

Parallel computing with Elmer

Antti Pursula
CSC

Elmer User Meeting
29.5.2006

<http://www.csc.fi/elmer>
antti.pursula@csc.fi

Contents

- General issues on parallel computing
- Parallel Elmer
- Use of iterative solvers in parallel computations
- Challenges in parallel computing with Elmer
- Setting up parallel runs in practice
- Example

General issues on parallel computing

- In parallel computing the numerical simulation problem is divided and sent to more than one processor for simultaneous execution
- The processors need to communicate during the run
 - ▷ For this purpose a message passing library is used
 - ▷ Elmer uses the Message-Passing Interface (MPI)
- Parallel computing is beneficial when a serial run takes too long to complete
- Parallel runs require tuning and testing
 - ⇒ Think carefully do you really need to go parallel!

Parallel Elmer

- Parallel version of Elmer has been implemented using MPI
- The mesh is divided into partitions with, e.g., ElmerGrid (Metis library & native implementation)
 - ▷ Elements are divided in partitions
 - ▷ Interface nodes are shared by partitions
- All Elmer solvers can be used in parallel computations (in principle)
- Parallel Elmer works with Unix/Linux systems
- Both matrix assembly and linear equation solving are performed in parallel
 - ▷ Parallelization helps also in reducing the time used in assembling the system matrix

Iterative solvers in parallel

- In parallel runs the linear system of equations is solved with iterative methods
- The iterative solvers do not need the whole matrix, instead for each partition the following are specified
 - ▷ matrix-vector multiplication
 - ▷ dot product for vectors
 - ▷ a norm for vectors
- Results of the above operations are communicated for the shared nodes
- Also the result is returned by partitions

The conjugate gradient method

- For example, the algorithm for conjugate gradient solver is shown on the right
- The matrix and right hand side entries on shared nodes need to be communicated before the iteration
- The operations (matrix-vector multiplication, dot product and norm) are computed locally and communicated on the shared nodes

$$r^0 = b^0 - Ax^0, \quad p^0 = r^0, \quad k = 0$$

do

$$\alpha_k = \frac{\langle r^k, r^k \rangle}{\langle p^k, Ap^k \rangle}$$

$$x^{k+1} = x^k + \alpha_k p^k$$

$$r^{k+1} = r^k - \alpha_k Ap^k$$

$$\beta_k = \frac{\langle r^{k+1}, r^{k+1} \rangle}{\langle r^k, r^k \rangle}$$

$$p^{k+1} = r^{k+1} + \beta_k p^k$$

$$k = k + 1$$

until $\|r^{k+1}\| < \epsilon$

Challenges in parallel runs

- Preconditioning is performed locally on each partition, except multigrid preconditioning
 - ▷ Reduces the effectiveness of preconditioning
 - ▷ Especially problematic with difficult equations (Navier-Stokes) or with a large number of partitions
- Parallel runs possible only with “normal” elements
 - ▷ p-elements, face and edge elements and discontinuous Galerkin elements are not supported
- Saving parameters from results with `SaveData` and `SaveLine` does not work in parallel at the moment

Setting up parallel runs

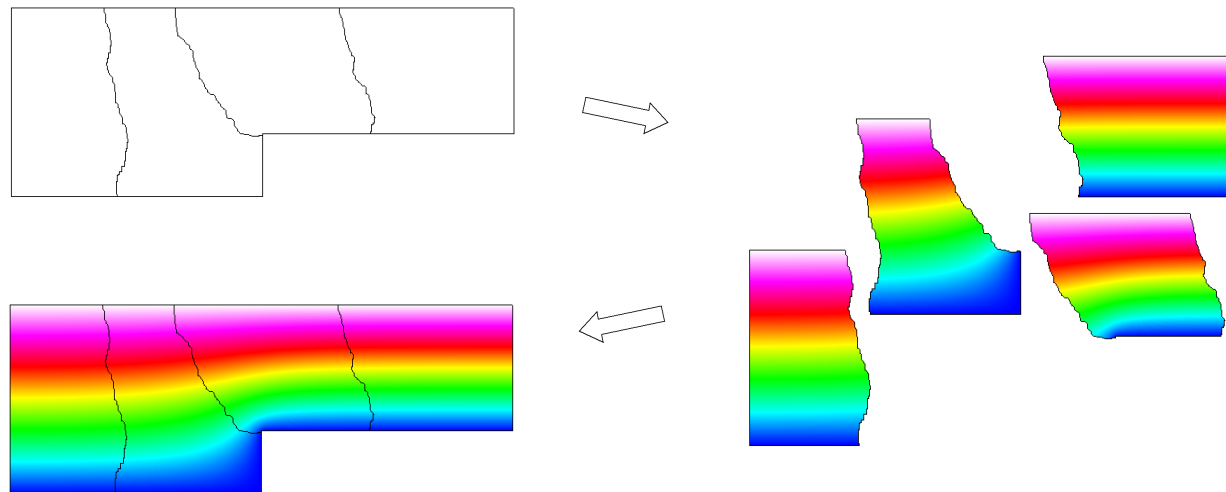
- MPI-version of Elmer may be compiled with the configure utility by giving the path to MPI libraries: `./configure --prefix=... --with-mpi-dir=...`
- Mesh needs to be partitioned (with ElmerGrid)
- The same sif-file and ELMERSOLVER_STARTINFO apply than with serial runs!
- Execute with `mpirun` command
- Construct a global post-processing file with `ElmerGrid`, if desired

Parallel runs on CSC's cluster

- MPI version of Elmer has been compiled for CSC's Linux cluster `sepeli.csc.fi`
- How to set up parallel runs on `sepeli`:
 - ▷ Set up a normal serial simulation
 - ▷ Initialize Elmer: use `elmer_pathf90`
 - ▷ Mesh partitioning: `ElmerGrid 2 2 mesh_dir -metis 4`
 - ▷ Log in to a computing node or prepare a batch job file
(see <http://www.csc.fi/proj/mgrid/docs/sepeli-pikaopas.html>)
 - ▷ Initialize parallel environment: use `mpich-path64`
 - ▷ Run the parallel solver: `mpirun -np 4 -machinefile $MFILE
/v/linux24_x8664/appl/math/elmer/pathf90/bin/ElmerSolver_mpi`
 - ▷ Construct global result file from the pieces: `ElmerGrid 15 3 filename.ep`

Example

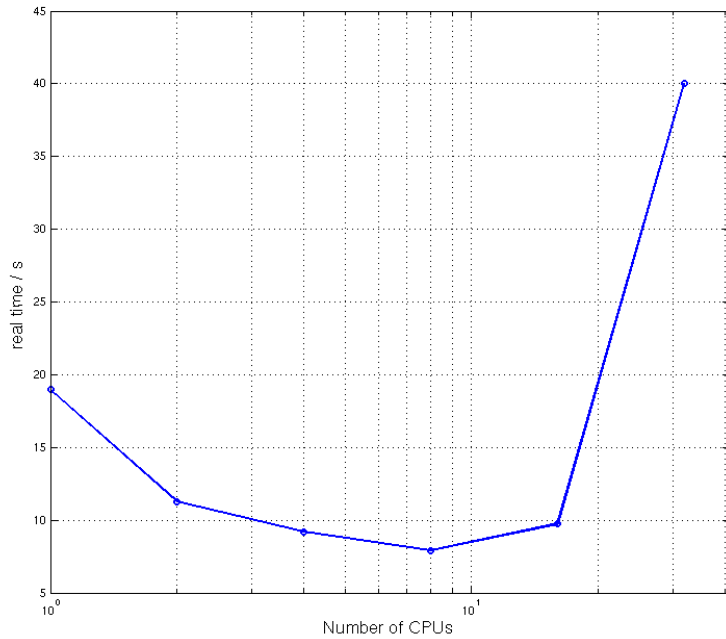
- As an example a 2D electrostatic problem was solved with varying number of processors
- The test had 100,000 degrees of freedom and the computations were done on sepeli Linux cluster



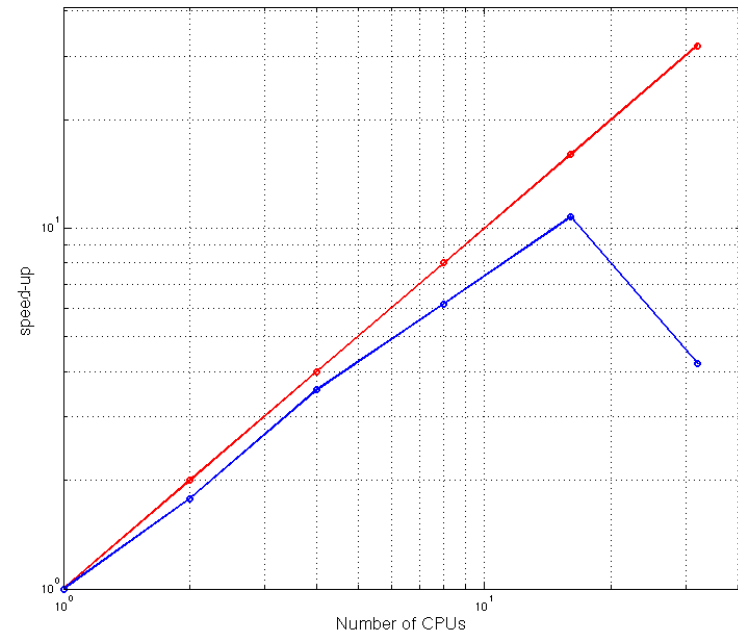
The mesh is divided into four partitions on which the problem is solved. Finally the pieces are put back together.

Results – I

- Results in a test case



Real time used in computations in seconds
as a function of CPUs used



Speed-up as a function of CPUs used;
theoretic (red) and experimental lines (blue)

Results – II

- Results in a test case

processors	cpu-time / sec	real-time / sec	speed-up	efficiency
1	18.45	18.97	1.00	1.00
2	20.74	11.27	1.78	0.89
4	20.70	9.19	3.57	0.89
8	23.95	7.90	6.16	0.88
16	27.48	9.73	10.74	0.67
32	140.43	40.03	4.20	0.13