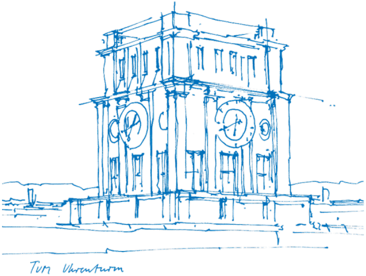
Experiences with Earthquake and Tsunami Simulation on Xeon Phi Platforms

Special session on MIC experience and best practice

Michael Bader (and co-authors listed with chapters)
Technical University of Munich

LRZ, 29 June 2016



Overview and Agenda

Dynamic Rupture and Earthquake Simulation with SeisSol:

- unstructured tetrahedral meshes
- high-order ADER-DG discretisation
- compute-bound performance via optimized matrix kernels

Optimising SeisSol for Xeon Phi Platforms

- offload scheme: 1992 Landers Earthquake as landmark simulation, scalability on SuperMUC, Tianhe-2, Stampede
- optimisation for Knights Corner and Landing
- towards simulations in symmetric mode (1st results on Salomon)

Tsunami Simulation on SuperMIC:

- parallel adaptive mesh refinement with sam(oa)^2
- enable vectorization via introducing patches
- towards load balancing on heterogeneous systems

Part I

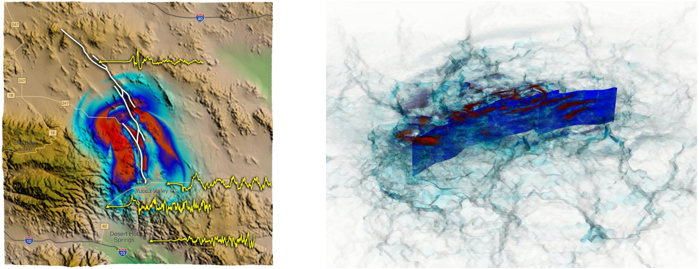
Dynamic Rupture and Earthquake Simulation with SeisSol

`http://www.seissol.org/`

Dumbser, Käser et al. [9]
An arbitrary high-order discontinuous Galerkin method . . .

Pelties, Gabriel et al. [12]
Verification of an ADER-DG method for complex dynamic rupture problems

Dynamic Rupture and Earthquake Simulation

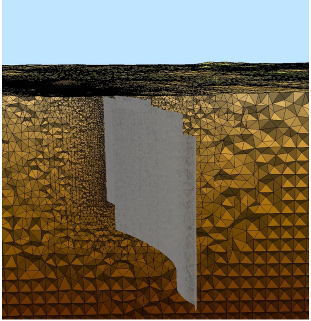


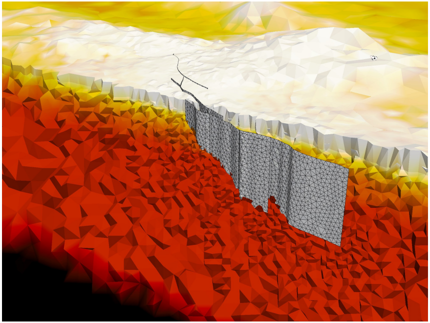
Landers fault system: simulated ground motion and seismic waves [3]

SeisSol – ADER-DG for seismic simulations:

- adaptive tetrahedral meshes
  → complex geometries, heterogeneous media, multiphysics
- complicated fault systems with multiple branches
  → non-linear multiphysics dynamic rupture simulation
- ADER-DG: high-order discretisation in space and time

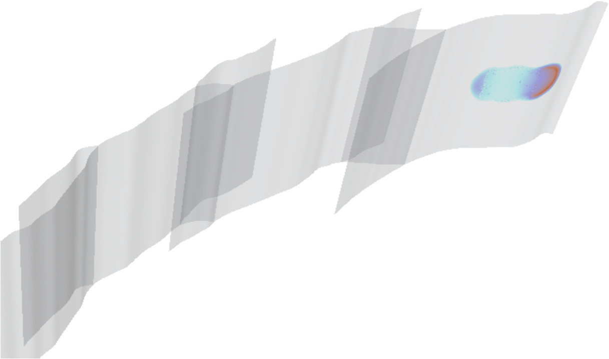
Example: 1992 Landers M7.2 Earthquake





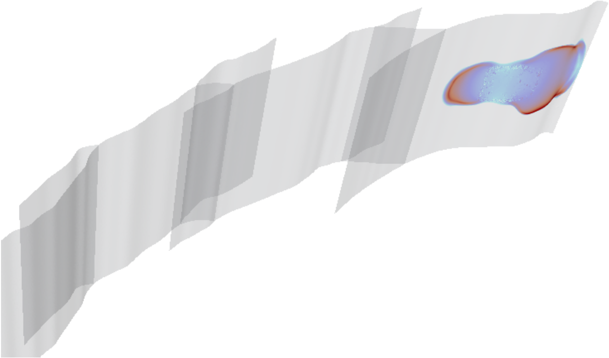
- multiphysics simulation of dynamic rupture and resulting ground motion of a M7.2 earthquake
- fault inferred from measured data, regional topography from satellite data, physically consistent stress and friction parameters
- static mesh refinement at fault and near surface

Multiphysics Dynamic Rupture Simulation



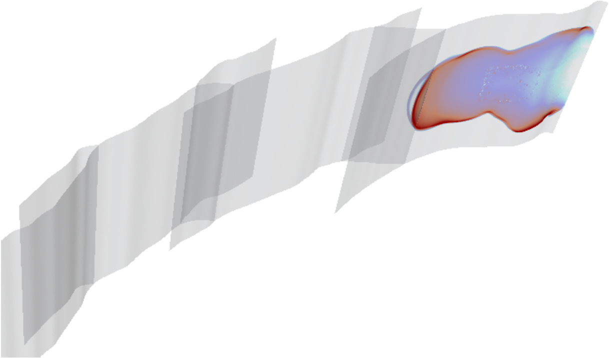
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



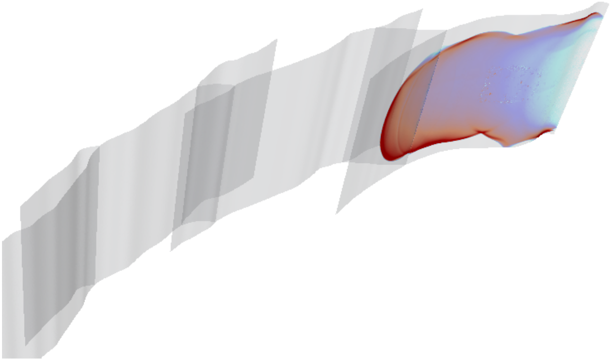
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



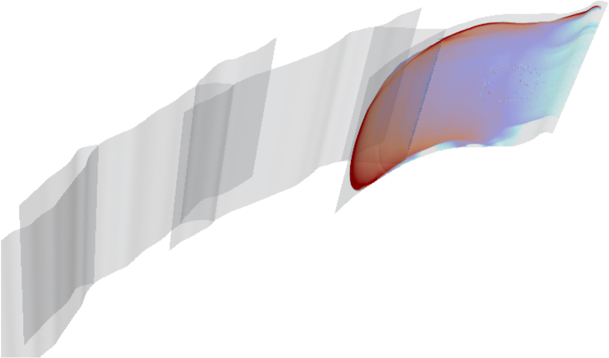
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



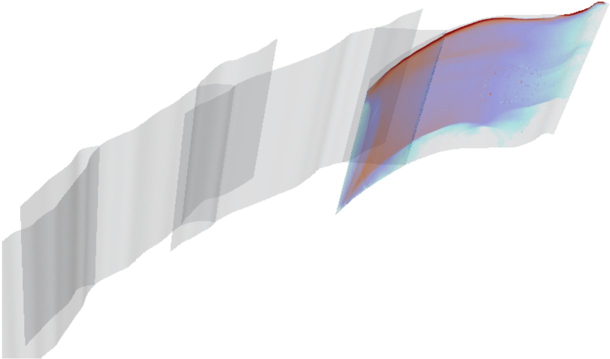
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



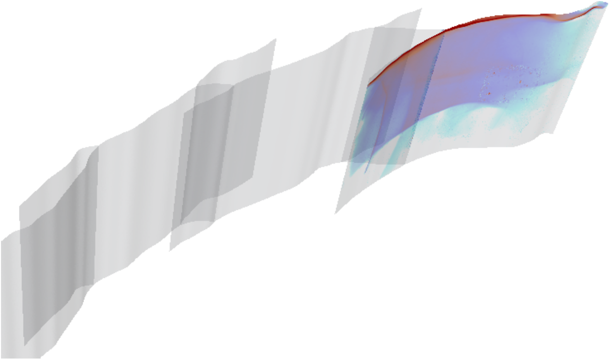
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



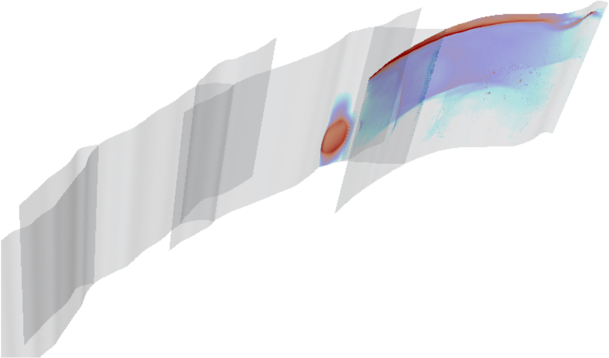
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



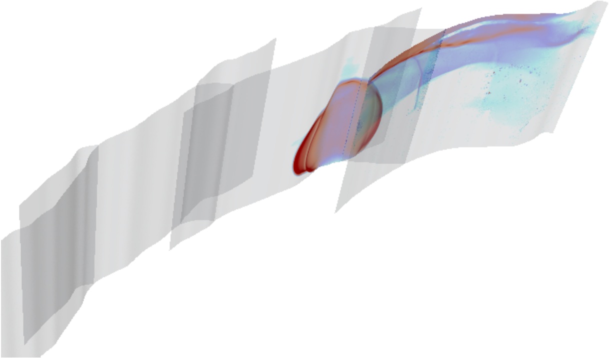
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



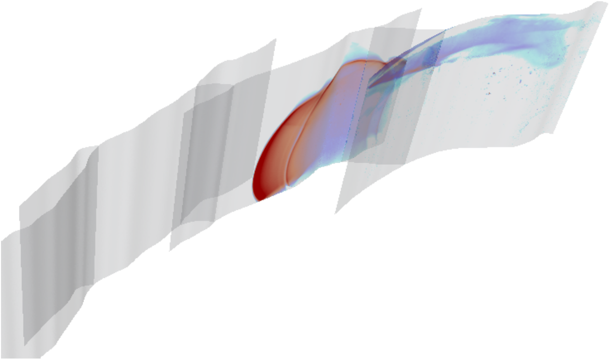
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



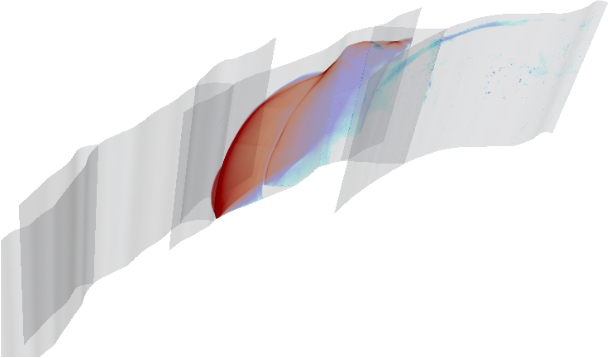
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



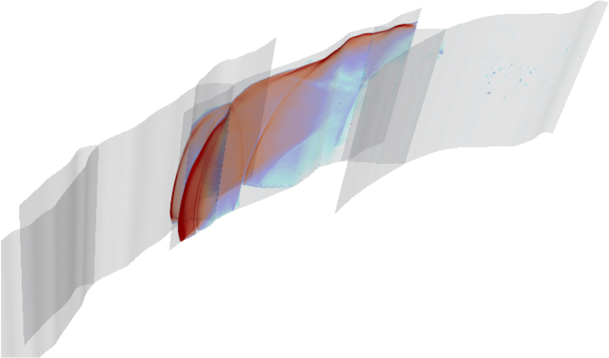
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



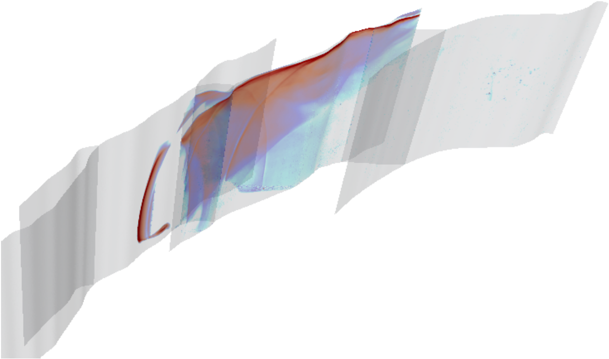
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



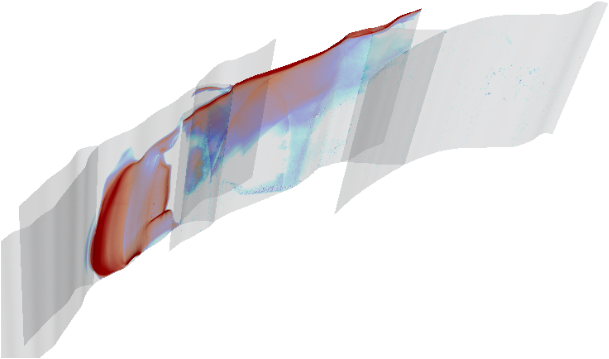
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



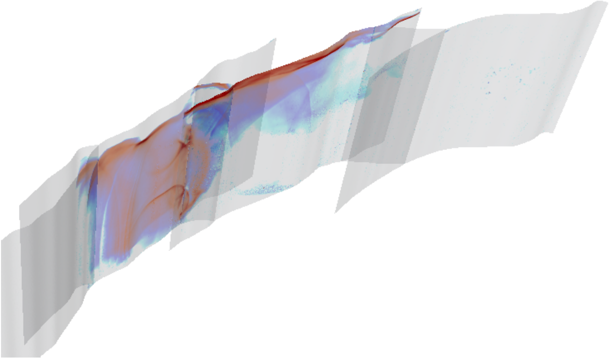
- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Multiphysics Dynamic Rupture Simulation



- spontaneous rupture, non-linear interaction with wave-field
- featuring rupture jumps, fault branching, etc.
- tackles fundamental questions on earthquake dynamics
- realistic rupture source for seismic hazard assessment

Part II

SeisSol as a Compute-Bound Code: Code Generation for Matrix Kernels

Breuer, Heinecke, Rannabauer, Bader [1]: High-Order ADER-DG Minimizes Energy- and Time-to-Solution of SeisSol (ISC'15)
Uphoff, Bader [6]: Generating high performance matrix kernels for earthquake simulations with viscoelastic attenuation (HPCS 2016)

Seismic Wave Propagation with SeisSol

Elastic Wave Equations: (velocity-stress formulation)

$$q_t + Aq_x + Bq_y + Cq_z = 0$$

with $q = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{13}, u, v, w)^T$

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -\lambda - 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mu \\ -\rho^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\rho^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\rho^{-1} & 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda - 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mu \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\rho^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\rho^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\rho^{-1} & 0 & 0 & 0 & 0 \end{pmatrix}$$

- high order discontinuous Galerkin discretisation
- **ADER-DG**: high approximation order in space and time:
- additional features: local time stepping, high accuracy of earthquake faulting (full frictional sliding)

→ Dumbser, Käser et al., e.g. [9]

SeisSol in a Nutshell – ADER-DG

Update scheme

$$
\begin{aligned}
Q_k^{n+1} = Q_k - \frac{|S_k|}{|J_k|} M^{-1} \Bigg(& \sum_{i=1}^{4} F^{-,i} I(t^n, t^{n+1}, Q_k^n) N_{k,i} A_k^+ N_{k,i}^{-1} \\
& + \sum_{i=1}^{4} F^{+,i,j,h} I(t^n, t^{n+1}, Q_{k(i)}^n) N_{k,i} A_{k(i)}^- N_{k,i}^{-1} \Bigg) \\
& + M^{-1} K^{\xi} I(t^n, t^{n+1}, Q_k^n) A_k^* \\
& + M^{-1} K^{\eta} I(t^n, t^{n+1}, Q_k^n) B_k^* \\
& + M^{-1} K^{\zeta} I(t^n, t^{n+1}, Q_k^n) C_k^*
\end{aligned}
$$

Cauchy Kovalewski

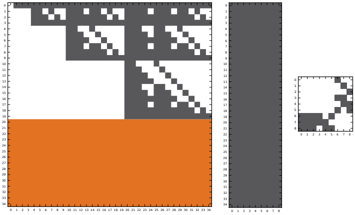
$$
I(t^n, t^{n+1}, Q_k^n) = \sum_{j=0}^{J} \frac{(t^{n+1} - t^n)^{j+1}}{(j+1)!} \frac{\partial^j}{\partial t^j} Q_k(t^n)
$$

$$
(Q_k)_t = -M^{-1} \left((K^{\xi})^T Q_k A_k^* + (K^{\eta})^T Q_k B_k^* + (K^{\zeta})^T Q_k C_k^* \right)
$$

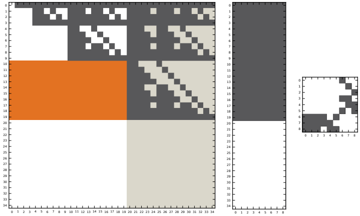
Optimisation of Matrix Operations

Apply sparse matrices to multiple DOF-vectors Q_k

$\hat{K}^\zeta$ $\quad Q_k^n$ $\quad C_k^\star$



$\hat{K}^\zeta$ $\quad \frac{\partial^1}{\partial t^1} Q_k$ $\quad C_k^\star$

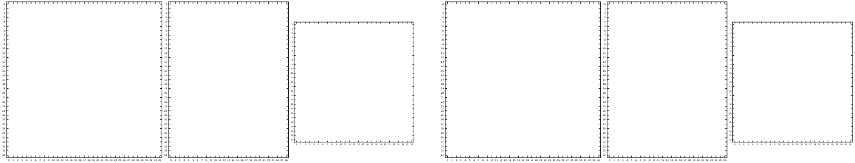


Dense vs. Sparse Kernels: (Breuer et al. [2])

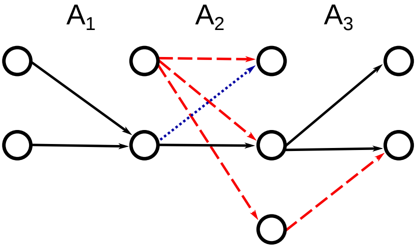
- most kernels fastest, if executed as dense matrix multiplications
- exploit zero-blocks generated during recursive CK computation
- switch to sparse kernels depending on achieved time to solution

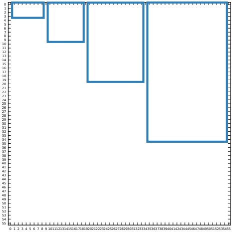
Sparse, Dense → Block-Sparse

Consider equivalent sparsity patterns: (Uphoff, [6])

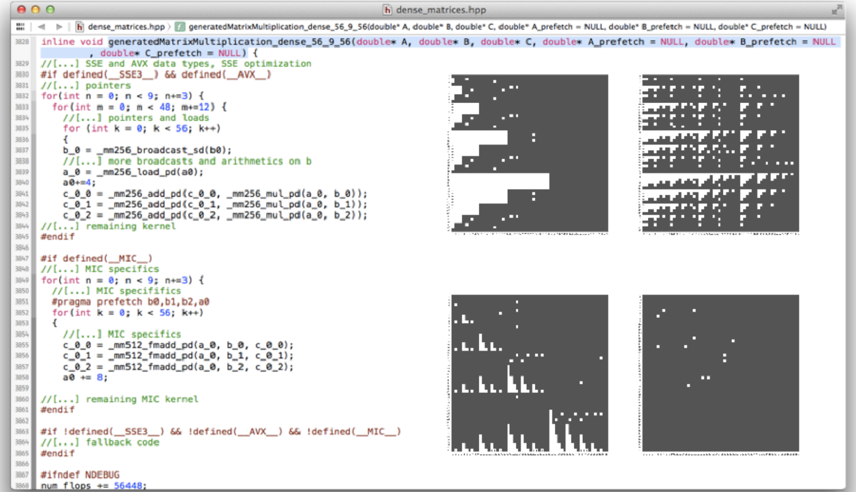


Graph representation and block-sparse memory layouts





Code Generator: Instrinsics → Assembler



dense_matrices.hpp
dense_matrices.hpp
generatedMatrixMultiplication_dense_56_9_56(double* A, double* B, double* C, double* A_prefetch = NULL, double* B_prefetch = NULL, double* C_prefetch = NULL)

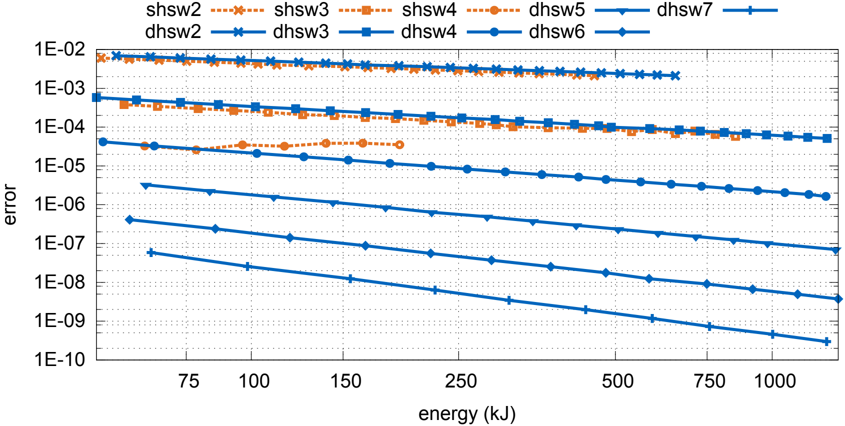
```
inline void generatedMatrixMultiplication_dense_56_9_56(double* A, double* B, double* C, double* A_prefetch = NULL, double* B_prefetch = NULL
            , double* C_prefetch = NULL) {
//[...] SSE and AVX data types, SSE optimization
#if defined(__SSE3__) && defined(__AVX__)
//[...] pointers
for(int n = 0; n < 9; n+=3) {
  for(int m = 0; m < 48; m+=12) {
    //[...] pointers and loads
    for (int k = 0; k < 56; k++)
    {
    b_0 = _mm256_broadcast_sd(b0);
    //[...] more broadcasts and arithmetics on b
    a_0 = _mm256_load_pd(a0);
    a0+=4;
    c_0_0 = _mm256_add_pd(c_0_0, _mm256_mul_pd(a_0, b_0));
    c_0_1 = _mm256_add_pd(c_0_1, _mm256_mul_pd(a_0, b_1));
    c_0_2 = _mm256_add_pd(c_0_2, _mm256_mul_pd(a_0, b_2));
//[...] remaining kernel
#endif

#if defined(__MIC__)
//[...] MIC specifics
for(int n = 0; n < 9; n+=3) {
  //[...] MIC specififics
  #pragma prefetch b0,b1,b2,a0
  for(int k = 0; k < 56; k++)
  {
    //[...] MIC specifics
    c_0_0 = _mm512_fmadd_pd(a_0, b_0, c_0_0);
    c_0_1 = _mm512_fmadd_pd(a_0, b_1, c_0_1);
    c_0_2 = _mm512_fmadd_pd(a_0, b_2, c_0_2);
    a0 += 8;

//[...] remaining MIC kernel
#endif

#if !defined(__SSE3__) && !defined(__AVX__) && !defined(__MIC__)
//[...] fallback code
#endif

#ifndef NDEBUG
num_flops += 56448;
```

Code Generator – Programming Interface

```
db = Tools.parseMatrixFile(’matrices.xml’)
Tools.memoryLayoutFromFile(’layout.xml’, db)
arch = Arch.getArchitectureByIdentifier(’dhsw’)
volume = db[’kXiDivM’]
       * db[’timeIntegrated’]
       * db[’AstarT’]
       + db[’timeIntegrated’]
       * db[’ET’]
kernels = [(’volume’, volume)]
Tools.generate(
  ’path/to/output’,
  db,
  kernels,
  ’path/to/libxsmm_gemm_generator’,
  arch
)
```

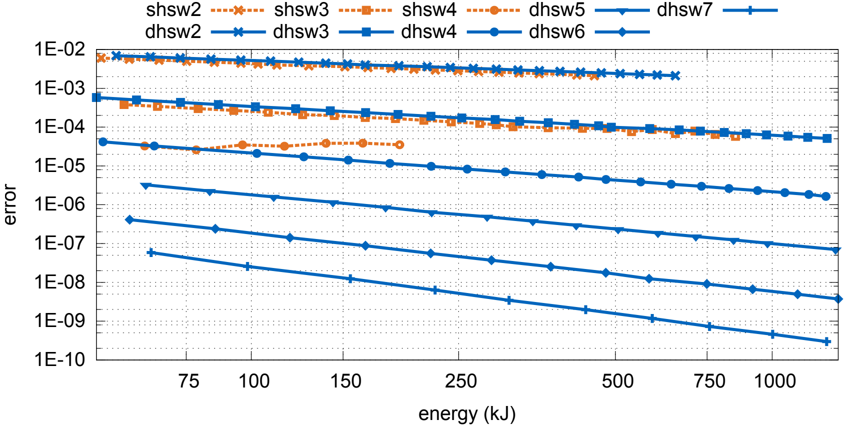
Exploit efficient backend: libxsmm library [11]

Benefit of High Order ADER-DG – Energy-Efficient



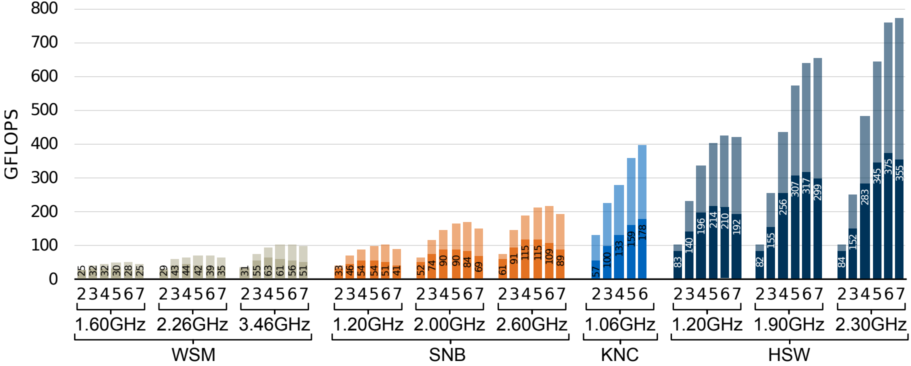
- mesasure maximum error vs. consumed energy
- for increasing discretisation order on regular meshes
- here: dual-socket “Haswell” server, 36 cores @1.9 GHz

Benefit of High Order ADER-DG – Energy-Efficient



- high order (“compute”) beats high resolution (“memory”)
- ≈ 35% gain in energy-to-solution for single precision, but only for low order

Benefit of High Order ADER-DG – Compute-Bound



- mesasured “GFlop/s” and “MFlop/s per Watt” for Westmere, Sandy Bridge, Knights Corner and Haswell architectures [1]
- at selected clock frequencies and for different order
- preference towards high order and low frequency on newest architectures

Part III

Accelerators – Dynamic Rupture Simulation on Xeon Phi Supercomputers

Heinecke, Breuer, Rettenberger, Gabriel, Pelties et al. [3]:
Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers (Gordon Bell Prize Finalist 2014)

On the Road from Peta- to Exascale?

SuperMUC @ LRZ, Munich

- 9216 compute nodes (18 “thin node” islands)
  147,456 Intel SNB-EP cores (2.7 GHz)
- Infiniband FDR10 interconnect (fat tree)
- #20 in Top 500: 2.897 PFlop/s

Image: LRZ

Stampede @ TACC, Austin

- 6400 compute nodes, **522,080 cores**
  2 SNB-EP (8c) + **1 Xeon Phi SE10P** per node
- Mellanox FDR 56 interconnect (fat tree)
- #8 in Top 500: 5.168 PFlop/s

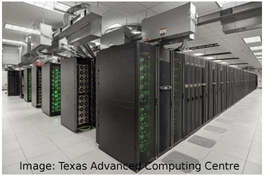


Image: Texas Advanced Computing Centre

Tianhe-2 @ NSCC, Guangzhou

- 8000 compute nodes used, **1.6 Mio cores**
  2 IVB-EP (12c) + **3 Xeon Phi 31S1P** per node
- TH2-Express custom interconnect
- #1 in Top 500: 33.862 PFlop/s

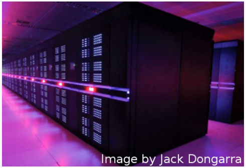


Image by Jack Dongarra

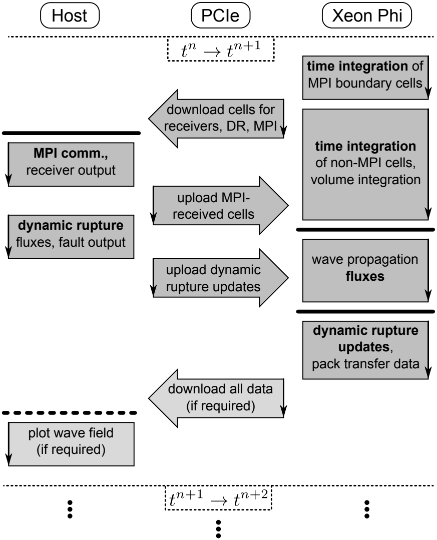
Optimization for Intel Xeon Phi Platforms

Offload Scheme:

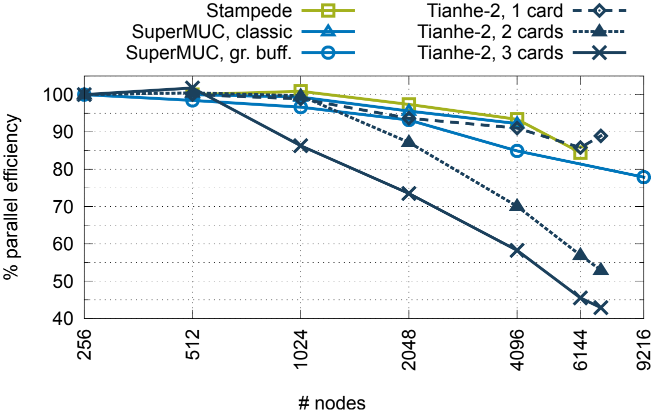
- hide2 communication with Xeon Phi and between nodes
- use “heavy” CPU cores for dynamic rupture

Hybrid parallelism:

- on 1–3 Xeon Phis and host CPU(s)
- reflects multiphysics simulation
- manycore parallelism on Xeon Phi

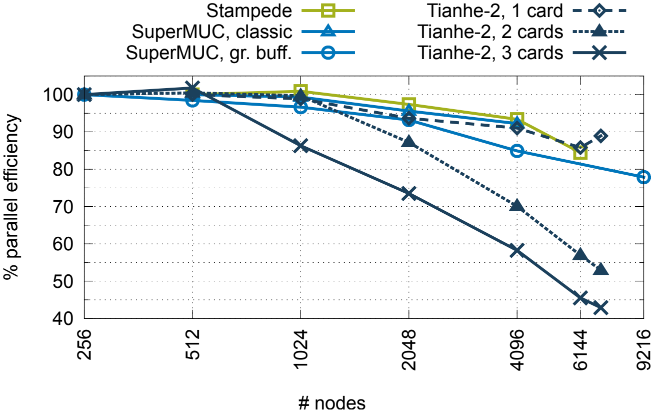

Host
PCIe
Xeon Phi
$t^n \rightarrow t^{n+1}$
time integration of MPI boundary cells
download cells for receivers, DR, MPI
MPI comm., receiver output
time integration of non-MPI cells, volume integration
upload MPI-received cells
dynamic rupture fluxes, fault output
wave propagation **fluxes**
upload dynamic rupture updates
dynamic rupture updates, pack transfer data
download all data (if required)
plot wave field (if required)
$t^{n+1} \rightarrow t^{n+2}$

Strong Scaling of Landers Scenario



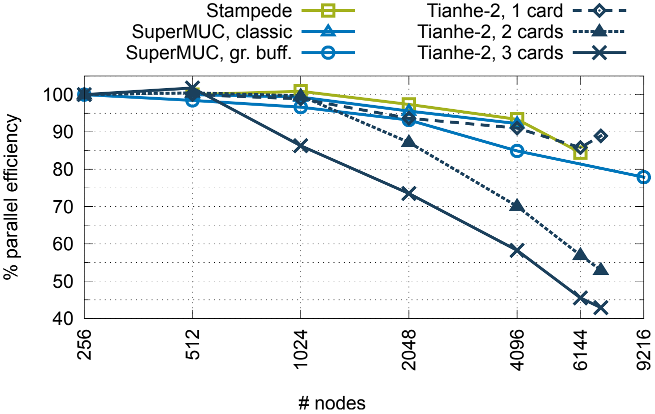
- 191 million tetrahedrons; 220,982 element faces on fault
- 6th order, 96 billion degrees of freedom

Strong Scaling of Landers Scenario



- more than 85 % parallel efficiency on Stampede and Tianhe-2 (when using only one Xeon Phi per node)
- multiple-Xeon-Phi performance suffers from MPI communication

Strong Scaling of Landers Scenario



- 3.3 PFlop/s on Tianhe-2 (7000 nodes)
- 2.0 PFlop/s on Stampede (6144 nodes)
- 1.3 PFlop/s on SuperMUC (9216 nodes)

Optimizing SeisSol for Xeon Phi (Knights Landing)

Heinecke et al., ISC 16 [7]

Code Generation:

- 512-bit wide vector processing unit
- profits from Knights Landing optimization of libxsmm library [11]

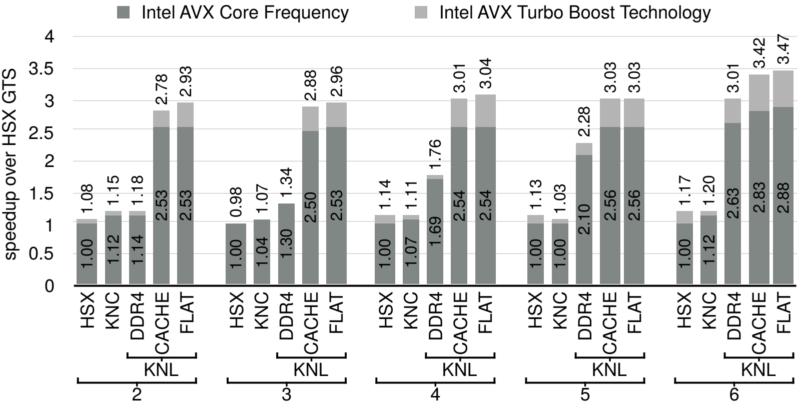
Memory Optimization:

- examine impact of DRAM-only, CACHE and FLAT mode
- FLAT mode: careful placement of element-local matrices in local MCDRAM (table from [7]):

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

Performance Results on Knights Landing

Heinecke et al., ISC 16 [7]



Landers scenario, 466,574 elements

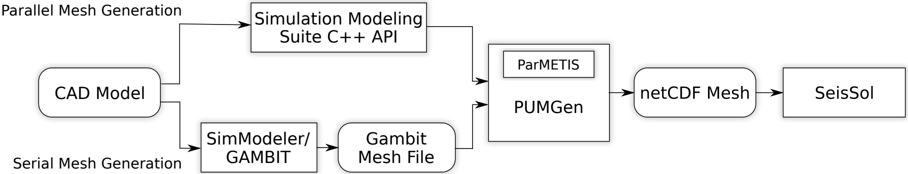
Part IV

Current Work – Simulations in Symmetric Mode on Salomon

Rettenberger, Uphoff, Rannabauer;
Project CzeBaCCA: Czech-Bavarian Competence Centre for Supercomputing Applications

Modify Mesh Input and Load Distribution

Scalable Mesh Partitioning and Input Pipeline:

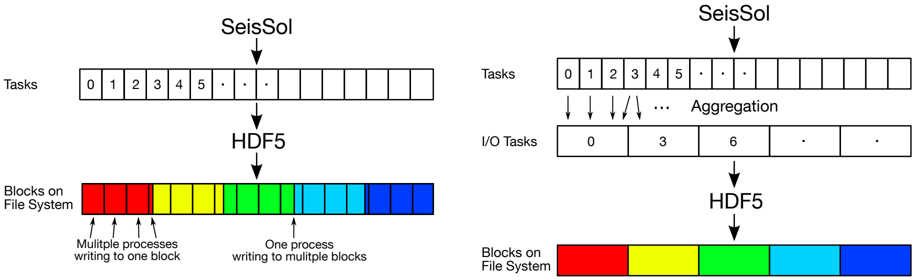


Towards Symmetric Mode:

- Modify weights for METIS graph partitioning: compensate speed differences between host CPU and Xeon Phi
- Work in progress: modify input of meshes
  → Xeon Phi mesh partitions may be read by host and sent via MPI
  → in case of bad I/O bandwidth (library support) of Xeon Phis

Work in Progress: Modify Wave Field Output

Aggregation of MPI ranks to speed up I/O: (Rettenberger [5])

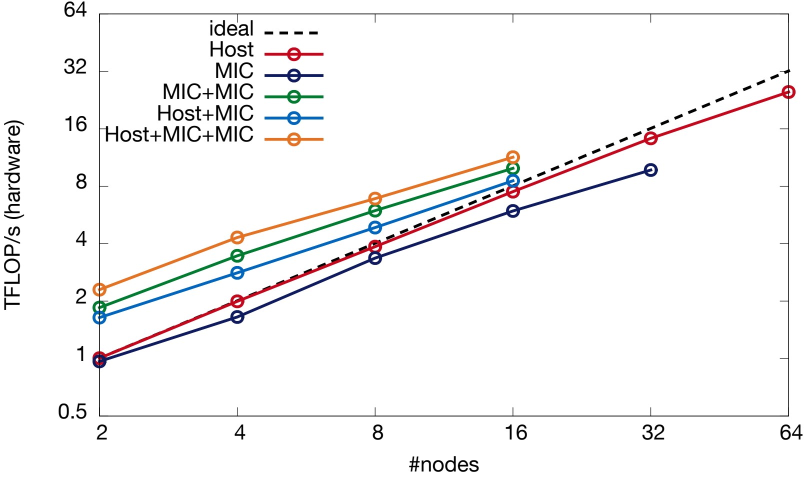


Towards Symmetric Mode:

- Output routines aggregate data from several MPI ranks
  → match I/O block size to achieve substantial speedup
- Only use host MPI ranks for output
  → again in case of bad I/O bandwidth (library support) of Xeon Phis

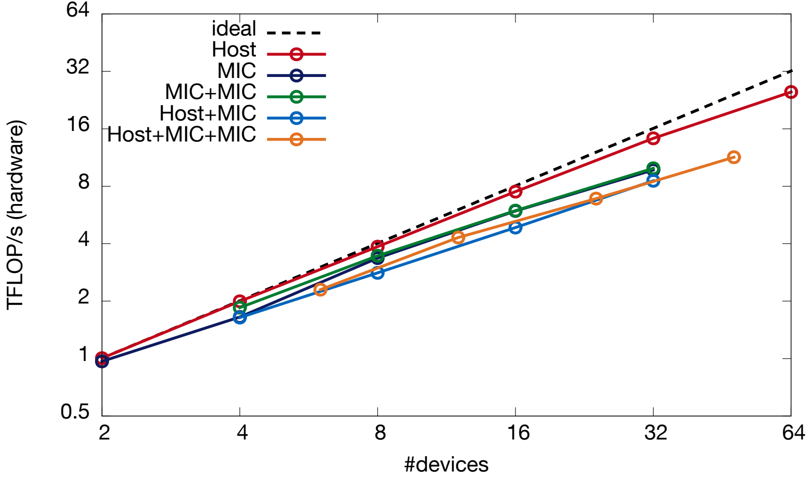
First Runs on Salomon – Native and Symmetric

Setup: LOH4 benchmark, 250k elements, order 6, no output yet



First Runs on Salomon – Native and Symmetric

Setup: LOH4 benchmark, 250k elements, order 6, no output yet



Part V

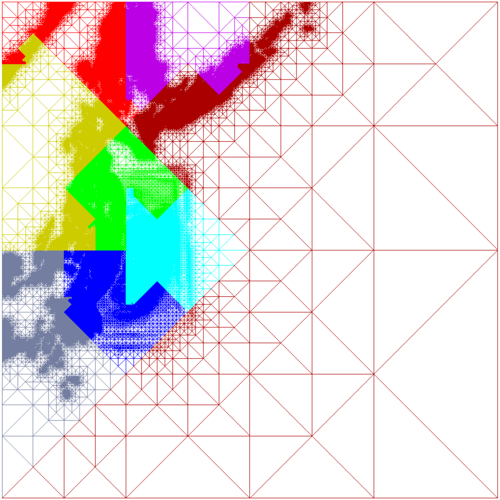
sam(oa)^2

Parallel Adaptive Mesh Refinement using Sierpinski Space Filling Curves

O. Meister, K. Rahnema, M. Bader [4]
Parallel Memory Efficient Adaptive Mesh Refinement
on Structured Triangular Meshes with Billions of Grid Cells

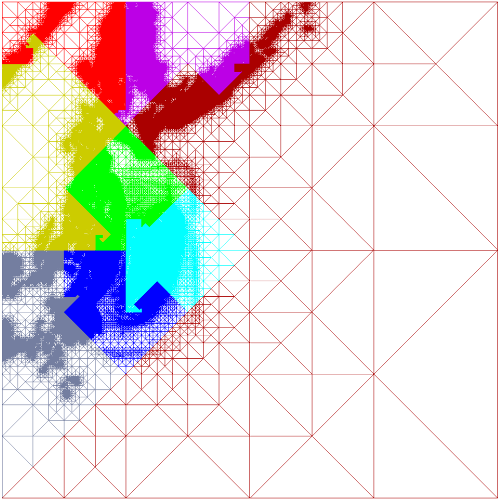
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



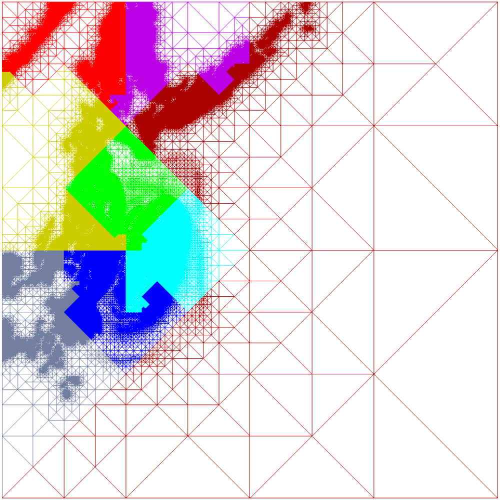
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



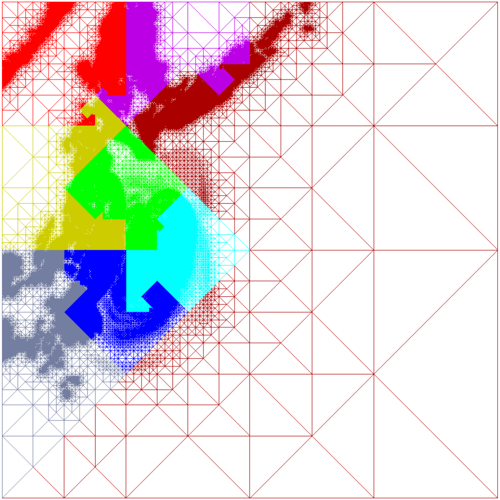
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



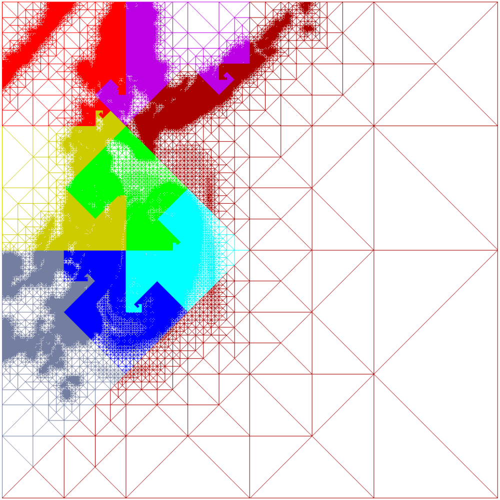
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



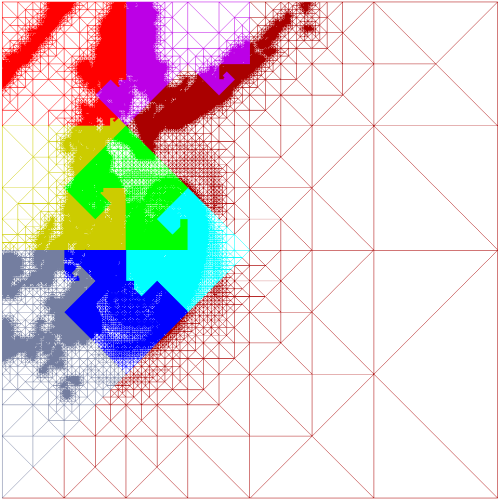
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



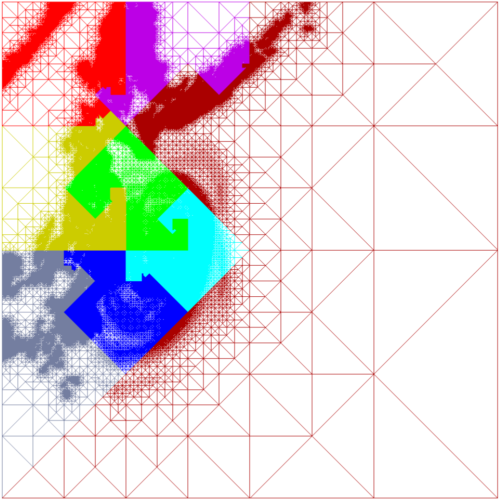
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



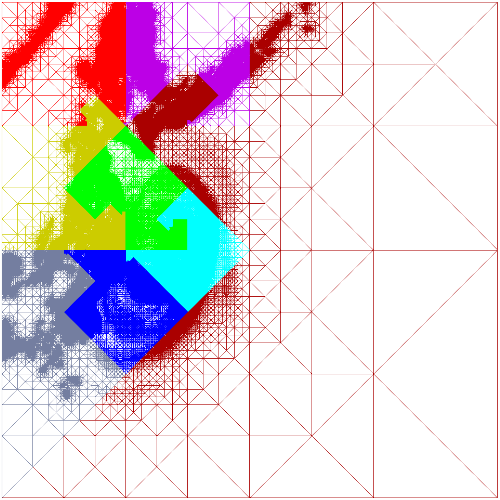
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



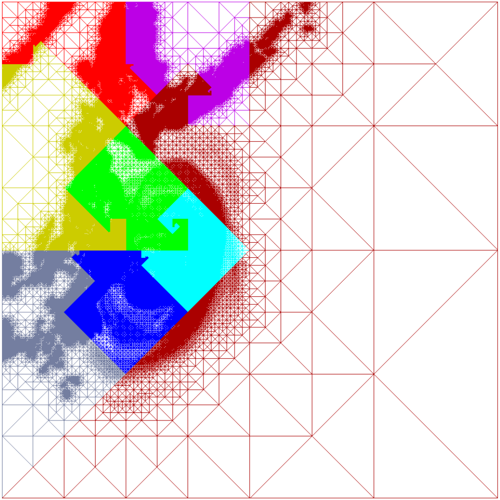
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



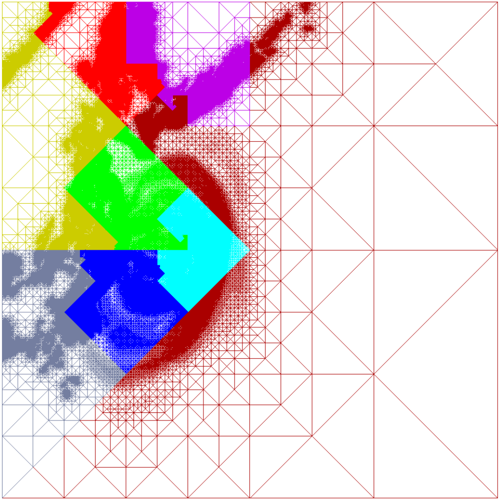
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



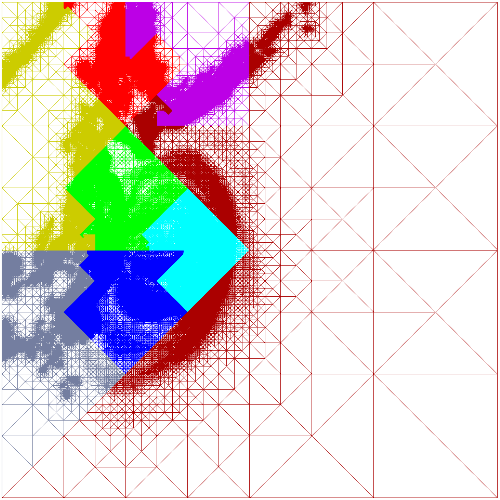
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



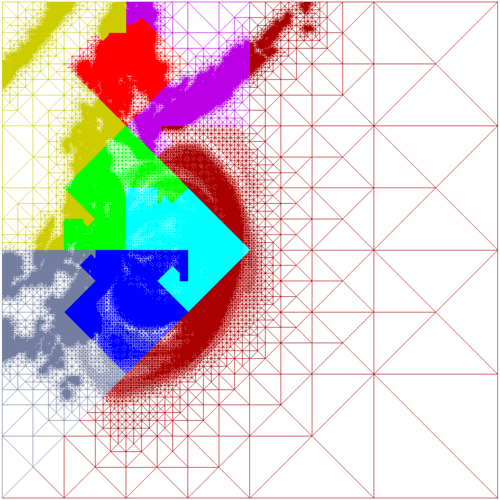
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



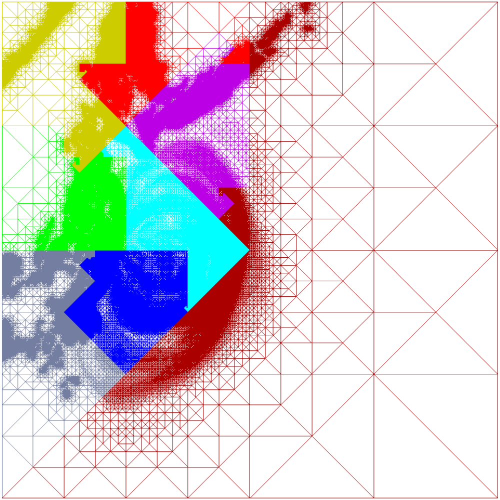
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



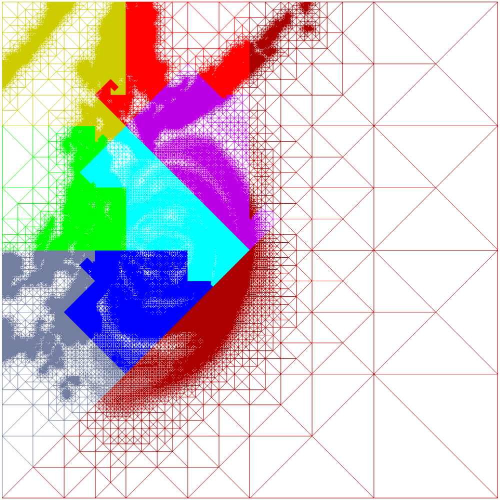
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



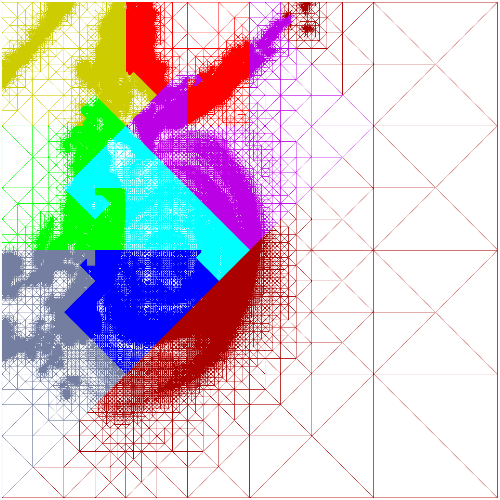
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



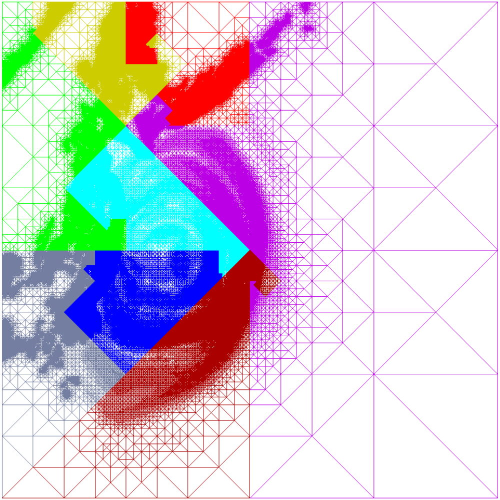
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



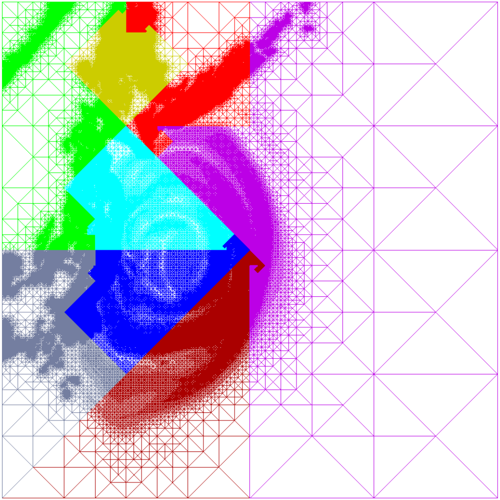
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



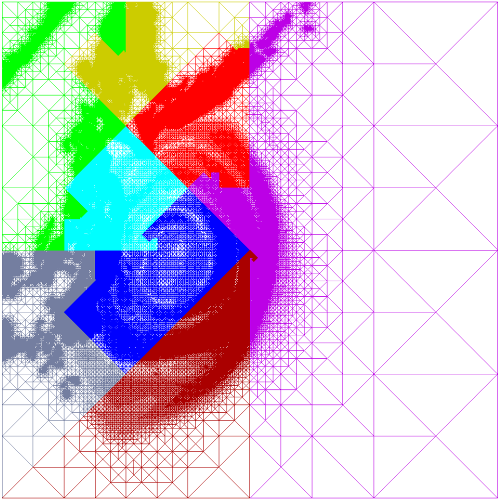
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



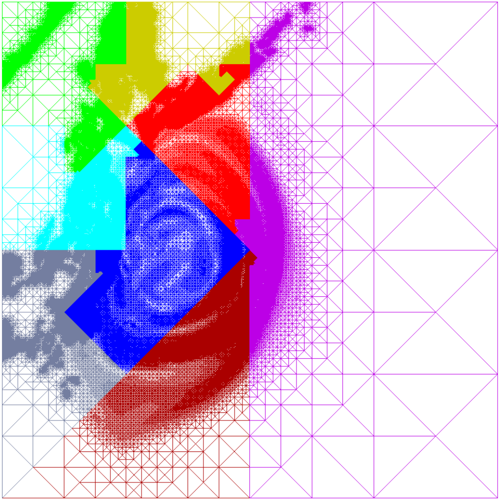
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



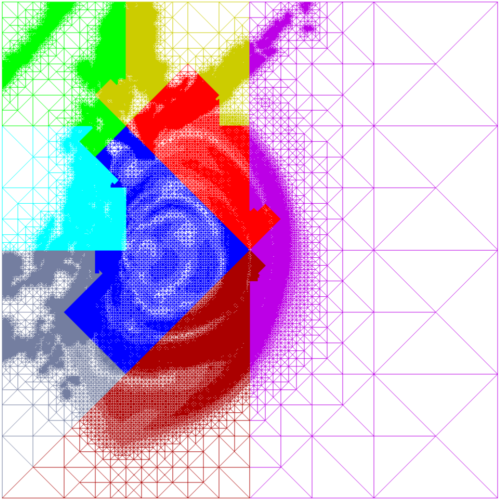
sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve

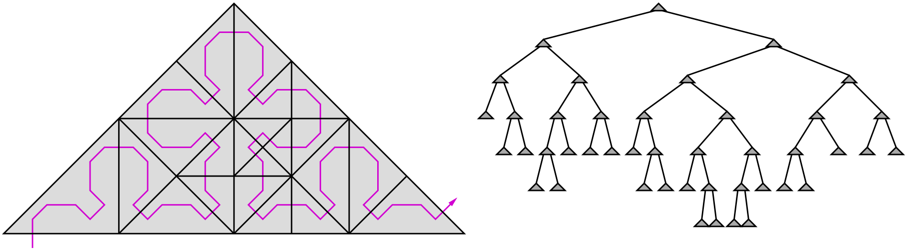


sam(oa)^2: Scalable Dynamic Adaptivity

Using Structured Triangular Meshes and Sierpinski Space-Filling Curve



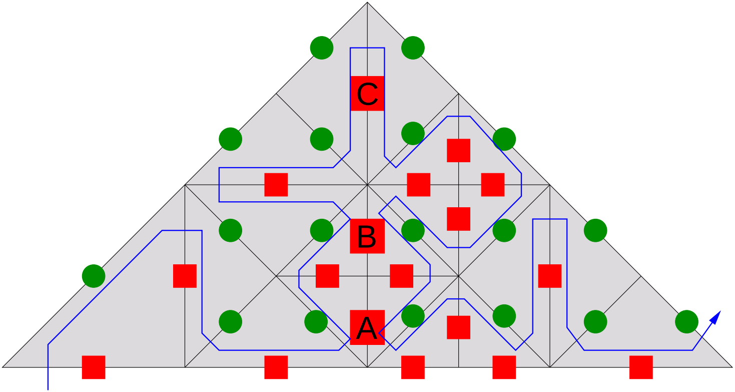
Triangular Meshes Generated by Bisection



“Newest Vertex Bisection” Refinement & Sierpinski Curves:

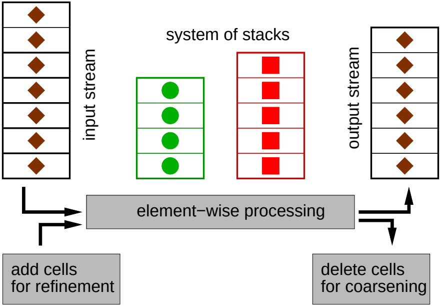
- fully adaptive grid described by a corresponding refinement tree
- tree and grid cells traversed in Sierpinski order
- minimum memory requirements (1 bit per tree node?!)
  → **triangle strips** as data structure
- exploit for cache efficiency and parallelisation

The Stack Principle for Data Exchange on Edges



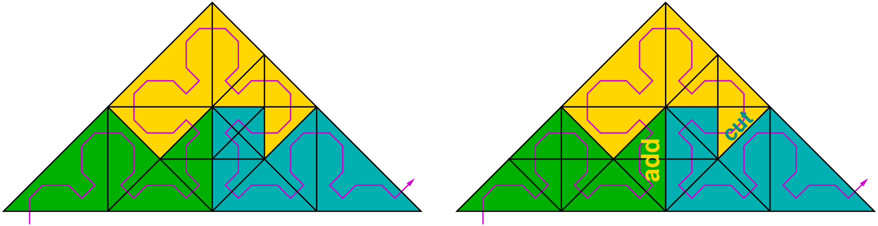
- to compute **numerical fluxes** in Finite Volume and DG methods
- to synchronise refinement of neighbour cells (**conforming grids**)

Stream- and Stack-based Processing



- drop fixed memory location of variables!
- **persistent** (degrees of freedom) vs. **non-persistent** (residuals, “old” variables) data
- aim: reduce memory footprint and number of traversals
- strongly element-oriented; impedes vectorization over elements

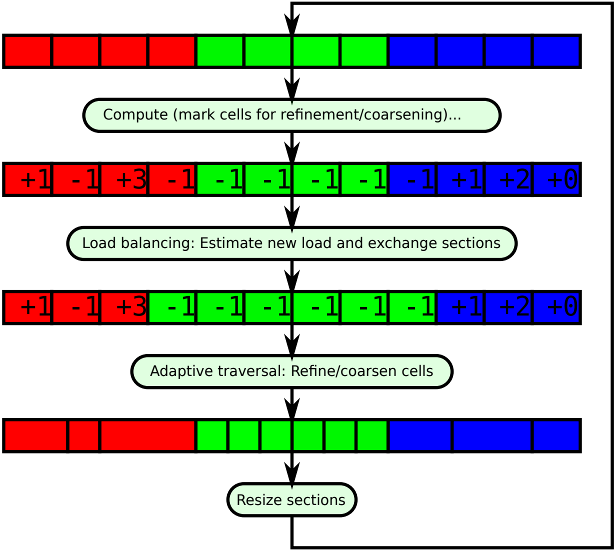
Partitions & Load Balancing: Sierpinski Sections



Use Sierpinski Space-Filling Curve for Partitioning:

- remeshing requires three sections: “add”, “keep”, and “cut”
- generalized towards restructuring of **Sierpinski sections**: turn *M* sections into *N* (balanced) sections
- migrate only sections to simplify data transfer (tolerate sacrifice on load balancing)
- flexible concept of “load”: number of cells, weight per cell, average measured runtime

Load Balancing Using Sierpinski Index Sections



Compute (mark cells for refinement/coarsening)...

+1 -1 +3 -1 -1 -1 -1 -1 -1 +1 +2 +0

Load balancing: Estimate new load and exchange sections

+1 -1 +3 -1 -1 -1 -1 -1 -1 +1 +2 +0

Adaptive traversal: Refine/coarsen cells

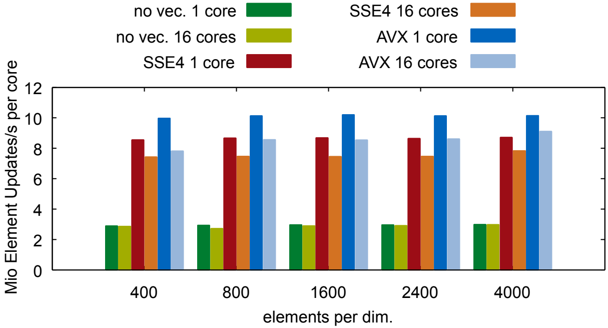
Resize sections

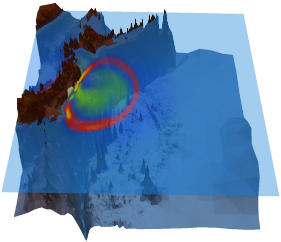
Part VI

Dynamically Adaptive Tsunami Simulation – Performance, Scalability, Patches

Oliver Meister, Kaveh Rahnema, Chaulio Ferreira

Tsunami Simulation: Vectorization vs. AMR



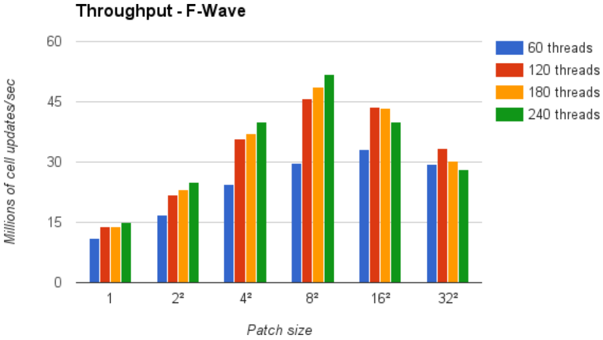


Vectorization Imperative on Modern CPU Architectures:

- example above: shallow water simulation on static **Cartesian mesh**
- speed-up due to intrinsics-implementation of augm. Riemann solver
- **sam(oa)^2:** mesh adaptivity impedes vectorization over grid cells
- possible remidy: **introduce regularly refined patches**

Using Patches – First Results on Xeon Phi

SuperMIC: 2 × Xeon Phi 5110P (60 cores, 1.1 GHz) ⇝ Chaulio Ferreira

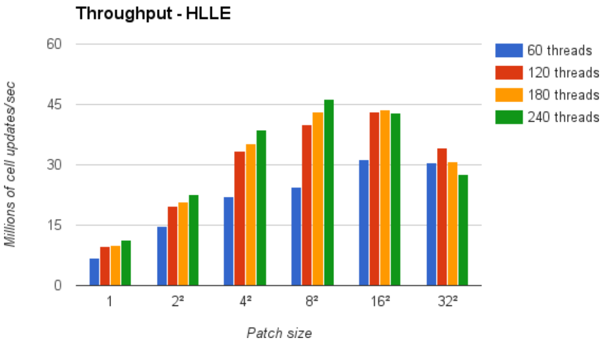


Simple f-wave Riemann solver:

- solid speedup, but away from optimum (vector width) and not perfectly scaling (single node, shared memory bandwidth)
- need to study influence of memory-bound parts, etc.

Using Patches – First Results on Xeon Phi

SuperMIC: 2 × Xeon Phi 5110P (60 cores, 1.1 GHz) ⇝ Chaulio Ferreira

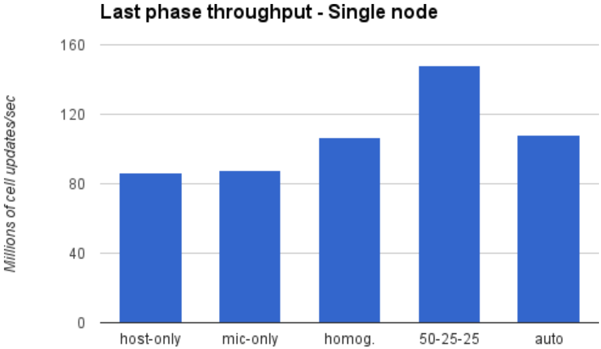


Vector HLLE solver: (compared against augmented Riemann)

- relatively larger speed-up (more compute-bound than f-wave)
- best time-to-solution for patch-size 8 ⇝ detailed analysis “to do”
- from ∼12 to ∼45 Mio element updates per second per node

Load Balancing for Symmetric Mode

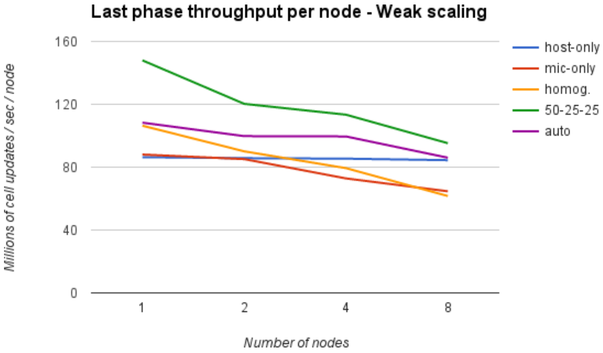
SuperMIC: 2× Ivy Bridge (8 cores, 2.6 GHz) plus 2× Xeon Phi 5110P (60 cores, 1.1 GHz)



- guided (2:1:1) clearly superior to homogeneous (1:1:1) load balancing between hosts (1 MPI rank) and Xeon Phi (2 ranks)
- “automatic” (measure runtime) load balancing finds 42:29:29
- suspect heavy losses due to communication

Symmetric Mode – Multiple Nodes

SuperMIC: 2× Ivy Bridge (8 cores, 2.6 GHz) plus 2× Xeon Phi 5110P (60 cores, 1.1 GHz)



- strong scaling currently impeded by slow communication
- withdraw to load balancing every 10 time steps
- will need to test with communication proxy/on Salomon

Part VII

Conclusions and Outlook

Conclusions on SeisSol

Performance Optimisation on Multi&Manycore Platforms:

- high convergence order and high computational intensity of ADER-DG → compute-bound performance on current and imminent CPUs
- code generation to accelerate element kernels
- careful tuning and parallelisation of the entire simulation pipeline (scalable mesh input, output and checkpointing)

Xeon Phi Platforms

- offload scheme scaled to 1.5 million cores (Tianhe-2, Stampede)
- our goal: scale in symmetric mode on heterogeneous supercomputers → current work on SuperMIC and esp. Salomon
- heterogeneity challenges exist in load balancing and scalable I/O
- SeisSol runs on Knights Landing → ISC'16 [7] and KNL-Book [8]

(Preliminary) Conclusions on sam(oa)^2

High-Performance Parallel AMR:

- making dynamically adaptive simulation codes achieve high performance is hard work
- space-filling-curve approaches for hybrid parallelism → sam(oa)^2
- load balancing in each time step is feasible (and hybrid parallism helps)

Performance Challenges for Modern Heterogeneous Platforms:

- crucial performance question #1:

  Where can I apply SIMD parallelism for dynamic adaptivity?

- patches offer additional opportunity for vectorization
  (similar: layered models → 2D adaptivity for 3D problems)
- crucial performance question #2:

  How do I effectively load balance with heterogeneity?

- hoping for automagic (runtime-based) load balancing;
  → prescribed weights currently more successful

Acknowledgements

Special thanks go to . . .

- the entire SeisSol team and all contributors, esp.:
  - Alex Breuer, Sebastian Rettenberger, Carsten Uphoff
  - Alex Heinecke
  - Alice Gabriel, Christian Pelties, Stephanie Wolherr
- the sam(oa)^2 team:
  - Oliver Meister, Kaveh Rahnema, Chaulio Ferreira
- all colleagues from the Leibniz Supercomputing Centre
- all colleagues from the IT4Innovations Supercomputing Centre
- for financial and project support:
  - Intel Corporation (IPCC ExScaMIC)
  - Volkswagen Foundation (project ASCETE)
  - BMBF (project CzeBACCA)

Publications

[1] A. Breuer, A. Heinecke, L. Rannabauer, M. Bader: *High-Order ADER-DG Minimizes Energy- and Time-to-Solution of SeisSol*. In: High Performance Computing, Proceedings of ISC 15, LNCS 9137, p. 340–357, 2015.

[2] A. Breuer, A. Heinecke, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties: *Sustained Petascale Performance of Seismic Simulations with SeisSol on SuperMUC.* In: Supercomputing, LNCS 8488, p. 1–18. PRACE ISC Award 2014.

[3] A. Heinecke, A. Breuer, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties, A. Bode, W. Barth, X.-K. Liao, K. Vaidyanathan, M. Smelyanskiy, P. Dubey: *Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers*. Gordon Bell Prize Finalist 2014.

[4] O. Meister, K. Rahnema, M. Bader: *Parallel Memory Efficient Adaptive Mesh Refinement on Structured Triangular Meshes with Billions of Grid Cells*.
ACM Transactions on Mathematical Software TOMS, accepted.

[5] S. Rettenberger, M. Bader: *Optimizing Large Scale I/O for Petascale Seismic Simulations on Unstructured Meshes* 2015 IEEE International Conference on Cluster Computing (CLUSTER), p. 314–317. IEEE Xplore, 2015.

[6] C. Uphoff, M. Bader: *Generating high performance matrix kernels for earthquake simulations with viscoelastic attenuation*. The 2016 International Conference on High Performance Computing & Simulation (HPCS 2016), p. 908–916.

Publications and References

[7] A. Heinecke, A. Breuer, M. Bader: *High Order Seismic Simulations on the Intel Xeon Phi Processor (Knights Landing)*. ISC High Performance, 2016.

[8] A. Heinecke, A. Breuer, M. Bader: *High Performance Seismic Simulations*. In J. Jeffers, J. Reinders, A. Sodani (ed.), Intel Xeon Phi Processor High Performance Programming – Knights Landing Edition, ch. 21. Morgan Kaufmann, 2016.

[9] M. Dumbser, M. Käser: *An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes – II. The three-dimensional isotropic case.* Geophys. J. Int. 167(1), 2006.

[10] D. George: *Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation.*
J. Comput. Phys. 227(6), 2008.

[11] A. Heinecke, G. Henry, M. Hutchinson, H. Pabst: *LIBXSMM: Accelerating Small Matrix Multiplications by Runtime Code Generation*, SC16, accepted.

[12] C. Pelties, A.-A. Gabriel, J.-P. Ampuero: *Verification of an ADER-DG method for complex dynamic rupture problems*, Geoscientific Model Development, 7(3), p. 847–866.

[13] C. Pelties, J. de la Puente, J.-P. Ampuero, G. B. Brietzke, M. Käser: *Three-dimensional dynamic rupture simulation with a high-order discontinuous Galerkin method on unstructured tetrahedral meshes.* J. Geophys. Res.: Solid Earth, 117(B2), 2012.