2017 IXPUG US
Annual Meeting

Harnessing the Intel Xeon Phi x200 Processor for Earthquake Simulations

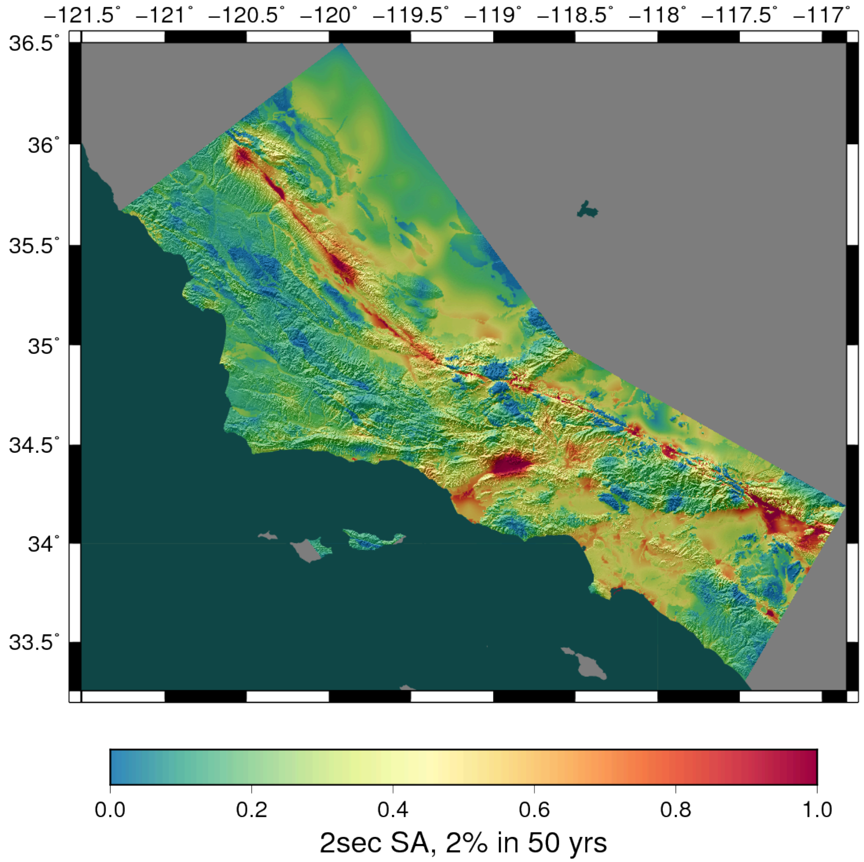
Alexander Breuer, Yifeng Cui, Alexander Heinecke (Intel),
Josh Tobin, Chuck Yount (Intel)

What is AWP-ODC-OS?

- AWP-ODC-OS (Anelastic Wave Propagation, Olsen, Day, Cui): Simulates seismic wave propagation after a fault rupture
- Used extensively by the Southern California Earthquake Center community (SCEC)

- License: BSD 2-Clause

  https://github.com/HPGeoC/awp-odc-os

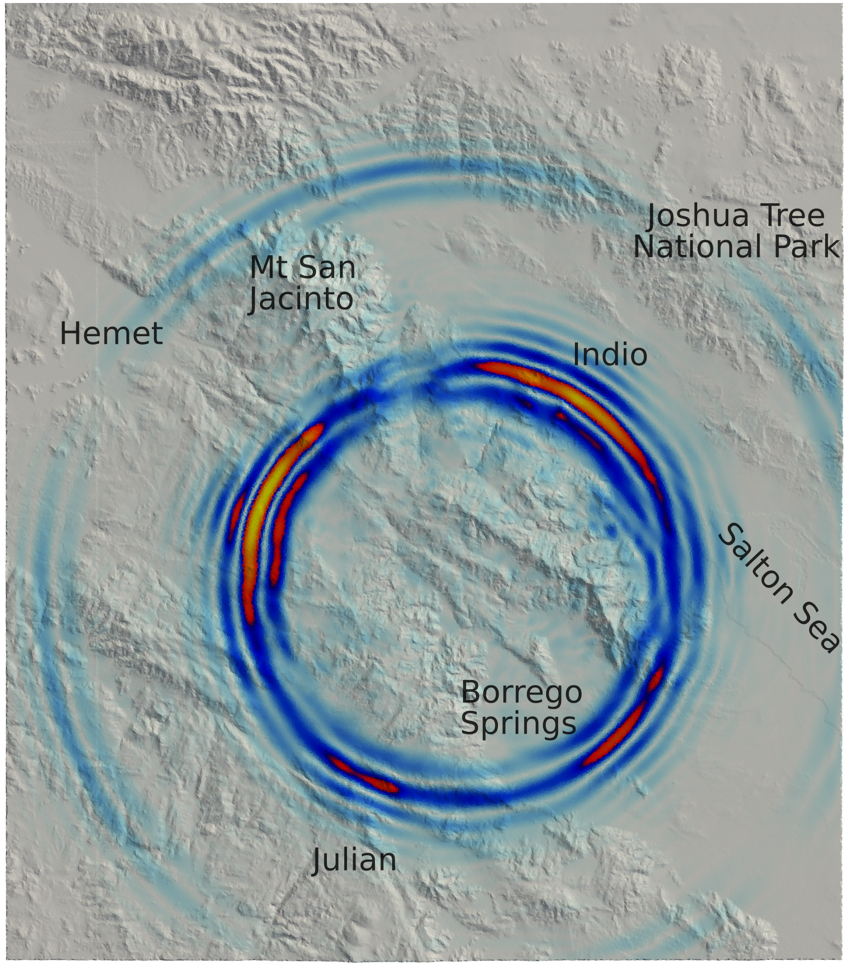


Combined Hazard map of CyberShake Study 15.4 (LA, CVM-S4.26) and CyberShake Study 17.4 (Central California, CCA-06). AWP-ODC simulations are used to generate hazard maps. Colors show 2 seconds period spectral acceleration (SA) for 2% exceedance probability in 50 years.

What is EDGE?

- **Extreme-scale Discontinuous Galerkin Environment (EDGE): Seismic wave propagation through DG-FEM**
- **Focus: Problem settings with high geometric complexity, e.g., mountain topography**
- **“License”: BSD 3-Clause (software), CC0 for supporting files (e.g., user guide)**

http://dial3343.org

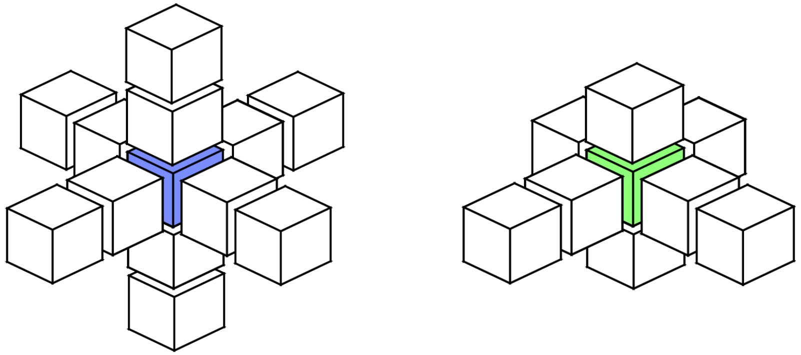


Example of hypothetical seismic wave propagation with mountain topography using EDGE. Shown is the surface of the computational domain covering the San Jacinto fault zone between Anza and Borrego Springs in California. Colors denote the amplitude of the particle velocity, where warmer colors correspond to higher amplitudes.

Two Representative Codes

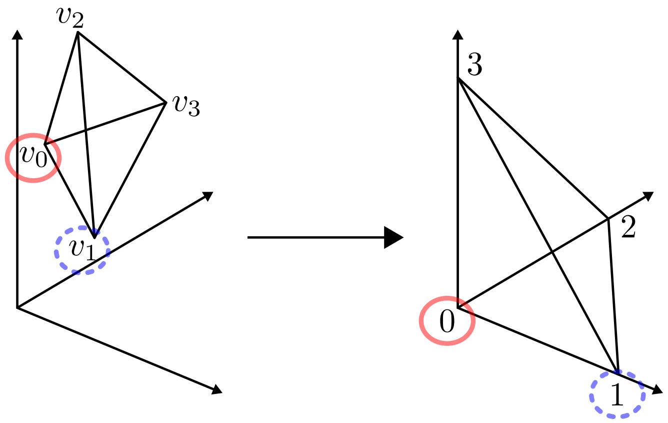
AWP-ODC-OS

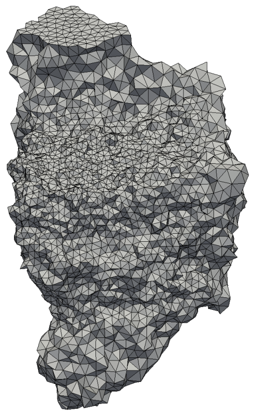
- Finite difference scheme: 4th order in space, 2nd order in time
- Staggered-grid, velocity/stress formulation of elastodynamic eqns with frequency dependent attenuation
- Memory bandwidth bound



EDGE

- Discontinuous Galerkin Finite Element Method (DG-FEM)
- Unstructured tetrahedral meshes
- Small matrix kernels in inner-loop
- Compute bound (high orders)





AWP-ODC-OS

Boosting Single-Node Performance: Vector Folding

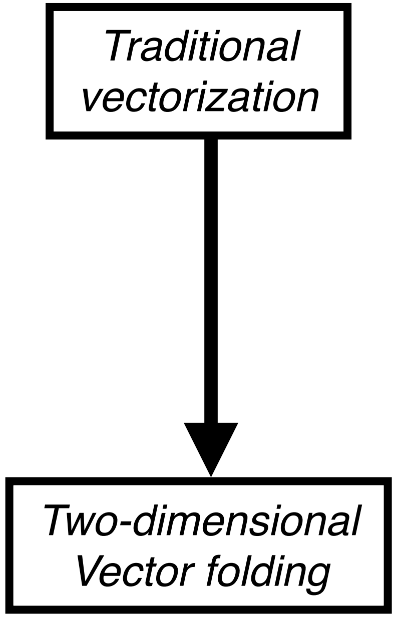
- Vector folding data layout
  - Stores elements in small SIMD-sized multi-dimensional blocks
  - Reduces memory bandwidth demands by increasing reuse

- YASK (Yet Another Stencil Kernel)
  - Open-source (MIT License) framework from Intel
  - Inputs scalar stencil code
  - Creates optimized kernels using vector folding and other optimizations

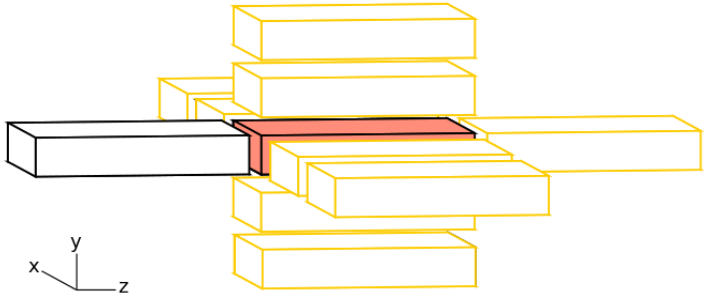
Y*A*S*K

Yet Another Stencil Kernel

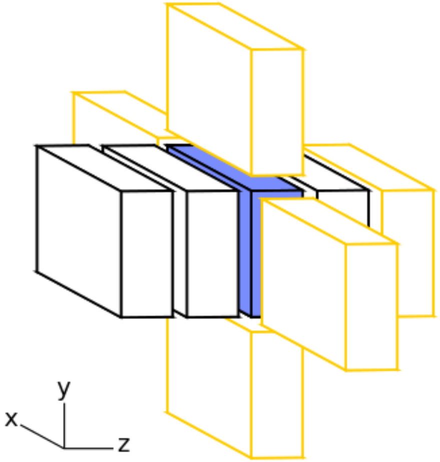
https://github.com/01org/yask


Traditional vectorization

Two-dimensional Vector folding



Requires 9 cache loads per SIMD result

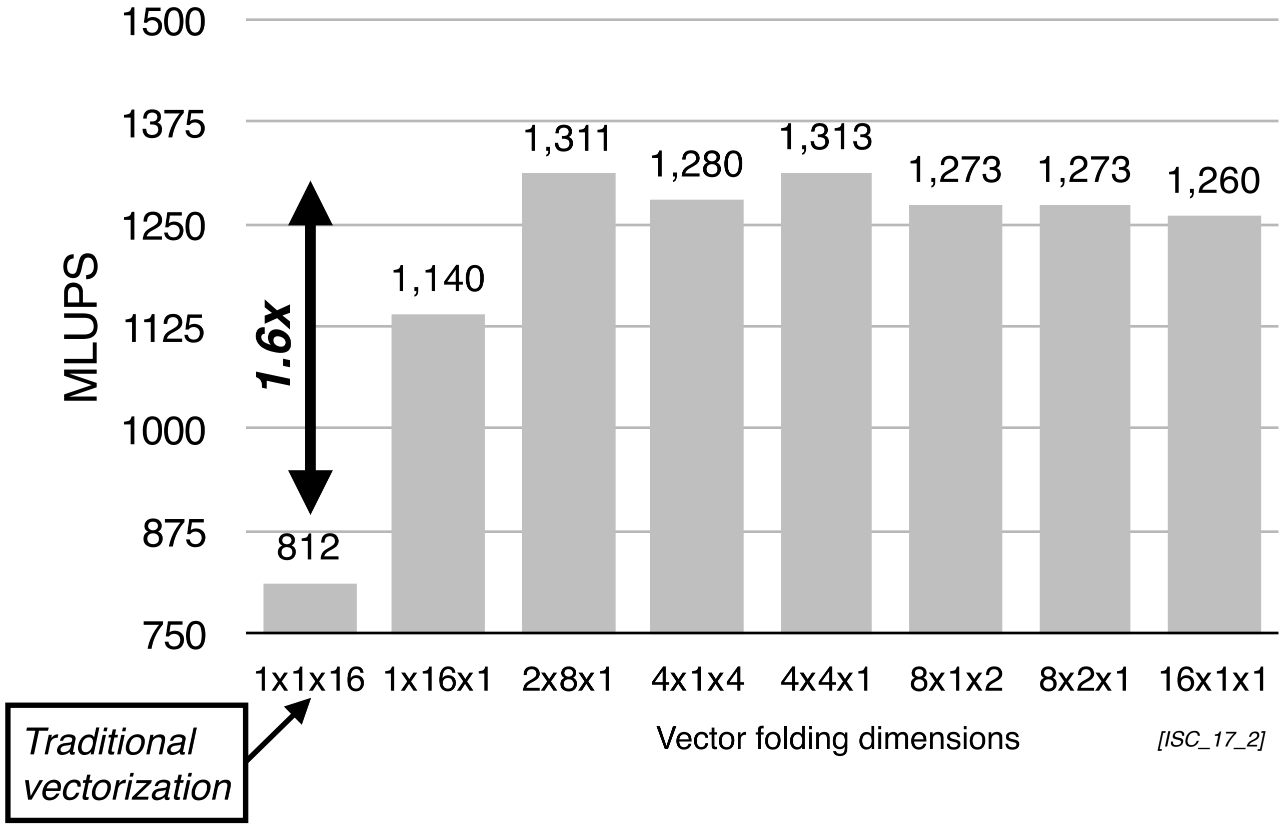


Requires only 5 cache loads

AWP-ODC-OS

Vector Folding: Performance

- Hardware: Intel Xeon Phi 7210
- Domain size: 1024x1024x64
- Single precision: Vector blocks of 16 elements
- Performance measured by YASK proxy
- Performance in Mega Lattice Updates per Second (MLUPS) out of MCDRAM (flat-mode)
- Insight: Vector folding achieves a speedup of up to 1.6x

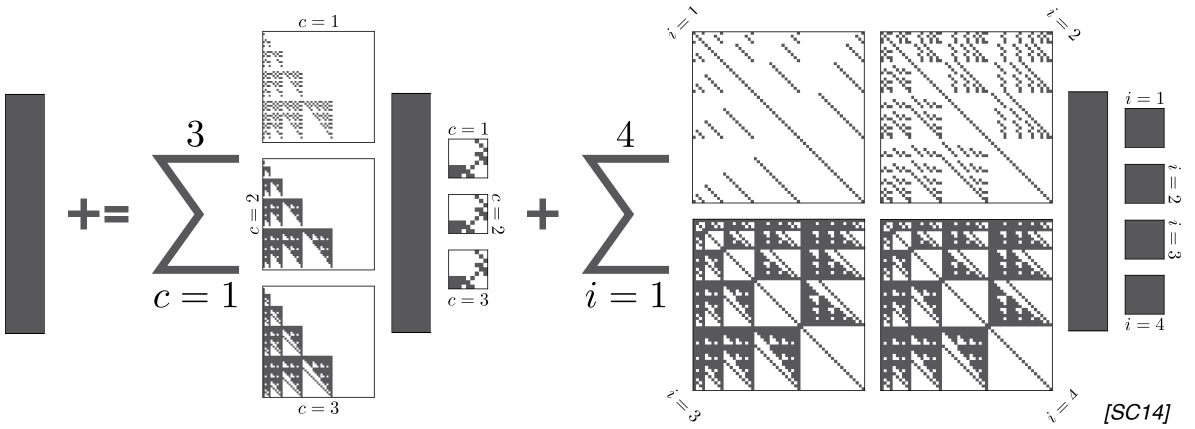


[ISC_17_2]

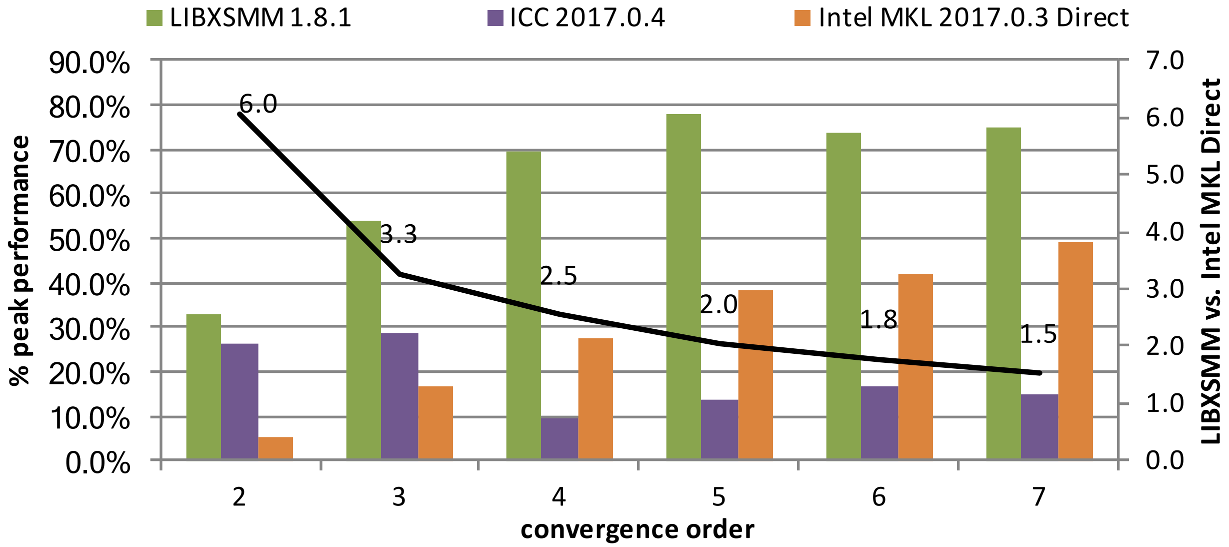
LIBXSMM

- LIBXSMM: Library for small sparse and dense matrix-matrix multiplications, BSD 3-Clause
- JIT code generation of matrix kernels
- Hardware: Intel Xeon Phi 7250, flat
- Insight: Close to peak performance out of a hot cache

https://github.com/hfp/libxsmm



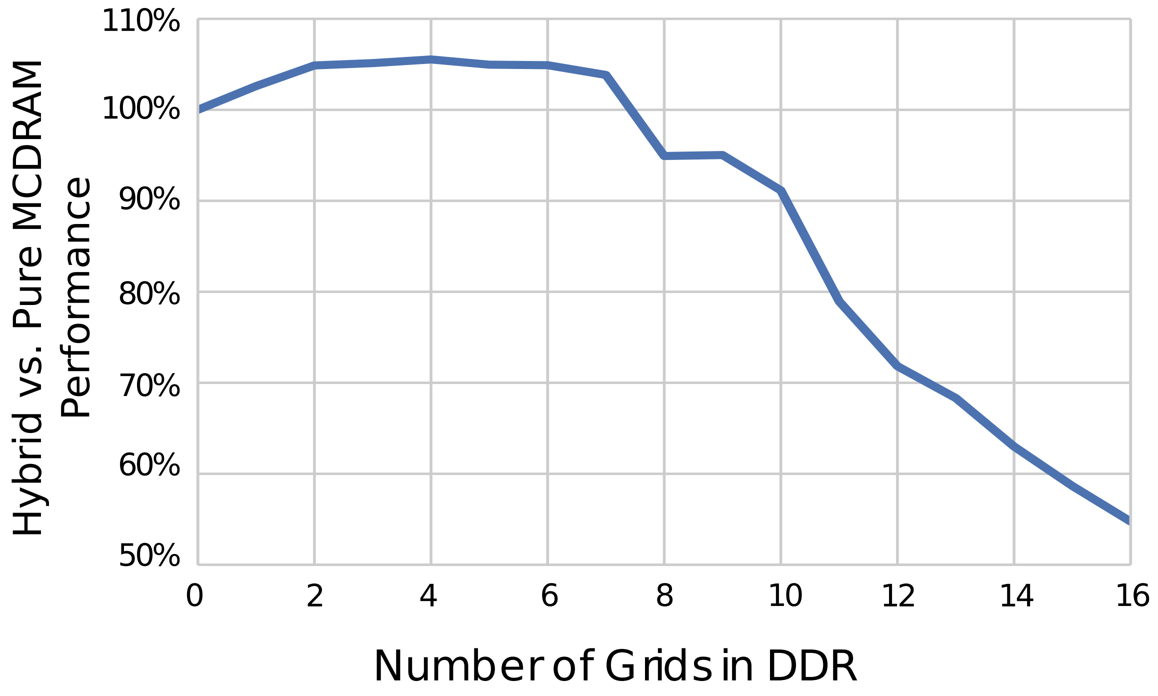
[SC14]



Performance comparison of dense matrix-matrix multiplications in LIBXSMM on Knights Landing at 1.2 GHz with autovectorized code (compiler) and Intel MKL in version 2017.0.3 out of a hot cache. Shown is the stiffness or flux matrix multiplied with the DOFs

Leveraging KNL Memory Modes

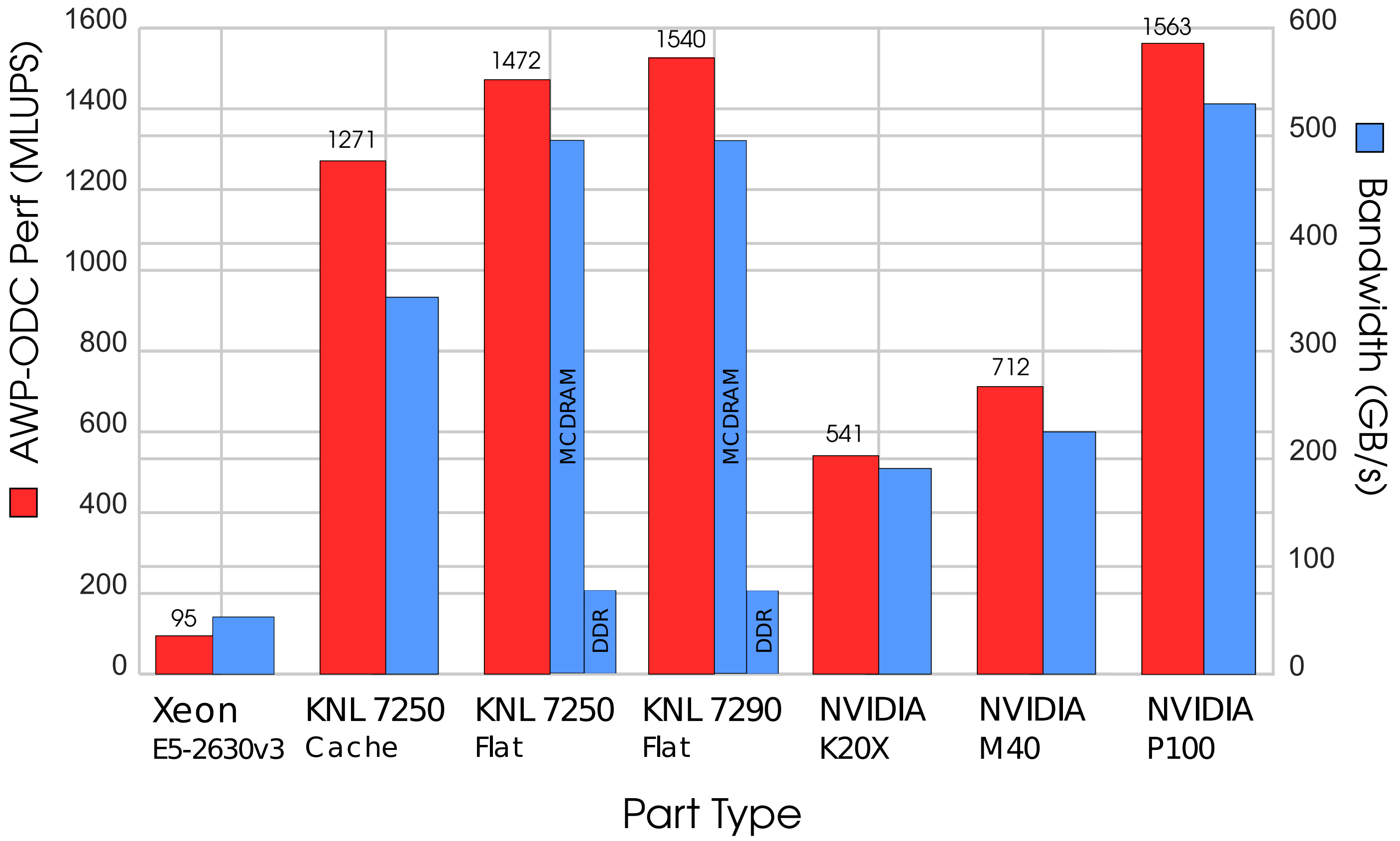
- 26 three-dimensional arrays, 17 of which are read-only or read-heavy
- Heuristically identified: Arrays which are good candidates to be placed in DDR
- Hybrid memory placement:
  - Option 1: Increase available memory by 26% and improve overall performance
  - Option 2: Increase available memory to 46 GB with 50% of optimal performance



Relative performance of AWP-ODC-OS as we move arrays from MCDRAM to DDR. In each case, the best performing combination was found via heuristics and simple search [ISC_17_2].

Architecture Comparison

- Xeon Phi KNL 7290: **2x speedup over NVIDIA K20X; 97% of NVIDIA Tesla P100** performance
- Memory bandwidth accurately predicts performance of architectures (as measured by STREAM and HPCG-SpMv)

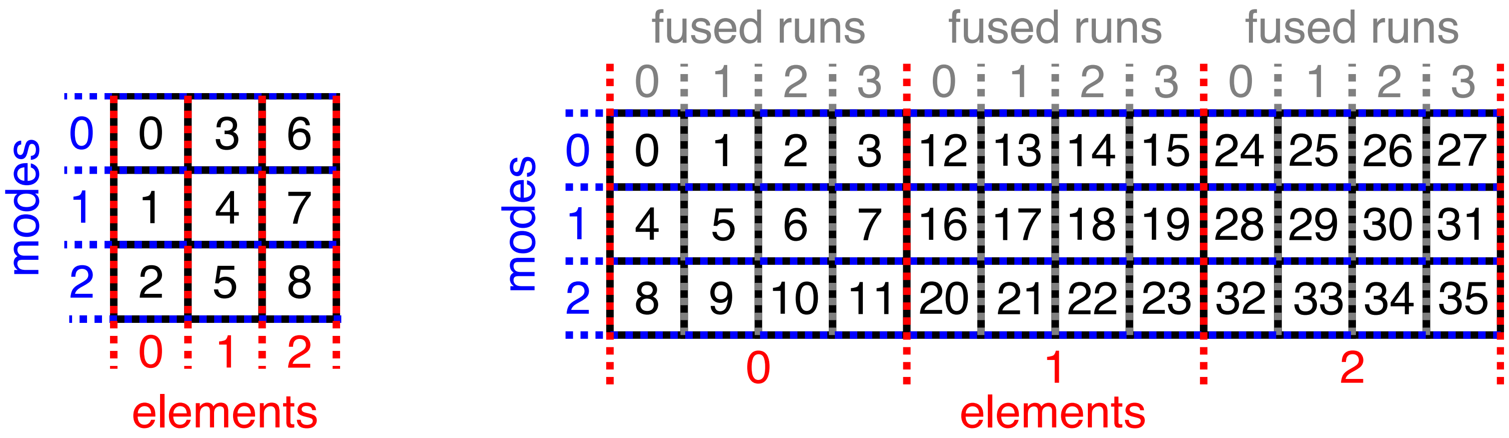


Single node performance comparison of AWP-ODC-OS on a variety of architectures. Also displayed is the bandwidth of each architecture, as measured by a STREAM and HPCG-SpMv [ISC_17_2].

EDGE

Fused Simulations

- **Exploits inter-simulation parallelism:**
  - **Full vector operations, even for sparse matrix operators**
  - **Automatic memory alignment**
  - **Read-only data shared among all runs**
  - **Lower sensitivity to latency (memory & network)**



modes			
0	0	3	6
1	1	4	7
2	2	5	8
	0	1	2
	elements		

	fused runs				fused runs				fused runs			
modes	0	1	2	3	0	1	2	3	0	1	2	3
0	0	1	2	3	12	13	14	15	24	25	26	27
1	4	5	6	7	16	17	18	19	28	29	30	31
2	8	9	10	11	20	21	22	23	32	33	34	35
	0				1				2			
	elements											

Illustration of the memory layout for fused simulations in EDGE. Shown is a third order configuration for line elements and the advection equation. Left: Single forward simulation, right: 4 fused simulations

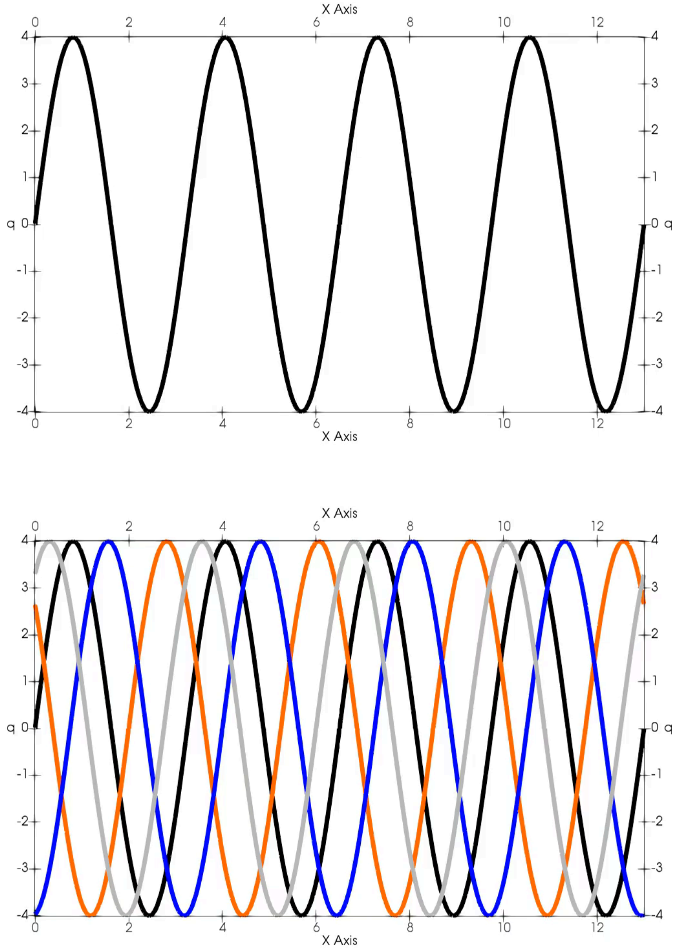
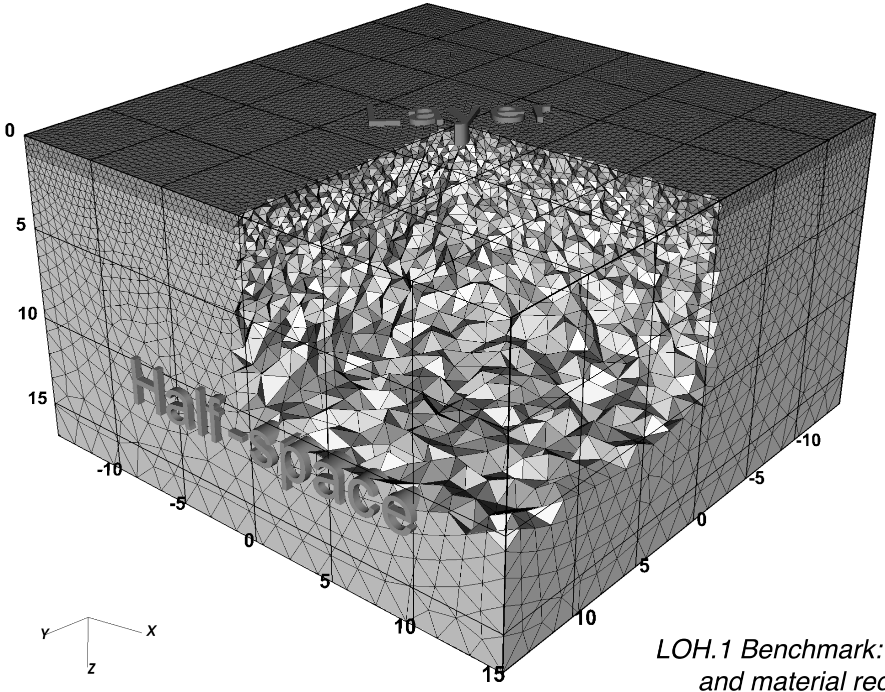


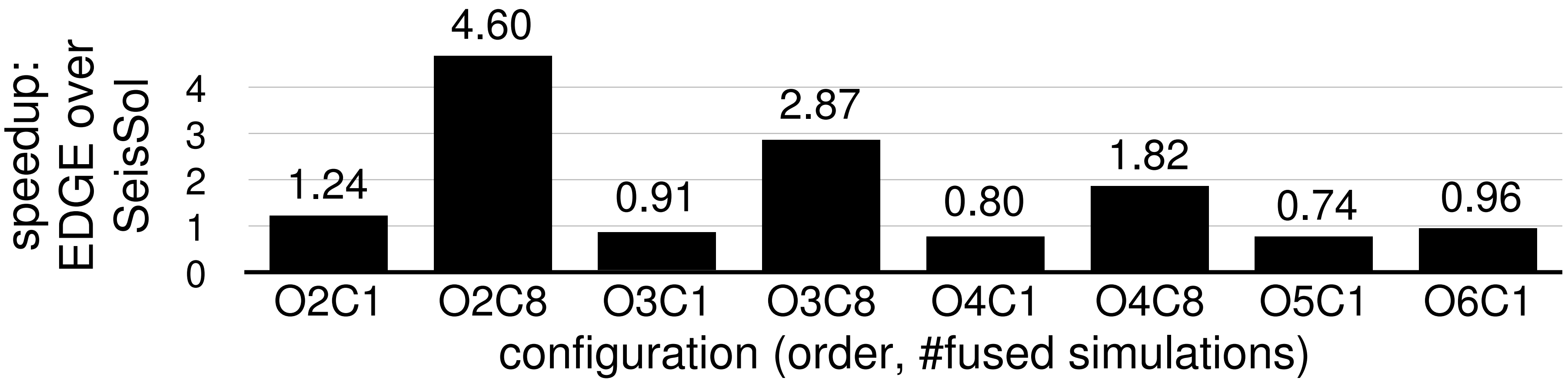
Illustration of fused simulations in EDGE for the advection equation using line elements. Top: Single forward simulation, bottom: 4 fused simulations.

Fused Simulations: Performance

- Orders: 2-6 (non-fused), 2-4 (fused)
- Unstructured tetrahedral mesh: 350,264 elements
- Single node of Cori-II (68 core Intel Xeon Phi x200, code-named Knights Landing)
- EDGE vs. SeisSol (GTS, git-tag 201511)
- Speedup: 2-5x



LOH.1 Benchmark: Example mesh and material regions [ISC16_1]

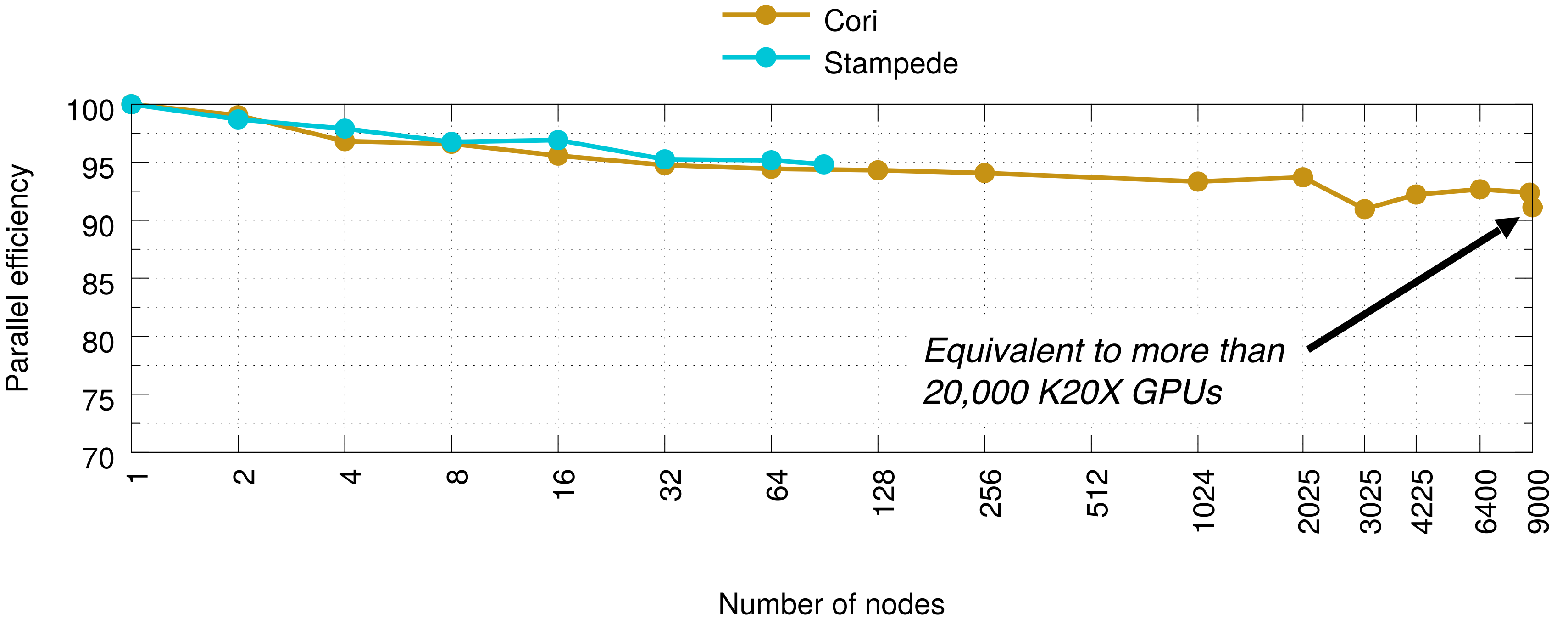


Speedup of EDGE over SeisSol (GTS, git-tag 201511). Convergence rates O2 – O6: single non-fused forward simulations (O2C1-O6C1). Additionally, per-simulation speedups for orders O2–O4 when using EDGE's full capabilities by fusing eight simulations (O2C8-O4C8). [ISC17_1]

AWP-ODC-OS

Outperforming 20K GPUs

- Weak scaling studies on NERSC Cori Phase II and TACC Stampede Extension
- Parallel efficiency of over 91% from 1 to 9000 nodes (9000 nodes = 612,000 cores)
- Problem size of 512x512x512 per node (14 GB per node)
- Performance on 9000 nodes of Cori equivalent to performance of over 20,000 K20X GPUs at 100% scaling



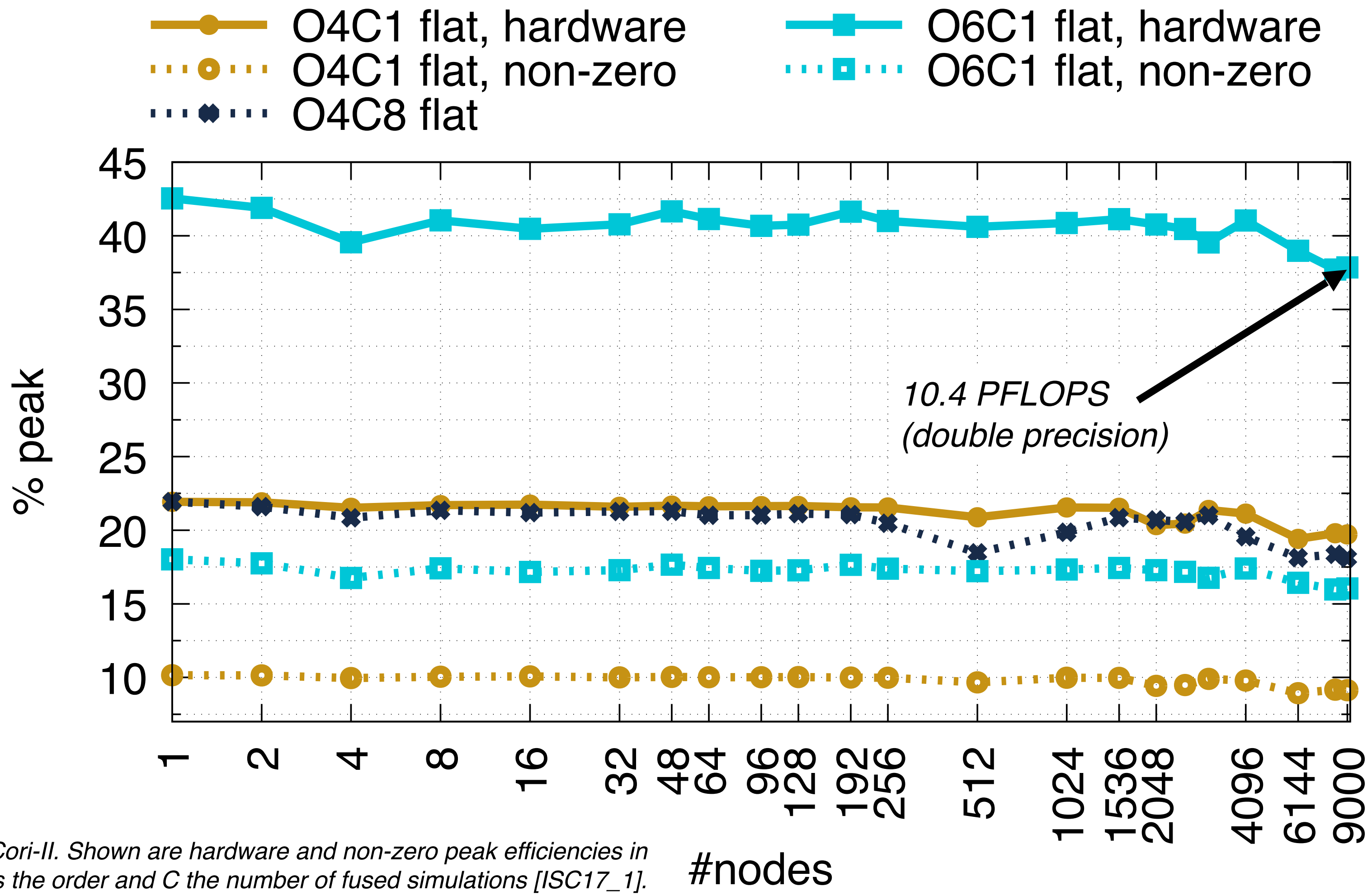
Equivalent to more than 20,000 K20X GPUs

AWP-ODC-OS weak scaling on Cori Phase II and TACC Stampede. We attain 91% scaling from 1 to 9000 nodes. The problem size required 14GB on each node [ISC_17_2].

EDGE

Reaching 10+ PFLOPS

- **Regular cubic mesh, 5 Tets per Cube, 4th order (O4) and 6th order (O6)**
- **Imitates convergence benchmark**
- **276K elements per node**
- **1-9000 nodes of Cori-II (9000 nodes = 612,000 cores)**
- **O6C1 @ 9K nodes: 10.4 PFLOPS (38% of peak)**
- **O4C8: @ 9K nodes: 5.0 PFLOPS (18% of peak)**
- **O4C8 vs. O4C1 @ 9K nodes: 2.0x speedup**



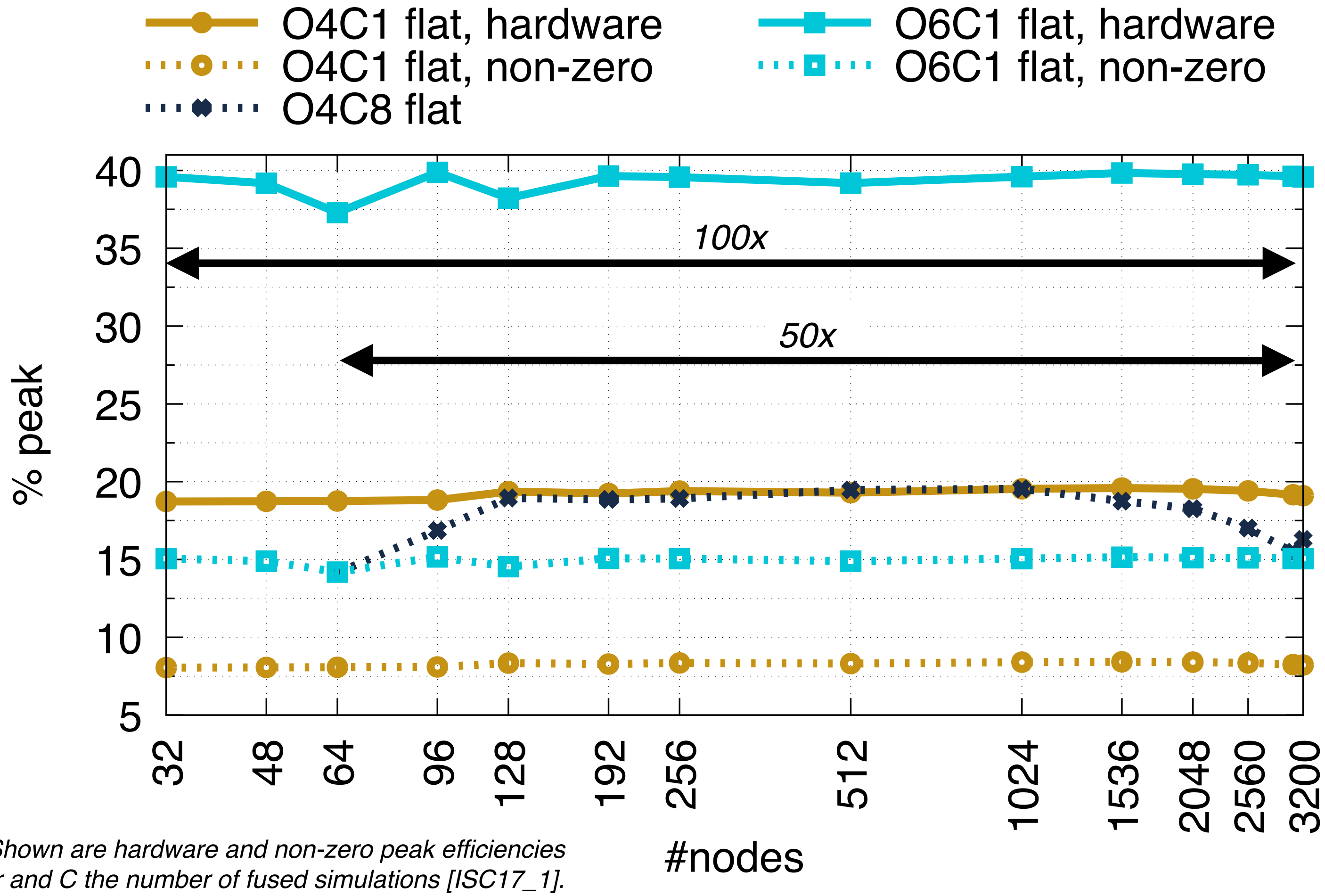
10.4 PFLOPS (double precision)

Weak scaling study on Cori-II. Shown are hardware and non-zero peak efficiencies in flat mode. O denotes the order and C the number of fused simulations [ISC17_1].

EDGE

Strong at the Limit: 50x and 100x

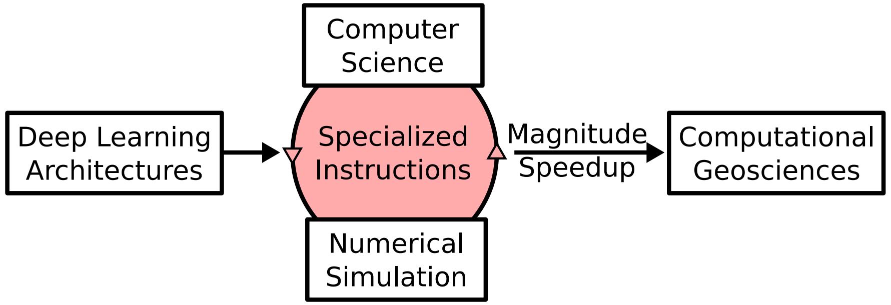
- **Unstructured tetrahedral mesh: 172,386,915 elements**
- **32-3200 nodes of Theta (64 core Intel Xeon Phi x200, code-named Knights Landing)**
- **3200 nodes = 204,800 cores**
- **O6C1 @ 3.2K nodes: 3.4 PFLOPS (40% of peak)**
- **O4C8 vs. O4C1 @ 3.2K nodes: 2.0x speedup**

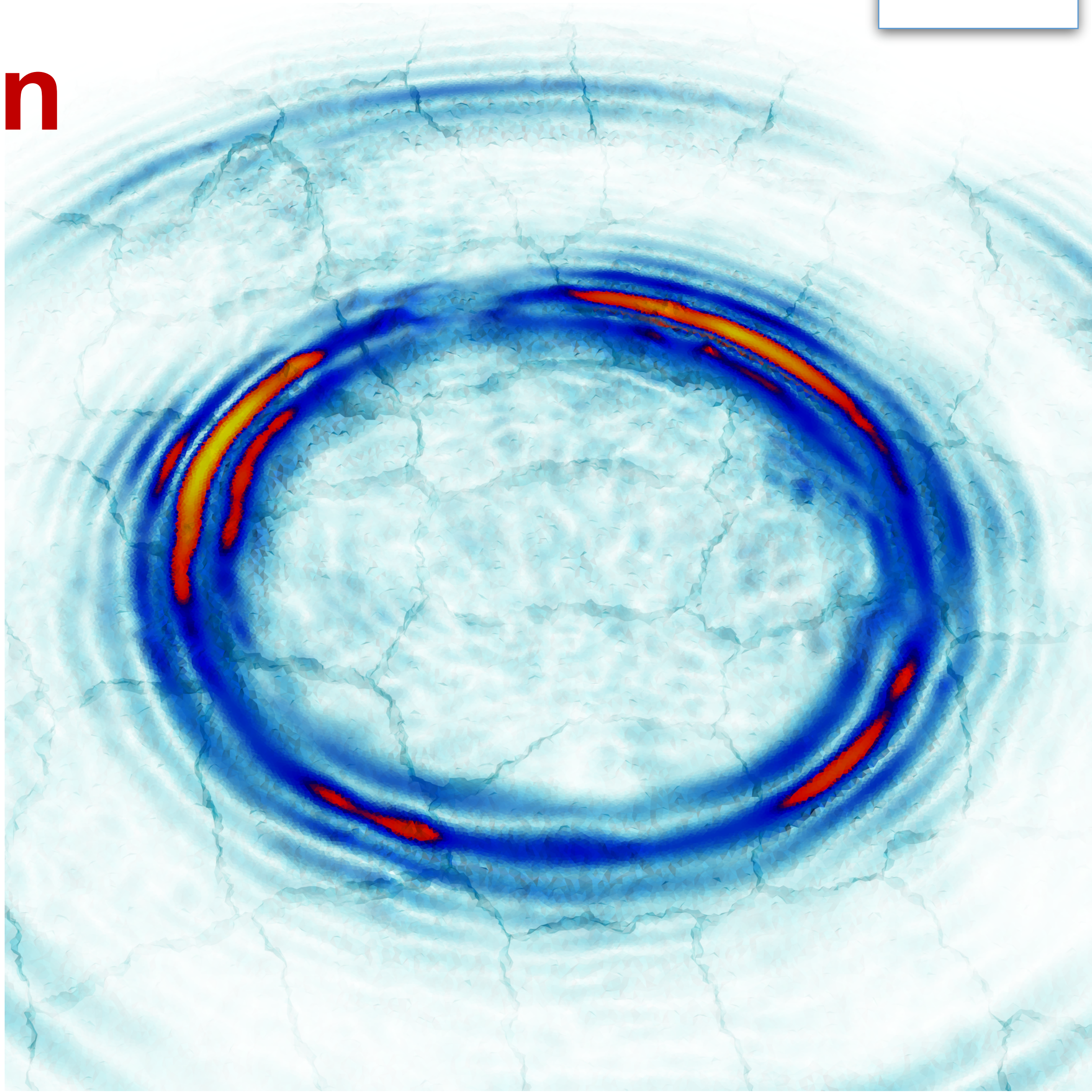


Strong scaling study on Theta. Shown are hardware and non-zero peak efficiencies in flat mode. O denotes the order and C the number of fused simulations [ISC17_1].

Outlook: AI Revolution

- EDGE is a prime candidate for merging traditional HPC and AI
- Work in progress: LIBXSMM for AVX512_4FMAPS (KnightsMill)
- Future work: AVX512_4VNNIW for seismic simulations (KnightsMill)
- Future work: Fused simulations to address high-dimensional parameter spaces (“crunching data”):
  - EDGElearn: (Deep) Learning from seismic simulations
- Future work: LIBXSMM in TensorFlow





References

- [ISC17_1] A. Breuer, A. Heinecke, Y. Cui: EDGE: Extreme Scale Fused Seismic Simulations with the Discontinuous Galerkin Method. Proceedings of International Super Computing (ISC) High Performance 2017
- [ISC17_2] J. Tobin, A. Breuer, C. Yount, A. Heinecke, Y. Cui: Accelerating Seismic Simulations Using the Intel Xeon Phi Knights Landing Processor
  Proceedings of International Super Computing (ISC) High Performance 2017
- [ISC16_1] A. Heinecke, A. Breuer, M. Bader: High Order Seismic Simulations on the Intel Xeon Phi Processor (Knights Landing). High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings. http://dx.doi.org/10.1007/978-3-319-41321-1_18
- [ISC16_2] A. Heinecke, A. Breuer, M. Bader: Chapter 21 - High Performance Earthquake Simulations. In Intel Xeon Phi Processor High Performance Programming Knights Landing Edition.
- [IPDPS16] A. Breuer, A. Heinecke, M. Bader: Petascale Local Time Stepping for the ADER-DG Finite Element Method. In Parallel and Distributed Processing Symposium, 2016 IEEE International. http://dx.doi.org/10.1109/IPDPS.2016.109
- [ISC15] A. Breuer, A. Heinecke, L. Rannabauer, M. Bader: High-Order ADER-DG Minimizes Energy- and Time-to-Solution of SeisSol. In 30th International Conference, ISC High Performance 2015, Frankfurt, Germany, July 12-16, 2015.
- [SC14] A. Heinecke, A. Breuer, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties, A. Bode, W. Barth, X.-K. Liao, K. Vaidyanathan, M. Smelyanskiy and P. Dubey: Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers. In Supercomputing 2014, The International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, New Orleans, LA, USA, November 2014. Gordon Bell Finalist.
- [ISC14] A. Breuer, A. Heinecke, S. Rettenberger, M. Bader, A.-A. Gabriel and C. Pelties: Sustained Petascale Performance of Seismic Simulations with SeisSol on SuperMUC. In J.M. Kunkel, T. T. Ludwig and H.W. Meuer (ed.), Supercomputing — 29th International Conference, ISC 2014, Volume 8488 of Lecture Notes in Computer Science. Springer, Heidelberg, June 2014. 2014 PRACE ISC Award.
- [PARCO13] A. Breuer, A. Heinecke, M. Bader and C. Pelties: Accelerating SeisSol by Generating Vectorized Code for Sparse Matrix Operators. In Parallel Computing — Accelerating Computational Science and Engineering (CSE), Volume 25 of Advances in Parallel Computing. IOS Press, April 2014.

Acknowledgements

Only the great support of experts at **NERSC, ALCF and TACC** made our extreme-scale results possible. In particular, we thank **J. Deslippe, S. Dosanjh, R. Gerber, K. Kumaran, and Dan Stanzione**. This work was supported through **NSF** Awards ACI-1450451, OCI-1148493, EAR-1349180 (AWP-ODC-OS).This work was supported by the Southern California Earthquake Center (**SCEC**) through contribution #16247 (EDGE). This work was supported by the **Intel Parallel Computing Center** program (AWP-ODC-OS and EDGE). This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. This work used the Extreme Science and Engineering Discovery Environment (**XSEDE**), which is supported by National Science Foundation grant number ACI-1053575. This research is part of the **Blue Waters** sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

EDGE heavily relies on contributions of many authors to open-source software.
This software includes, but is not limited to: **ASan** (https://clang.llvm.org/docs/AddressSanitizer.html, debugging), **Catch** (https://github.com/philsquared/Catch, unit tests), **CGAL** (http://www.cgal.org, surface meshes), **Clang** (https://clang.llvm.org/, compilation), **Cppcheck** (http://cppcheck.sourceforge.net/, static code analysis), **Easylogging++** (https://github.com/easylogging/, logging), **ExprTk** (http://partow.net/programming/exprtk, expression parsing), **GCC** (https://gcc.gnu.org/, compilation), **Git** (https://git-scm.com, versioning), **Git LFS** (https://git-lfs.github.com, versioning), **gitbook** (https://github.com/GitbookIO/gitbook, documentation), **Gmsh** (http://gmsh.info/, volume meshing), **GoCD** (https://www.gocd.io/, continuous delivery), **HDF5** (https://www.hdfgroup.org/HDF5/, I/O), **jekyll** (https://jekyllrb.com, homepage), **LIBXSMM** (https://github.com/hfp/libxsmm, matrix kernels), **MOAB** (http://sigma.mcs.anl.gov/moab-library/, mesh interface), **NetCDF** (https://www.unidata.ucar.edu/software/netcdf/, I/O), **ParaView** (http://www.paraview.org/, visualization), **pugixml** (http://pugixml.org/, XML interface), **SCons** (http://scons.org/, build scripts), **Valgrind** (http://valgrind.org/, memory debugging), **Visit** (https://wci.llnl.gov/simulation/computer-codes/visit, visualization).