_small	10,23	3,22	7,46
3Delas_med	72,53	6,17	11,24
3Delas_large	303,4	9,87	15,85
NUMA aware implementation			
3Delas_small	10,19	0,35	0,59
3Delas_med	73,12	1,86	1,69
3Delas_large	306,2	7,14	4,51

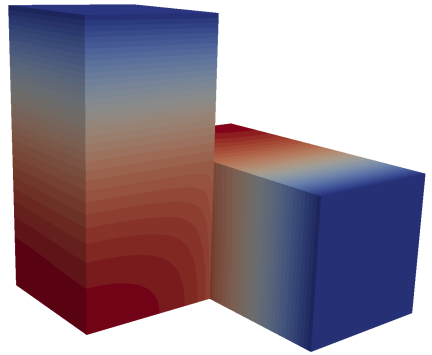
Xeon Phi & Intel MKL Pardiso

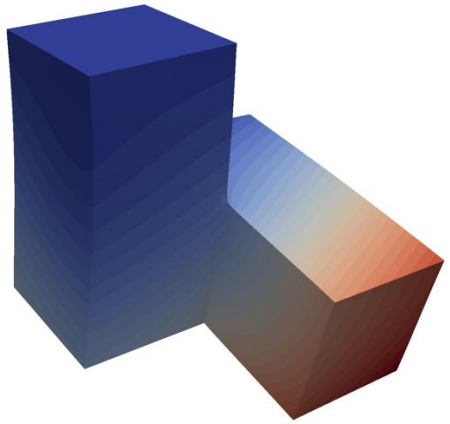
- Pardiso is one of the most advanced multifrontal direct solvers
- MKL version is multithreaded for Xeon Phi

Case	Time(s) nt=1	nt=60	nt=120	Speedup nt=60	nt=120
P1	10.93	3.82	4.38	2.86	2.50
P2	67.86	12.36	12.90	5.49	5.26
P3	429.55	49.02	48.23	8.76	8.91
E1	12.92	3.14	3.50	4.12	3.70
E2	169.91	18.63	19.28	9.12	8.81
E3	1492.79	86.46	79.28	17.26	18.83

Test problems for multithreading

Equation	Case	n	nz(A)	nz(A)/n
Poisson	P1	35,721	896,761	25.1
	P2	105,300	2,702,656	25.6
	P3	291,060	7,580,368	26.0
Elasticity	E1	24,843	1,786,545	71.9
	E2	107,163	8,070,849	75.3
	E3	315,900	23,323,904	77.0





Mikko Byckling, *Solving sparse linear systems in a many-core environment*, Sparse Days Meeting 2013, CERFACS, Toulouse.

Thank you!

Numerics

- Saddle-point system - stabilization:

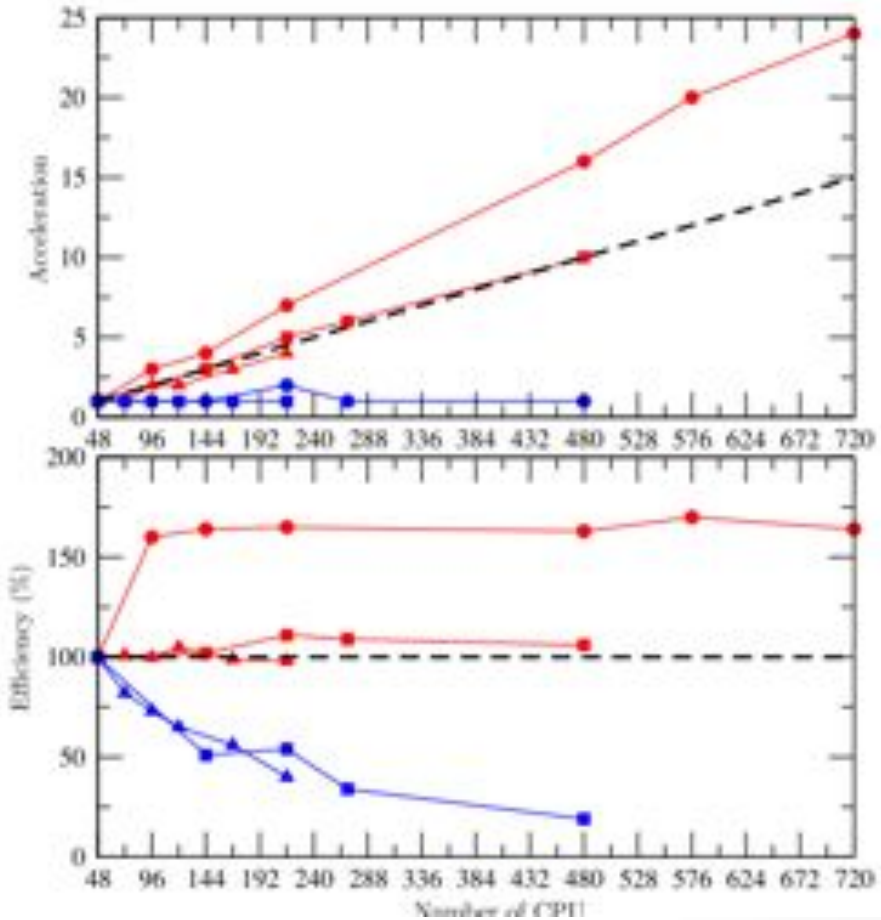
$$\begin{pmatrix} A & B^{\mathrm{T}} \\ B & C \end{pmatrix} \cdot \begin{pmatrix} v \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

- Bad condition number – direct solver
- Block pre-conditioner:

$$P = \begin{pmatrix} A & B^{\mathrm{T}} \\ 0 & M \end{pmatrix}$$

  – inverse, P^{-1}, still requires the exact solution of linear systems with A and M

Numerics



- Strong scaling
  - 10k, 100k, 1M
  - Comparison MUMPS and BPC
  - Super-linear scaling for BPC

Numerics

Massive parallel computing @Fabien Gillet-Chaulet, LGGE

1 900 000 nodes on 400 partitions
~7 000 000 dofs