Sustained Petascale Performance of Seismic Simulations with SeisSol on SuperMUC

Alexander Breuer[1], Alexander Heinecke[1], Sebastian Rettenberger[1], Michael Bader[1], Alice-Agnes Gabriel[2], and Christian Pelties[2]

[1] Department of Informatics, Technische Universität München, Germany
[2] Department of Earth and Environmental Sciences, Geophysics, Ludwig-Maximilians-Universität München, Germany

Abstract. Seismic simulations in realistic 3D Earth models require peta- or even exascale computing power to capture small-scale features of high relevance for scientific and industrial applications. In this paper, we present optimizations of SeisSol – a seismic wave propagation solver based on the Arbitrary high-order accurate DERivative Discontinuous Galerkin (ADER-DG) method on fully adaptive, unstructured tetrahedral meshes – to run simulations under production conditions at petascale performance. Improvements cover the entire simulation chain: from an enhanced ADER time integration via highly scalable routines for mesh input up to hardware-aware optimization of the innermost sparse-/dense-matrix kernels. Strong and weak scaling studies on the SuperMUC machine demonstrated up to 90% parallel efficiency and 45% floating point peak efficiency on 147k cores. For a simulation under production conditions (10^8 grid cells, $5 \cdot 10^{10}$ degrees of freedom, 5 seconds simulated time), we achieved a sustained performance of 1.09 PFLOPS.

Keywords: seismic wave and earthquake simulations, petascale, vectorization, ADER-DG, parallel I/O

1 Introduction and Related Work

The accurate numerical simulation of seismic wave propagation through a realistic three-dimensional Earth model is key to a better understanding of the Earth's interior. It is thus utilized extensively for earthquake simulation and seismic hazard estimation and also for oil and gas exploration [15, 34, 47]. Although the physical process of wave propagation is well understood and can be described by a system of hyperbolic partial differential equations (PDEs), the numerical simulation of realistic settings still poses many methodological and computational challenges. These may include complicated geological material interfaces, faults, and topography. In particular, to resolve the high-frequency content of the wave field, which is desired for capturing crucial small-scale features, computational resources on peta- and probably exascale level are required [16, 28, 45]. Therefore, respective simulation software must be based on geometrically flexible and high-order accurate numerical methods combined with highly scalable and computationally efficient parallel implementations.

Appeared in:
J.M. Kunkel, T. Ludwig, and H.W. Meuer (Eds.): ISC 2014, LNCS 8488, pp. 1–18, 2014.
http://link.springer.com/chapter/10.1007%2F978-3-319-07518-1_1

In this paper we present optimizations of the seismic wave propagation software SeisSol to realize such high-order adaptive simulations at petascale performance. SeisSol is based on the Arbitrary high-order accurate DERivative Discontinuous Galerkin (ADER-DG) method on unstructured tetrahedral meshes [8]. It has been successfully applied in various fields of seismology [23, 41, 50], exploration industry [24] and earthquake physics [36, 37, 39] – demonstrating advantages whenever the simulations need to account for highly complicated geometrical structures, such as topography or the shapes of geological fault zones.

With our optimizations, we consequently target high-frequency, large-scale seismic forward simulations. To achieve high sustained performance on respective meshes consisting of 10^8–10^9 elements with resulting small time step sizes, it is essential to leverage all levels of parallelism on the underlying supercomputer to their maximum extent. The related question of sustained peak performance on application level has received growing interest over the last years [26, 33]. For example, the BlueWaters project aims on sustained petascale performance on application level and thus chose to not have their supercomputer listed in the Top500 list [31], despite its aggregate peak performance of 13 PFLOPS.

The demand for sustained performance at the petascale and scaling to hundreds of thousands of cores is also well reflected by related work on large-scale seismic simulations. Cui et al. [5] demonstrated respective earthquake and seismic hazard simulations, recently also utilizing heterogeneous platforms: for simulations on the GPU-accelerated Titan supercomputer, they reported a speed-up of 5 due to using GPUs [4]. That the reported performance is less than 5% of the theoretical peak performance of the GPU accelerators shows the challenge of optimizing production codes for these platforms. SpecFEM [40], a well-known community seismic simulation code, has been finalist in the Gordon Bell Award 2008 for simulations on 62k cores [3] and an achieved 12% peak performance on the Jaguar system at Oak Ridge National Laboratory. In 2013, SpecFEM was executed with 1 PFLOPS sustained performance on the BlueWaters supercomputer utilizing 693,600 processes on 21,675 compute nodes which provide 6.4 PFLOPS theoretical peak performance[3].

To achieve sustained petascale performance with SeisSol, optimizations in its entire simulation chain were necessary. In Sec. 2, we outline the ADER-DG solver for the wave equations and present an improved ADER time integration scheme that substantially reduced the time to solution for this component. Sec. 3 describes a highly-scalable mesh reader and the hardware-aware optimization of innermost (sparse and dense) matrix operators – two key steps for realizing simulations with more than a billion grid cells at a performance of 1.4 PFLOPS (44% peak efficiency) on the SuperMUC machine. Respective strong and weak scaling tests, as well as a high-resolution benchmark scenario of wave propagation in the Mount Merapi volcano under production conditions, are presented in Sec. 4.

[3] http://www.ncsa.illinois.edu/news/stories/PFapps/

2 An ADER-DG Solver for the Elastic Wave Equation

SeisSol solves the three-dimensional elastic wave equations, a system of hyperbolic PDEs, in velocity-stress formulation. In the following we give a short description of the fully-discrete update scheme, with focus on issues relevant for implementation at high performance. Details about the mathematical derivation of the underlying equations, handling of boundary conditions, source terms, viscoelastic attenuations, and more of SeisSol's features, can be found in [7–9, 25].

For spatial discretization SeisSol employs a high-order Discontinuous Galerkin (DG) Finite Element method with the same set of hierarchical orthogonal polynomials as ansatz and test functions [20, 25] on flexible unstructured tetrahedral meshes. For every tetrahedron T_k, the matrix $Q_k(t)$ contains the degrees of freedom (DOFs) that are the time-dependent coefficients of the modal basis on the unique reference tetrahedron for all nine physical quantities (six stress and three velocity components) [8]. The ADER time integration in SeisSol allows arbitrary high order time discretization. Adopting the same convergence order $\mathcal{O}$ in space and time results in convergence order $\mathcal{O}$ for the overall ADER-DG scheme. $\mathcal{O} = 5$ and $\mathcal{O} = 6$ are the typically chosen orders for production runs.

2.1 Compute Kernels and Discrete Update Scheme

The ADER-DG scheme in SeisSol is formulated as time, volume and boundary integrations applied to the element-local matrices $Q_k^n = Q_k(t^n)$, which contain the DOFs for tetrahedron T_k at time step t^n. Application of the different integrations leads to the DOFs Q_k^{n+1} at the next time step t^{n+1}. In fully explicit formulation each of these integration steps is formulated as a compute kernel that may be expressed as a series of matrix-matrix multiplications.

Time Kernel: The first compute kernel, consisting of the ADER time integration, derives an estimate $\mathcal{I}(t^n, t^{n+1}, Q_k^n)$ of the DOFs Q_k^n integrated in time over the interval $[t^n, t^{n+1}]$:

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

where the time derivates $\partial^j / \partial t^j Q_k(t^n)$ are computed recursively by:

$$\frac{\partial^{j+1}}{\partial t^{j+1}} Q_k = -\hat{K}^\xi \left(\frac{\partial^j}{\partial t^j} Q_k \right) A_k^\star - \hat{K}^\eta \left(\frac{\partial^j}{\partial t^j} Q_k \right) B_k^\star - \hat{K}^\zeta \left(\frac{\partial^j}{\partial t^j} Q_k \right) C_k^\star, \qquad (2)$$

with initial condition $\partial^0 / \partial t^0 Q_k(t^n) = Q_k^n$. Matrices $\hat{K}^{\xi_c} = M^{-1} (K^{\xi_c})^T$, with the reference coordinates $\xi_1 = \xi$, $\xi_2 = \eta$ and $\xi_3 = \zeta$, are the transpose of the stiffness matrices K^{ξ_c} over the reference tetrahedron multiplied (during preprocessing) by the inverse diagonal mass matrix M^{-1}. $A_k^\star$, $B_k^\star$ and $C_k^\star$ are linear combinations of the element-local Jacobians.

Volume Kernel: SeisSol's volume kernel $\mathcal{V}_k(\mathcal{I}_k^{n,n+1})$ accounts for the local propagation of the physical quantities inside each tetrahedron using the time integrated unknowns $\mathcal{I}(t^n, t^{n+1}, Q_k^n)$ computed by the time kernel expressed in Eqs. (1) and (2):

$$\mathcal{V}_k\left(\mathcal{I}_k^{n,n+1}\right) = \tilde{K}^{\xi}\left(\mathcal{I}_k^{n,n+1}\right)A_k^{\star} + \tilde{K}^{\eta}\left(\mathcal{I}_k^{n,n+1}\right)B_k^{\star} + \tilde{K}^{\zeta}\left(\mathcal{I}_k^{n,n+1}\right)C_k^{\star}. \tag{3}$$

Analog to the time kernel, the matrices $\tilde{K}^{\xi_c} = M^{-1}K^{\xi_c}$ are defined over the reference tetrahedron. Matrices $A_k^{\star}$, $B_k^{\star}$ and $C_k^{\star}$ are defined as in Eq. (2).

Boundary Kernel: The boundary integration connects the system of equations inside each tetrahedron to its face-neighbors. Here the DG-characteristic Riemann problem is solved, which accounts for the discontinuity between the element's own quantities and the quantities of its face-neighbors. The kernel uses the time integrated DOFs $\mathcal{I}_k^{n,n+1}$ of the tetrahedron T_k itself and $\mathcal{I}_{k(i)}^{n,n+1}$ of the four face-neighbors $T_{k(i)},\ i = 1, \ldots, 4$:

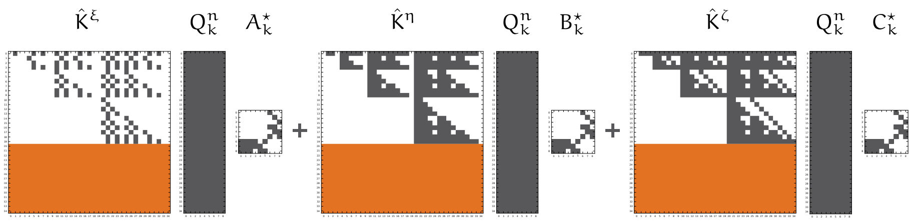
$$\begin{aligned}\mathcal{B}_k\left(\mathcal{I}_k^{n,n+1}, \mathcal{I}_{k(1)}^{n,n+1}, \ldots, \mathcal{I}_{k(4)}^{n,n+1}\right) = &\sum_{i=1}^{4}\left(M^{-1}F^{-,i}\right)\mathcal{I}_k^{n,n+1}\left(\frac{|S_k|}{|J_k|}N_{k,i}A_k^{+}N_{k,i}^{-1}\right)\\ &+\sum_{i=1}^{4}\left(M^{-1}F^{+,i,j_k(i),h_k(i)}\right)\mathcal{I}_{k(i)}^{n,n+1}\left(\frac{|S_k|}{|J_k|}N_{k,i}A_{k(i)}^{-}N_{k,i}^{-1}\right).\end{aligned} \tag{4}$$

$F^{-,i}$ and $F^{+,i,j_k(i),h_k(i)}$ are 52 integration matrices defined over the faces of the reference tetrahedron [8]. The choice of these matrices depends on the three indices $i = 1\ldots4$, $j_k(i) = 1\ldots4$ and $h_k(i) = 1\ldots3$, which express the different orientations of two neighboring tetrahedrons relative to each other with respect to their projections to the reference tetrahedron [1]. $N_{k,i}A_k^{+}N_{k,i}^{-1}$ and $N_{k,i}A_{k(i)}^{-}N_{k,i}^{-1}$ denote the element-local flux solvers, multiplied by the scalars $|J_k|$ and $|S_k|$, which are the determinant of the Jacobian of the transformation to reference space $\xi - \eta - \zeta$ and the area of the corresponding face, during initialization. The flux solvers solve the face-local Riemann problems by rotating the time integrated quantities to the face-local spaces and back via the transformation matrices $N_{k,i}$ and $N_{k,i}^{-1}$.

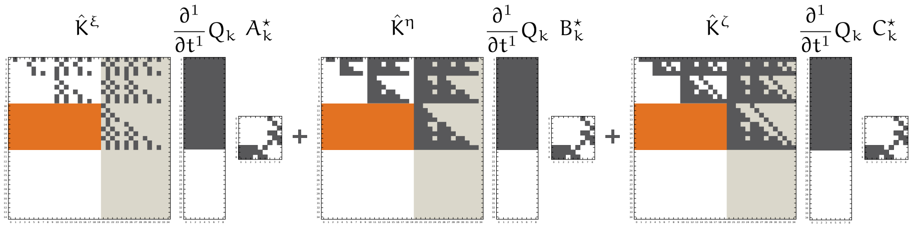
Update Scheme: Combining time, volume and boundary kernel, we get the complete update scheme from one time step level t^n to the next t^{n+1}:

$$Q_k^{n+1} = Q_k^n - \mathcal{B}_k\left(\mathcal{I}_k^{n,n+1}, \mathcal{I}_{k(1)}^{n,n+1}, \ldots, \mathcal{I}_{k(4)}^{n,n+1}\right) + \mathcal{V}_k\left(\mathcal{I}_k^{n,n+1}\right) \tag{5}$$

Matrix Computations: SeisSol's update scheme is heavily dominated by matrix-matrix operations. The size $B_{\mathcal{O}} \times B_{\mathcal{O}}$ of all matrices over the reference tetrahedron – $\hat{K}^{\xi_c}, \tilde{K}^{\xi_c}$, $F^{-,i}$ and $F^{+,i,j_k(i),h_k(i)}$ – depends on the order $\mathcal{O}$ of the scheme. In turn, the order $\mathcal{O}$ defines the number of used basis functions $B_{\mathcal{O}}$: $B_1 = 1$, $B_2 = 4$, $B_3 = 10$, $B_4 = 20$, $B_5 = 35$, $B_6 = 56$, $B_7 = 84$, ... The



(a) Computation of the first time derivative $\partial^1/\partial t^1 Q_k$.


$\hat{K}^{\xi}$
$\frac{\partial^1}{\partial t^1} Q_k$
$A_k^{\star}$
+
$\hat{K}^{\eta}$
$\frac{\partial^1}{\partial t^1} Q_k$
$B_k^{\star}$
+
$\hat{K}^{\zeta}$
$\frac{\partial^1}{\partial t^1} Q_k$
$C_k^{\star}$

(b) Computation of the second derivative $\partial^2/\partial t^2 Q_k$.

Fig. 1. First two recursions of the ADER time integration for a fifth-order method. Orange blocks generate zero blocks in the derivatives, light-gray blocks hit zero blocks.

element-local matrices – $A^{\star}$, $B^{\star}$, $C^{\star}$, $N_{k,i} A_k^{+} N_{k,i}^{-1}$ and $N_{k,i} A_{k(i)}^{-} N_{k,i}^{-1}$ – are of size 9×9, which correlates to the nine quantities of the elastic wave equations. A detailed description of the involved sparsity patterns is given in [1].

2.2 Efficient Evaluation of the ADER Time Integration

In this subsection, we introduce an improved scheme that reduces the computational effort of the ADER time integration formulated in [25] and in discrete form in Sec. 2.1.

Analyzing the sparsity patterns of the involved stiffness matrices, we can substantially reduce the number of operations in the ADER time integration. This is especially true for high orders in space and time: The transposed stiffness matrices $(K^{\xi})^T, (K^{\eta})^T$ and $(K^{\zeta})^T$ of order $\mathcal{O}$ contain a zero block starting at the first row that accounts for the new hierarchical basis functions added to those of the preceding order $\mathcal{O}-1$. According to Eq. (1) we have to compute the temporal derivatives $\partial^j/\partial t^j Q_k$ for $j \in 1 \ldots \mathcal{O}-1$ by applying Eq. (2) recursively to reach approximation order $\mathcal{O}$ in time.

The first step (illustrated in Fig. 1a) computes $\partial^1/\partial t^1 Q_k$ from the initial DOFs Q_k^n. The zero blocks of the three matrices $\hat{K}^{\xi_c}$ generate a zero block in the resulting derivative and only the upper block of size $B_{\mathcal{O}-1} \times 9$ contains non-zeros. In the computation of the second derivative $\partial^2/\partial t^2 Q_k$ we only have to account for the top-left block of size $B_{\mathcal{O}-1} \times B_{\mathcal{O}-1}$ in the matrices $\hat{K}^{\xi_c}$. As illustrated in Fig. 1b the additional non-zeros hit the previously generated zero block of derivative $\partial^1/\partial t^1 Q_k$. The following derivatives $\partial^j/\partial t^j Q_k$ for $j \in 3 \ldots \mathcal{O}-1$ proceed analogously and for each derivative j only the $\hat{K}^{\xi_c}$-sub-blocks of size

$B_{\mathcal{O}-j+1} \times B_{\mathcal{O}-j+1}$ have to be taken into account. Obviously, the zero blocks also appear in the multiplication with the matrices $A^\star$, $B^\star$ and $C^\star$ from the right and additional operations can be saved.

3 Optimizing SeisSol for Sustained Petascale Performance

In order to ensure shortest execution time and highest efficiency of SeisSol, all levels of parallelism of the considered petascale machine have to be leveraged. This starts with an efficient usage of the parallel file system and ends at a carefully tuned utilization of each core's vector instruction set. Additionally, we have to ensure that the problem's decomposition is efficiently mapped onto the computing resources. The state-of-the-art is to join several thousands of shared memory multi-core systems to large distributed-memory cluster installations. This system layout needs to be addressed for best performance, cf. [42, 43]: the shared memory within each node must be leveraged, which results in a faster intra-node communication and less partitions of the problem under investigation. Friedley et al. [10] presented strategies on how to address the first item within a pure distributed-memory programming model (MPI) [30]. Although such an approach utilizes the fast shared memory for intra-node communications, it still requires a partitioning of the problem handling all processing elements separately. We address this issue in SeisSol by switching to the explicit shared-memory programming model OpenMP [35] for parallel subtasks, such as loops over all process-local elements or gathering elements from the unstructured mesh into data exchange buffers.

In the following two subsections, we present the two major steps of our rigorous performance re-engineering process of turning SeisSol into a sustained petascale application, even on entry-level petascale systems. First, we describe how to read input data in a highly-parallel and efficient manner, and second, we elaborate how to achieve close-to-peak node-level performance by a highly-tuned BLAS3 implementation for dense and sparse matrix operations of small rank.

3.1 Highly Scalable Mesh Reader

SeisSol's approach is especially well-suited for the simulation of seismic wave propagation in highly complex geometries and heterogeneous 3D Earth models, relying on the flexibility offered by fully adaptive, unstructured tetrahedral meshes [8]. Comparable approaches, which first generate coarse unstructured meshes and successively refine these in a structured way (such as [2, 13]), do not provide the same approximation quality in terms of geometry and thus in the accuracy of the obtained results [38].

SeisSol's workflow for the generation and parallel input of high-resolution discretization grids requires three main steps: (1) Starting from a sufficiently detailed CAD model of the system geometry, a fully adaptive, unstructured tetrahedral mesh needs to be generated using a suitable meshing software; the desired high-quality meshes (e.g., without degenerated mesh cells) consisting of 10s to

100s of million mesh cells can only be generated by few packages – our preferred software being SimModeler by Simmetrix[4]. The mesh information is stored to be used for multiple simulations: for these mesh files we used the ASCII-based GAMBIT format, so far. (2) To generate compact and balanced parallel partitions, we use the multilevel k-way partitioning algorithm [21, 22] implemented in ParMETIS (or similar partitioning software). The output is typically a partitioning file that maps each grid cell to a specific partition. (3) From the mesh file and the partition-mapping file, not only the partitions need to be generated for each MPI process, but also the communication structures and ghost layers (virtual elements of neighboring MPI domains) for communication. In previous versions of SeisSol, this step had been part of the initialization phase. However, it limited simulations to meshes with only a few million grid cells, because computing the required information did not scale to several thousand MPI ranks and did not allow parallel I/O.

In order to retain the flexibility regarding mesh generator and partitioning software for individual users, we decided not to integrate the mesh generator into SeisSol, as for example proposed in [3]. Instead, we reorganized steps (2) and (3) of our workflow into an offline part to compute partitions and communication structures, and a highly-scalable input phase that remains in SeisSol.

In the offline part, we read the GAMBIT mesh file and use ParMETIS to construct the so-called dual graph of the mesh. As the dual graph reflects the data exchange between grid cells in the ADER-DG scheme, both the partitioning and the computation of communication structures are based on this graph. After the partitioning with ParMETIS, we sort the elements and vertices according to their partition and duplicate vertices that belong to multiple partitions. The small overhead due to boundary vertex doubling easily pays off by the simplifications in the resulting new format. Then, we precompute the communication structures and ghost layers required by SeisSol from the dual graph. The ordered elements and vertices are stored together with the precomputed data in a customized format. This new file format is based on netCDF [44], a generic binary file format for multidimensional arrays that supports parallel I/O. Our offline tool is parallelized to satisfy the memory requirements of large meshes. To convert a mesh with 99,831,401 tetrahedral elements for our strong scaling setup (Sec. 4.2) from GAMBIT to netCDF we required 47.8 minutes on 64 cores and consumed 32.84 GB memory. 65% of this time was spent reading the original file.

Reading the netCDF file during the redesigned input phase exploits netCDF's efficient MPI-I/O-based implementation and strongly profits from parallel file systems available on the respective supercomputer. For our largest simulation with a mesh of 8,847,360,000 tetrahedrons the parallel mesh input required only 23 seconds on 9,216 nodes (using the GPFS file system, see the description of the weak scaling setup, Sec. 4.1).

[4] http://simmetrix.com/

3.2 (Sparse-)DGEMM Functions for Matrices of Small Rank

According to the equations and matrix structures elaborated in Sec. 2, several matrix operations with different sparsity patterns and small ranks had to be optimized. Before the re-engineering process, SeisSol stored all sparse matrices in a simple coordinate format, regardless of the actual sparsity pattern of a certain matrix. Dense matrix structures were only used for the DOFs, Q_k. The goal of our optimization was to replace the kernel operations by high-performance and specialized (sparse-)DGEMM functions within SeisSol. Throughout the integration schemes (compare Sec. 2.1), the following types of matrix-matrix multiplications are required: *sparse* $\times$ *dense* = *dense*, *dense* $\times$ *sparse* = *dense* and *dense* $\times$ *dense* = *dense*.

In previous work [1], we adopted the original approach of SeisSol and handled all matrices except DOF matrices as sparse. Since the sparsity patterns of all matrices are known up-front, we generated specialized sparse-DGEMM routines that efficiently leverage the hardware's vector instruction extensions. This was achieved by hard-wiring the sparsity pattern of each operation into the generated code through unrolling [29]. Note that such a generation has to take place in an offline step, because compilers' auto-vectorizer, source-to-source vectorizers such as Scout [27] or highly-efficient BLAS libraries such as Intel MKL [19] or Blaze [18] regard the index-structures of the sparse matrices as variables since they are runtime parameters. Through this approach we were able to accelerate SeisSol by a factor of 2.5–3.0 as presented in [1].

Besides handling either operand matrix A or B of a DGEMM call as sparse, it is profitable to generate optimal code for dense matrix kernels as well, because many of the operands feature dense blocks, cf. [32, 49, 53]. Due to the small size of the matrices (56×56, 56×9, 9×9 for $\mathcal{O} = 6$), calling highly-efficient BLAS3 functions offered by vendor implementations does not lead to satisfying results. From Sec. 2 we know that the number of columns N of at least two operands is always 9, so we hard-wired $N = 9$ into our code. Additionally, this optimization prevents reusing inner-most block-operations of processing 4×4 blocks or 8×4 blocks as proposed in [14, 54, 55]. However, we applied identical ideas to an inner-most kernel of 12×3 blocks which perfectly match with the size of Intel's AVX registerfile (16 32-byte registers).

In order to select the fastest alternative between the operation-specific sparse kernel and its highly-tuned dense counterpart, we extended our time, volume and boundary kernels by a sparse-dense switch: during initialization we hard-wire function pointers to an optimal matrix kernel (sparse or dense) for each individual matrix operator appearing in the different kernels. In order to decide whether sparse or dense kernels for a specific matrix lead to shorter time to solution, we performed dry-runs on the first 100 time steps of the representative SCEC LOH.1 benchmark [6] with 7,252,482 elements for approximation orders $\mathcal{O} \in \{2, 3, \ldots, 6\}$. Such a tuning process for linear algebra is well-known from projects such as ATLAS [51, 52] or OSKI [48]. For each run we identified the percentage of non-zeros required to make the corresponding matrix kernel perform better as dense (instead of sparse) kernel. These automated training runs in the preparation phase of SeisSol returned the desired collection of kernel operations that provides the shortest time to solution. The identified kernels were used afterwards to compile the final SeisSol binary for a certain order of approximation. A detailed discussion of this switch can be found in [17], which in general suggests to switch to the discussed sparse computing kernels if more than 80% of the matrix's entries are zeros.

In conclusion we want to highlight that both of the implemented kernel variants support the improved ADER time integration scheme presented in Sec. 2.2. This was realized by injecting jumps that skip the processing of unused sub-blocks in the case of generated sparse-DGEMM kernels. When using their dense counterpart, no change was required as our highly-optimized DGEMM kernel features variable values for M and K (BLAS notation). However, M is rounded towards the next multiple of the blocking width of our inner-most kernel. Such a 'zero-padding' is required to ensure best-possible performance, as it reduces the complexity of the instruction mix and thus optimally exploits level 1 instruction caches. Note that such a rounding is not needed along K since there is no vectorization in place.

4 Performance Evaluation on SuperMUC

In this section we analyze the behavior of SeisSol in strong and weak scaling benchmarks executed on SuperMUC[5] at Leibniz Supercomputing Centre. SuperMUC features 147,456 cores and is at present one of the biggest Intel Sandy Bridge systems worldwide. It comprises two eight-core Intel Xeon E5-2680 processors per node at 2.7 GHz. With a collective theoretical double-precision peak performance of more than 3 PFLOPS, it was ranked #10 on the November 2013 Top500 list [31]. In contrast to supercomputers offered by Cray, SGI or IBM's BlueGene, the machine is based on a high-performance Infiniband commodity network organized in a fat tree with 18 islands that consist of 512 nodes each. All nodes can communicate within each island at full IB-FDR-10 data-rate. In case of inter-island communication, four nodes share one uplink to the spine switch, leading to reduced bandwidth for inter-island communication.

For all of the following results we derived the number of double-precision floating point operations (FLOP) in a preprocessing step. In addition, we validated the obtained numbers by aggregating FLOP-counters integrated into SeisSol's compute kernels. We followed this combined approach for all strong-scaling runs and small weak-scaling runs (< 512 cores). For reasons of practicability we restricted the application of the runtime method to smaller weak-scaling jobs. The obtained floating point operations per second (FLOPS) rates allow us to calculate directly the peak efficiency of SeisSol on SuperMUC.

Before scaling SeisSol to the full SuperMUC petascale machine, we again ran the SCEC LOH.1 benchmark in a strong-scaling setting to determine the performance increase of our optimized version of SeisSol in comparison to the classical SeisSol implementation in Fig. 2. We see a factor-5 improvement in time to solu-

[5] http://www.lrz.de/services/compute/supermuc/

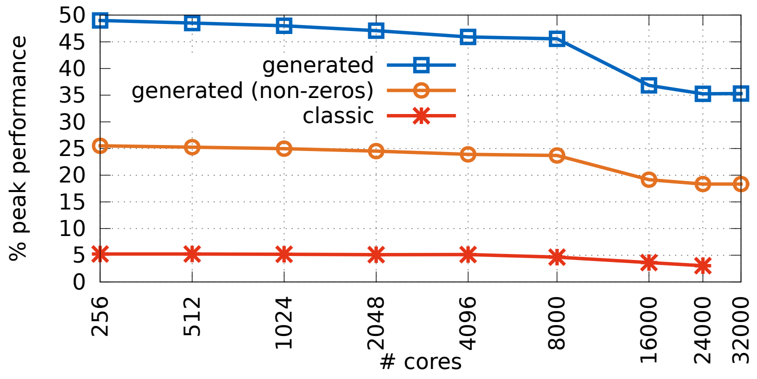


Fig. 2. Strong scaling of the SCEC LOH.1 benchmark with 7,252,482 elements using approximation order $\mathcal{O} = 6$ for the classic version of SeisSol (red) and our highly tuned derivate (blue and orange). The slight efficiency decrease for >8,000 cores is mostly due to SuperMUC's island concept.

tion, which translates into 25% peak performance without accounting for vector instruction set padding and nearly 50% machine efficiency on vector instruction level. Note that the kink in all three performance series, when increasing the core count from 8,000 to 16,000 cores, is due to SuperMUC's island concept. Besides a raw performance gain of 5× on the same numbers of cores, our optimized version of SeisSol is able to strong-scale to larger numbers of cores due to its hybrid OpenMP and MPI parallelization approach leading to an aggregate speedup of nearly 10 in this setting.

4.1 Weak Scaling

For a weak scaling study, and also to test SeisSol's limits in terms of mesh size, we discretized a cubic domain with regularly refined tetrahedral meshes. The computational load was kept constant with 60,000 elements per core and a total of 100 time steps per simulation. To complete the setup, we used a sinusoidal wave as initial condition and 125 additional receivers distributed across the domain. We performed the weak scaling simulations starting at 16 cores as a baseline. Then, we doubled the number of cores in every next run up to 65,536 cores and performed an additional run utilizing all of the available 147,456 cores. Fig. 3 shows the fraction of the theoretical peak performance. We observed the maximum performance using all 147,456 cores with a sustained performance of 1.42 PFLOPS, which correlates to 44.5% of theoretical peak performance and a parallel efficiency of 89.7%.

In contrast to the observations made for the LOH.1 strong scaling benchmark presented in the previous section, a performance kink is identifiable when more than two islands are employed, specifically, when moving from 16,000 to

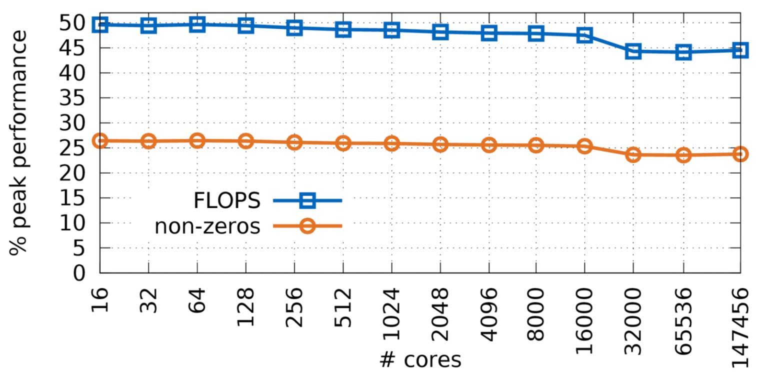


Fig. 3. Weak scaling of the cube benchmark with 60,000 elements per core using approximation order $\mathcal{O} = 6$. Shown are hardware FLOPS (blue) and non-zero operations (orange). The slight efficiency decrease for >16,000 cores is mostly due to SuperMUC's island concept.

32,000 cores. Since inter-island communication only happens in one dimension of the computational domain and we performed a weak scaling study, we have lower requirements on the communication infrastructure compared to the LOH.1 benchmark. Therefore, in the case of two islands only 100 ($< \frac{1}{4}$ of the nodes per island) inter-island communication channels are used. However, when utilizing four or more islands the 128 up-links per island, given by the islands' pruned tree organization, become a limiting factor as these runs demand more than 200 channels.

4.2 Strong Scaling

The strong scaling setup was based on a typical scenario setup as used in geophysical forward modeling. We discretized the volcano Mount Merapi (Java, Indonesia) with a total number of 99,831,401 tetrahedrons. The setup contained two layers of different material properties (density, seismic wave speeds), topography obtained from local digital elevation models [11, 12] and a moment tensor representation of a double-couple seismic point source approximation. We elaborate the geophysical specifications in Sec. 4.3, where a run under production conditions is presented.

In the strong scaling runs, we performed 1000 time steps on 1024 to 65,536 cores by doubling the number of cores in every next run. Finally, an additional run on all 147,456 cores was executed. In the extreme case of using the entire SuperMUC, the load per core was only ≈ 677 elements. Fig. 4 shows the fraction of theoretical peak performance for all runs. The sustained performance on all 147,456 cores was 1.13 PFLOPS, which is 35.58% of theoretical peak performance

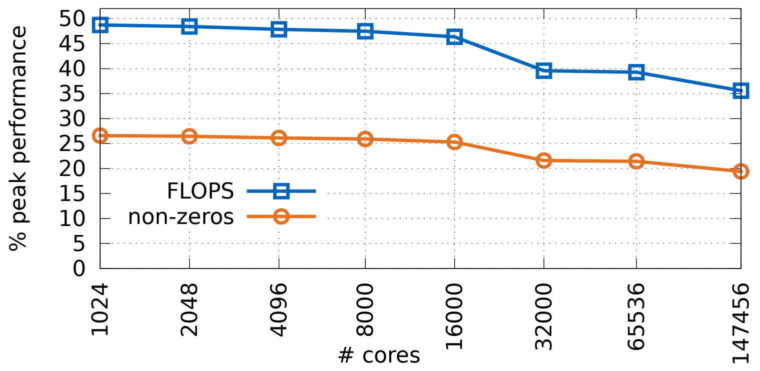


Fig. 4. Strong scaling of the Mount Merapi benchmark with 99,831,401 tetrahedrons using approximation order $\mathcal{O} = 6$. Shown are hardware FLOPS (blue) and non-zero operations (orange).

and correlates to 19.42%, if only non-zero operations are considered. The parallel efficiency with 1024 cores as baseline was at 73.04%.

4.3 Production Run

Finally, we performed a simulation under production conditions on all 147,456 cores of SuperMUC. As a test case scenario, we chose the volcano Mount Merapi to demonstrate the solver's capabilities and that its optimizations regarding parallel I/O, hybrid parallelization as well as vectorization are sustained for arbitrary mesh and model complexity.

Several eruptions of Mount Merapi have caused fatalities and the stratovolcano is still highly active. Thus, Mount Merapi is subject of ongoing research and a network of eight seismographs monitors tremors and earthquakes, with the goal of implementing a functional early warning system in the future. Seismic forward modeling could help to locate such tremors and to gain a better image of the geological subsurface structure. Therefore, synthetic time series and 3D wave field visualization are required to support the interpretation.

Our simulation setup was identical to the strong scaling runs, with the addition of a set of 59 receivers distributed throughout the domain, sampling the physical wave field every 0.001 seconds. Also, in contrast to the strong scaling runs, we now computed 166,666 time steps to accomplish 5 seconds in simulation time. The total simulation took 3 hours and 7.5 minutes with 1 minute and 22 seconds being initialization. The remainder was spent in the time marching loop, which ran at 1.09 PFLOPS. This correlates to 34.14% theoretical peak performance. Thus, we conclude that our optimization can be fully exploited under realistic research conditions.

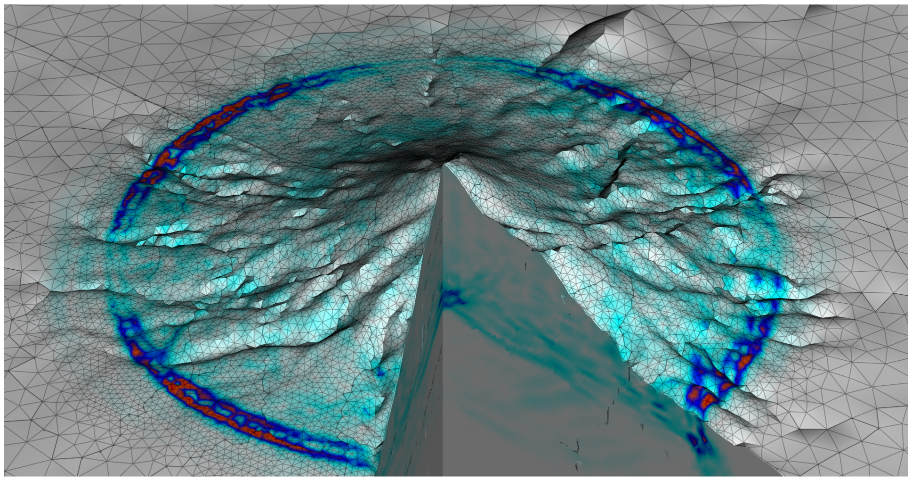
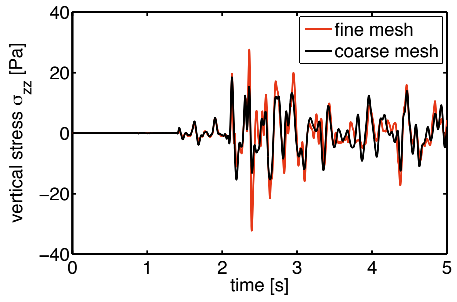
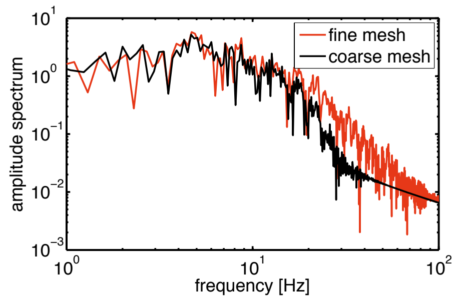


Fig. 5. Wave field of the Mount Merapi benchmark after 4 seconds. For illustration purposes a smaller mesh with 1,548,496 tetrahedrons was used. The tetrahedral approximation of the topography is shown as overlay on the surface. Insight into the interior is provided by virtually removing a section in the front. Warmer colors denote higher wave field energy.

Fig. 5 illustrates the model setup and shows the wave field at 4 seconds simulation time. The achieved high resolution (due to applied order and element size) allows accurate modeling of the highest frequencies (more than 20 Hz) capturing small-scale details in the wave-form modulation due to topography or wave interaction at material contrasts. Wave field complexity is already visible in Fig. 5, although the image is based on data from a much coarser mesh for visualization purposes. In Fig. 6 we compare results from the presented high-resolution simulation with a coarser simulation. The coarser mesh discretized the model domain by 100 m in the shallower region and 500 m in the remainder resulting in 1,548,496 tetrahedrons, whereas the fine meshes used an element edge length of 28 m and 100 m, respectively resulting in 99,831,401 tetrahedrons. Both simulations were performed with $\mathcal{O} = 6$. The receiver station for Fig. 6 was randomly picked and is located in the interior of the volcano. As an example, we plot the vertical stress component σ_{zz}. In (a) the time series are plotted, showing a higher amount of wave form details for the fine-mesh simulation. The increased frequency content is also reflected by plotting the spectrum (b) of the time series. Due to the decreased numerical errors by using the fine mesh more details and small scale features are uncovered.



(a)



(b)

Fig. 6. In (a) we compare the time series produced with the fine mesh (red) and the coarse mesh (black). Clearly, the fine mesh shows larger amplitudes (e.g. at second 2.3) that might be damped out for the coarse mesh simulation due to increased numerical dissipation. More difficult to identify, but visible in the spectrum (b), is the increased frequency content beyond 20 Hz of the fine mesh.

5 Conclusions

Performing a head-to-toe performance re-engineering, we have enabled SeisSol for seismic simulations at petascale. The improved mesh input allows SeisSol to run simulation settings on meshes with billions of grid cells and (for order $\mathcal{O} = 6$) more than 10^{12} degrees of freedom. Based on the generation of optimized sparse- and dense-matrix kernels, we achieved 20–25% effective peak efficiency, i.e., disregarding instructions due to vector padding or dense computations. These results demonstrate that SeisSol is competitive with earlier discussed codes reaching 10–16% peak efficiency. The aggregate performance gain due to hardware-aware single-node optimizations and hybrid OpenMP and MPI parallelization leads to speedups of 5–10 for typical production runs. The accomplished 5 seconds in simulation time for the Mount Merapi scenario show that the optimizations presented in this paper enable SeisSol to run not only scaling, but also production runs at petascale performance.

Considering the trend towards supercomputing architectures that are based on accelerators and many-core hardware, in general (GPUs, Intel MIC, e.g.), we demonstrated that substantial performance gains that address vectorization and efficient use of hybrid programming models also lead to substantial performance gains on commodity CPUs. In fact, the respective optimizations have prepared SeisSol for current and future many-core chips, such as Intel Xeon Phi [46].

6 Acknowledgements

We would like to thank Mikhail Smelyanskiy at Intel's Parallel Computing Lab for assisting us in the development of hardware-aware kernels, as well as all colleagues of the Leibniz Supercomputing Centre who supported us in the execution of the large-scale SuperMUC runs.

Our project was supported by the Volkswagen Stiftung (project ASCETE: Advanced Simulation of Coupled Earthquake-Tsunami Events) and by the Bavarian Competence Network for Technical and Scientific High Performance Computing (KONWIHR). Computing resources for this project have been provided by the Leibniz Supercomputing Centre under grant: pr83no.

References

1. Breuer, A., Heinecke, A., Bader, M., Pelties, C.: Accelerating SeisSol by Generating Vectorized Code for Sparse Matrix Operators. In: International Conference on Parallel Computing (ParCo) 2013. Technische Universität München, Munich, Germany (Sep 2013)
2. Burstedde, C., Stadler, G., Alisic, L., Wilcox, L.C., Tan, E., Gurnis, M., Ghattas, O.: Large-scale adaptive mantle convection simulation. Geophysical Journal International 192(3), 889–906 (2013)
3. Carrington, L., Komatitsch, D., Laurenzano, M., Tikir, M.M., Michéa, D., Le Goff, N., Snavely, A., Tromp, J.: High-frequency simulations of global seismic wave propagation using SPECFEM3D_GLOBE on 62K processors. In: International Conference for High Performance Computing, Networking, Storage and Analysis, 2008. pp. 60:1–60:11. IEEE Press, Austin, Texas (2008)
4. Cui, Y., Poyraz, E., Olsen, K., Zhou, J., Withers, K., Callaghan, S., Larkin, J., Guest, C., Choi, D., Chourasia, A., Shi, Z., Day, S.M., Maechling, J.P., Jordan, T.H.: Physics-based Seismic Hazard Analysis on Petascale Heterogeneous Supercomputers. In: International Conference for High Performance Computing, Networking, Storage and Analysis, 2013. IEEE Press, Denver (2013)
5. Cui, Y., Olsen, K.B., Jordan, T.H., Lee, K., Zhou, J., Small, P., Roten, D., Ely, G., Panda, D.K., Chourasia, A., Levesque, J., Day, S.M., Maechling, P.: Scalable Earthquake Simulation on Petascale Supercomputers. In: International Conference for High Performance Computing, Networking, Storage and Analysis, 2010. pp. 1–20. IEEE Press, Washington, DC (2010)
6. Day, S.M., Bielak, J., Dreger, D., Graves, R., Larsen, S., Olsen, K., Pitarka, A.: Tests of 3D elastodynamic codes: Final report for Lifelines Project 1A02. Tech. rep., Pacific Earthquake Engineering Research Center (2003)
7. De La Puente, J., Käser, M., Dumbser, M., Igel, H.: An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes–IV. Anisotropy. Geophysical Journal International 169(3), 1210–1228 (2007)
8. Dumbser, M., Käser, M.: An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes – II. The three-dimensional isotropic case. Geophysical Journal International 167(1), 319–336 (2006)
9. Dumbser, M., Käser, M., Toro, E.F.: An arbitrary high-order Discontinuous Galerkin method for elastic waves on unstructured meshes–V. Local time stepping and p-adaptivity. Geophysical Journal International 171(2), 695–717 (2007)
10. Friedley, A., Bronevetsky, G., Lumsdaine, A., Hoefler, T.: Hybrid MPI: Efficient Message Passing for Multi-core Systems. In: International Conference for High Performance Computing, Networking, Storage and Analysis, 2013. IEEE Press, Denver (2013)
11. Gerstenecker, C., Läufer, G., Steineck, D., Tiede, C., Wrobel, B.: Digital Elevation Models for Merapi. In: Microgravity at Merapi Volcano: Results of the first two campaigns, 1st Merapi-Galeras-Workshop (1999), DGG Special Issue

12. Gerstenecker, C., Läufer, G., Steineck, D., Tiede, C., Wrobel, B.: Validation of Digital Elevation Models around Merapi Volcano, Java, Indonesia. Natural Hazards and Earth System Sciences 5, 863–876 (Nov 2005)
13. Gmeiner, B., Köstler, H., Stürmer, M., Rüde, U.: Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters. Concurrency and Computation: Practice and Experience 26(1), 217–240 (2014)
14. Goto, K., Van De Geijn, R.: High-performance implementation of the level-3 BLAS. ACM Transactions on Mathematical Software 35, 1–14 (2008)
15. Graves, R., Jordan, T.H., Callaghan, S., Deelman, E., Field, E., Juve, G., Kesselman, C., Maechling, P., Mehta, G., Milner, K., Okaya, D., Small, P., Vahi, K.: CyberShake: A Physics-Based Seismic Hazard Model for Southern California. Pure and Applied Geophysics 168, 367–381 (Mar 2011)
16. Guest, M.: The Scientific Case for HPC in Europe 2012 - 2020. Tech. rep., Partnership for Advanced Computing in Europe (PRACE) (2012)
17. Heinecke, A., Breuer, A., Rettenberger, S., Bader, M., Gabriel, A., Pelties, C.: Optimized Kernels for large scale earthquake simulations with SeisSol, an unstructured ADER-DG code. In: International Conference for High Performance Computing, Networking, Storage and Analysis, 2013. IEEE Press, Denver (2013), poster abstract.
18. Iglberger, K., Hager, G., Treibig, J., Rude, U.: High performance smart expression template math libraries. In: International Conference on High Performance Computing and Simulation, 2012. pp. 367–373 (2012)
19. Intel Cooperation: Intel Math Kernel Library (Intel MKL) 11.0. Tech. rep., Intel Cooperation (2013), http://software.intel.com/en-us/intel-mkl
20. Karniadakis, G.E., Sherwin, S.J.: Spectral/hp Element Methods for Computational Fluid Dynamics. Oxford University Press (2007)
21. Karypis, G., Kumar, V.: Parallel Multilevel series k-Way Partitioning Scheme for Irregular Graphs. SIAM Review 41(2), 278–300 (1999)
22. Karypis, G., Kumar, V.: A Coarse-Grain Parallel Formulation of Multilevel k-way Graph Partitioning Algorithm. In: SIAM Conference on Parallel Processing for Scientific Computing. SIAM (1997)
23. Käser, M., Mai, P., Dumbser, M.: On the Accurate Treatment of Finite Source Rupture Models Using ADER-DG on Tetrahedral Meshes. Bulletin of the Seismological Society of America 97(5), 1570–1586 (2007)
24. Käser, M., Pelties, C., Castro, C., Djikpesse, H., Prange, M.: Wave field modeling in exploration seismology using the discontinuous galerkin finite element method on hpc-infrastructure. The Leading Edge 29, 76–85 (2010)
25. Käser, M., Dumbser, M., De La Puente, J., Igel, H.: An arbitrary high-order Discontinuous Galerkin method for elastic waves on unstructured meshes–III. Viscoelastic attenuation. Geophysical Journal International 168(1), 224–242 (2007)
26. Kramer, W.T.: Top500 Versus Sustained Performance: The Top Problems with the Top500 List – and What to Do About Them. In: 21st International Conference on Parallel Architectures and Compilation Techniques. pp. 223–230. ACM, New York (2012)
27. Krzikalla, O., Feldhoff, K., Müller-Pfefferkorn, R., Nagel, W.E.: Scout: A Source-to-Source Transformator for SIMD-Optimizations. In: Euro-Par 2011: Parallel Processing Workshops, pp. 137–145. Springer Berlin Heidelberg (2012)
28. Lay, T., Aster, R., Forsyth, D., Romanowicz, B., Allen, R., Cormier, V., Wysession, M.E.: Seismological grand challenges in understanding Earth's dynamic systems. Report to the National Science Foundation, IRIS Consortium, 46 (2009)

29. Mellor-Crummey, J., Garvin, J.: Optimizing Sparse Matrix-Vector Product Computations Using Unroll and Jam. Int. J. High Perform. Comput. Appl. 18(2), 225–236 (May 2004)
30. Message Passing Interface Forum: MPI: A Message-Passing Interface Standard, Version 3.0. Specification, Message Passing Interface Forum (2012)
31. Meuer, H., Strohmaier, E., Dongarra, J., Simon, H.: Top500 list, November 2013 (2013), http://www.top500.org
32. Nishtala, R., Vuduc, R., Demmel, J., Yelick, K.: When cache blocking of sparse matrix vector multiply works and why. Applicable Algebra in Engineering, Communication and Computing 18(3), 297–311 (2007)
33. Oliker, L., Canning, A., Carter, J., Iancu, C., Lijewski, M., Kamil, S., Shalf, J., Shan, H., Strohmaier, E., Ethier, S., Goodale, T.: Scientific Application Performance on Candidate PetaScale Platforms. In: Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. pp. 1–12. IEEE International (2007)
34. Olsen, K.B., Day, S.M., Minster, J.B., Cui, Y., Chourasia, A., Faerman, M., Moore, R., Maechling, P., Jordan, T.: Strong shaking in Los Angeles expected from southern San Andreas earthquake. Geophysical Research Letters 33(7) (2006)
35. OpenMP Architecture Review Board: OpenMP Application Program Interface Version 4.0 (2013), http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf
36. Pelties, C., Gabriel, A.A., Ampuero, J.P.: Verification of an ADER-DG method for complex dynamic rupture problems. Geoscientific Model Development Discussion 6, 5981–6034 (2013)
37. Pelties, C., Huang, Y., Ampuero, J.P.: Pulse-like rupture induced by three-dimensional fault zone flower structures. Journal of Geophysical Research: Solid Earth, to be submitted (2014)
38. Pelties, C., Käser, M., Hermann, V., Castro, C.E.: Regular versus irregular meshing for complicated models and their effect on synthetic seismograms. Geophysical Journal International 183(2), 1031–1051 (2010)
39. Pelties, C., la Puente, J.D., Ampuero, J.P., Brietzke, G.B., Käser, M.: Three-Dimensional Dynamic Rupture Simulation with a High-order Discontinuous Galerkin Method on Unstructured Tetrahedral Meshes. Journal of Geophysical Research: Solid Earth 117 (2012)
40. Peter, D., Komatitsch, D., Luo, Y., Martin, R., Le Goff, N., Casarotti, E., Le Loher, P., Magnoni, F., Liu, Q., Blitz, C., Nissen-Meyer, T., Basini, P., Tromp, J.: Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes. Geophys. J. Int. 186(2), 721–739 (2011)
41. Pham, D.N., Igel, H., de la Puente, J., Käser, M., Schoenberg, M.A.: Rotational motions in homogeneous anisotropic elastic media. Geophysics 75(5), D47–D56 (2010)
42. Rabenseifner, R., Hager, G., Jost, G.: Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In: 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing. pp. 427–436. IEEE Press, Washington, DC (2009)
43. Rabenseifner, R., Wellein, G.: Comparison of Parallel Programming Models on Clusters of SMP Nodes. In: Modeling, Simulation and Optimization of Complex Processes, pp. 409–425. Springer Berlin Heidelberg (2005)
44. Rew, R., Davis, G.: NetCDF: an interface for scientific data access. Computer Graphics and Applications, IEEE 10(4), 76–82 (1990)
45. Southern California Earthquake Center: 2014 Science Collaboration Plan. Tech. rep. (2013)

Appeared in:
J.M. Kunkel, T. Ludwig, and H.W. Meuer (Eds.): ISC 2014, LNCS 8488, pp. 1–18, 2014.
http://link.springer.com/chapter/10.1007%2F978-3-319-07518-1_1

46. Vaidyanathan, K., Pamnany, K., Kalamkar, D.D., Heinecke, A., Smelyanskiy, M., Park, J., Kim, D., Shet G, A., Kaul, B., Jo'o, B., Dubey, P.: Improving Communication Performance and Scalability of Native Applications on Intel Xeon Phi Coprocessor Clusters. In: 28th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2014. Phoenix (2014), accepted for publication
47. Virieux, J., Calandra, H., Plessix, R.Ã.: A review of the spectral, pseudo-spectral, finite-difference and finite-element modelling techniques for geophysical imaging. Geophysical Prospecting 59(5), 794–813 (2011)
48. Vuduc, R., Demmel, J.W., Yelick, K.A.: OSKI: A library of automatically tuned sparse matrix kernels. In: Scientific Discovery through Advanced Computing, 2005. Journal of Physics: Conference Series, Institute of Physics Publishing, San Francisco (2005)
49. Vuduc, R.W., Moon, H.J.: Fast Sparse Matrix-vector Multiplication by Exploiting Variable Block Structure. In: First International Conference on High Performance Computing and Communications. pp. 807–816. HPCC'05, Springer-Verlag, Berlin, Heidelberg (2005)
50. Wenk, S., Pelties, C., Igel, H., Käser, M.: Regional wave propagation using the discontinuous Galerkin method. Journal of Geophysical Research: Solid Earth 4(1), 43–57 (2013)
51. Whaley, R.C., Dongarra, J.J.: Automatically Tuned Linear Algebra Software. In: Conference on Supercomputing, 1998. pp. 1–27. IEEE Press, Washington (1998)
52. Whaley, R.C., Petitet, A., Dongarra, J.J.: Automated empirical optimizations of software and the ATLAS project. Parallel Computing 27(1–2), 3–35 (2001)
53. Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., Demmel, J.: Optimization of Sparse Matrix-vector Multiplication on Emerging Multicore Platforms. In: International Conference for High Performance Computing, Networking, Storage and Analysis, 2007. pp. 38:1–38:12. New York (2007)
54. Van Zee, F.G., van de Geijn, R.A.: BLIS: A Framework for Rapidly Instantiating BLAS Functionality. ACM Transactions on Mathematical Software (2013), accepted pending minor modifications
55. Van Zee, F.G., Smith, T., Igual, F.D., Smelyanskiy, M., Zhang, X., Kistler, M., Austel, V., Gunnels, J., Low, T.M., Marker, B., Killough, L., van de Geijn, R.A.: The BLIS Framework: Experiments in Portability. ACM Transactions on Mathematical Software (2013), accepted pending modifications

Appeared in:
J.M. Kunkel, T. Ludwig, and H.W. Meuer (Eds.): ISC 2014, LNCS 8488, pp. 1–18, 2014.
http://link.springer.com/chapter/10.1007%2F978-3-319-07518-1_1