FIGHTING THE FLOP/BYTE GAP IN SCIENTIFIC COMPUTING BY SOLVING PDES IN HIGH ORDER

Alexander Heinecke
Parallel Computing Lab, Intel Labs, USA
2016-04-18
Imperial College London, United Kingdom

Legal Disclaimer & Optimization Notice

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

Outline

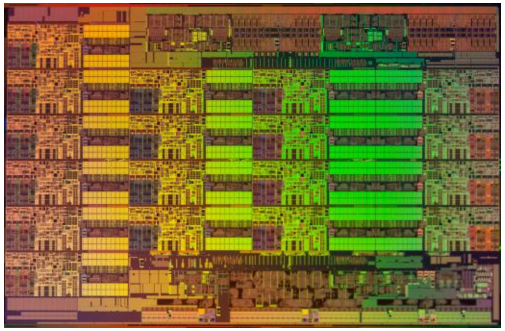
- Challenges and Settings in current HPC and Intel's Parallel Computing Lab
- High Order Earthquake Simulation – SeisSol
- High Order CFD – NekBox/Nek5000
- LIBXSMM
- Conclusion

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

Xeon*

Latest released – Broadwell (14nm process)

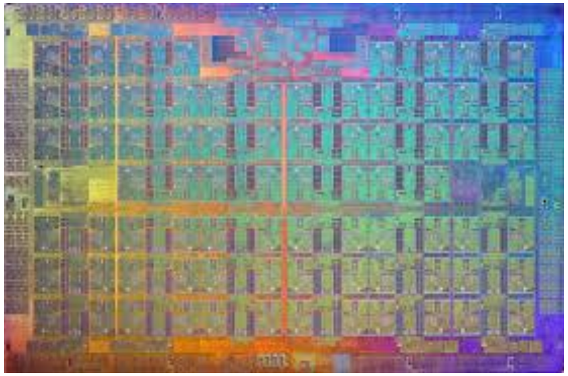
- Intel's Foundation of HPC Performance
- Up to 22 cores, Hyperthreading
- ~66 GB/s stream memory BW (4 ch. DDR4 2400)
- AVX2 – 256-bit (4 DP, 8 SP flops) -> 0.7 TFLOPS



Xeon Phi*

Knights Landing (Future generation, 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
- 60+ cores, 400+ GB/s stream BW, on-die 2D mesh
- AVX512– 512-bit (8 DP, 16 SP flops) -> >3 TFLOPS
- https://software.intel.com/en-us/articles/what-disclosures-has-intel-made-about-knights-landing



Modern HPC….

$$Speedup = \left(\frac{1}{Serial_{frac} + \frac{1 - Serial_{frac}}{\boldsymbol{NumCores}}}\right) * \left(\frac{1}{Scalar_{frac} + \frac{1 - Scalar_{frac}}{\boldsymbol{VectorLength}}}\right)$$

Goal: *Reduce* ***Serial Fraction*** and *Reduce* ***Scalar Fraction*** of Code

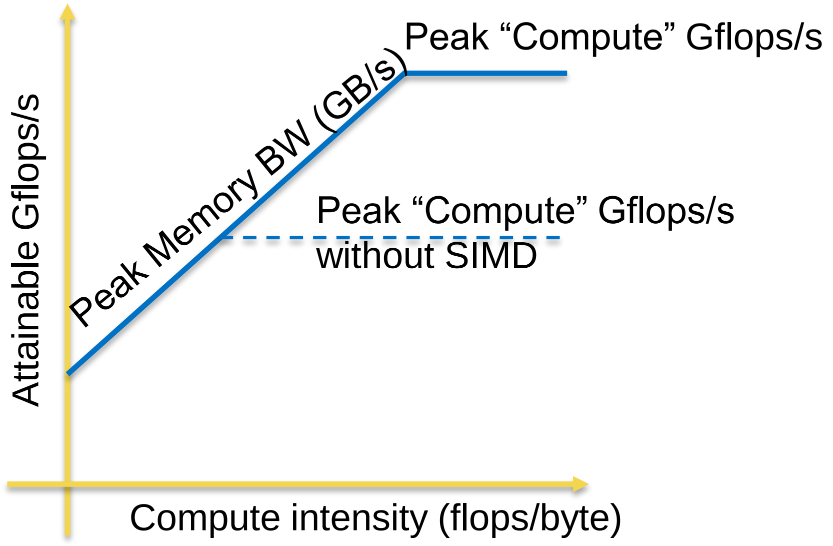
Ideal Speedup: **NumCores*VectorLength (requires zero scalar, zero serial work)**

Bandwidth Bound Performance

Most kernels of (CFD) codes are memory BW bound i.e. not enough compute intensity (flops/byte)

But: modern chips are unbalanced (high flops/ low byte/s)

Can we make better "use" of data after we have moved it into the chip?

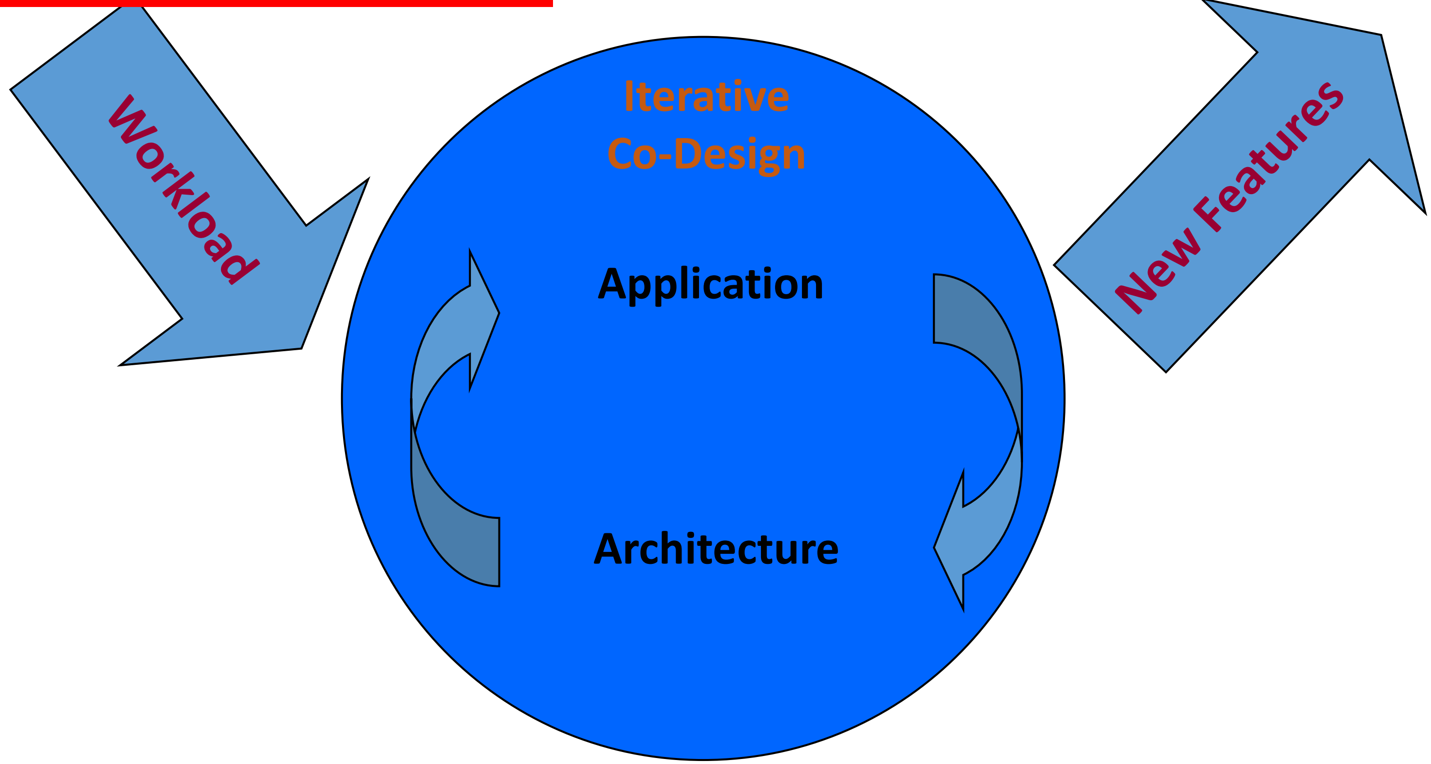


HPC is hard -> Our Approach at PCL

Existing and Emerging Workloads:

- **HPC**
- **Finance, Medical**
- **Big Data**

Parallel Algorithms, supporting hardware features and parallel libraries for single and multinode



SeisSol

High Order Earthquake Simulations

http://www.seissol.org

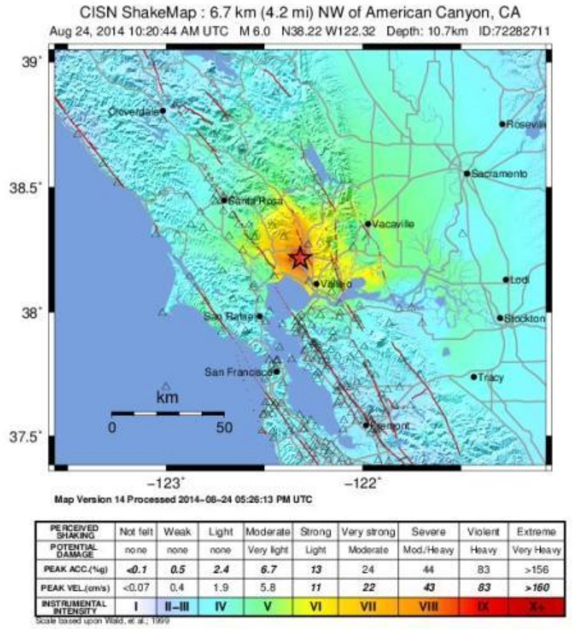
Joint Work with Michael Bader (TUM), Alexander Breuer (TUM, SDSC)

This slide deck is based on the ISC’15 paper with an ISC’16 preview

Motivation

“Development of more realistic implementations of dynamic or kinematic representations of fault rupture, including simulation of higher frequencies (up to 10+ Hz).”

Research Topics in GMP, 2014 Science Collaboration Plan, Southern California Earthquake Center (SCEC).

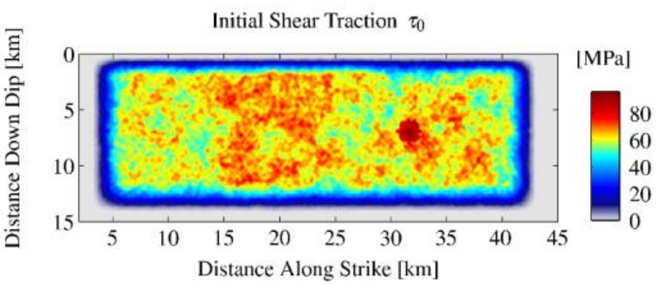


ShakeMap, M6.0, 2014-08-24, 3:20 am

Downtown Napa, Aug 24th, 2014. source: cnn.com

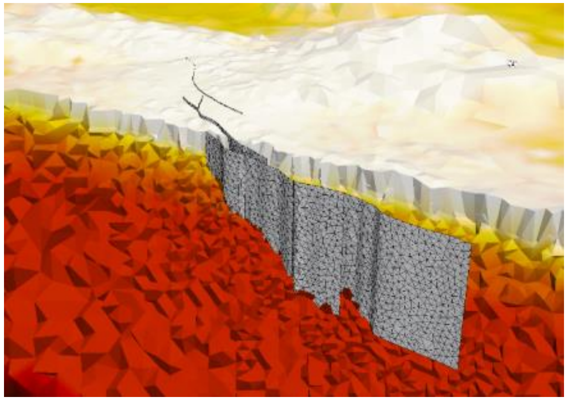
Downtown Napa, Nov 28th, 2015.

Seismic Wave Propagation & Dynamic Rupture Simulation

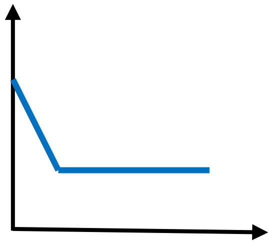


(Brietzke et al. (2009))

Initial Fault Stresses



Geological Structure
(Fault Geometry & Material Properties)



Failure Criterion

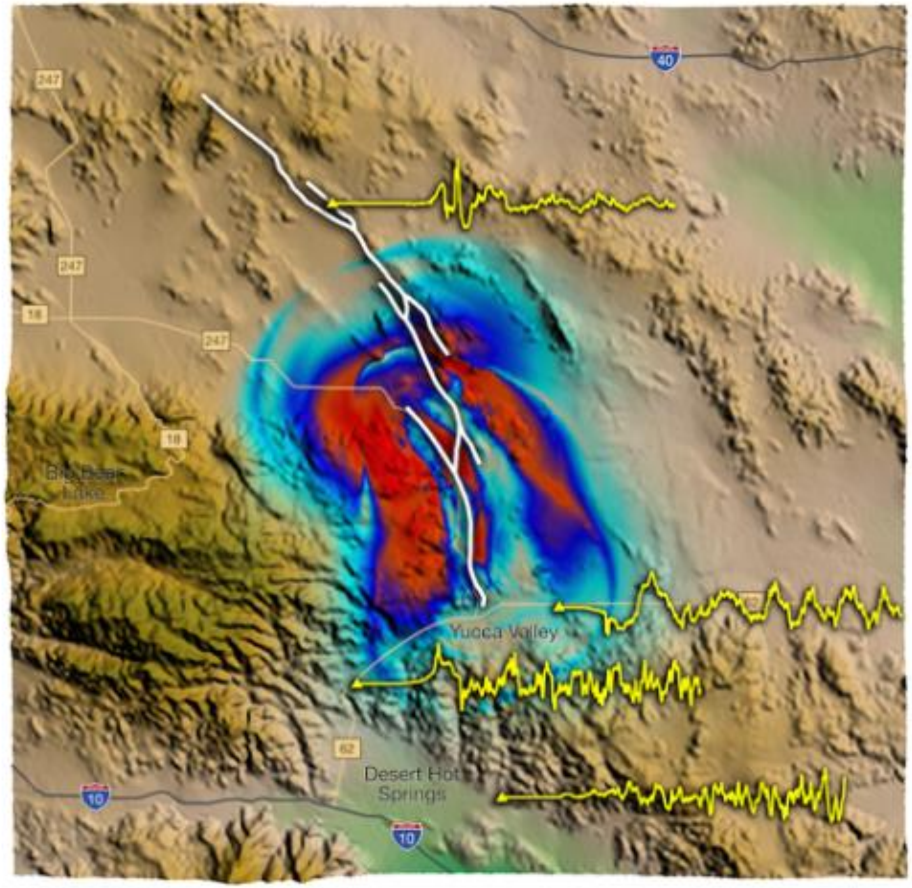
SeisSol

Ground Shaking
(Seismograms),
Fault Split, etc.

- 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
- Massively parallel

The M7.2 Landers 1992 Earthquake Simulation – Gordon Bell Finalist 14

- 191,098,540 tetrahedrons (~1300 per core of SuperMUC, ~130 per thread of Xeon Phi on Stampede)
- Production run SuperMUC:
  - 234,567 time steps equaling 42s simulated time
  - Output: 23 pick-points + high-res fault
  - 7h 15m @ 147,456 SNB-EP cores
  - 1.25 PFLOPs incl. setup and output!! (96.7% of scaling without setup and output), SuperMUC
  - 2 PFLOPS on Stampede
  - Frequencies up to 10Hz



Taken from a)

Discontinuous Galerkin in a Small Nutshell

- Most-flexible method
- seen as a mixed/hybrid method
  - FEM: polynomial (high-order) approximation within the element
  - FVM: inter-element convection: up winded numerical fluxes
- pretty new: first serious codes in the 2000s, first ever paper on Navier-Stokes with DG (CFD) in 1997.
- Very compact element-centric formulation
- Inter-element communication just across adjacent faces
  - h- and p-refinement simple
  - Unstructured meshes automatically supported
  - Local-Time-Stepping relatively simple
- high-order can lead to GEMM-like/demanding routines (e.g. SeisSol modal basis and Tets)

Deriving SeisSol's Compute Kernels

$$Q_k^{n+1} = Q_k^n - \mathcal{B}_k\left(\mathcal{J}_k^{n,n+1}, \mathcal{J}_{k(1)}^{n,n+1}, \ldots, \mathcal{J}_{k(4)}^{n,n+1}\right) + \mathcal{V}_k\left(\mathcal{J}_k^{n,n+1}\right)$$

SeisSol's Compute Kernels

$$\hat{q}_b^{n+1} = \hat{q}_b^n - \frac{1}{|J| m_b}\left(\int_{t^n}^{t^{n+1}} \int_{\partial T_k} \phi_b f(q) \cdot n \, d\vec{x} dt - \int_{t^n}^{t^{n+1}} \int_{T_k} \nabla \phi_b \cdot f(q) \, d\vec{x} dt\right)$$

DG-Formulation

$$q_t + A(\vec{x}) q_x + B(\vec{x}) q_y + C(\vec{x}) q_z = 0$$

Elastic Wave Equations

Taken from talk for a)

SeisSol's Compute Kernels – Time Kernel

Recursive Scheme

$$T_k(t^n, t^{n+1}, Q_k^n) = \sum_{j=0}^{O-1} \frac{(t^{n+1} - t^n)^{j+1}}{(j+1)!} \frac{\partial^j}{\partial t^j} Q_k(t^n)$$

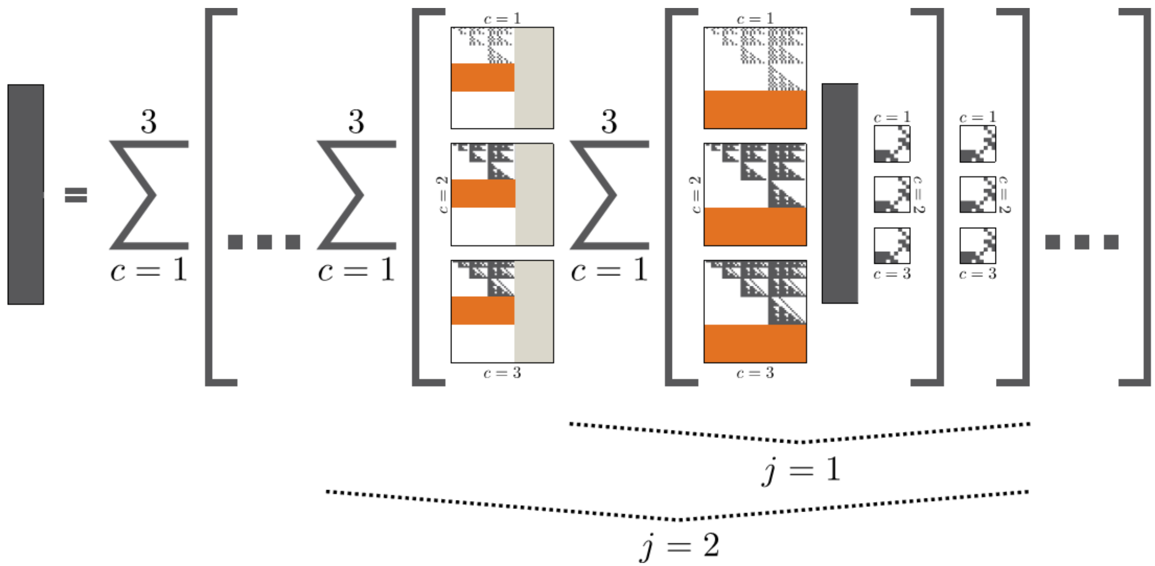
$$\frac{\partial^{j+1}}{\partial t^{j+1}} Q_k = -\hat{K}^\xi \left(\frac{\partial^j}{\partial t^j} Q_k \right) A_k^* - \hat{K}^\eta \left(\frac{\partial^j}{\partial t^j} Q_k \right) B_k^* - \hat{K}^\zeta \left(\frac{\partial^j}{\partial t^j} Q_k \right) C_k^*$$

Zero blocks in $\hat{K}^\xi$, $\hat{K}^\eta$ and $\hat{K}^\eta$ 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^*, B_k^* and C_k^*

Typical Matrix sizes of production runs (converge



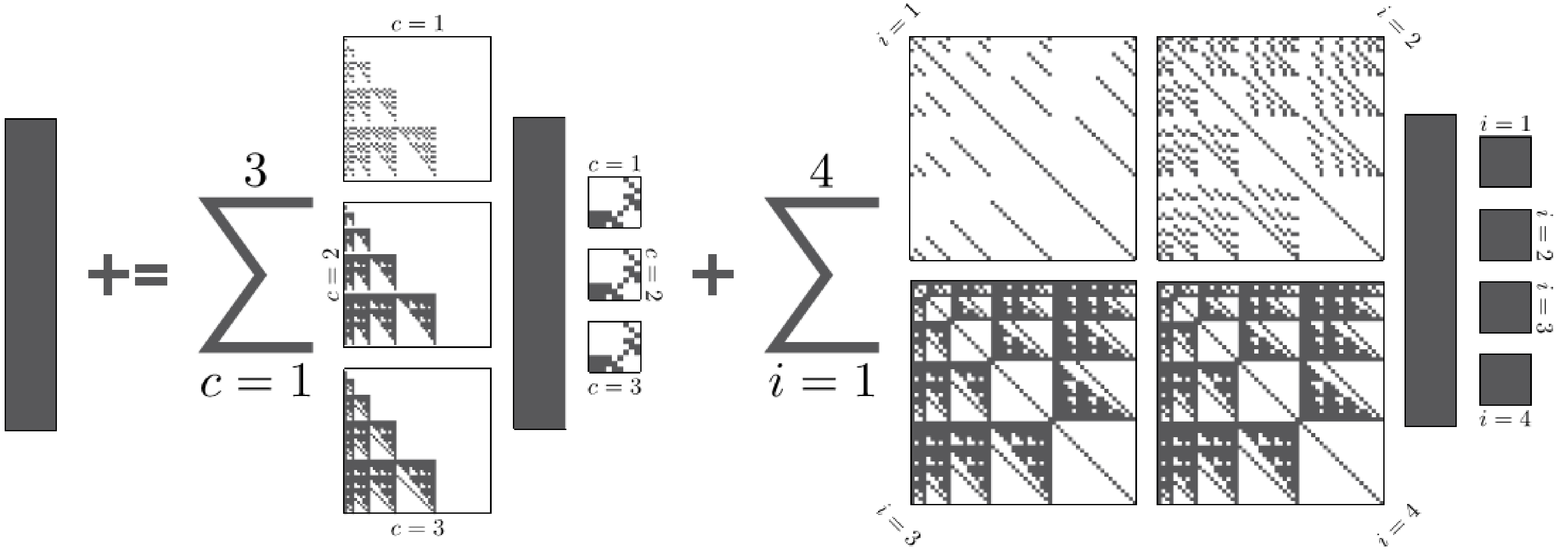
SeisSol's Compute Kernels – Local Integration

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

Sparse/Dense Matrix-Matrix multiplications

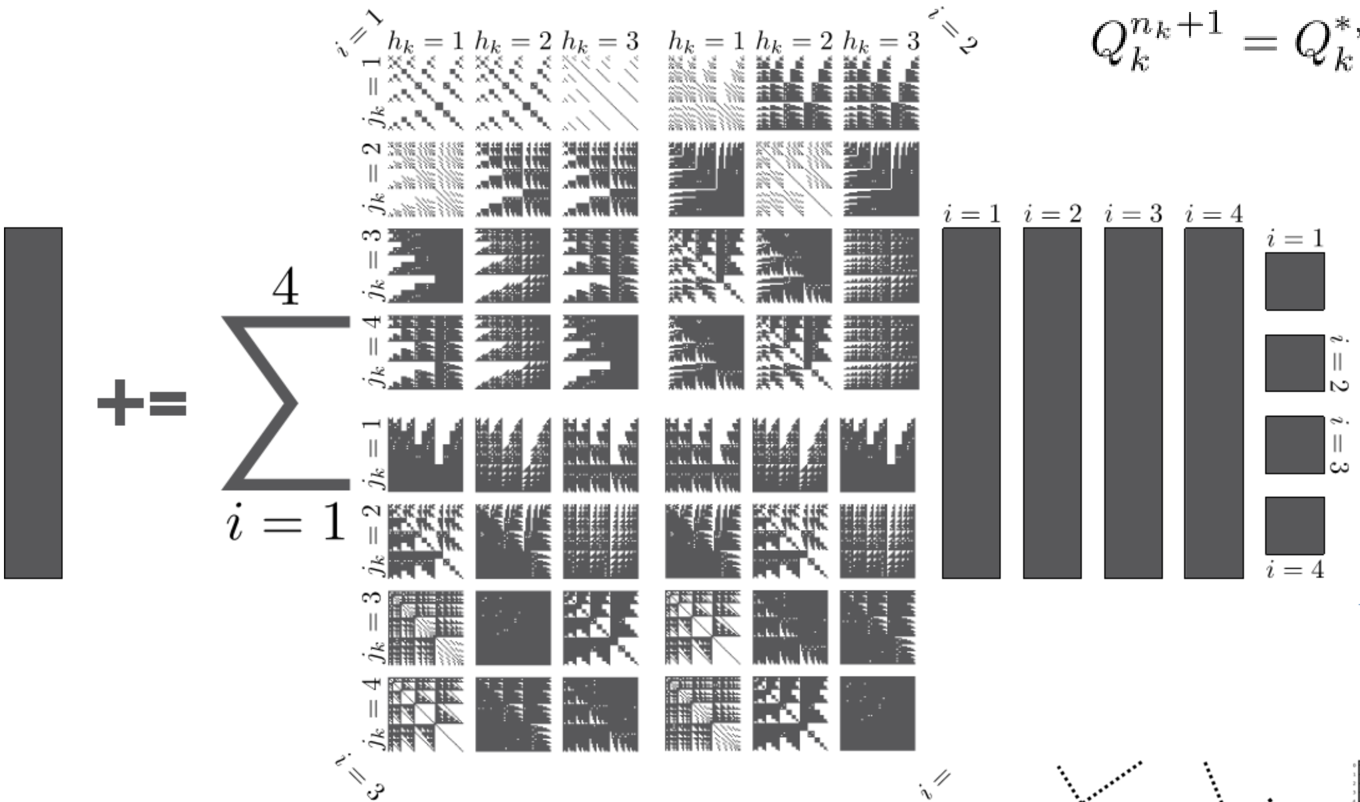
Typical Matrix sizes of production runs (convergence order 6): $9x9, 56x9, 56x35$

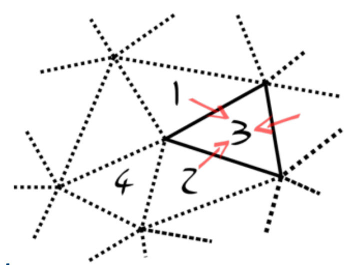
A priori known sparsity patterns

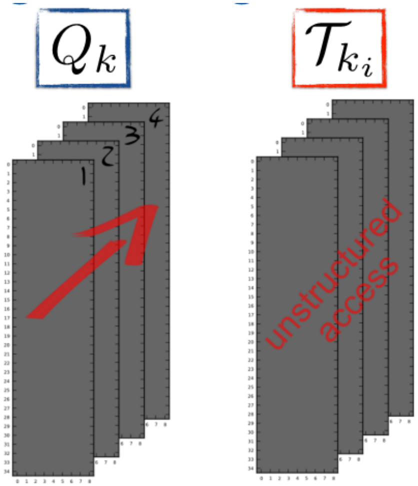


SeisSol's Compute Kernels – Boundary Kernel

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







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 sparse 24NNZ, 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. 7.5** (sixth order)
- 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
  - Code generator which hard-wires sparsity patterns

  → **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.**

  → **Due to switch between dense and sparse implementations: highest efficiency doesn’t correspond to shortest time to solution**

Systems used in our Performance Comparison

WSM A dual-socket Intel® Xeon® X5690 server, 12 cores @3.46 GHz, 48 GB of DDR3-1333 memory, 128-bit SSE4.2 vector instruction set, 41 GiB/s memory bandwidth, 166 GFLOPS double precision peak performance, idle power consumption of 160 W and DGEMM power consumption of 350 W.

SNB A dual-socket Intel® Xeon® E5-2670 server, 16 cores @2.6 GHz, 128 GB of DDR3-1600 memory, 256-bit AVX vector instruction set, 75 GiB/s memory bandwidth, 333 GFLOPS double precision peak performance, idle power consumption of 100 W and DGEMM power consumption of 280 W.

HSW A dual-socket Intel® Xeon® E5-2699 v3 server, 36 cores @1.9 GHz (guaranteed, P0-frequency is 2.3 GHz), 128 GB of DDR4-1866 memory, 256-bit AVX2 vector instruction set, 105 GiB/s (cluster-on-die enabled) memory bandwidth, 1.1 TFLOPS double precision peak performance @1.9 GHz, idle power consumption of 75 W and DGEMM power consumption of 400 W.

KNC A Intel® Xeon Phi™ 5110P coprocessor, 60 cores @1.06 GHz, 8 GB of GDDR5 memory, 512-bit MIC vector instruction set, 150 GiB/s memory bandwidth and 1 TFLOPS double precision peak performance, idle power consumption of 100 W and DGEMM power consumption of 225 W.

Powermeters:

- Watts-Up?: HSW and WSM
- Megware® Clustsafe®: SNB
- micsmc: (software, co-processor only): KNC

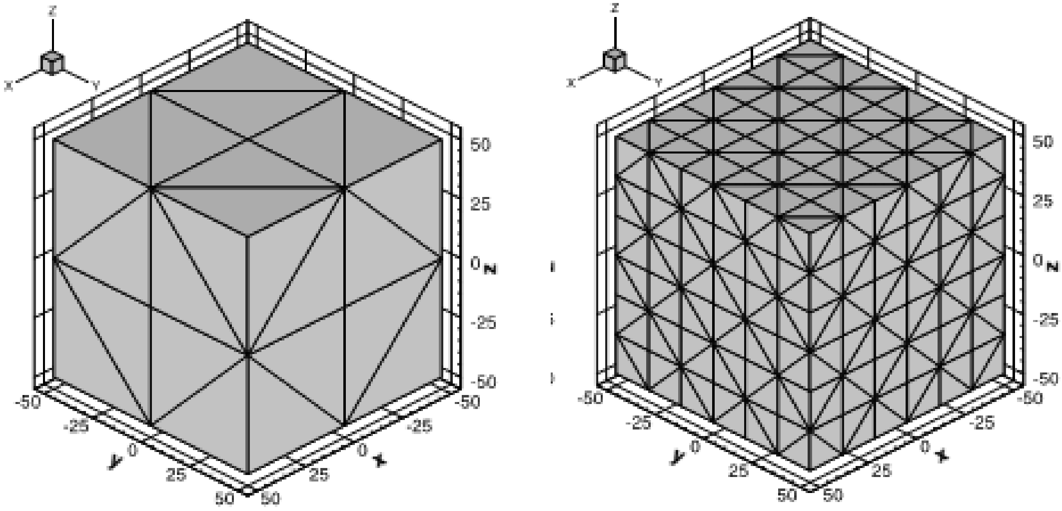
Optimized Matrix Kernels – LIBXSMM

- Highly optimized sparse and dense matrix kernels by offline code generation and auto-tuning (publically available):
  - Intel SSE3, AVX, AVX2, KNC, AVX512F

Fig. 1. Double precision GEMM kernel performance for $\mathcal{O}2$ to $\mathcal{O}7$ on WSM, SNB and HSW. DGEMM performance for $B_{\mathcal{O}} \times 9 \times B_{\mathcal{O}}$ shapes (left) and $B_{\mathcal{O}} \times 9 \times 9$ cases (right) is shown. GFLOPS are depicted as bars whereas fraction of peak performance is given by lines. DGEMMs were executed for 10,000 times on a hot L1 cache.

Regular Convergence Test

- Regular cubic mesh with 5 tets per cube
- Error-Norms: Wave propagation in diagonal direction with periodic boundaries
- Hardware: HSW @ 1.9 GHz
- Convergence: Mesh width



Dumbser, Käser: An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes – II. The three-dimensional isotropic case

What are the benefits from using a higher-order method?

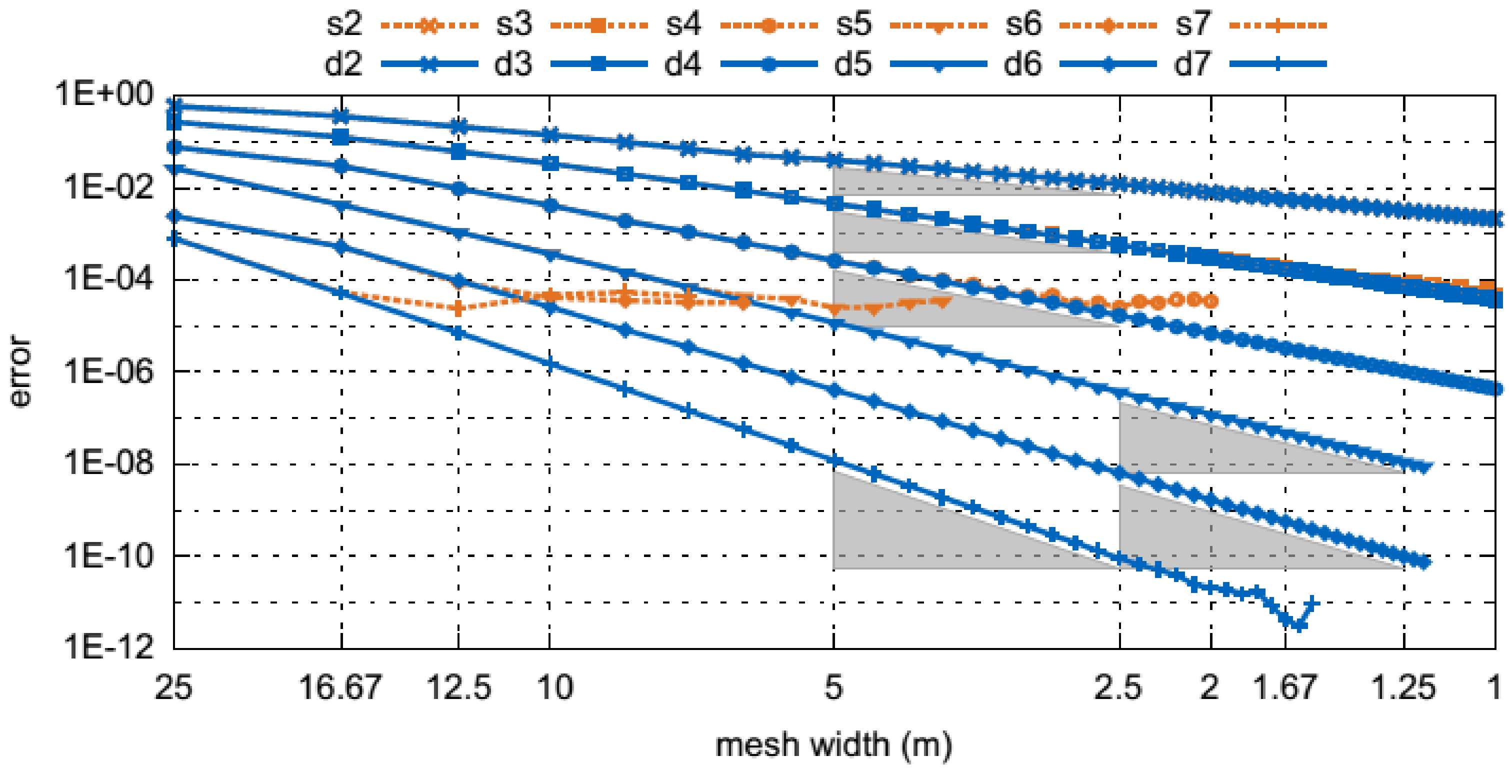


Fig. 2. Convergence of the synthetic benchmark for orders $\mathcal{O}2 - \mathcal{O}7$. Shown is the mesh width $100/N_{1D}$ against the error in the L^∞-norm for the variable σ_{yz}. Orange dashed lines represent single precision, blue solid lines double precision. The slopes of the gray triangles next to the respective curves illustrate the mathematical convergence rate.

What does this mean for the energy consumption?

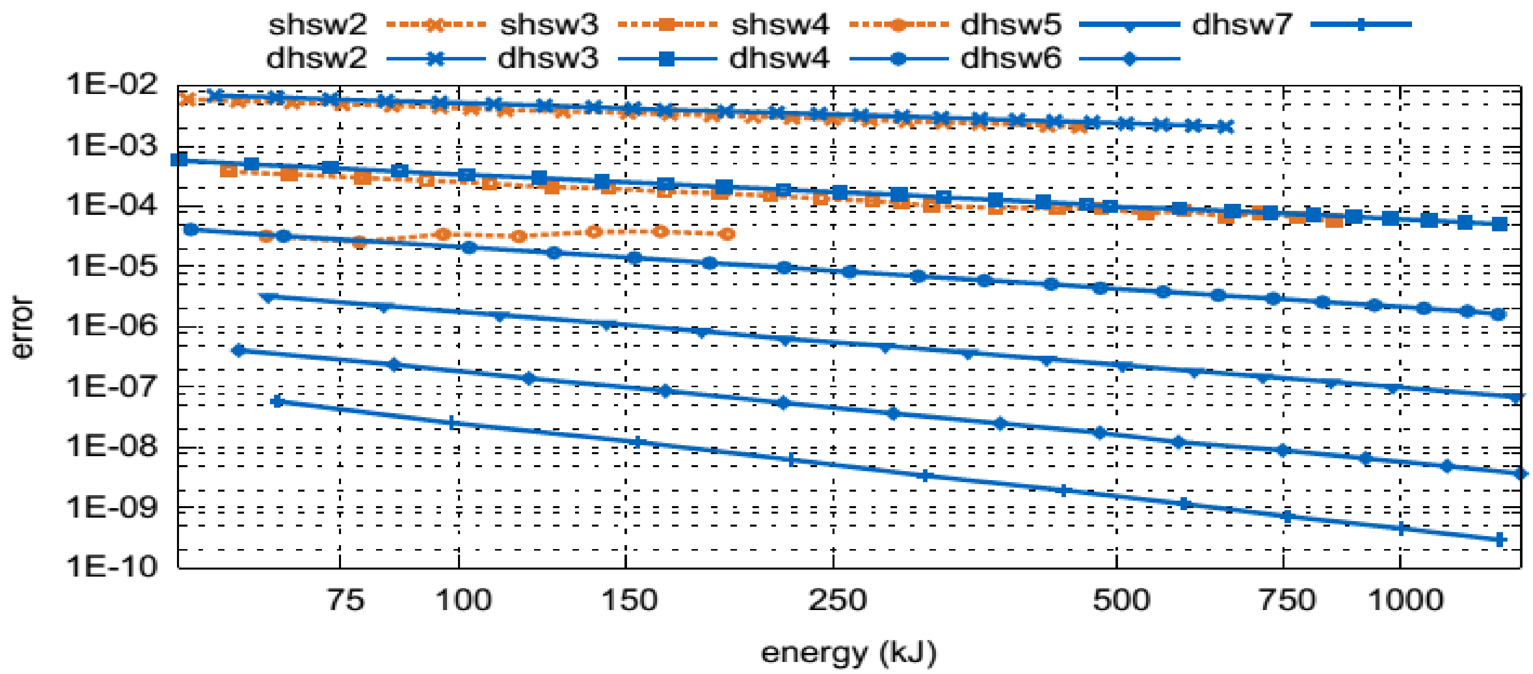


Fig. 3. L^{∞}-error of variable σ^{yz} in dependency of the consumed energy for the HSW machine in single- and double-precision.

Benchmark	@1.2 GHz		@1.9 GHz	
	Performace	Power	Performace	Power
STREAM - Triad	105 GiB/s	295 W	105 GiB/s	345 W
DGEMM - 60k×60k×192	610 GFLOPS	250 W	950 GFLOPS	400 W

HSW: Dual –socket Xeon E5-2699v3, @1.9 GHz

What reduction in error is possible in a given energy-budget?

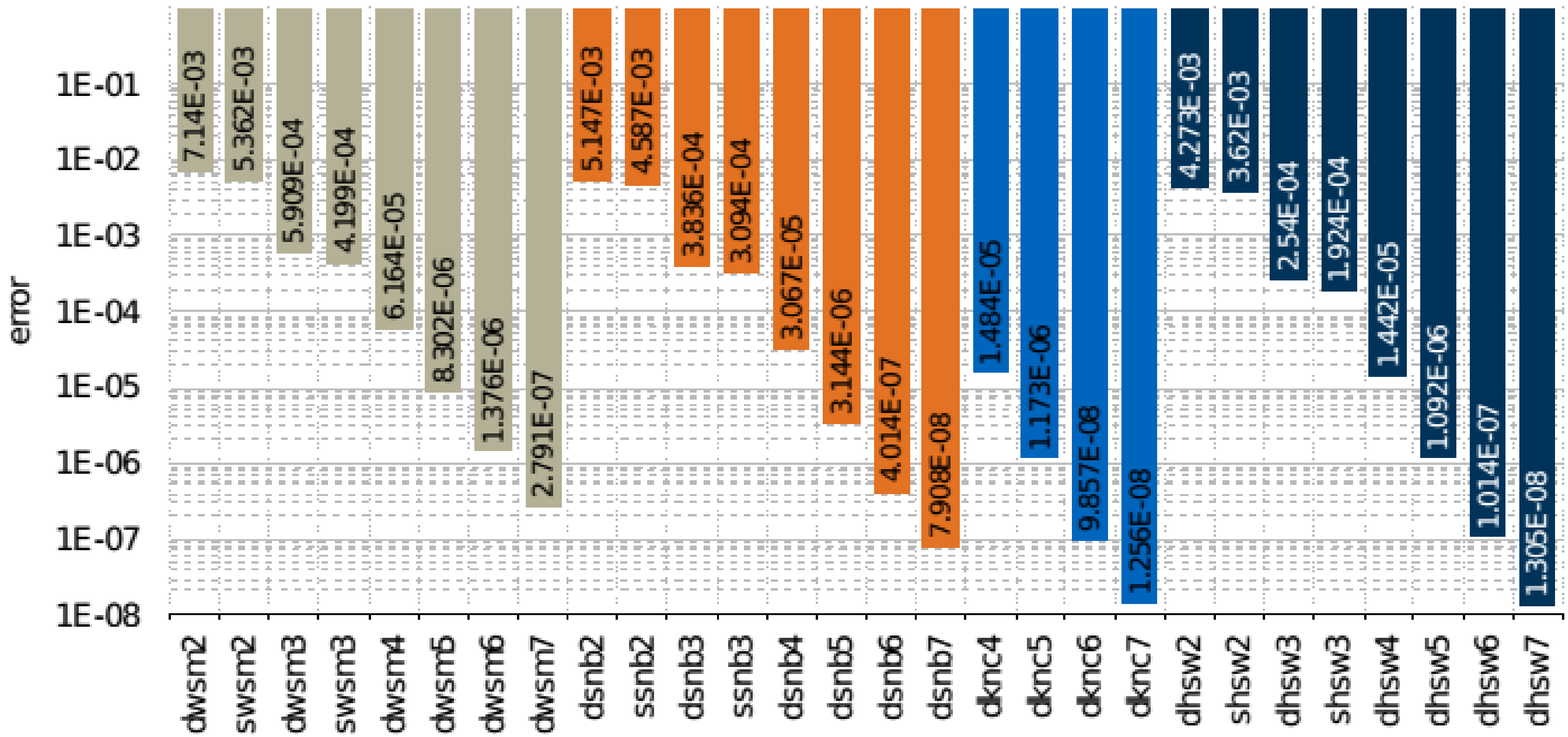


Fig. 4. Error with respect to a 150 kJ energy-budget for all matching settings. Shown is the interpolated L^∞-error of variable σ^{yz} for the different architectures, orders of convergence and single- and double-precision.

The LOH.1 Benchmark

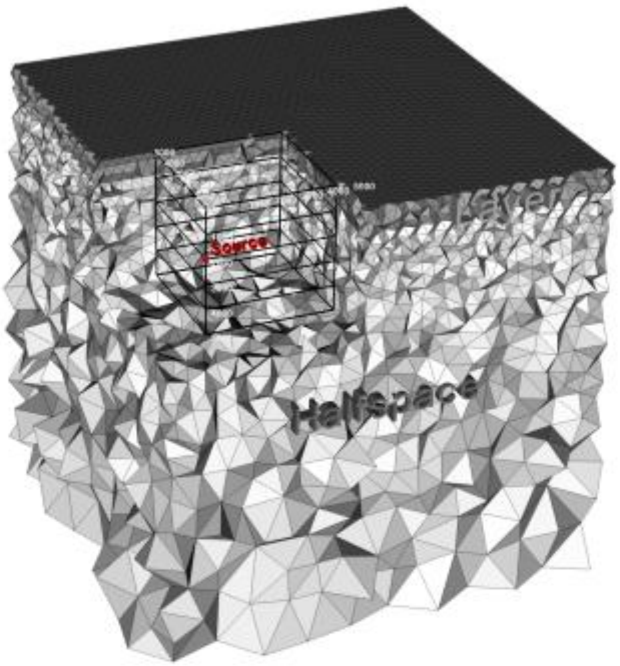


Fig. 5. Setup of the LOH.1 scenario.

- 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]

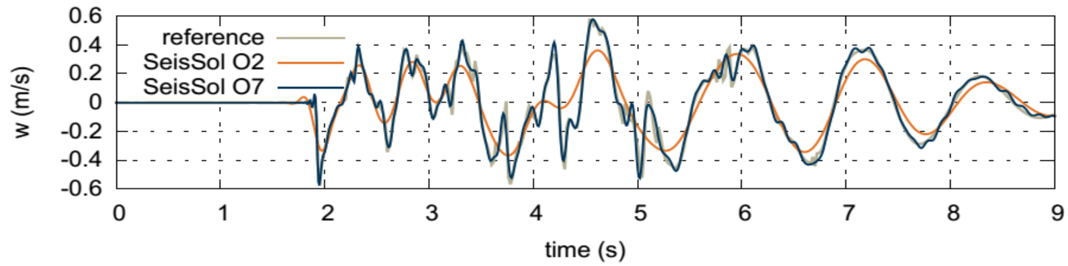
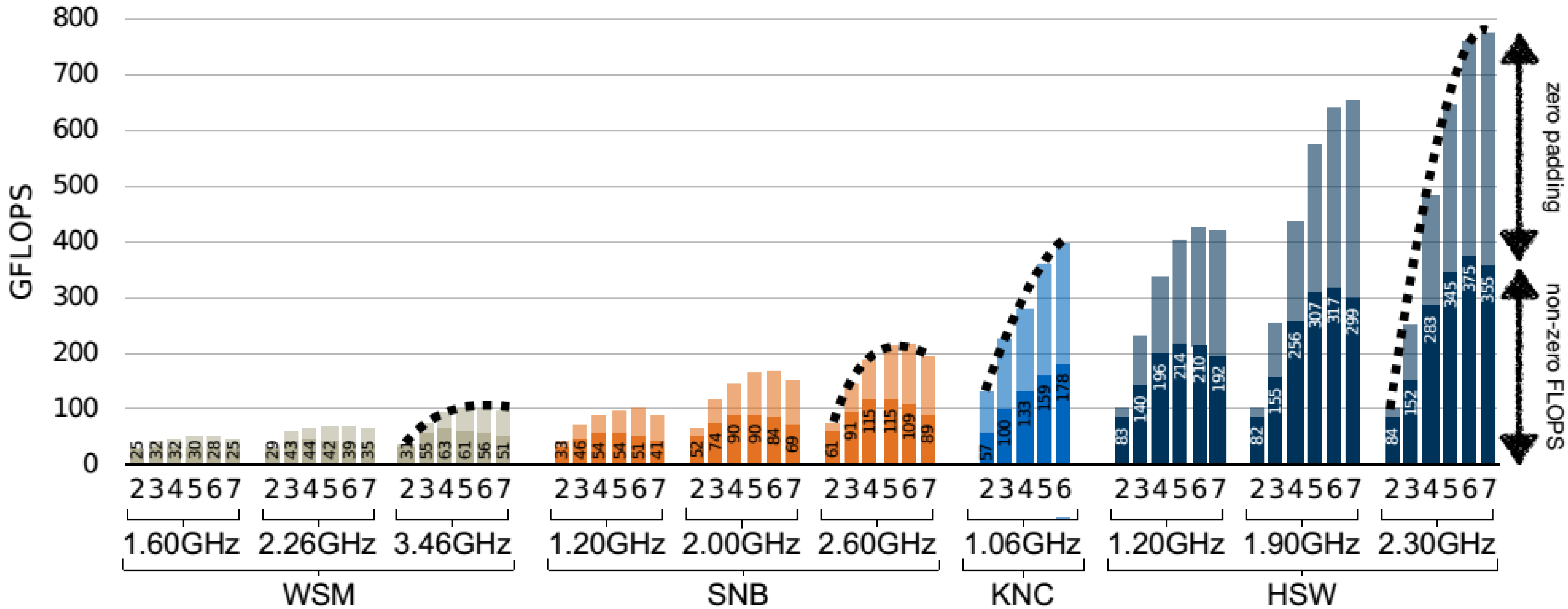


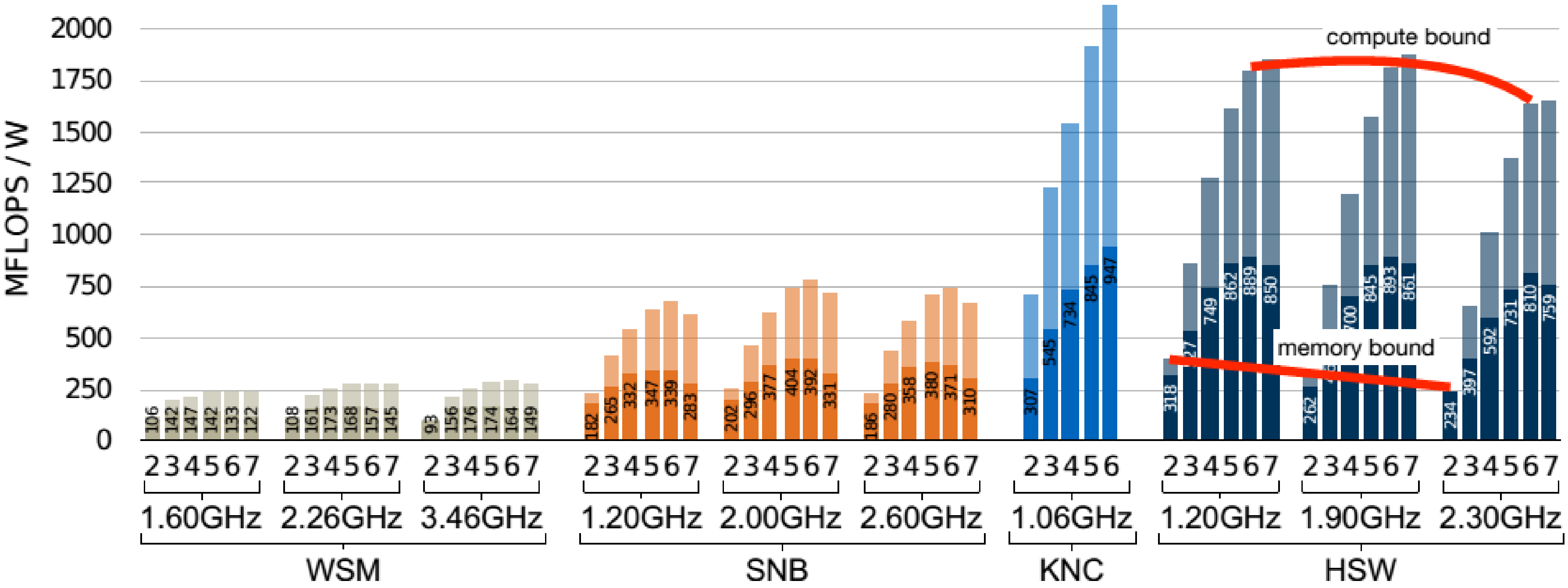
Fig. 6. Comparison of the vertical particle velocity w at receiver nine. Shown is the reference solution of the SISMOWINE project against Seissol using $\mathcal{O}2$ and $\mathcal{O}7$.

GFLOPS for LOH.1



→ Switching to newer and denser platforms, the memory bound to compute bound ratios increase in higher convergence orders.

MFLOPS/W for LOH.1



→ In case of lower order runs, the lowest frequency results in the best energy to performance ratio (recall micro benchmark results!)

Power Consumption in dependence on the VEX on Haswell

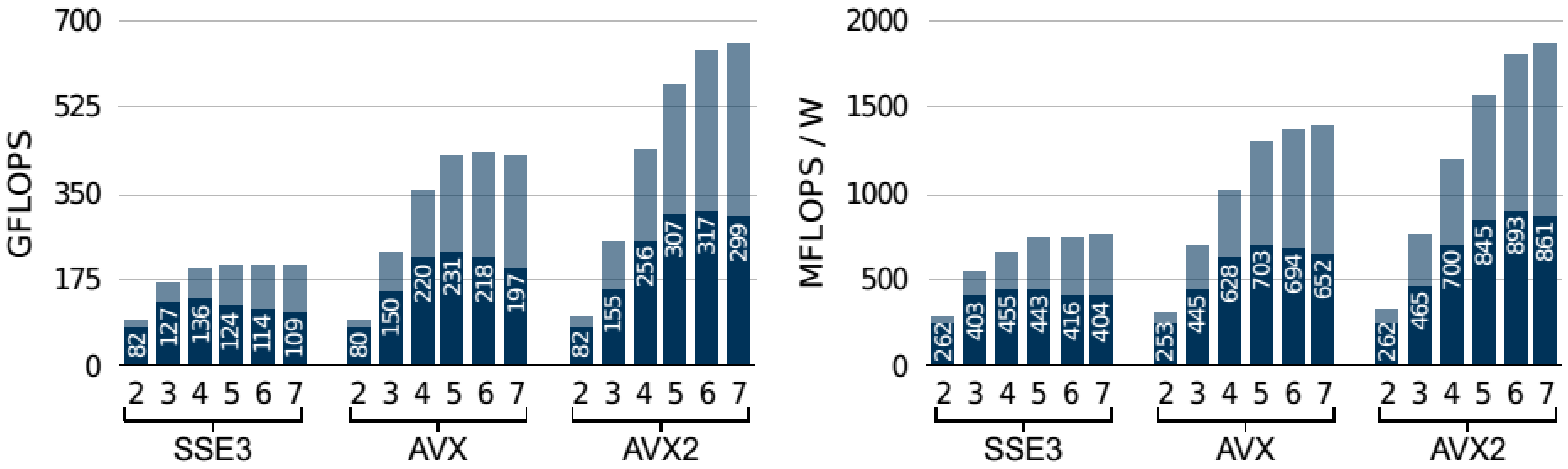


Fig. 8. GFLOPS (left) and the MFLOPS per Watt (right) over all convergences orders and instructions set on HSW @1.9 GHz.

- Shorter vectors result in less performance and less power → good for old code
- FMA significantly increases power efficiency

→ whenever we can, we should use long vectors and FMA

What are Architectural Learnings?

- **Low-order energy consumption**: memory-bound new architectures minor impact: **2X** Westmere to Haswell.
- **High-order energy consumption**: >**10X** Westmere to Haswell. **10^5 better than low-order**.
- Very good perf./watt, however: **up to 50% useless work**!
- Westmere -> Haswell: 14 sparse matrices reduced to 2
  - Vector Performance vs. L1 bandwidth
  - Using gather/scatter or sparsity code gen. turn DGEMMs in to sparse DGEMMs which are an L1 bandwidth benchmark: 2 reads, 1 store per cycle
  - Max. 50% peak since 2nd FPU is empty
  - 50% is not reachable due to address calc. and jumps -> ~25-30% achieved in practice
- Lots of opportunities for architectural or algorithmic (UQ) improvements
- For best time-to-solution, we also need tricks such as LTS

Nek5000 / NekBox

High Order CFD Simulations

http://nek5000.mcs.anl.gov/

https://github.com/maxhutch/nek

Joint Work with Maxwell Hutchinson (U Chicago), David Keyes (KAUST)

ISC'16 paper preview

Nek Overview

Nek solves the incompressible Navier Stokes equations:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\frac{1}{\rho}\nabla p + \nu \nabla^2 u + f \qquad \nabla \cdot u = 0$$

Incl. advection-diffusion equations for scalar variables such as temperature or mass fractions.

Nek use the spectral element method (SEM) which is a two-level discretization:

a) A tensor product construction of Gauss-Lobatto-Legendre (GLL) quadrature points within each element -> N^3 DOFs per element

b) Continuity across elements -> forming a mesh

Operators are written as element local operators:

$$A = (A_x \times I_y \times I_z) + (I_x \times A_y \times I_z) + (I_x \times I_y \times A_z)$$

And direct stiffness summation ensure continuity. Furthermore, this reduces the complexity form O(N^6) to O(N^4).

NekBox's main compute routines

A typical NekBox run spends <1% in sparse computations & communications, ~40% in vector-vector or matrix-vector operations, ~60% matrix-matrix operations.

Helmholtz operator:

```
Hu(:,:,:) = gx(:,:,:) * matmul(Kx(:,:), reshape(u, (/N, N*N/)))
do i = 1, n
  Hu(:,:,i) += gy(:,:,i) * matmul(u(:,:,i), KyT(:,:))
enddo
Hu(:,:,:) += gz(:,:,:) * matmul(reshape(u, (/N*N, N/)), KzT(:,:))
Hu(:,:,:) = h1 * Hu(:,:,:) + h2 * M(:,:,:) * u(:,:,:)
```

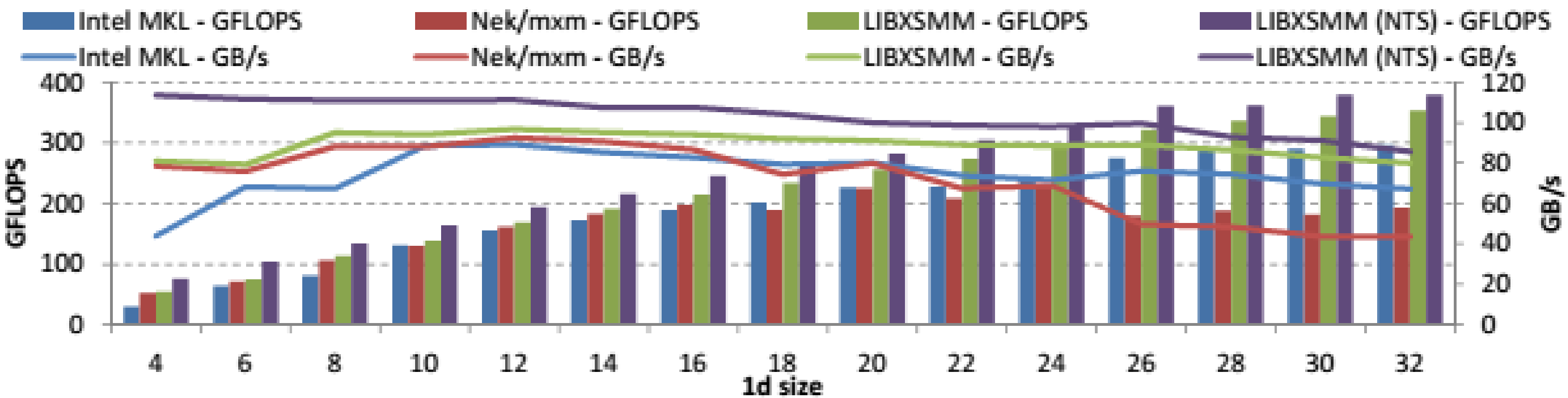
Basis transformation:

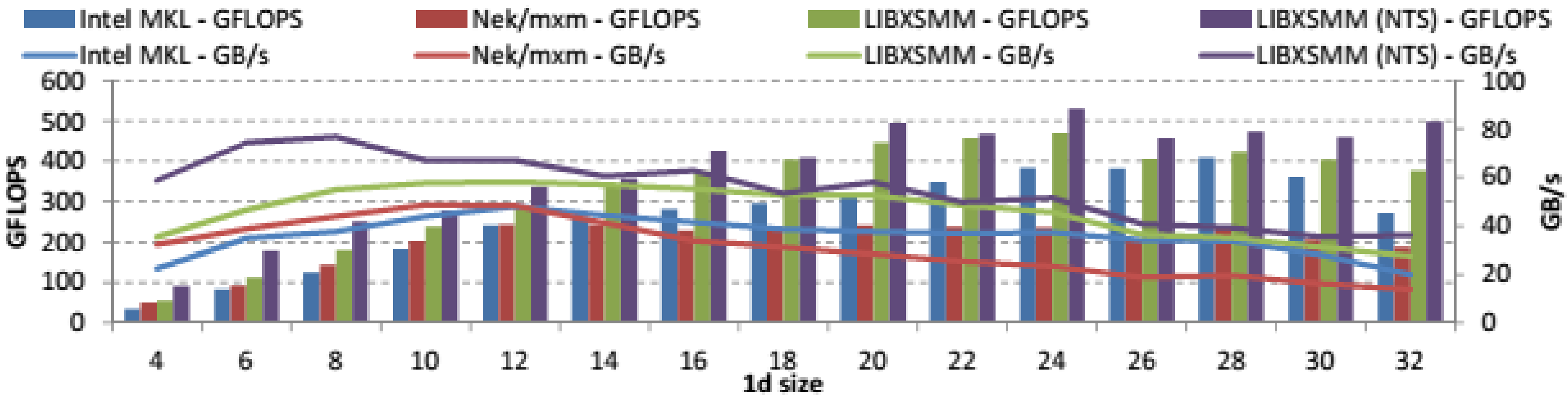
```
tmp_x = matmul(Ax, u)
do i = 1, n
  tmp_y(:,:,i) = matmul(tmp_x(:,:,i), AyT)
enddo
v = matmul(tmp_y, AzT)
```

Gradient calculation:

```
dudx = matmul(Dx, u)
do i = 1, n
  dudy(:,:,i) = matmul(u(:,:,i), DyT)
enddo
dudz = matmul(u, DzT)
```

Helmholtz Operator / Basis Transformation

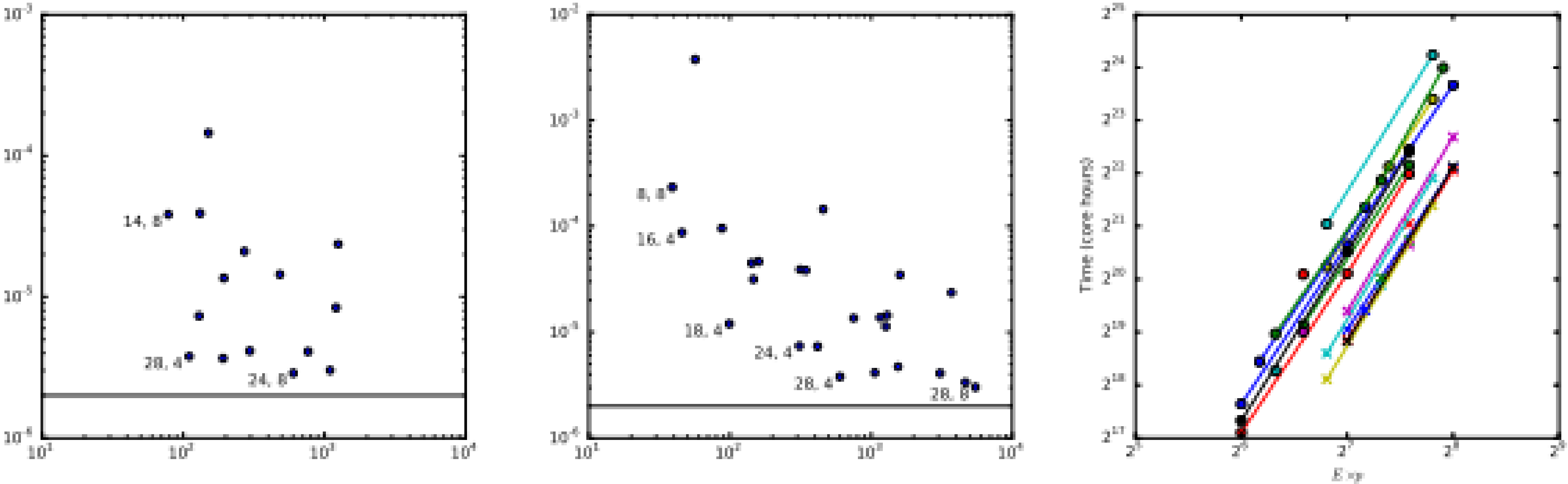




Performance of the Helmholtz operator reproducer (up) and Basis Transformation (bottom) using different implementation for the small matrix multiplications. NTS denotes the usage of non-temporal stores. Measured on Shaheen (32 cores of HSW-EP, 2.3 GHz)

Some Early High Order Efficiency Results

We model the single-mode Rayleigh-Taylor Instability (Boussinesq equations)



(a) Shaheen (b) Mira (c) Cost vs DOF

Log-base 10 error of bubble height and mix volume:

$$H = \sup\left\{ z : \min_{x,y} T(x,y,z) < T_0 \right\} \qquad \Theta = \int |T - T_0|\, dV$$

on Shaheen.

We can see that high orders are favorable as they better match modern hardware and have superior convergence speed.

LIBXSMM

Library for small matrix multiplications on Intel Architecture

https://github.com/hfp/libxsmm

Joint Work with Hans Pabst (Intel), Greg Henry (Intel)

Abstract and Motivation

"Improving Performance for Small GEMM Size Problems."

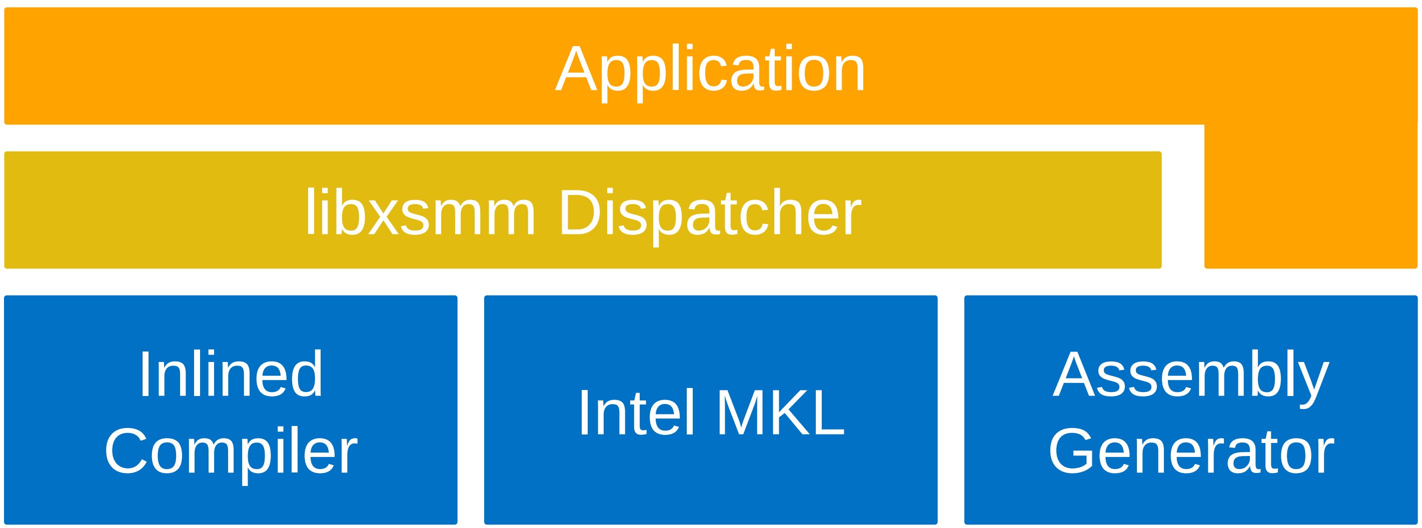
- Problem size is characterized by the M, N, and K parameters
  - Common building block for high order methods
  - Common building block for blocked Sparse Linear Algebra
- A suitable problem size may fall within (M N K)^(1/3) <= 60
  - Intel® Math Kernel Library (Intel® MKL) uses MKL_DIRECT_CALL
  - These sizes are smaller than regular S/DGEMM blocked macro-kernels, therefore MKL_DIRECT_CALL helps, but is only the tip of the iceberg- a lot more performance is necessary/possible

LIBXSMM

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)



License

- Open Source Software (BSD 3-clause license)*

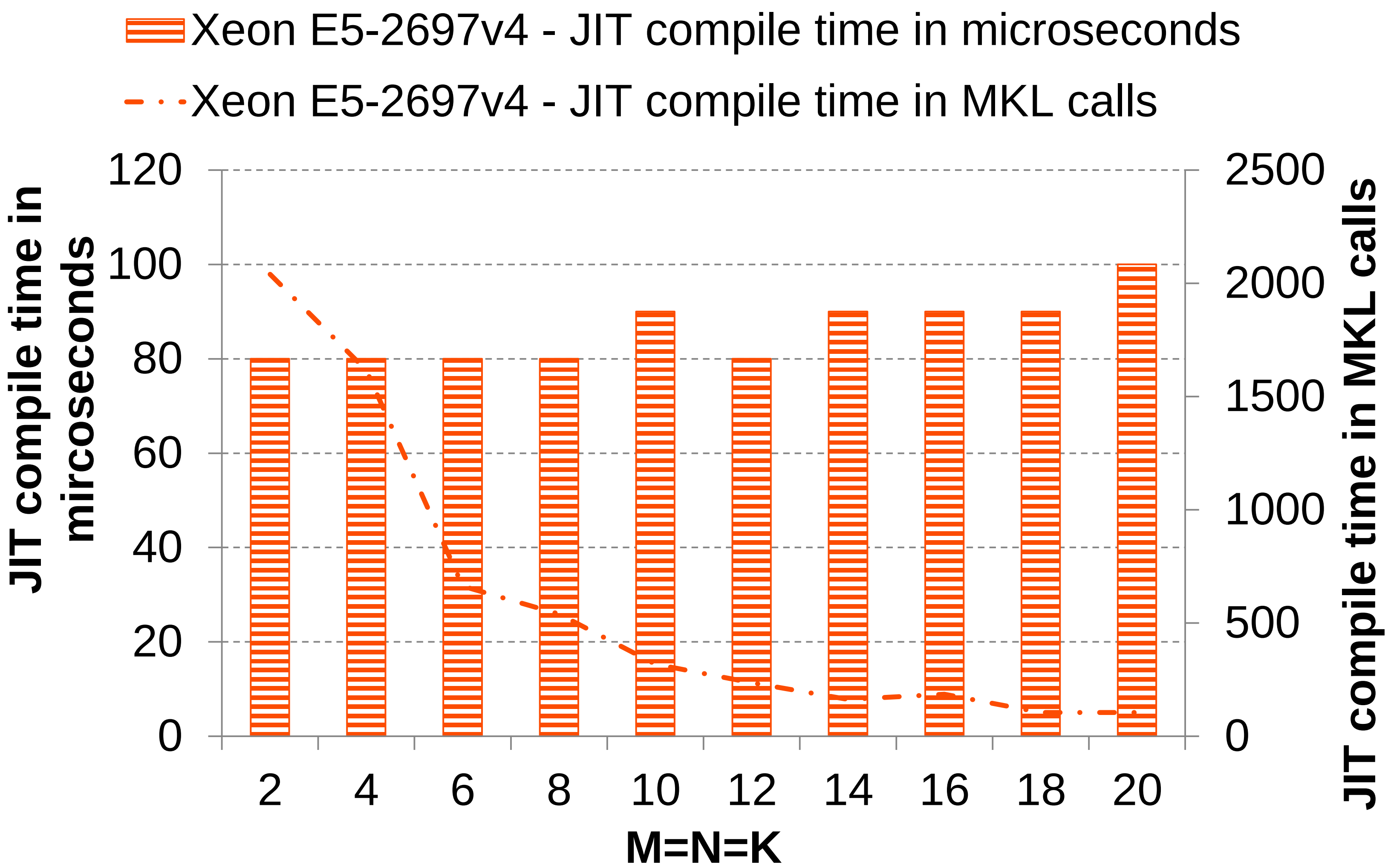
* https://github.com/hfp/libxsmm

LIBXSMM Implementation

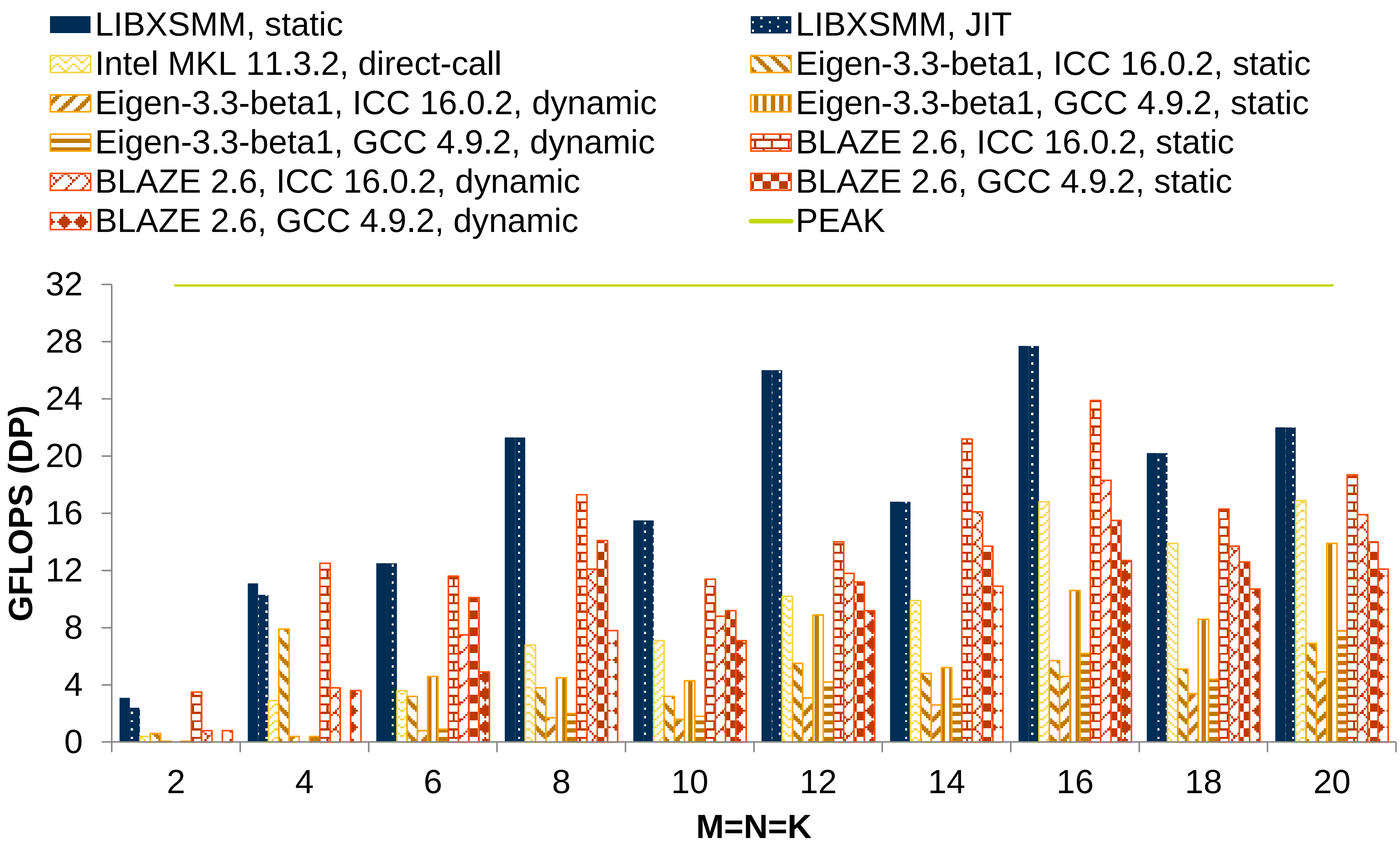
Three Critical Parts of Technology:

- Highly efficient Frontend (Hans Pabst)
  - BLAS compatible (DGEMM, SGEMM) (even LD_PRELOAD)
  - Support for F77, C89, F2003, C++
  - 2-level code caching
  - Zero-overhead calls into assembly
- Code Generator (Alex Heinecke)
  - Supports all Intel Architectures since 2005, special focus on AVX-512
  - Prefetching across small GEMMs
  - Can generate *.s, inline assembly into *.h/*.c of the feed the JIT encoder
- JIT (Just-In-Time) Encoder (Greg Henry)
  - Encodes an instruction based on basic blocks
  - Very fast as no compilation is involved

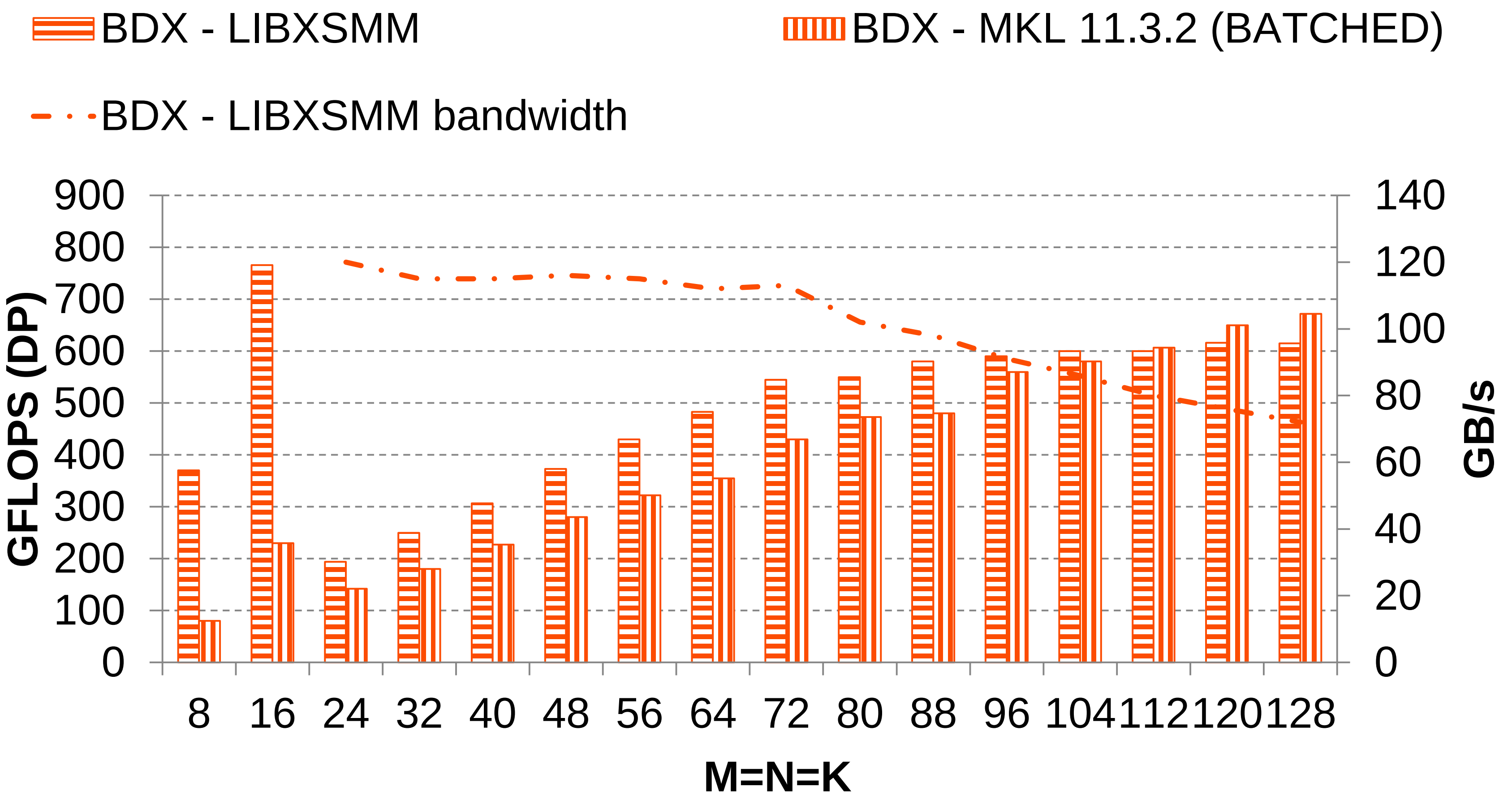
JIT overhead (incl. OS overheads)



LIBXSMM Performance on 1c Xeon E5-2697v4 (BDX)



LIBXSMM vs. MKL DGEMM_BATCH
2x Xeon E5-2697v4 (BDX)



Conclusion and Future/Current Work

Conclusions

- High order simulations can leverage both memory bandwidth and compute
- Due to faster convergence, they are more energy efficient
- Problem solved?
- No!
  - We need a better understanding where these techniques are applicable and where possible traps are.
  - We still need optimal hardware, SeisSol & Nek!
- Therefore:
  - We started a collaboration with U Chicago/ ANL on Nek5000 -> first results have been submitted to ISC'16 and SC'16
  - And of course we are looking for more☺

intel®

experience
what's inside™