OpenMP threading and vectorization of MPI Finite element code Elmer

Mikko Byckling, Intel Corporation   Juhani Kataja, CSC - Finnish IT Center for Science

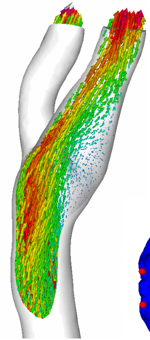
CSC – IT Center for Science

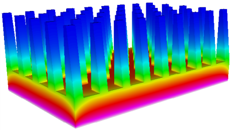
6th March 2018

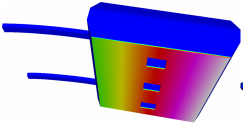
CSC

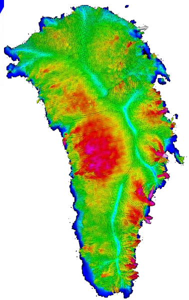
Legal Disclaimers and Optimization Notices

CSC

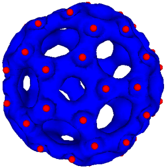


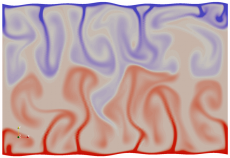


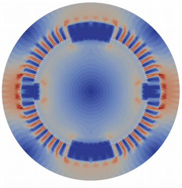


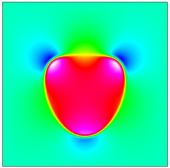


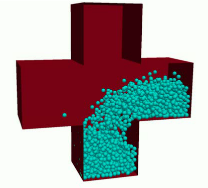
Elmer finite element software for multiphysical problems

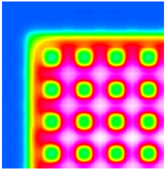












Figures by Esko Järvinen, Mikko Lyly, Peter Råback, Timo Veijola (TKK) & Thomas Zwinger

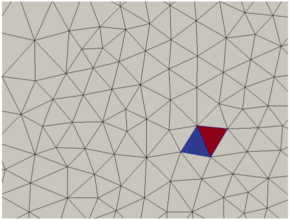
Elmer codebase

- Software
  - ~600k lines of code, $\frac{3}{4}$ Fortran and rest in C/C++
  - ~500 consistency tests
  - ~750 pages of documentation
  - ~1000 commits per year
- Community
  - ~20k downloads of Windows binary yearly
  - ~2k forum posts yearly
  - ~100 participants on Elmer courses yearly
  - Several Elmer related scientific visits on CSC yearly

CSC

The finite element method

1. Element loop
   1.1 Local matrix assembly
   1.2 Local to global glueing: gather/scatter store in a compressed compressed sparse row (CSR) matrix structure
2. Solve the global linear system



1. Loop $K \in \mathcal{K}$
   1.1 $\hat{A}_{ij}^K = \int_K a(\phi_i, \phi_j)dx$
   1.2 $\boldsymbol{A}_{\sigma_i^K,\sigma_j^K} = \boldsymbol{A}_{\sigma_i^K,\sigma_j^K} + \hat{A}_{ij}^K$
2. $\boldsymbol{Ax} = \boldsymbol{b}$

Previously

Local matrix assembly (1.1 only)

- *Optimizing Elmer finite element software on KNL – IXPUG annual Spring Conference 2017*
  - Vectorization of high order basis function evaluations
  - Optimization of local matrix assembly
  - Comparison of HSW and KNL performance after and before optimizations

GEMM Assembly + vectorized basis functions

Multicore speedup 128 threads on KNL, 24 threads on HSW, P=6				
Element (#ndofs, #quadrature points)	Speedup		Optimized local matrix evaluations / s	
	KNL	HSW	KNL	HSW
Line (7, 8)	1.5	2	363k	560k
Triangle (28, 64)	8.7	11	500k	627k
Quadrilateral (30, 64)	7.2	6.2	507k	598k
Tetrahedron (84, 512)	8.0	5.7	11.1k	11.8k
Prism (93, 512)	8.9	5.6	10.6k	10.8k
Hexahedron (105, 512)	8.9	6.2	9.4k	10.2k

Glueing

CSR matrix is a triple

- $(\texttt{vals} \in \mathbb{R}^{\mathrm{NNZ}}, \texttt{rows} \in \mathbb{N}^{N_{\mathrm{rows}}+1}, \texttt{cols} \in \mathbb{N}^{\mathrm{NNZ}})$

CSR add value algorithm in FEM: $(\boldsymbol{A}_{\sigma_i^K,\sigma_j^K} = \boldsymbol{A}_{\sigma_i^K,\sigma_j^K} + \hat{A}_{ij}^K)$

1. Local to global index map: each local j corresponds to global index σ_j.
2. Row value bounds: $\boldsymbol{r}_{\mathrm{bottom}} = \texttt{rows}(\sigma_i) \ldots \boldsymbol{r}_{\mathrm{top}} = \texttt{rows}(\sigma_i + 1) - 1$.
3. Value indices: Let c_{ij} be location of σ_j in $\texttt{cols}(\boldsymbol{r}_{\mathrm{bottom}}:\boldsymbol{r}_{\mathrm{top}})$.
4. Add $\hat{A}_{ij}$ to $\texttt{vals}(c_{ij})$.

Optimizations:

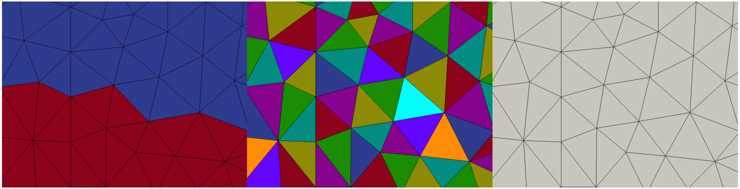
3. Switch binary search to linear search.
4. Prefetch $\texttt{vals}(c_{ij})$.

Parallelization of matrix assembly

The part “4. Add $\hat{A}_{ij}$ to `vals(`c_{ij}`)`” is the only place for possible race conditions.

Possible solutions:

1. MPI: Partition mesh and assemble in serial in each rank. Avoids race conditions by construction.
2. Coloring
   - Color elements so that elements of certain color never meet.
   - Looping over elements of one color guarantees no race conditions.
   - Coloring can be done in threaded fashion.
   - No need for minimal coloring.
3. Concurrent access (`!$omp atomic`)



Linear solver

Krylov methods: Find solution from Krylov subspace

$$\mathcal{K}_N = (b, Ab, A^2b, \dots, A^N).$$

OpenMP

- Only one "interior" M-V product required for global product Ab
- Use `mkl_dcsrgemv` if MKL is available, otherwise use OpenMP pragmas:

MPI

- One CSR matrix per MPI rank and information about data shared with neighboring partitions.
- One "interior" M-V product and exchange with `mpi_recv`/`mpi_send`.

```
!$omp parallel do private(j,rsum)
  DO i=1,n
    rsum = 0.0d0
!DIR$ IVDEP
    DO j=Rows(i),Rows(i+1)-1 ...
```

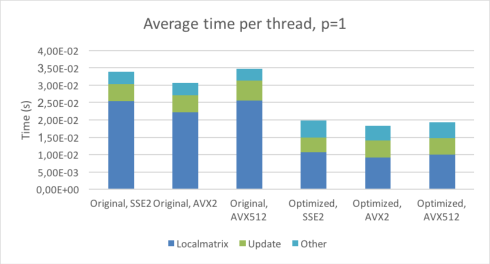
csc

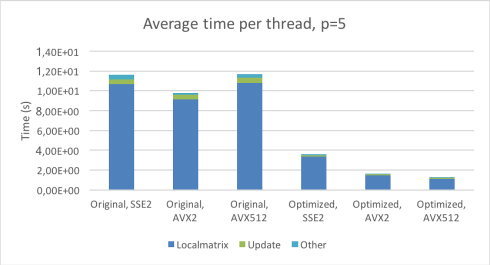
Test platforms

- HSW: 2 $\times$ 12 core Intel Haswell E5-2690v3 node.
- SKL: 2 $\times$ Intel® Xeon® Gold 6148, 2.4GHz, 20 cores/socket, 2 threads/core
- KNL: Colfax development platform with 7210 Intel Xeon Phi, at 1.3 GHz, 16GB of MCDRAM, 96GB DDR4 memory. Cache mode.

More details in appendices.

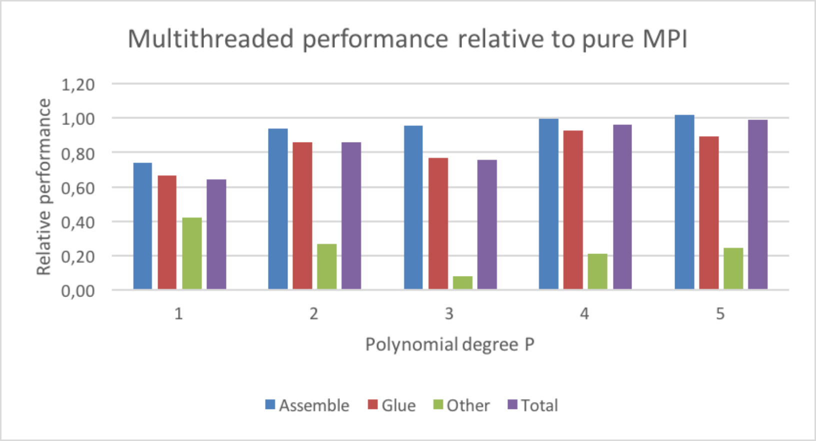
Assembly results on SKL





CSC

OMP/MPI comparison on SKL

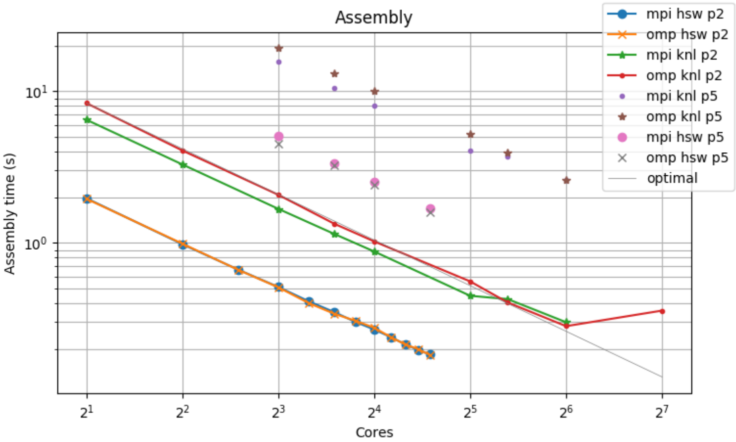


Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. For configuration info, see System Setup.

Results: model problem

- Poisson problem in unit cube, 64,000 hexas
  - 3rd order elements: 472,361 dofs
  - 5th order elements: 875,801 dofs
- Measure assembly and linear solve separately
  - Strong scalability on one node
- Parallelization scheme: MPI, OpenMP

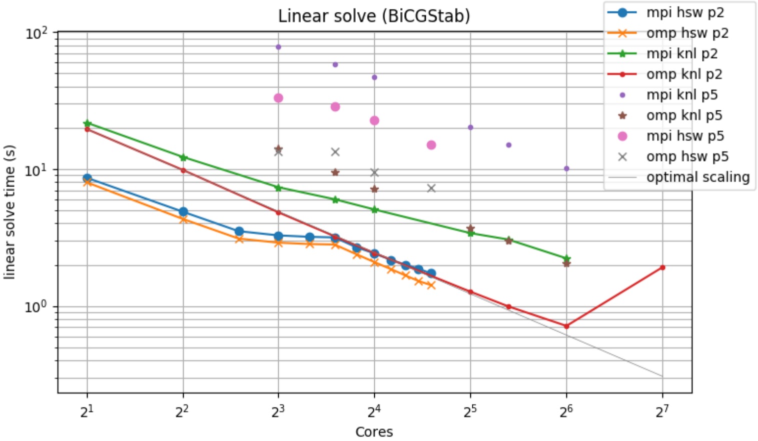
Assembly: KNL vs HSW



Results: linear solve

- Fixed number (100) of BiCGSTab iterations
- Utilizing full node with OpenMP:

$$\texttt{t(KNL p2)} / \texttt{t(HSW p2)} \approx 0.5$$



Challenges

- Elmer has been originally designed to be a pure MPI code
  - All $O(\#\mathrm{Elements})$ algorithms without communication parallelize automatically with MPI
- Threading challenges
  - ILU(n) preconditioning
  - Disk IO
  - Construction of internal element data structures
  - CSR matrix formatting

Conclusion

- Threading critical path FEM:
  - OpenMP assembly on par with MPI (optimally scaling) assembly
  - Data layout of local assembly optimized for modern SIMD processors
  - Krylov solvers parallelize easily with OpenMP pragmas
    - CSR matrix vector product: call `mkl_dcsrgemv`
- Linear solve on KNL node faster than on HSW
  - Memory bound algorithm: BiCGSTab
- Given recent developments Elmer is now a hybrid OpenMP-MPI code
  - Threading controlled with environmental variables.

http://elmerfem.org https://github.com/ElmerCSC/elmerfem

CSC

Details (KNL and HSW)

- KNL:
  - Colfax development platform with 7210 Intel Xeon Phi ,at 1.3 GHz, 16GB of MCDRAM, 96GB DDR4 memory.
  - Cache mode
  - Compile flags `-vecabi=cmdtarget -xMIC-AVX512 -align array64byte`
- HSW:
  - 2 × 12 core Intel Haswell E5-2690v3 node
  - 8 × 16 GB DDR4 memory at 2133 MHz
  - No hyperthreading
  - Compile flags `-xAVX -axCORE-AVX2,CORE-AVX-I -align array64byte -vecabi=cmdtarget`
- Software stack:
  - Intel Fortran 17.0.4 (20170411)
  - IntelMPI 17 update 3 build 20170405
  - MKL 20170003
- Environment variables:
  - `OMP_PROC_BIND=TRUE`
  - `I_MPI_FABRICS`: default (shm intranode)

CSC

Details (SKL)

- Intel® Xeon® Gold 6148, Dual socket server, 2.4GHz, 20 cores/socket, 40 cores, 80 threads
- Intel® Hyper-Threading Technology enabled
- Intel® Turbo Boost Technology enabled
- DDR4 192 GB, 2666 MHz
- RHEL* 7.3
- Intel® Composer 2017U4
- nr_hugepages=8000
- `KMP_AFFINITY=scatter,granularity=fine`
  `I_MPI_FABRICS=shm:tmi`
  `I_MPI_PIN_PROCESSOR_LIST=allcores:map=bunch`