

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

Legal Disclaimers and Optimization Notices

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

All products, platforms, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See www.intel.com/products/processor_number for details. 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 information go to www.intel.com/benchmarks. Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3E instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.

The Intel Core and Itanium processor families may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

The code names Arrandale, Bloomfield, Boazman, Boulder Creek, Calpella, Chief River, Clarkdale, Cliffside, Cougar Point, Gultown, Huron River, Ivy Bridge, Kilmer Peak, King's Creek, Lewisville, Lynnfield, Maho Bay, Montevina, Montevina Plus, Nehalem, Penryn, Puma Peak, Rainbow Peak, Sandy Bridge, Sugar Bay, Tylersburg, and Westmere presented in this document are only for use by Intel to identify a product, technology, or service in development, that has not been made commercially available to the public, i.e., announced, launched or shipped. It is not a "commercial" name for products or services and is not intended to function as a trademark.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

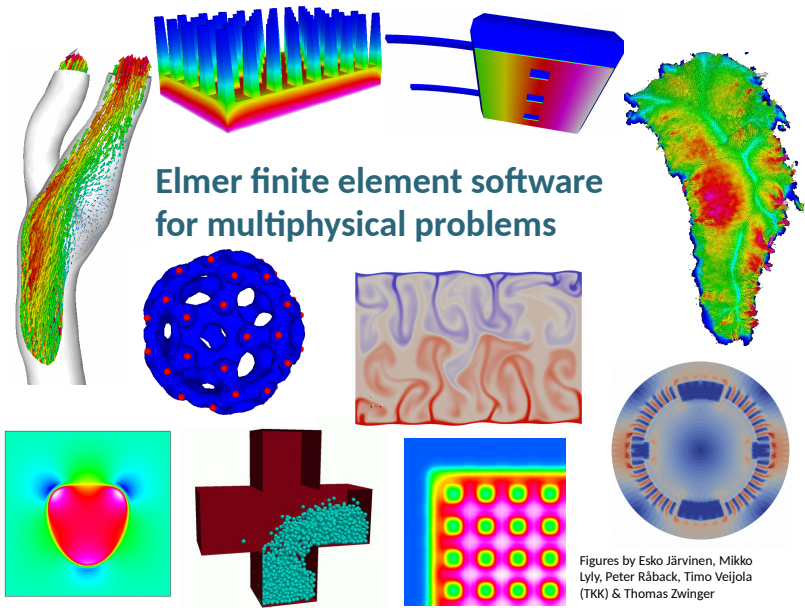
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Intel, Intel Core, Core Inside, Itanium, and the Intel Logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others. Intel collects and uses personal information from employees as part of SES, including capturing audio recording of sessions (both presenters and audience QA) as well as photographs and video recording of various event activities during the event. By registering and attending the SES conference, you give your consent for this capture. This includes both speakers and attendees. Intel will not retain your personal information longer than is necessary for the purposes for which it is collected.



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

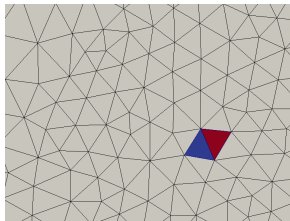
- ▶ Community

- ▶ ~20k downloads of Windows binary yearly
- ▶ ~2k forum posts yearly
- ▶ ~100 participants on Elmer courses yearly
- ▶ Several Elmer related scientific visits on CSC yearly

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 $K \in \mathcal{K}$

$$1.1 \hat{\mathbf{A}}_{ij}^K = \int_K a(\phi_i, \phi_j) dx$$

$$1.2 \mathbf{A}_{\sigma_i^K, \sigma_j^K} = \mathbf{A}_{\sigma_i^K, \sigma_j^K} + \hat{\mathbf{A}}_{ij}^K$$

2. $\mathbf{Ax} = \mathbf{b}$

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



Element (#ndofs, #quadrature points)	Multicore speedup 128 threads on KNL, 24 threads on HSW, P=6			
	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

- ▶ $(\mathbf{vals} \in \mathbb{R}^{\text{NNZ}}, \mathbf{rows} \in \mathbb{N}^{\text{N}_{\text{rows}}+1}, \mathbf{cols} \in \mathbb{N}^{\text{NNZ}})$

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

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

Optimizations:

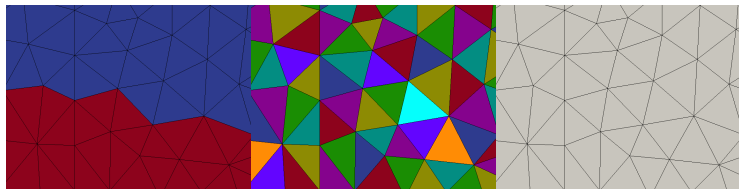
3. Switch binary search to linear search.
4. Prefetch $\mathbf{vals}(c_{ij})$.

Parallelization of matrix assembly

The part “4. Add \hat{A}_{ij} to $\text{vals}(c_{ij})$ ” 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 `mk1_dcsrgev` if MKL is available, otherwise use OpenMP pragmas:

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

MPI

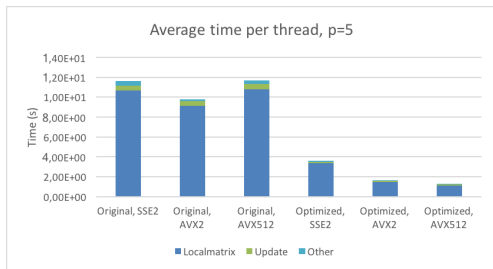
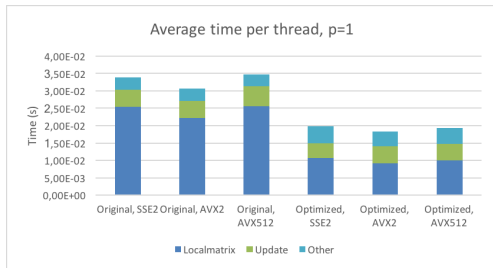
- ▶ One CSR matrix per MPI rank and information about data shared with neighboring partitions.
- ▶ One “interior” M-V product and exchange with `mpi_recv/mpi_send`.

Test platforms

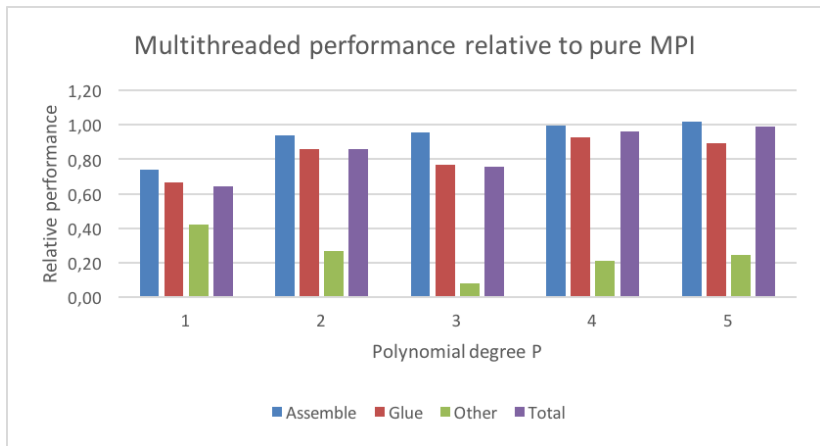
- ▶ HSW: 2 × 12 core Intel Haswell E5-2690v3 node.
- ▶ SKL: 2 × Intel® Xeon® Gold 6148, 2.4GHz, 20 cores/socket, 2 threads/core
- ▶ KNL: Colfax development platform with 7210 Intel Xeon Phi, at 1.3 GHz, 16GB of MCDRAM, 96GB DDR4 memory. Cache mode.

More details in appendices.

Assembly results on SKL



OMP/MPI comparison on SKL

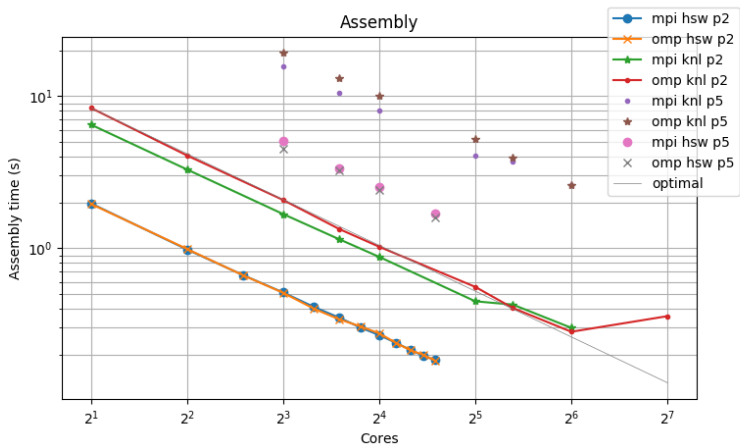


Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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

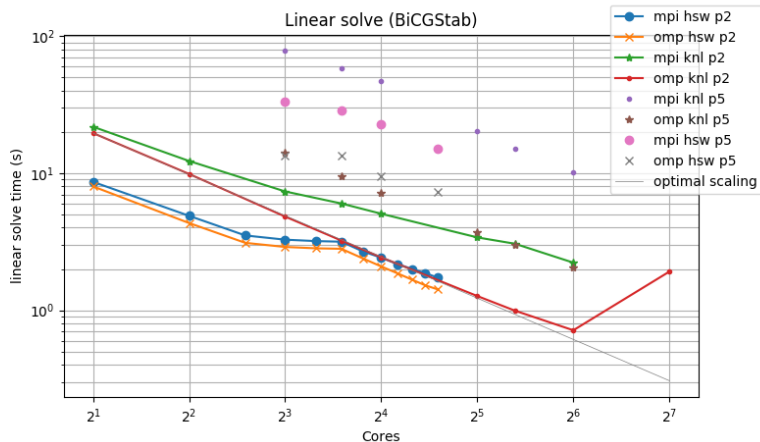
Assembly: KNL vs HSW



Results: linear solve

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

$$\tau(\text{KNL p2}) / \tau(\text{HSW p2}) \approx 0.5$$



Challenges

- ▶ Elmer has been originally designed to be a pure MPI code
 - ▶ All $O(\#Elements)$ 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 `mk1_dcsrgev`
- ▶ 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>

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`
 - ▶ `T MPI FABRTCS: default (shm intranode)`

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