HIGH ORDER SEISMIC SIMULATIONS ON THE INTEL® XEON PHI™ PROCESSOR (KNIGHTS LANDING)

Alexander Heinecke, Alexander Breuer, Michael Bader, Pradeep Dubey

Parallel Computing Lab, Intel Labs, USA
2016-06-22 International Supercomputing Conference (ISC)
Frankfurt, Germany

Legal Disclaimer & Optimization Notice

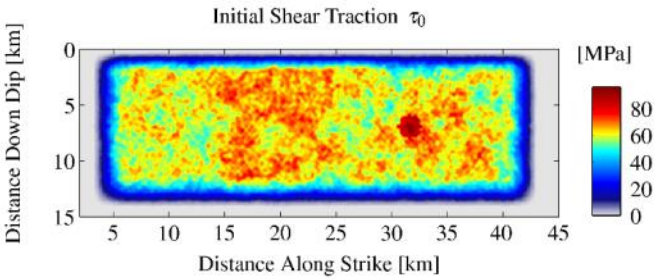
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may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

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 SSSE3 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.

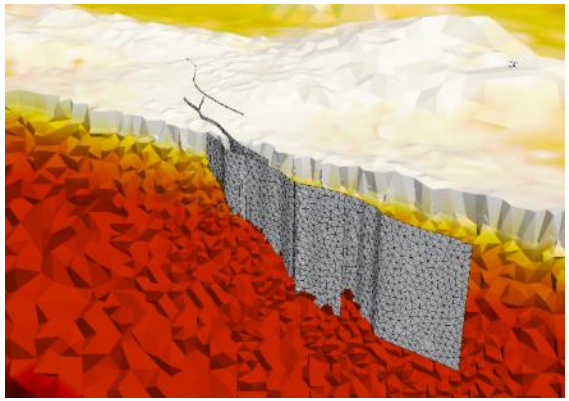
Notice revision #20110804

Seismic Wave Propagation & Dynamic Rupture Simulation

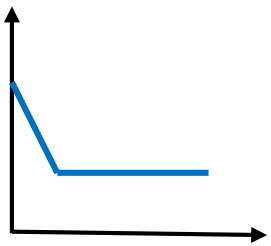


(Brietzke et al. (2009))

Initial Fault Stresses



Geological Structure
(Fault Geometry & Material Properties)



Failure Criterion

SeisSol

Ground Shaking
(Seismograms),
Fault Split, etc.

$$Q_k^{n+1} = Q_k^n + \mathcal{V}_k - \sum_{i=1}^{4} \mathcal{F}_{k,i}.$$

- Full elastic wave equations in 3D and complex heterogeneous media
- Dynamic Rupture without artificial oscillations
- High order: ADER(time)-DG(space)

- Unstructured tetrahedral meshes
- Highly Optimized Compute Kernels
- multi PFLOPS, GB14 finalist
- Local Time Stepping

Outline

- Reprise: Knights Landing Architecture
- SeisSol’s compute Routines in a Nutshell
- SeisSol’s Optimizations for Intel Xeon Phi processor – Knights Landing
- Performance Discussion

Intel® Xeon Phi™ Processor

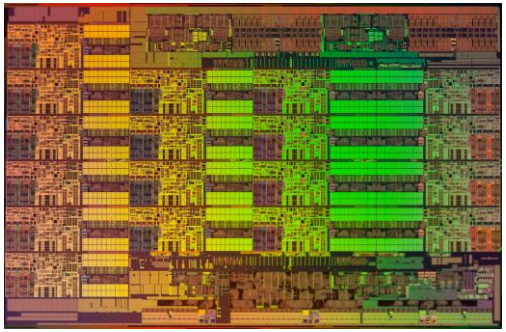
(code-named Knights Landing)

Current & Next Generation Intel® Xeon and Xeon Phi™ Platforms

Xeon*

Latest released – Broadwell (14nm process)

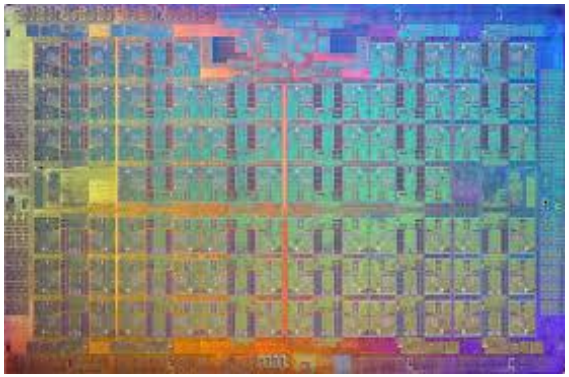
- Intel's Foundation of HPC Performance
- Up to 22 cores, Hyperthreading
- ~66 GB/s stream memory BW (4 ch. DDR4 2400)
- AVX2 – 256-bit (4 DP, 8 SP flops) -> >0.7 TFLOPS
- 20 PCIe lanes



Xeon Phi*

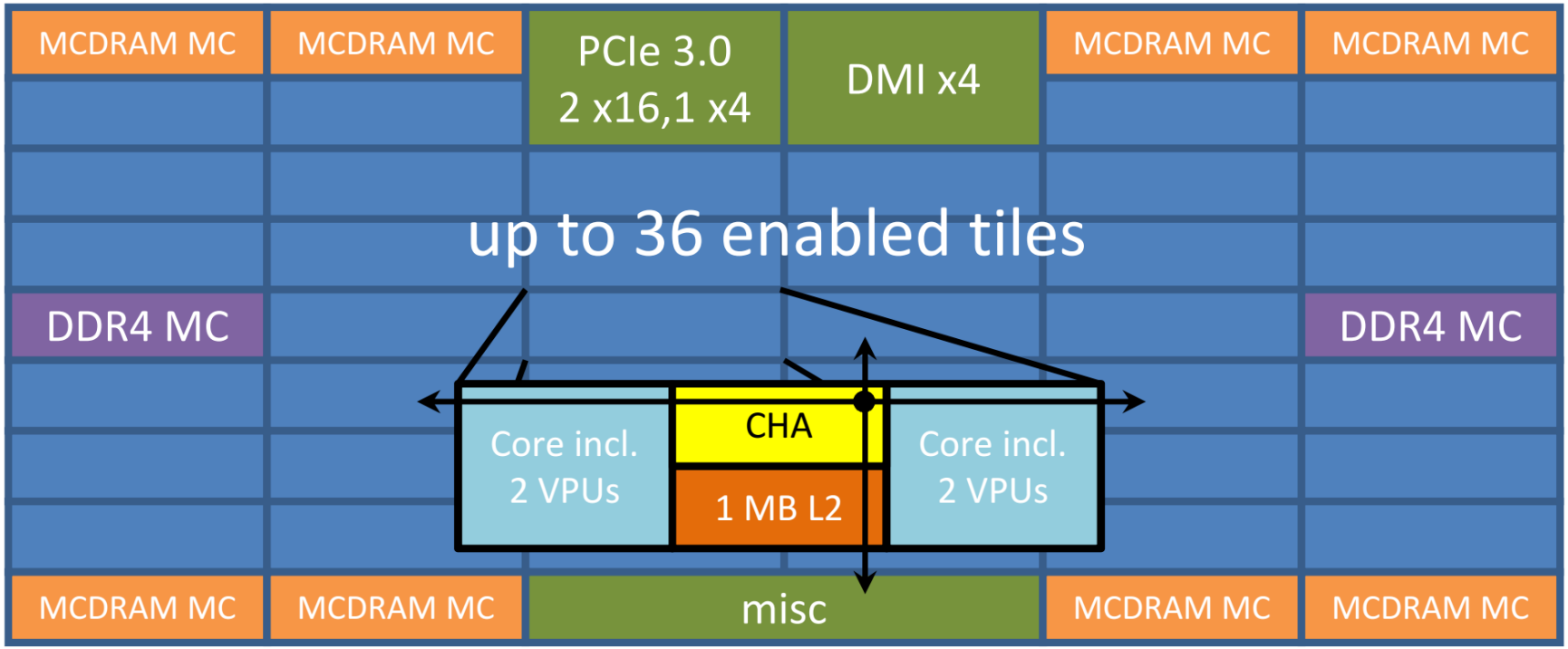
Knights Landing (14nm process)

- Optimized for highly parallelized compute intensive workloads
- Common programming model & S/W tools with Xeon processors, enabling efficient app readiness and performance tuning
- up to 72 cores, 490 GB/s stream BW, on-die 2D mesh
- AVX512– 512-bit (8 DP, 16 SP flops) -> >3 TFLOPS
- 36 PCIe lanes



*Intel Xeon and Intel Xeon Phi are trademarks of Intel Coporation in the US and/or other countries.

Intel Xeon Phi processor (Knights Landing)



- Self boot and binary compatible with main line IA (x87, SSE, AVX, AVX512)
- Boots standard OS, such as Linux or Windows.
- Significant improvement in scalar (heavily-modified Intel Atom core) and vector (2x AVX512 VPU per core) performance.
- Memory on package: innovative memory architecture for high bandwidth and high capacity
- Fabric on package connected with 32 PCIe Gen 3 lanes for 200 Gbps

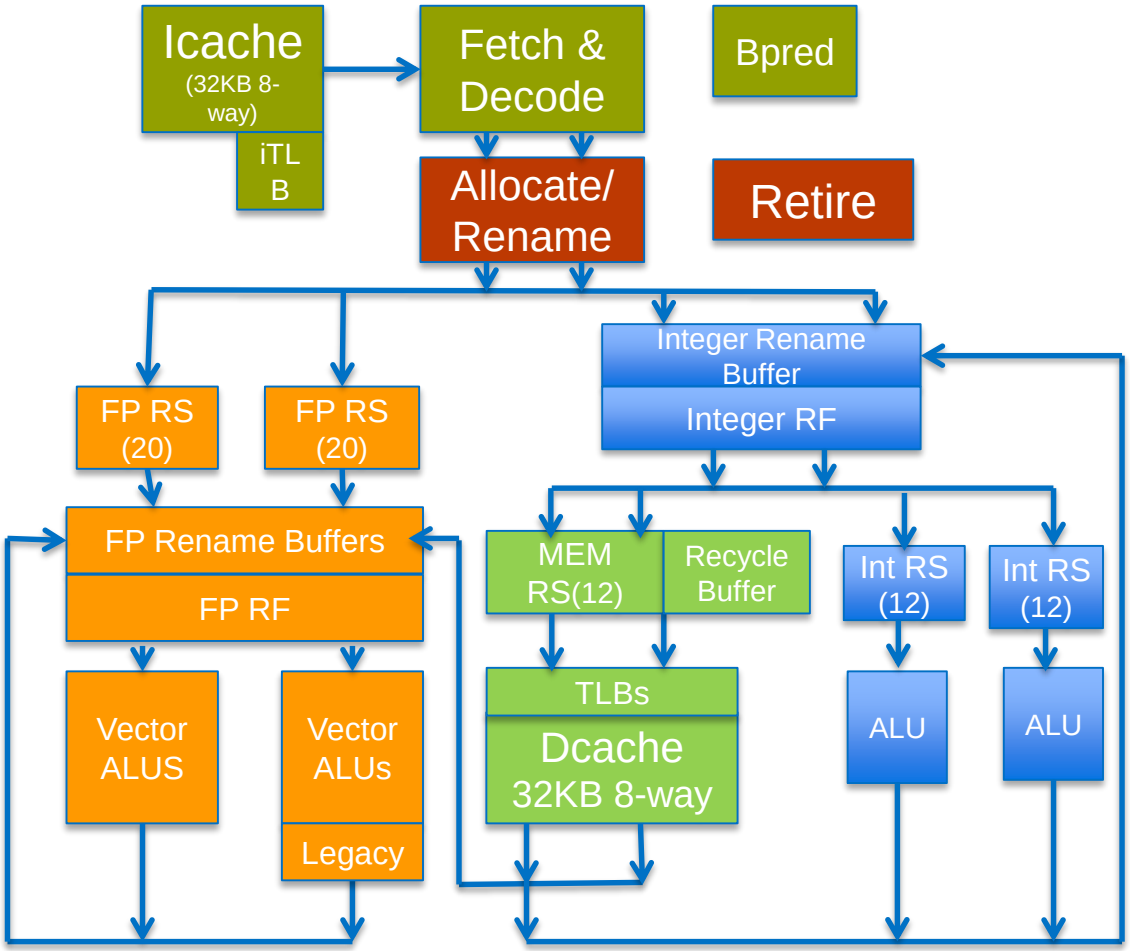
Core & VPU

Balanced power efficiency, single thread performance and parallel performance

2-wide Out-of-order core

4 SMT threads

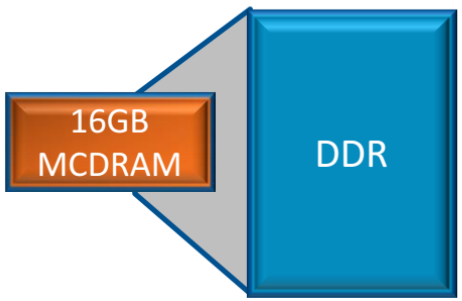
- 72 in-flight instructions.
- 6-wide execution
- 64 SP and 32 DP Flop/cycle
- Dual ported DL1 → to feed 2 VPU
- Two-level TLB. Large page support
- Gather/Scatter engine
- Unaligned load/store support
- Core resources **shared** or **dynamically repartitioned** between active threads
- General purpose IA core



Memory Modes of Xeon Phi 72xx

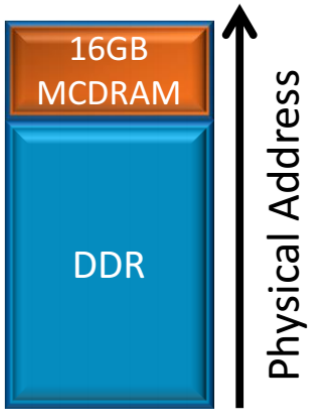
Three Modes. Selected at boot

Cache Mode



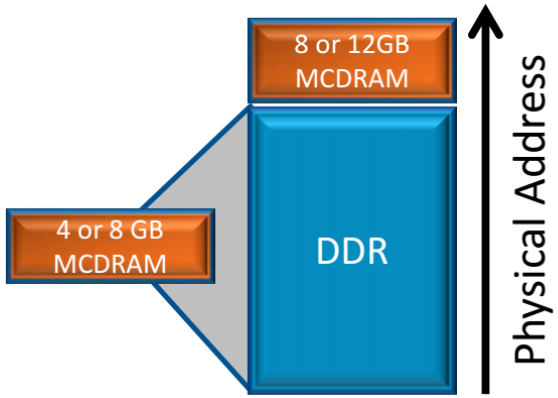
- SW-Transparent, Mem-side cache
- Direct mapped. 64B lines.
- Tags part of line
- Covers whole DDR range

Flat Mode



- MCDRAM as regular memory
- SW-Managed
- Same address space

Hybrid Mode



- Part cache, Part memory
- 25% or 50% cache
- Benefits of both

Taken from: A. Sodani: Knights Landing (KNL): 2nd Generation Intel Xeon Phi Processor, Hotchips '15

SeisSol and LIBXSMM

SeisSol's Compute Kernels – Time Integration

Recursive Scheme

$$\mathcal{T}_k := \mathcal{T}_k\left(Q_k^n, \Delta t\right) = \sum_{j=0}^{\mathcal{O}-1} \frac{\Delta t^{j+1}}{(j+1)!} \frac{\partial^j}{\partial t^j} Q_k(t^n)$$

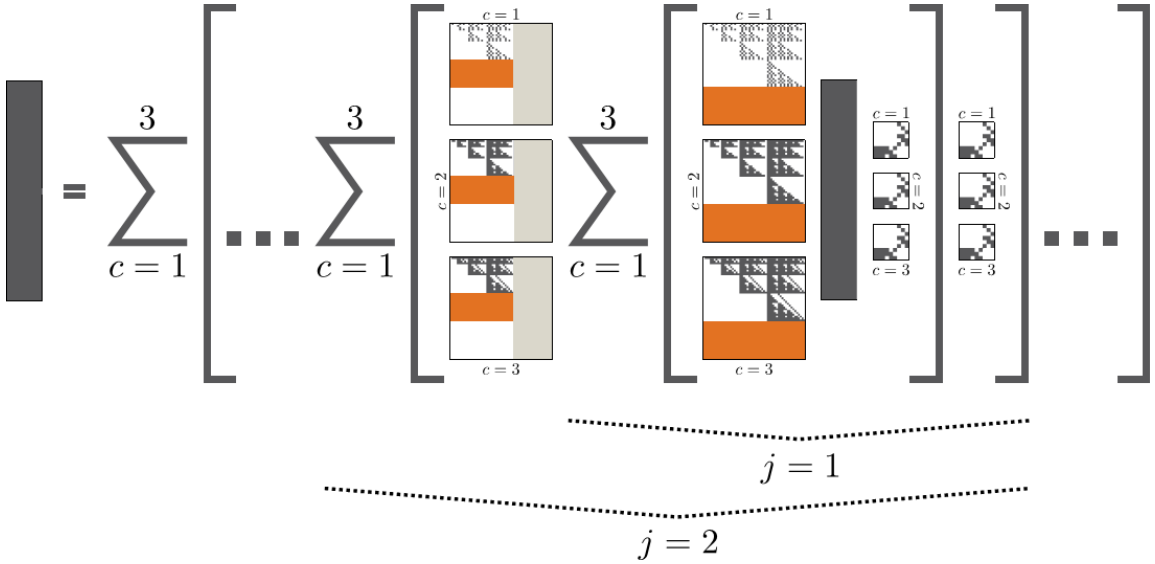
$$\frac{\partial^{j+1}}{\partial t^{j+1}} Q_k = -\sum_{c=1}^{3} \hat{K}^{\xi_c} \left(\frac{\partial^j}{\partial t^j} Q_k\right) A^{\star}_{k,c}. \quad \text{with} \quad \frac{\partial^0}{\partial t^0} Q_k = Q_k^n$$

Zero blocks in $\widehat{K}^{\xi_1}$, $\widehat{K}^{\xi_2}$ and $\widehat{K}^{\xi_3}$ lead to zero values in the degrees of freedom Q_k^n

Matrix size is reduced in each recursive step

Zero values in Q_k^n also appear in the multiplications with $A^*_{k,1}$, $A^*_{k,2}$, and $A^*_{k,3}$

Typical Matrix sizes of production runs (converge

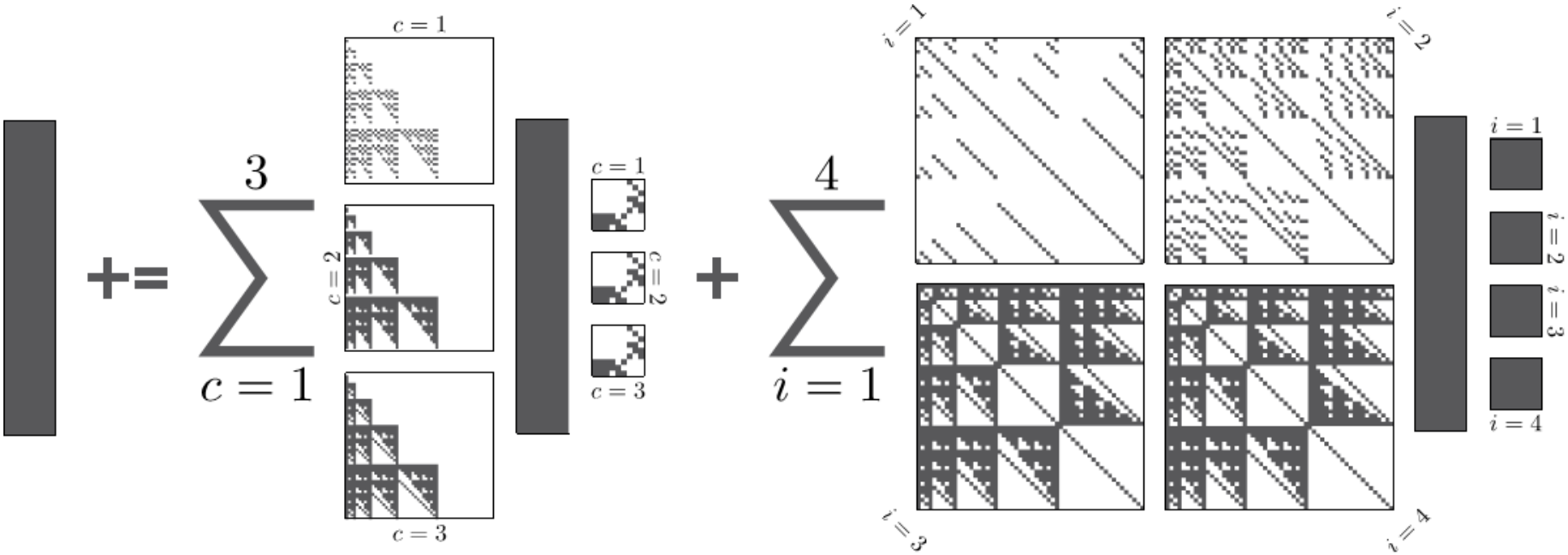


SeisSol's Compute Kernels – Local Integration

Small Matrix-Matrix multiplications

Typical Matrix sizes of production runs (convergence order 6): $9x9, 56x9, 56x35$

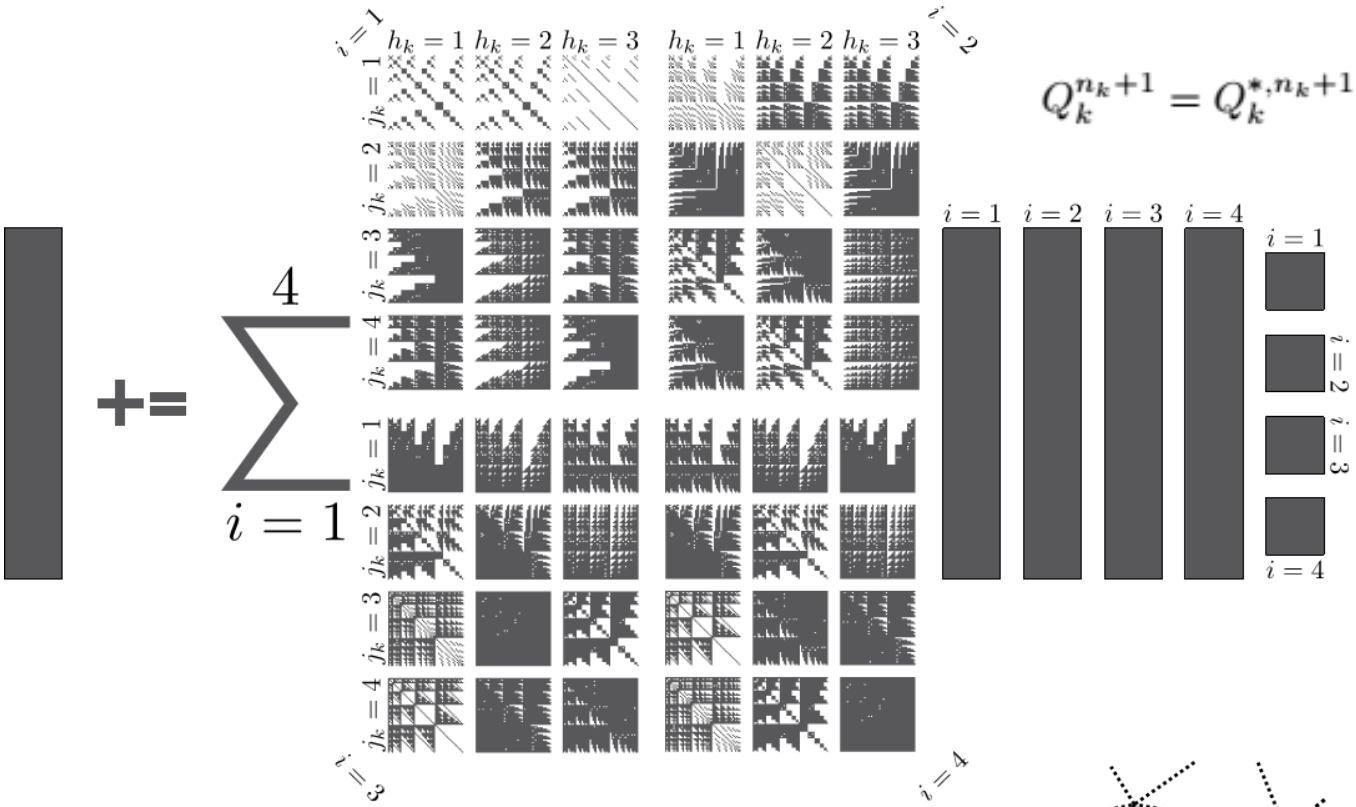
A priori known sparsity patterns



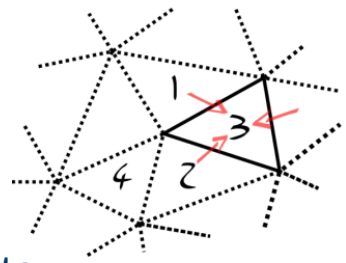
$$\mathcal{V}_k(\mathcal{T}_k) = \sum_{c=1}^{3} \tilde{K}^{\xi_c}(\mathcal{T}_k) A_k^{\xi_c}$$

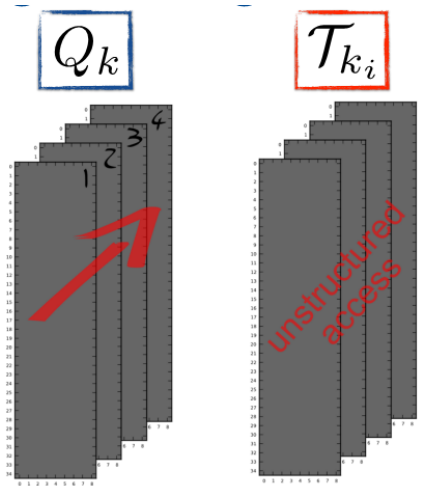
$$Q_k^{*,n_k+1} = Q_k^{n_k} + \mathcal{V}_k - \sum_{i=1}^{4} \hat{F}^{-,i}(\mathcal{T}_k) \hat{A}_{k,i}^{-}$$

SeisSol's Compute Kernels – Neighbor Integration



$$Q_k^{n_k+1} = Q_k^{*,n_k+1} - \sum_{i=1}^{4} \hat{F}^{+,i,j_k,h_k}\left(\mathcal{T}_{k_i}\right) \hat{A}_{k,i}^{+}$$





Unstructured Access to:

- Flux matrices
- Time integrated DOFs of neighboring elements

SeisSol Key Compute Kernels

- **Local Integration** (partly L1-cache BW bound, linear memory accesses)
  - Consist of time integration, volume integration and local part of boundary integration
  - 9 element local matrices (3 9x9, 4 9x9, 2 56x9) and 10 global matrices (3 40x56, 3 35x56, 4 56x56) (all number for sixth order)
  - **Flop/byte approx. 15** (sixth order)
- **Neighbor Integration** (in theory compute bound, irregular memory accesses and higher bandwidth)
  - 10 element local matrices (4 9x9, 6 56x9) and 4 out of 48 global matrices (4 56x56)
  - **Flop/byte approx. 2** (sixth order) on Xeon Phi as we cannot keep the 48 matrices in L2
- Sparse and Dense Matrix Multiplication of small sizes:
  - Due to unstructured meshes we need a prefetching stream that matches the mesh structure (not a regular DGEMM prefetching strategy)
  - Global (irregularly accessed) operators occupy close to 1.5 MB

  → **Intel MKL cannot by used, due to blocking and padding overheads. Using MKL instead of our highly tuned kernels results into 1.5-3X speed-down depending on order.**

  → **Batched GEMM is not beneficial as it would mean loosing locality**

  → **We leverage and optimized LIBXSMM to speed-up SeisSol's compute kernels**

Simultaneous Use of DDR4 and MCDRAM

- Low bandwidth requirements in local integration, high bandwidth requirements in neigh integration (behavior changes with convergence order!)
- Idea: Place as few as possible data structures in MCDRAM to make efficient use of MCDRAM, e.g. running large problem exceeding MCDRAM, we use memkind for this, https://github.com/memkind/memkind

order	Q_k	$\mathcal{B}_k, \mathcal{D}_k$	$A_k^{\xi_c}, \hat{A}_k^{-,i}, \hat{A}_k^{+,i}$	$\hat{K}^{\xi_c}, \tilde{K}^{\xi_c}, \hat{F}^{-,i}, \hat{F}^{+,i,j,h}$
2	MCDRAM	MCDRAM	MCDRAM	MCDRAM
3	MCDRAM	MCDRAM	MCDRAM	MCDRAM
4	DDR4	MCDRAM	MCDRAM	MCDRAM
5	DDR4	MCDRAM	DDR4	MCDRAM
6	DDR4	MCDRAM	DDR4	MCDRAM

unknowns, element-local, e.g. 1x 56x9

unknowns, element-local, e.g. 1x 56x9

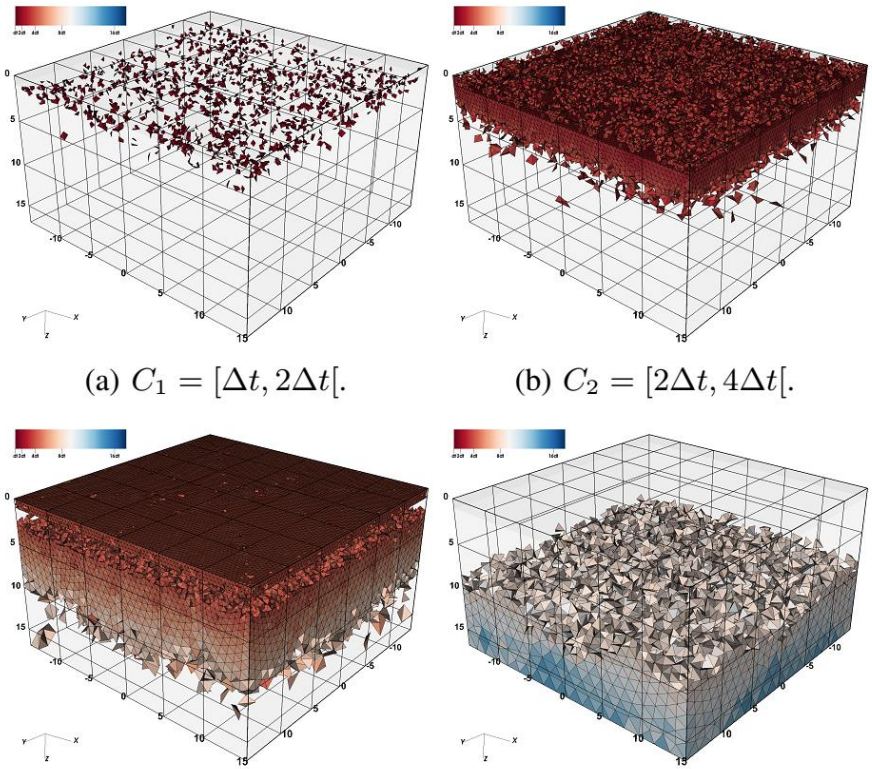
star, flux solver, element-local, 11x 9x9

Stiffness and Flux matrices, flux solver, global, e.g. 58x 56x56

→ **At high-order, ~70% of the data can be stored in DDR4 using an out-of-core scheme**

Global (GTS) vs. Local (LTS) Time Stepping

- SeisSol has to respect the CFL-condition (max. time step width is determined by a tet's in-sphere radius), this is element-local!
- An unstructured mesh with N elements might have N time step widths
- Global Time Stepping (GTS): all elements are forced to the minimal time step of the entire mesh! Not beneficial, but common practice in most of these codes!
- Local Time Stepping (LTS): all elements can advance in time independent from each other! Horrible, since not scalable!
- SeisSol: Clustered Local Time Stepping (cLTS)
    - instead of one big loop we have several smaller loops with different trip counts
    - We did all possible optimization to implement cLTS on modern multi/many core SIMD hardware -> IPDPS 16, scales to ~90K cores.



(a) $C_1 = [\Delta t, 2\Delta t[$. (b) $C_2 = [2\Delta t, 4\Delta t[$.

(c) $C_3 = [4\Delta t, 8\Delta t[$. (d) $C_4 = [8\Delta t, 16\Delta t[$.

Fig. 2: Spatial distribution of the first four clusters' elements for the LOH.1 setup and a rate-2 clustering. The location of the layer and the half-space is indicated by almost transparent gray boxes. The individual elements of the clusters are colored by their CFL time step.

LIBXSMM: Implementation

Interface (C/C++ and FORTRAN API)

Simplified interface for matrix-matrix multiplications

- $c_{m \times n} = c_{m \times n} + a_{m \times k} * b_{k \times n}$ (also full xGEMM)

Highly optimized assembly code generation (inline, *.s, JIT byte-code)

- SSE3, AVX, AVX2, IMCI (KNCni), and AVX-512
- AVX-512 code quality
  - Maximizes number of immediate operands
  - Limits instruction width to 16 Byte/cycle

High level code optimizations

- Implicitly aligned leading dimension (LDC) – allows aligned store instr.
- Aligned load instructions
- Sophisticated data prefetch

License

- Open Source Software (BSD 3-clause license)*, https://github.com/hfp/libxsmm

LIBXSMM AVX512 code for SeisSol (N=9)

```
vmovapd 1792(%rdi), %zmm4
vmovapd 2240(%rdi), %zmm5
vfmadd231pd 16(%rsi){1to8}, %zmm2, %zmm23
vfmadd231pd 16(%rsi,%r15,1){1to8}, %zmm2, %zmm24
vfmadd231pd 16(%rsi,%r15,2){1to8}, %zmm2, %zmm25
vfmadd231pd 16(%rax){1to8}, %zmm2, %zmm26
vfmadd231pd 16(%rsi,%r15,4){1to8}, %zmm2, %zmm27
vfmadd231pd 16(%rax,%r15,2){1to8}, %zmm2, %zmm28
vfmadd231pd 16(%rbx){1to8}, %zmm2, %zmm29
vfmadd231pd 16(%rax,%r15,4){1to8}, %zmm2, %zmm30
vfmadd231pd 16(%rsi,%r15,8){1to8}, %zmm2, %zmm31
vmovapd 2688(%rdi), %zmm6
vmovapd 3136(%rdi), %zmm7
vfmadd231pd 24(%rsi){1to8}, %zmm3, %zmm14
vfmadd231pd 24(%rsi,%r15,1){1to8}, %zmm3, %zmm15
vfmadd231pd 24(%rsi,%r15,2){1to8}, %zmm3, %zmm16
vfmadd231pd 24(%rax){1to8}, %zmm3, %zmm17
vfmadd231pd 24(%rsi,%r15,4){1to8}, %zmm3, %zmm18
vfmadd231pd 24(%rax,%r15,2){1to8}, %zmm3, %zmm19
vfmadd231pd 24(%rbx){1to8}, %zmm3, %zmm20
vfmadd231pd 24(%rax,%r15,4){1to8}, %zmm3, %zmm21
vfmadd231pd 24(%rsi,%r15,8){1to8}, %zmm3, %zmm22
vmovapd 3584(%rdi), %zmm0
```

→ **Max. theoretical efficiency: 90%!**

- column-major storage
- Working on all 9 columns and 8 rows simultaneously
- Loads to A (vmovapd) are spaced out to cover L1$ misses
- K-loop is fully unrolled
- B-elements are broadcasted within the FMA instruction to save execution slots
- SIB addressing mode to keep instruction size <= 8 byte for 2 decodes per cycle (16 byte I-fetch per cycle)
- Multiple accumulators (zmm31-xmm23 and zmm22-zmm14) for hiding FMA latencies

Performance Evaluation

Performance Benchmarking Systems

KNL: one Intel® Xeon Phi™ 7520 processor with 68 cores, 1.2 GHz AVX-base core-clock and 1.5 GHz all core Turbo frequency, 1.7 GHz mesh-clock, 16 GB MCDRAM@7.2 GT, 96\,GB DDR4-2400, FLAT/(CACHE or QUADRANT), one core reserved for OS

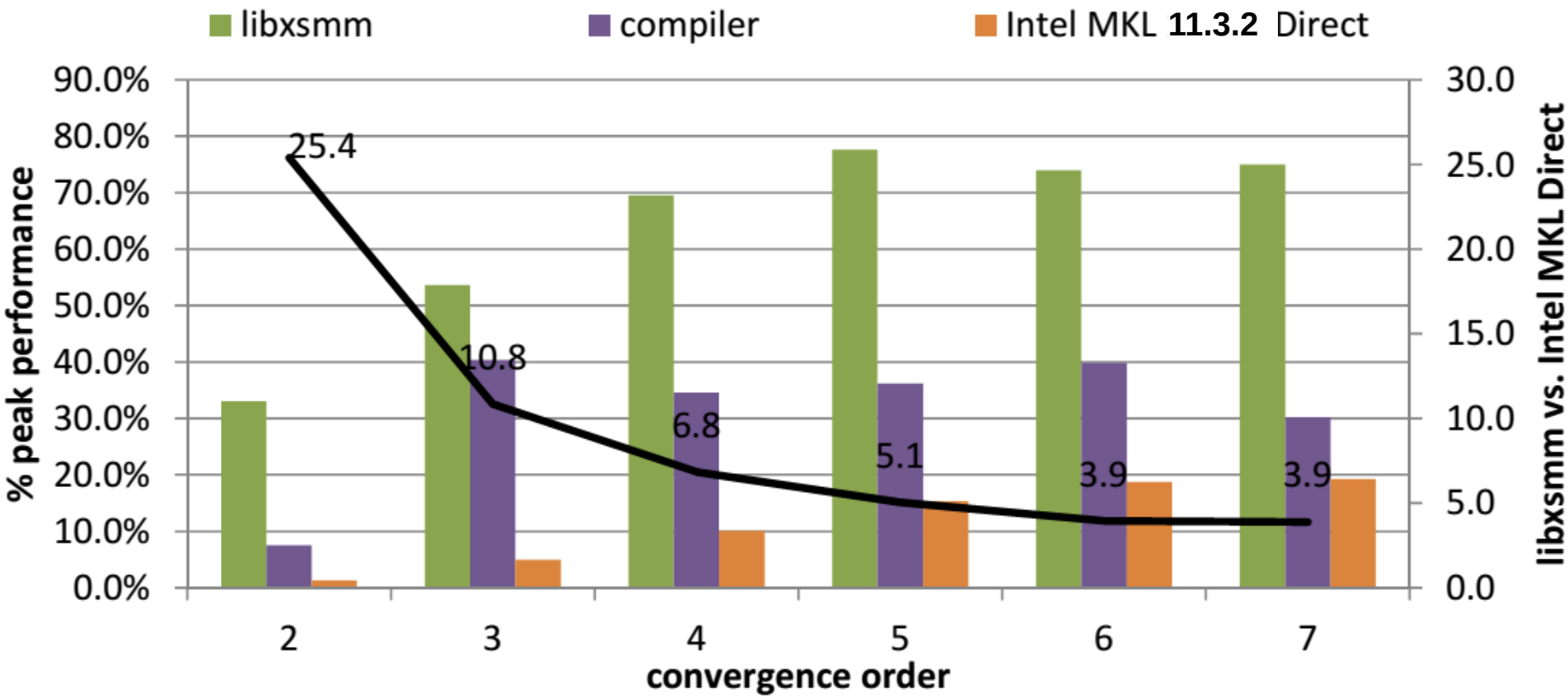
HSX: one Intel® Xeon® E5-2699v3 processor with 18 cores, 1.9 GHz at AVX-base frequency and up to 2.6 GHz Turbo frequency, 64 GB of DDR4-2133

KNC: one Intel® Xeon Phi™ 7120A coprocessor in native mode with 61 cores, 1.24 GHz base and 1.33 GHz Turbo frequency, 16 GB of GDDR5, one core reserved for OS

→We carry out single socket comparisons as Intel's reference platforms offer the same amount of KNL and HSX/BDX(Intel® Xeon® E5v4) sockets per rack-U!

Kernel Performance per Convergence Order

for selected, most important left-side operator, single core

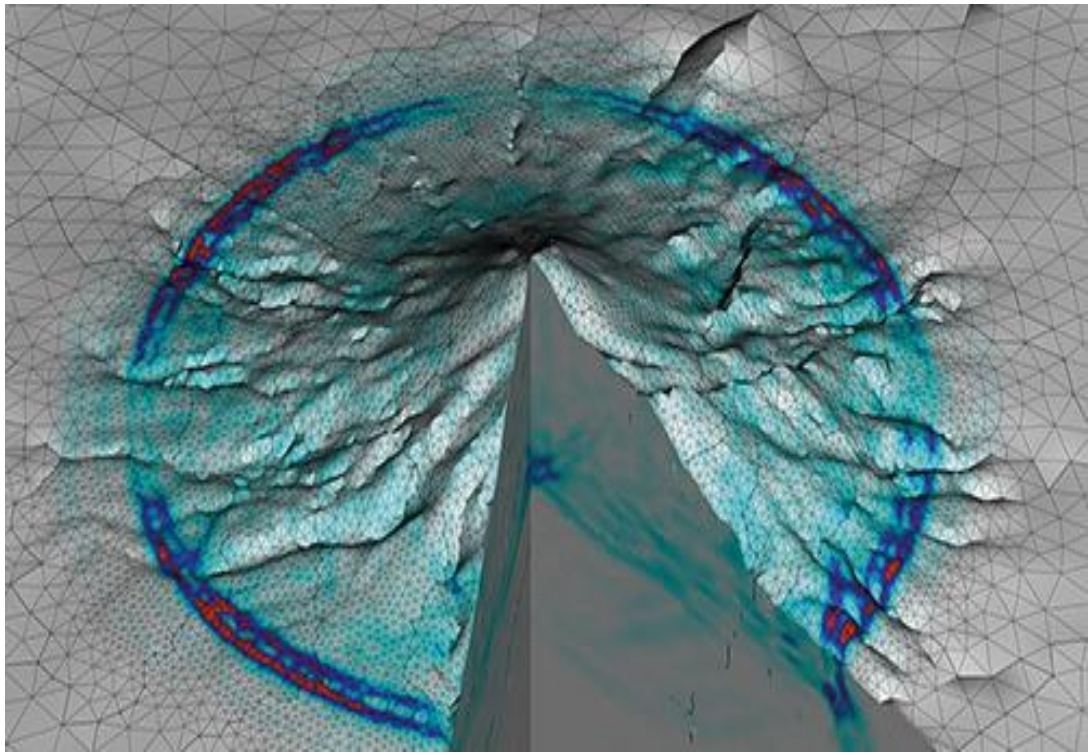


$B_{\mathcal{O}} \times 9 \times B_{\mathcal{O}}$ shapes $B_2 = 4$, $B_3 = 10$, $B_4 = 20$, $B_5 = 35$, $B_6 = 56$ and $B_7 = 84$

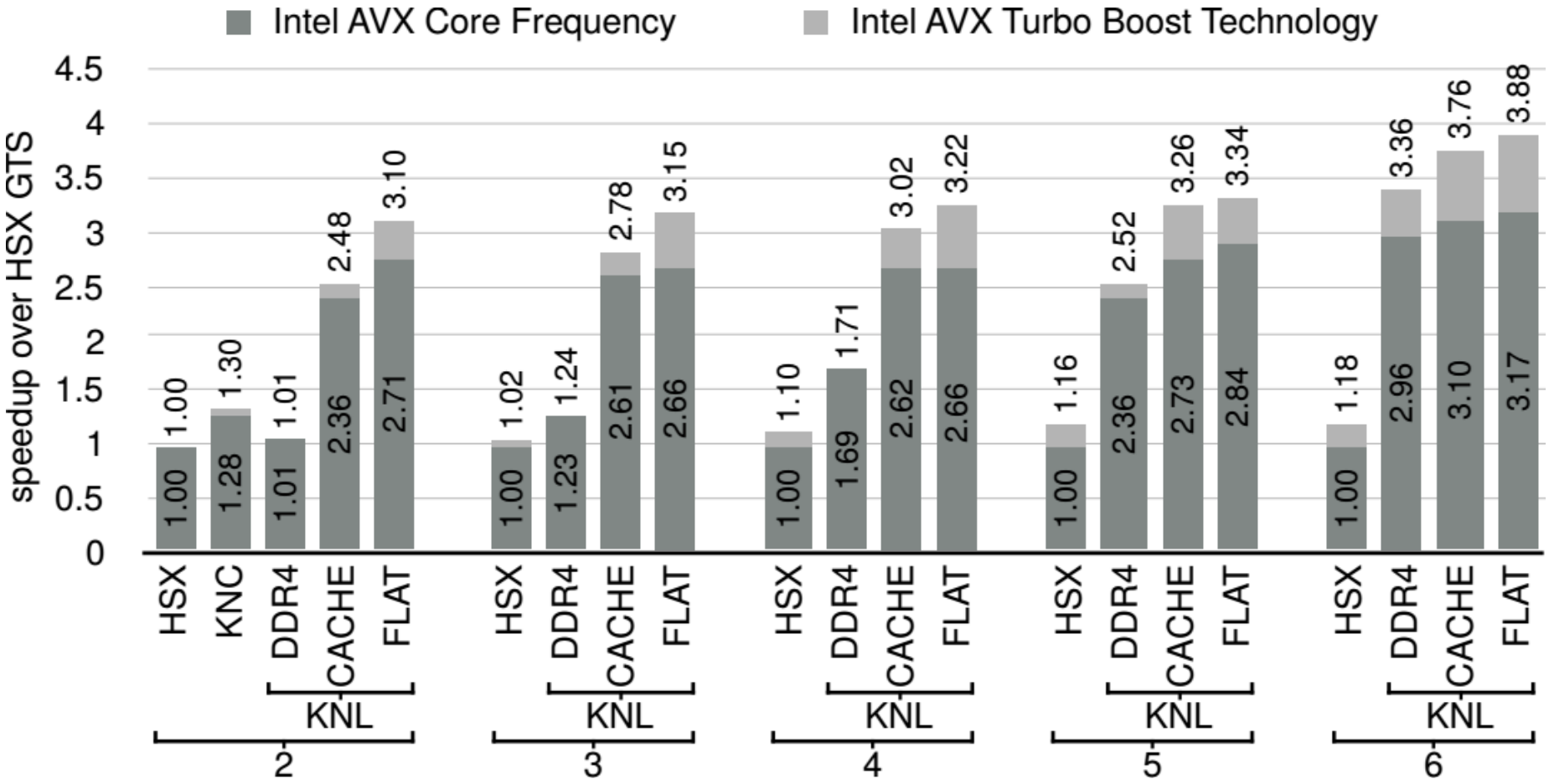
Not accounting for loop management, LD/ST to C matrix max. performance is 90% peak!
N=9 -> 1 LD + 9 FMAs is the basic block using all tricks gives 2 LD + 18 FMAs
-> 10 cycles out of which 9 are compute -> 90% peak.

Volcano Seismology: Mount Merapi in Indonesia

- Stripped down version of 2014 PRACE ISC Award winning simulation (ran on 156K cores) with 1.5M tetrahedrons (25-30 GB memory) and convergence order 6
- wave-propagation with free-surface boundary conditions on the surface and outflow boundary conditions
- GTS runs (cLTS in paper)
- Using MCDRAM and DDR4 simultaneously for optimal performance (“fast-running” data in MCDRAM, “slow-running” in DDR4



Mount Merapi GTS Performance



The LOH.1 Benchmark

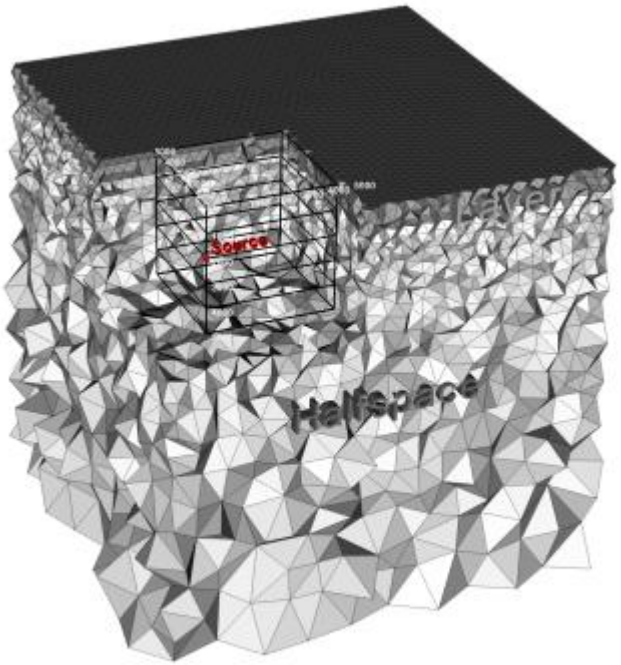
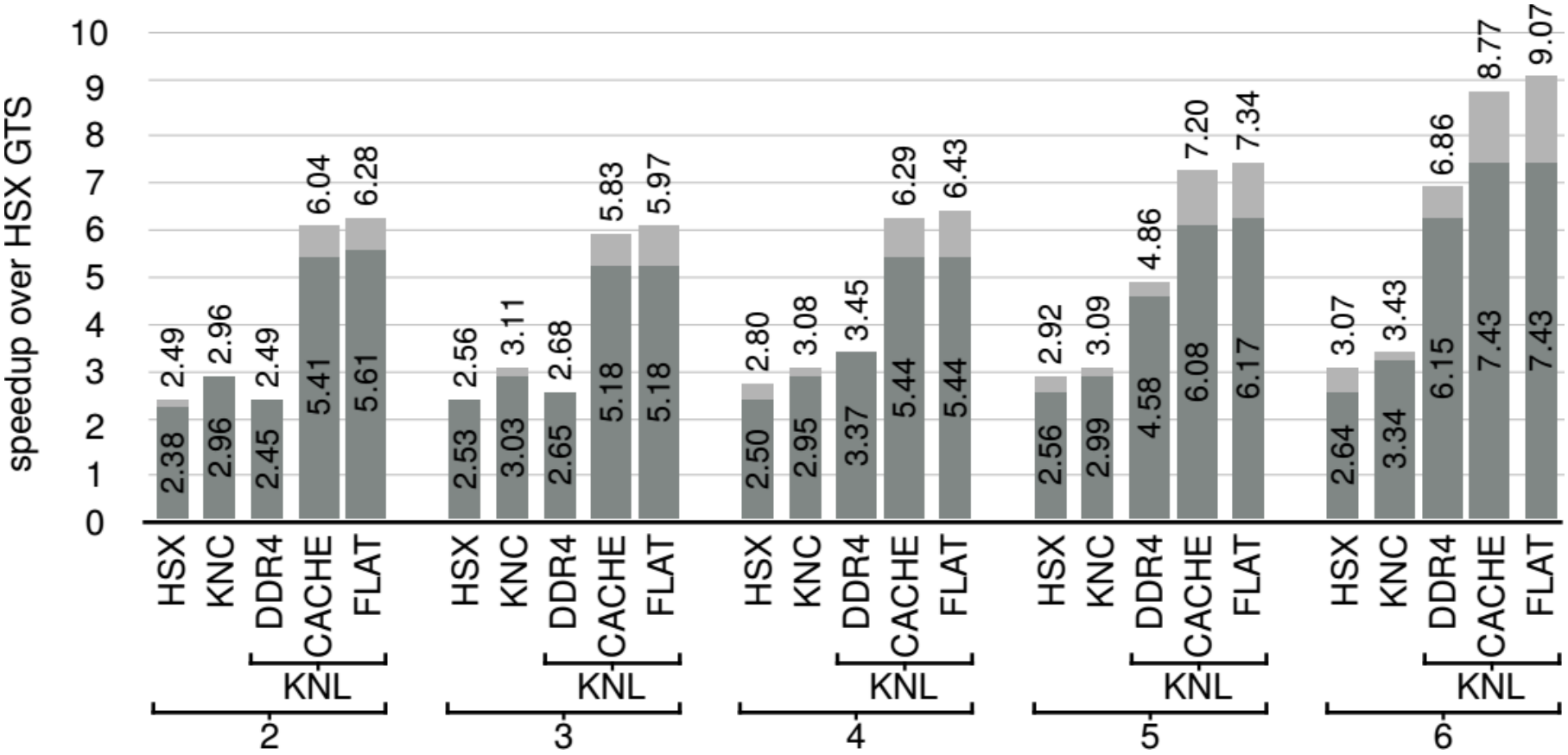


Fig. 5. Setup of the LOH.1 scenario.

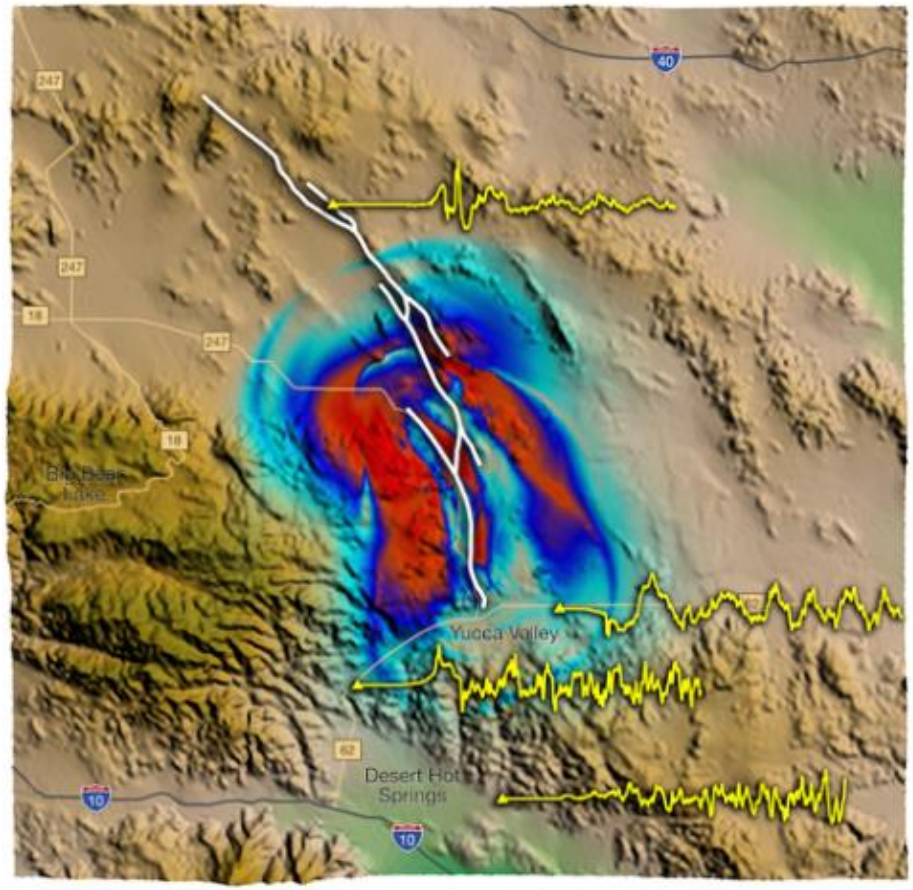
- computational domain extending [-15km, 15km]^2 X [0km, 17km]
- free-surface boundary conditions on the surface and outflow boundary conditions everywhere else. 386,518 elements and is unstructured
- the seismic source is located at (0,0,-2000). Shown is a only a part of size [-2km, 15km]^2 X [0,17 km]
- cLTS runs (GTS in paper)

LOH.1 cLTS Performance of GTS

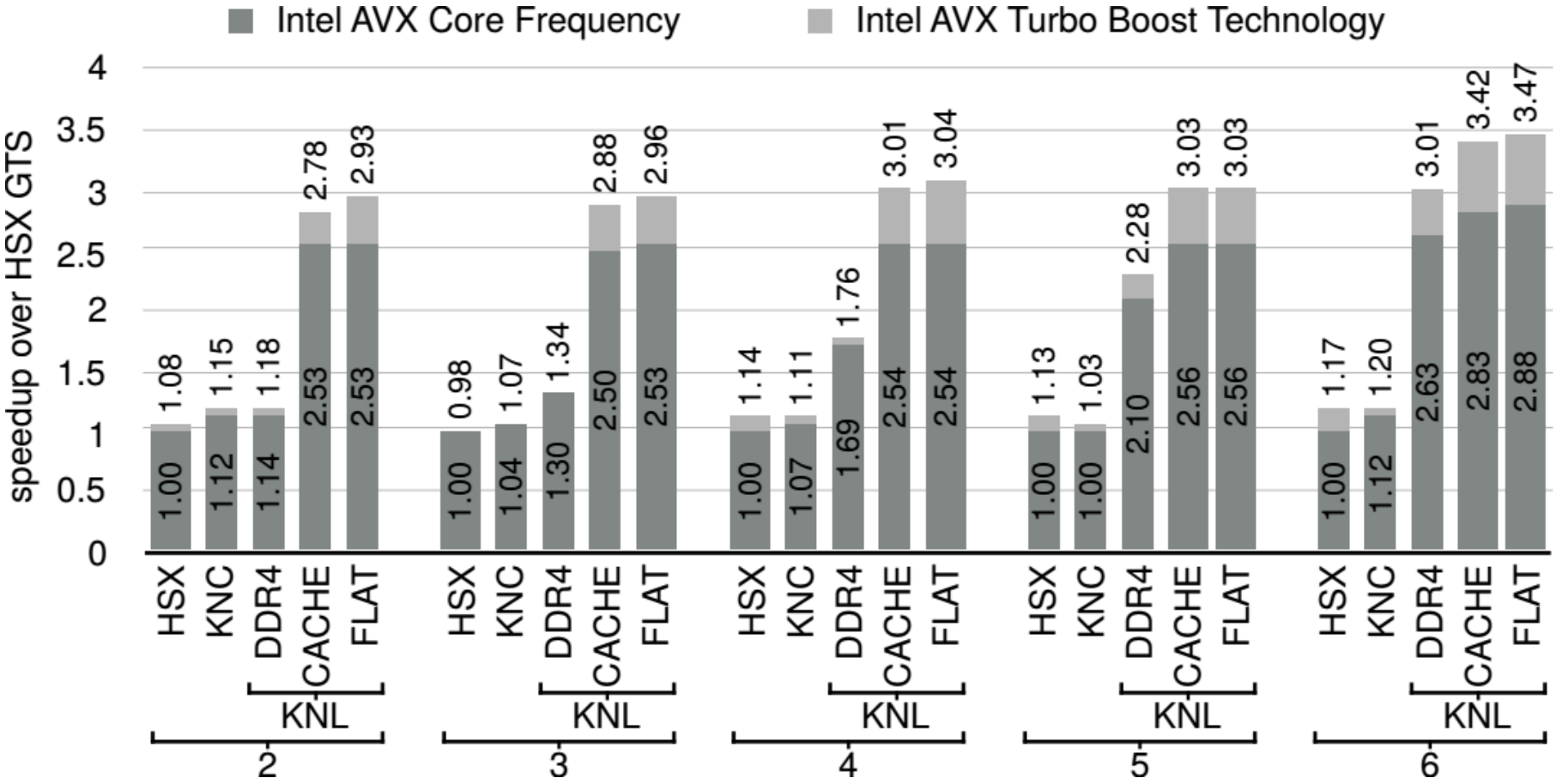


The M7.2 Landers 1992 Earthquake

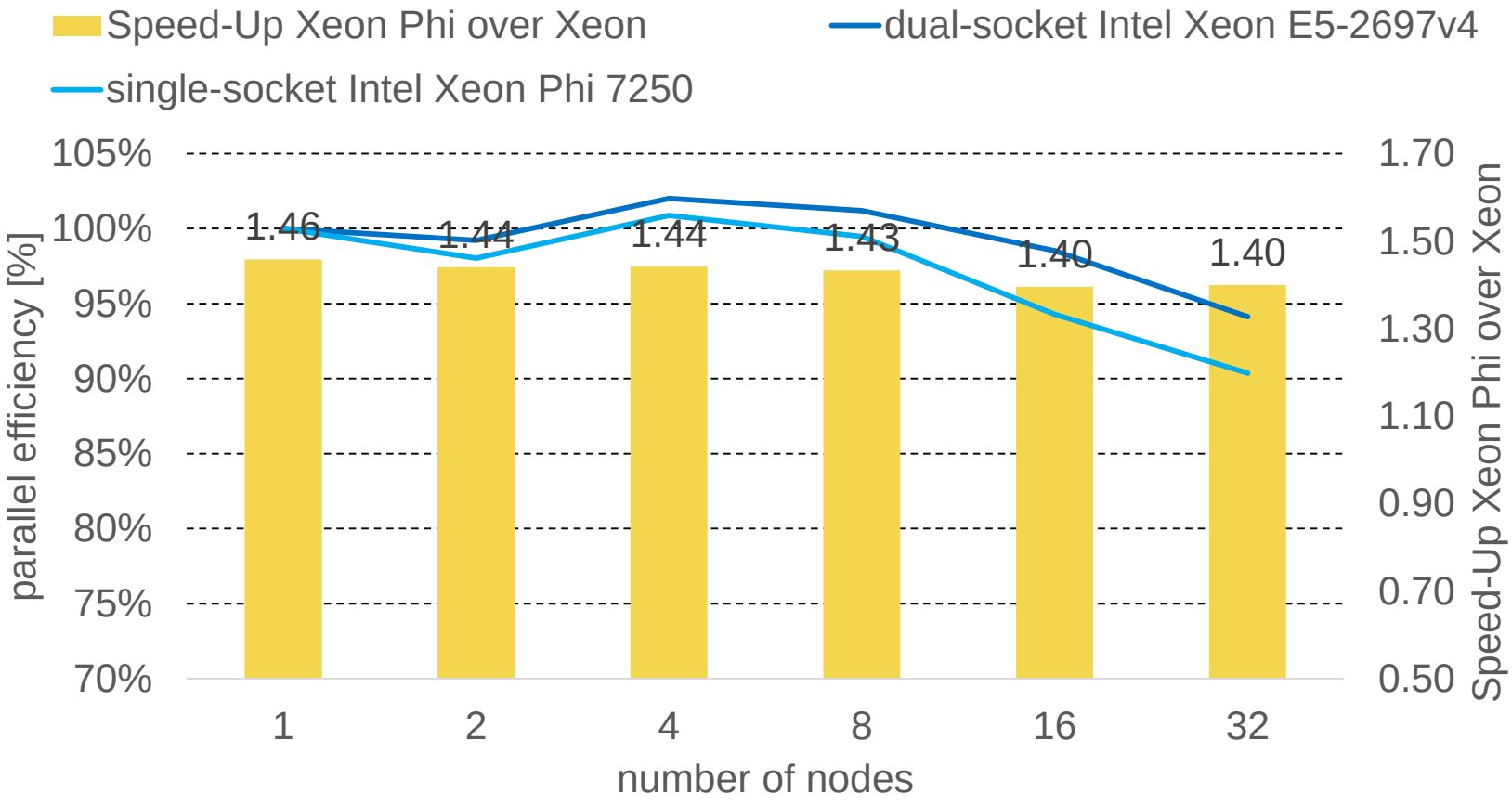
- Stripped down version of Gordon Bell Finalist 2014 run with 466,000 tetrahedrons
- Multi-physics dynamic rupture simulation
  - Scalar and branchy
  - In contrast to GB14 (Xeon did dynamic rupture and Xeon Phi wave propagation) we now run entirely on Xeon Phi
  - It's clearly visible that KNL's cores are much stronger than KNC's!
- GTS only, as theoretical foundation for (c)LTS+dynamic rupture is not stable (never been done before)



M7.2 Landers 1992 Performance



Cluster Strong Scaling of the Mount Merapi Setup



- On 32 nodes, only 47K elements per nodes, <1K per core!
- Measured on Intel's Endeavor Cluster. Each node is equipped with one Intel Omni-Path HFI PCIe x16 card offering 100 Gbps bandwidth.

Conclusion

- Intel Xeon Phi processor 72xx is able to outperform Intel Xeon E5v3 and Xeon Phi coprocessors 71xx by more than 3X in several setups!

- For best kernel level performance, we optimized the open source library small matrix multiplication library LIBXSMM for AVX512F, these optimization are automatically available to other workloads!

- We derived an out-of-core memory management scheme for ADER-DG FEM which allows to leverage Knights Landing innovative memory subsystem to run setups exceeding the high-bandwidth memory capacity without performance degradation!

- Although Xeon Phi offers a significant in single node performance, due to self-boot design, scalability doesn’t suffer!

intel®

experience
what's inside™