EDGE: Extreme Scale Fused Seismic Simulations with the Discontinuous Galerkin Method

Alexander Breuer[1], Alexander Heinecke[2], and Yifeng Cui[1]

[1] University of California, San Diego, 9500 Gilman Drive, La Jolla, 92093, CA, USA
[2] Intel Corporation, 2200 Mission College Blvd., Santa Clara 95054, CA, USA

Abstract. This article introduces EDGE, a solver package for fused seismic simulations. Fused seismic simulations are a novel technique addressing one of the grand challenges of computational seismology: large ensemble runs of geometrically similar forward simulations. Application fields include, but are not limited to: uncertainty quantification in the context of seismic hazard analysis or the accurate derivation of velocity models through tomographic inversion. For efficient and accurate handling of complex model geometries (topography, fault geometries, material heterogeneities), EDGE utilizes the Discontinuous Galerkin (DG) method for spatial and Arbitrary high order DERivatives (ADER) for time discretization, implemented for unstructured tetrahedral meshes. EDGE's ADER-DG scheme requires sparse and dense matrix-matrix multiplications at the kernel level. By choosing a sufficient memory layout and relying on runtime code generation and specialization, both, sparse and dense operations, can be efficiently vectorized on wide-SIMD machines. We present a convergence study of single and fused seismic simulations, code verification in an established benchmark, as well as a detailed performance assessment for different discretization orders. As target architecture we select the recently released Intel Xeon Phi processor, which powers the Theta and Cori-II supercomputers. For a single sixth order seismic forward simulation we achieved 10.4 PFLOPS of hardware performance and 5.0 PFLOPS for fused simulations in fourth order, both occupying 9,000 nodes of Cori-II. From a throughput perspective, fused seismic simulations can outperform a single forward simulation by 1.8 × to 4.6 ×, depending on the chosen order of the method.

1 Introduction

A popular approach for accurate numerical simulations of seismic wave propagation are Finite Difference Methods (FDM) [10, 8, 36, 28]. FDM approximate the partial derivatives through stencils, which combine adjacent grid points. While low dispersion errors can be reached through high-order stencils, accurate modeling of sharp material contrasts remains an ongoing challenge for FDM due to the underlying Cartesian meshes [29, 34, 3]. Further, the seismic wave field is often highly heterogeneous, resulting in inefficiencies for FDM since adaptive refinement in space and time is a highly non-trivial task, often limited to moderate patch-based adaptivity [19, 2, 32].

The final publication is available at Springer via http://doi.org/10.1007/978-3-319-58667-0_3.

Finite Element Methods (FEM) overcome many limitations intrinsic to FDM, if the mesh honors major material heterogeneities. Continuous Galerkin (CG-) FEM, often in combination with diagonal mass matrices, obtained through mass lumping or a special choice of quadrature and interpolation nodes, became a prominent option [16, 35, 26, 21]. Here, the widely used Spectral Element Methods (SEM) rely almost exclusively on hexahedral meshes and have been applied with great success on a global scale to forward runs and, more recently, to inverse problems [31, 21, 7]. However, on a local scale, the complexity of the resolved geometric features is limited by the difficult hexahedral meshing, leaving tetrahedral meshes as the only practical option [31, 33]. While the generalization of SEM to more flexible elements remains ongoing work, the CG scheme in [17] couples hexahedral and tetrahedral meshes, but is limited by low convergence rates. In contrast, Discontinuous Galerkin (DG-) FEM using tetrahedral meshes have reached a mature status in the last decade [9, 30, 11]. DG-FEM allow discontinuities in the numerical solution between elements and the corresponding discretized materials, which greatly simplifies the integration of sharp heterogeneities. Classical finite volume methods [5] are closely related to DG-FEM.

While the accurate numerical simulation of seismic wave propagation is already demanding, many of the grand challenges in computational seismology require large ensembles of geometrically similar forward simulations. In detail these ensembles cover few, but very complex model geometries with a broad range of variation influencing only the source descriptions. Important examples include uncertainty quantification in the context of seismic hazard analysis or the accurate derivation of velocity models through tomographic inversion. Interpretation of the similarities in the source descriptions as input parallelism offers large potential for reduced time-to-solution.

In this work we present EDGE, a new software package addressing some of the hardest challenges in computational seismology. EDGE's forward solver for seismic wave propagation relies on the flexibility of the ADER-DG scheme [9, 20]. Our software supports different element types and hyperbolic partial differential equations. However, in this work, we will focus on unstructured meshes with 4-node tetrahedral elements and the elastic wave equations. EDGE enables ensemble-based, high-dimensional studies with an unprecedented complexity by *fusing* multiple forward simulations into one execution of the solver. Therefore this paper makes following novel contributions: 1) EDGE as an open source solver package (BSD-3), which was created from scratch to support fused simulations for maximum throughput, and 2) a runtime code generation approach for highest performing kernels when running fused simulations on wide-SIMD architectures.

2 Discretization

For an isotropic medium the 3-D elastic wave equations in velocity-stress formulation are given by a system of hyperbolic partial differential equations:

$$q_t + A_1 q_{x_1} + A_2 q_{x_2} + A_3 q_{x_3} = 0. \tag{1}$$

Time is given by $t \in \mathbb{R}^+$ and location in space by $\boldsymbol{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3$. $q(\boldsymbol{x}, t) = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{13}, u, v, w)^T \in \mathbb{R}^9$ is the vector of quantities. Here, σ_{11}, σ_{22}, and σ_{33} are the normal stress components in x_1-, x_2-, and x_3-direction. The shear stresses are given respectively by σ_{12}, σ_{13}, and σ_{23}. $A_1(\boldsymbol{x}), A_2(\boldsymbol{x}), A_3(\boldsymbol{x}) \in \mathbb{R}^{9\times 9}$ are the three space-dependent Jacobians. The Jacobians characterize the wave propagation in our hyperbolic system and are derived from the material parameters, given by the mass density $\rho(\boldsymbol{x})$, and Lamé constants $\lambda(\boldsymbol{x})$ and $\mu(\boldsymbol{x})$. By applying the DG-machinery in space and the explicit ADER-scheme in time, we obtain the fully discrete form of Eq. 1 as a series of integration kernels. These kernels describe time-, volume-, and surface-integration and might be formulated as series of small matrix-matrix products.

Our fully discrete formulation divides the computational domain Ω into K pair-wise disjunct tetrahedral elements T_k: $\Omega = \bigcup_{k=1}^{K} T_k$. The numerical solution in every element k is given by a set of $9 \times \mathcal{B}(\mathcal{O})$ time-dependent Degrees Of Freedom (DOFs) $Q_k(t) \in \mathbb{R}^{9\times\mathcal{B}}$. $\mathcal{O}$ is the order of our ADER-DG discretization with $\mathcal{O} = \mathcal{P}-1$, where $\mathcal{P}$ is the degree of our orthogonal, hierarchical, polynomial basis. We use the same order in time and space, which can be arbitrarily high. Further, we assume piecewise constant material parameters in every element T_k, leading to per-element, constant Jacobians.

Time Kernel: Our first kernel uses the Cauchy-Kovalewski procedure to integrate the element-local DOFs Q_k for a full time step $t^n \rightarrow t^{n+1} = t^n + \Delta t$ in time:

$$\mathcal{I}_k^n = \mathcal{I}(Q_k^n) = \sum_{d=0}^{\mathcal{O}-1} \frac{\Delta t^{d+1}}{(d+1)!} \cdot \frac{\partial^d}{\partial t^d} Q_k, \tag{2}$$

where the time derivatives, with the DOFs Q_k^n at time step t^n as initial conditions, $\partial^0/\partial t^0 Q_k = Q_k^n = Q_k(t^n)$, are obtained recursively through:

$$\frac{\partial^{d+1}}{\partial t^{d+1}} Q_k = -\sum_{c=0}^{3} A_{k,c}^* \left(\frac{\partial^d}{\partial t^d} Q_k \right) (K_{\xi_c})^T. \tag{3}$$

Here, matrices $A_{k,c}^* \in \mathbb{R}^{9\times 9}$ are linear combinations of the element-local Jacobians, and matrices $K_{\xi_c} \in \mathbb{R}^{\mathcal{B}\times\mathcal{B}}$ the three stiffness matrices, formulated in terms of the unique reference tetrahedron T_{ref} and multiplied with the diagonal, inverse mass matrix in initialization.

Volume Kernel: The volume kernel computes the volume integration based on the element's time integrated DOFs:

$$\mathcal{V}_k^n = \mathcal{V}(\mathcal{I}_k^n) = \sum_{c=1}^{3} A_{k,c}^* \left(\mathcal{I}_k^n\right) K_{\xi_c}. \tag{4}$$

Surface Kernel: Our last kernel computes the net-updates of the surface integration based on the element's time-integrated DOFs $\mathcal{I}_k^n$ and those of the four face-adjacent elements $\mathcal{I}_{k_i}^n$:

$$\mathcal{S}_k^n = \mathcal{S}(\mathcal{I}_k^n, \mathcal{I}_{k_1}^n, \ldots, \mathcal{I}_{k_4}^n) = \sum_{i=1}^{4} A_{k,i}^- \left(\mathcal{I}_k^n\right) F^{-,i} + \sum_{i=1}^{4} A_{k,i}^+ \left(\mathcal{I}_k^n\right) F^{+,i,j_k,h_k} \tag{5}$$

$A^-_{k,i} \in \mathbb{R}^{9\times 9}$ and $A^+_{k,i} \in \mathbb{R}^{9\times 9}$ are the flux solvers, computing the numerical fluxes. Matrices $F^{-,i} \in \mathbb{R}^{\mathcal{B}\times\mathcal{B}}$ and $F^{-,i,j_k,h_k} \in \mathbb{R}^{\mathcal{B}\times\mathcal{B}}$ are the flux matrices. Index i is the local face of element k w.r.t. the reference element. Indices $j_k(i) \in \{1,2,3,4\}$ and $h_k(i) \in \{1,2,3\}$ depend on the vertices both adjacent elements k and k_i share with respect to their transformation to the reference element [9].
Time Step: Our ADER-DG scheme splits a time step $t^n \to t^{n+1}$ into two steps. First, we compute all element-local operations, not requiring any data from adjacent elements. This is the time kernel and the first update step consisting of the volume kernel $\mathcal{V}^n_k$, and the local part of the surface kernel $\mathcal{S}^n_k$:

$$\bar{Q}^{n+1}_k = Q^n_k + \mathcal{V}^n_k + \sum_{i=1}^{4} A^-_{k,i}\left(\mathcal{I}^n_k\right) F^{-,i} \tag{6}$$

Here, we use the recently computed time integrated DOFs $\mathcal{I}^n_k$ directly and store them for later use in our second step. The second step contains the remainder of the surface kernel, and thus updates the elements' DOFs with data of face-adjacent tetrahedrons:

$$Q^{n+1}_k = \bar{Q}^{n+1}_k + \sum_{i=1}^{4} A^+_{k,i}\left(\mathcal{I}^n_k\right) F^{+,i,j_k,h_k}. \tag{7}$$

3 Fused Simulations

A non-fused setup defines fixed input i, and runs the forward solver s to obtain observations $o = s(i)$. Now, if we are interested in results for n different inputs, e.g., different seismic sources, we would specify a set of inputs $I_n = (i_1, i_2, \ldots, i_n)$ and run the non-fused forward solver s on all these inputs to obtain the set of observations $O_n = (o_1, o_2, \ldots, o_n) = (s(i_1), s(i_2), \ldots, s(i_n))$. Typically, the n executions of the solver are completely decoupled, which means that potential parallelism and shared data between two instances $s(i_k)$ and $s(i_l)$ is not utilized.

Fused simulations in EDGE exploit this potential by integrating the concept of multiple but similar input parameters into the forward solver. Thus, we introduce a new forward solver S_m which is capable of handling a set of $m \leq n$ inputs $I_m = (i_1, i_2, \ldots, i_m)$ in a single execution: $O_m = (o_1, o_2, \ldots, o_m) = S_m(I_m)$. We achieve this by a fundamental paradigm in EDGE's data layout, which sets the m forward runs as the fastest dimension in all respective data structures. For example the two most important data structures in our ADER-DG solver for seismic simulations (see Sec. 2) are the DOFs Q^n_k and the time integrated DOFs $\mathcal{I}^n_k$. Here, we use the K elements as slowest dimension, followed by the 9 quantities, the $\mathcal{B}$ modes and finally the m simulations as fastest dimension. Each element is therefore represented by a 3D-tensor.

Note, that one might interpret the different input parameters as multiple right-hand sides of the PDEs, which would lead to the term *parallelization over multiple right-hand sides* in literature [4, 27]. However, in this work we prefer the more general term *fused simulation* due the diverse advantages of the approach,

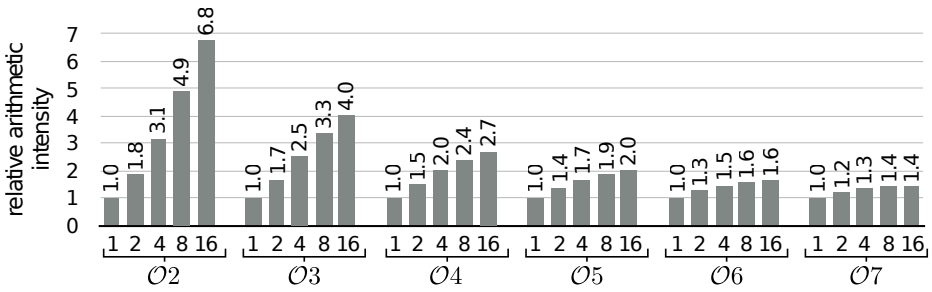

Fig. 1. Arithmetic intensity if the material parameters and mesh are shared in an elastic ADER-DG setup. Shown is the relative improvement over a non-fused simulation in dependency of the order ($\mathcal{O}2$-$\mathcal{O}7$) and the number of fused runs (1, 2, 4, 8, 16).

and settings where interpretation as a right-hand side is more complex, e.g., in multi-physics setups. We identify four key advantages of EDGE's fused approach over non-fused simulations:

1. By fusing multiples of the vector-width, we are able to perform full vector operations, even when using sparse matrix-operations, whereas non-fused settings require dense matrix operations (which have up-to a 50% zero padding overhead) for best performance [14].
2. Data structures are automatically aligned by fusing multiples of the cache line size. Zero-padding [14] for fast aligned loads and stores is not needed.
3. Read-only data structures might be shared among all runs. As illustrated in Fig. 1 for our seismic setup of Sec. 2, this results in substantially increased arithmetic intensities. For example, a non-fused fourth order accurate simulation theoretically requires 8,640 bytes per element. 67% of this requirement is read-only data. By fusing eight runs, we only need 28,800 bytes, which reduces this ratio to 20% and therefore increases the arithmetic intensity by 2.4 $\times$. Analogue, for a sixth order configuration, the memory footprint per element only increases from 13,824 to 70,272 bytes, which corresponds to a 1.6 $\times$ higher arithmetic intensity.
4. Fused simulations are less sensitive to memory latencies and conditional jumps, due to less frequent context switches. Here, the increased, fused workload per memory operation or conditional jump effectively reduces the performance penalty of start-up latencies or branch mispredictions. Analogue, they are less sensitive to network latencies due to larger MPI-messages having identical exchange-frequencies. See [1] for details on memory latencies of Knights Landing and [25] for the relation of message sizes and bandwidth.

However, there are also requirements and limitations. I_m, the set of m inputs has to be "similar enough" for exploitation in fused forward simulations. If a parameter space beyond the following fusing-limitations is studied, we simply distribute the $n \geq m$ inputs to respective fused and non-fused runs. Considering our seismic use case (see Sec. 2), we formulate the following requirements for EDGE on a set of input parameters to be fused into a single forward run:

1. The mesh needs to be identical for all m simulations. This ensures identical adjacency information and identical element sizes, used in our explicit solver's stability requirements.
2. Start- and end-time of all simulations are identical. Further, all simulations have the same order of convergence and share the same characteristics of wave field output (frequency) and seismic receivers (frequency and location).
3. All fused simulations share the same element-local material parameters. Thus, we obtain identical update patterns, since the resulting wave speeds, in combination with the shared mesh, determine the element-local time step.
4. All simulations are allowed to have arbitrary initial DOFs. The location of seismic sources is shared among all m simulations, but moment-rate time histories are private and thus arbitrary. A similar approach would apply if EDGE is extended with internal dynamic rupture boundary conditions [14] in future work.

Nevertheless, all of these limitation and requirements are fulfilled by ensemble simulations and therefore fused simulation are the perfect tool to increase hardware efficiency and simulation throughput.

4 EDGE in a Nutshell

4.1 Runtime Code Generation of DG-FEM Kernels

Sec. 2 shows that that the speed of EDGE's integrators heavily depends on the performance of small dense or sparse matrix-matrix operators. In the case of fused simulations, a sparse matrix needs to be multiplied with a 3D-tensor, which represents the DOFs Q_k^n, time derivatives $\partial^d/\partial t^d Q_k^n$, or time integrated DOFs $\mathcal{I}_k^n$ for a given mesh element. Previous work showed, that code generation is the ideal tool to speed up single forward runs and yields extremely high hardware efficiencies (greater than 50%) in a portable manner [14]. All previously discussed tricks (c.f., [6]) for an efficient implementation of the ADER-DG scheme for a single simulation have been enabled in EDGE and needed kernels are runtime code generated in EDGE's setup phase leveraging LIBXSMM [15]. As these techniques are covered in the literature, we are not recapping them here.

Instead, we focus on runtime code generation for the required 3D-tensor manipulations of fused simulations. Specifically, this requires two operations to be optimized:

- *K1*: sparse-matrix $\times$ 3D-tensor = 3D-tensor, this operation is needed for multiplication with Jacobians and flux-solvers. In BLAS-notation, the sparse matrix A is a 9×9 matrix, whereas B and C are dense 3D-tensors. Matrix A is applied to all planes enumerated by the inner-most dimension f of this tensor, which corresponds to the number of fused forward runs.
- *K2*: 3D-tensor $\times$ sparse-matrix = 3D-tensor, this operation is needed for multiplication with stiffness or flux matrices. The dimensions of the sparse matrix B depend on the order and which stage of the integration kernels is

Algorithm 1 Code generator sketch of kernel *K1*

```
1: for all m = 1 to #quantities do
2:    a_#Entries ← row_A[m + 1] − row_A[m]
3:    for k = 1 to a_#Entries do
4:       a ← A[row_A[m] + k]
5:       for all n = 1 to #modes do
6:          C[m][n][1 : f] ← fma(bcst(a), B[col_A[row_A[m] + k]][n][1 : f], C[m][n][1 : f])
7:       end for
8:    end for
9: end for
```

performed. Again, this matrix has to be applied to all forward simulations, which are stored in the inner-most dimension f of the 3D-tensor.

In this work, we focus on a length of f that matches the SIMD-length of the underlying architecture. As we target Intel's Xeon Phi processor, code-named Knights Landing, we leverage AVX-512, offering a double precision vector length of 8 entries. Thus, the number of fused simulations in this work is $f = 8$. We are using slightly different specifiers as in Sec. 2 to allow for a BLAS-related naming.

Under these assumptions, the code generator of *K1* can be realized straightforward and is sketched in Alg. 1. We store the entries of all sparse matrices in Compressed Sparse Row Format (CSR). However, the row pointer (row_A in Alg. 1) and column indices (col_A in Alg. 1) are only used for the runtime code genaration at EDGE's initialization. Thus, the loops hardwire the sparsity pattern of matrix A by fully unrolling *K1*'s implementation and therefore eliminating any access to row or column index structures. As A is sparse, we have unstructured accesses to full vectors over the fused quantities in input tensor B, c.f., line 6. Since the scalar entry of A can be reused across all fused forward simulations, we broadcast it and we can maintain a contiguous access pattern to the result tensor C. During the surface computation, matrix A, the flux solver, is a dense matrix. In this case we disable unrolling over the number of quantities to reduce code size. Additionally, for code used in the neighboring update (7), our code generator supports insertion of last-level cache software prefetching instructions. These help to accelerate EDGE by roughly 10% as the negative impact of accessing face-adjacent elements in the unstructured mesh can be mitigated.

Generating an efficient implementation of *K2* is more challenging and we cover the details by a step-by-step explanation of Alg. 2. From a high-level point of view, we follow the same approach as in Alg. 1. However, since now the right hand side operator, matrix B, is sparse we end up with unstructured accesses to the result tensor C, which depend on B's sparsity pattern. From a performance perspective we cannot afford frequent read and write access to C, as we already consume all L1 cache bandwidth for reading the input tensor A and matrix B. We therefore create an in-register scratchpad for a C accumulator set, indexed by the quantities, c.f., line 2 for loading and line 10 for storing this scratchpad. It contains all modes for all forward simulations for a given quantity. This now

Algorithm 2 Code generator sketch of kernel *K2*

```
1: for all m = 1 to #quantities do
2:    for all  n = 1 to #modes do c_n[1 : f] ← C[m][n][1 : f] end for
3:    for all k = 1 to #modes do
4:       b_#Entries ← row_B[k + 1] − row_B[k]
5:       for n = 1 to b_#Entries do
6:          j ← col_B[row_B[k] + n]
7:          c_j[1 : f] ← fma(A[m][k][1 : f], bcst(B[row_B[k] + n]), c_j[1 : f])
8:       end for
9:    end for
10:   for all  n = 1 to #modes do  C[m][n][1 : f] ← c_n[1 : f] end for
11: end for
```

allows us to implement unstructured access to C, as we only need to pick the corresponding register in the dot product calculation, c.f., lines 6 and 7.

In summary, both kernels *K1* and *K2* are able to achieve 25-40% of hardware efficiency on a single core of the Intel Xeon Phi processor using AVX512. However, they have a higher L2-cache pressure than dense kernels and therefore are limited by the shared L2 cache interface of two cores in Xeon Phi's computing tile for two reasons: a) latencies due to unstructured tensor entry accesses b) L2 cache bandwidth is shared. Therefore at full chip level we can expect a kernel compute efficiency of roughly 20%.

On an AVX512-capable processor, we can generate kernels for up to 31 modes efficiently without additionally blocking as the architecture offers 32 vector registers. In this work we limit ourself to a maximum of fourth order runs which have $\mathcal{B}(4) = 20$ modes. Having Fig. 1 in mind, this limitation is only minor as the expected runtime benefit decreases for higher orders. Nevertheless, an additional blocking is planned as future work. Such a feature will also allow to use older vector instruction sets such as AVX2 which offer a small register file with only 16 entries.

4.2 Parallelization and Data Layout

Our parallelization strategy strictly separates between shared and distributed memory parallelization. For the latter one we use the Message Passing Interface (MPI) and assign one rank to every of the P available nodes, sharing a memory space. Therefore, we require exactly P partitions of our unstructured mesh for utilization of P nodes. This reduces the pressure on the partitioner, e.g., the Metis-library [18], and reduces relative communication costs defined as the volume-to-surface ratio of the partitions.

In addition to using fused simulations as fastest dimension of the data layout, EDGE also follows the distributed memory parallelization for the sorting of entity-data in memory. Focusing on a single partition $p \in P$, we store inner-entities first, send-entities second and recv-entities last. Here, we follow the naming scheme of corresponding MPI-functions: Values of inner-entities are not

communicated, values of send-entities are send to other ranks and values of recv-entities received from other ranks. In terms of our ADER-DG solver for seismic wave propagation in Sec. 2, our MPI-partitions only exchange time integrated DOFs $\mathcal{I}_k^n$, required in the second update step (7). Here, our inner-elements are owned by partition p and are, within a time step, independent of element-data owned by other partitions. Send-elements are owned by p, but their associated $\mathcal{I}_k^n$ are required for application of Eq. (7) to send-elements of other partitions. Similar recv-elements are owned by an adjacent partition and the respective $\mathcal{I}_k^n$ are required for updating the DOFs of p's send-elements in Eq. (7). We further sort the send- and recv-elements by their corresponding neighboring rank. If one of these elements is connected to more than one MPI-rank through its faces, we logically duplicate the element in our data layout. Within the inner-elements and the per-rank groups of the send- and receive-elements, we sort the elements by an unique but arbitrary identifier. Therefore, we are able to directly use our data layout for sending and receiving MPI-messages without the need for artificial communication buffers.

Our shared memory parallelization uses the OpenMP library. Compared to other work [14], we only use minimal functionality of OpenMP in the time marching loop. After synchronization, e.g., after initialization or wave field output, we open a single parallel-region until we reach the next synchronization point. Out of a total of T threads, we use the first $1 \leq W < T$ threads as workers and the $W + 1$'th thread as management and communication thread. The workers perform the numerical operations described in Sec. 2. Here, the distribution of work, e.g., *"compute Eq. (6) for all send-elements"* to workers is performed statically at initialization. This approach is similar, to traditional, static OpenMP annotation of for-loops, but allows for fine-grained load balancing and removes unnecessary, implicit barriers. For example, a thread might directly continue with Eq. (6) for inner-elements, after finishing its part of the send-elements. The $W+1$'th thread initiates communication through `MPI_Isend` and `MPI_Irecv`, progresses communication through `MPI_test`, and ensures correctness by resolving dependencies and signaling the workers where to head next.

Considering different layers of memory, such as High Bandwidth Memory (HBM) and traditional DDR4 RAM in case of the Intel Xeon Phi x200 processor, we follow the general strategy of [13]. Here, we distribute data to the different layers, if our simulation size exceeds the size of near-memory and if the memory layers are available at application level, e.g., in flat-quadrant mode. In our seismic setup (see Sec. 2), we place the time-integrated DOFs $\mathcal{I}_k^n$, having high access frequencies and unstructured accesses, in near-memory. Further, EDGE provides high-bandwidth scratch memory for temporary storage of intermediate results, to avoid performance penalties of large stack-based memory chunks.

5 Experiments and Results

For the purpose of this work, we solely relied on double precision arithmetic for every of EDGE's floating point operations and used following machines:

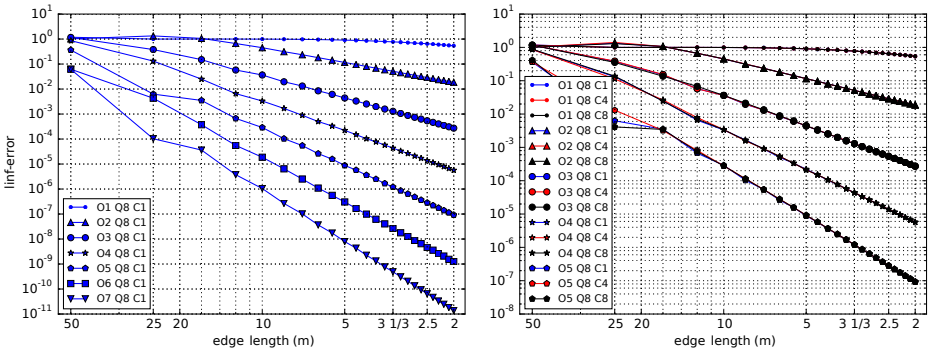

Fig. 2. Convergence of EDGE in the L^∞-norm. Shown are orders $\mathcal{O}1-\mathcal{O}7$ for the eighth quantity v (Q8) in non-fused runs on the left. The right plot shows orders $\mathcal{O}1-\mathcal{O}5$ for v (Q8) when utilizing EDGE's fusion capabilities with shifted initial conditions. For clarity, from the total of eight fused simulations, only errors of the first (C1), fourth (C4) and last simulation (C8) are shown.

- **Theta** is a Cray XC40 that comprises 3,200 Intel Xeon Phi 7230 64-core processors at 1.3 GHz (with Intel Turbo Boost enabled), 16 GB of in-package HBM and 192 GB of DDR4 RAM. Here, we used the performance-related modules `intel/17.0.1.132`, `craype/2.5.8`, `PrgEnv-intel/6.0.3`, `cray-mpich/7.5.0`, `cray-memkind`, `craype-mic-knl`, and the performance-related compile-flags `-O2`, `-xMIC-AVX512`, and `-qopenmp` for our scaling studies.
- **Cori-II** is a Cray XC40 that combines 9,304 Intel Xeon Phi 7250 68-core processors at 1.4 GHz (with Intel Turbo Boost enabled), 16 GB of in-package HBM and 96 GB of DDR4 RAM. Except for `craype/2.5.7` and `cray-mpich/7.4.4`, we used the same performance-related modules and flags, as on Theta, on Cori-II.

5.1 Benchmarks

Convergence Analysis Our first benchmark explores EDGE's high order convergence. Similar to [9], we use a cubic domain of size $[0,\ 100]^3$ and generate 24 setups by dividing the domain regularly into cubes with descending edge lengths: $\frac{100}{2} = 50, \frac{100}{4} = 25, \ldots, \frac{100}{50} = 2$. Within every setup, each of the cubes is then subdivided into five tetrahedral elements. Material parameters are $\rho = 1$, $\lambda = 2$, $\mu = 1$, while the initial DOFs discretize plane waves traveling in diagonal direction with a P-wave velocity of 2 and a S-wave velocity of 1. Additionally we use periodic boundary conditions, such that the solution of the setup can be derived analytically after a given time [9]. We simulate for a total time of $\sqrt{3} \cdot 100$. Therefore, the resulting exact solution is identical to our initial setup. Fig. 2 shows two convergence plots derived from our setups. The plot on the left shows convergence when executing EDGE in non-fused mode. The plot on the right presents convergence when running EDGE with $m = 8$

fused simulations. Here, we shifted the initial setup of the respective simulations by $(0,0,0)^T, (5,5,5)^T, \ldots, (35,35,35)^T$ to obtain true input-parallelism. We see that EDGE obtains the convergence rates reported in literature when ADER-DG is applied to seismic wave propagation [9]. The different fused simulations show almost identical convergence behavior.

Layer Over Halfspace Benchmark 1 (LOH.1) Our second configuration is the LOH.1 benchmark, which is part of *The Spice Code Validation* [22]. We used a domain covering $[-26\,\text{km}, 32\,\text{km}] \times [-26\,\text{km} \times 32\,\text{km}] \times [0\,\text{km}, 33\,\text{km}]$. All boundary-conditions are outflow, except for $z = 0$, where free-surface boundary conditions are set. The one seismic source of the benchmark is a point dislocation at $(0, 0, 693\,\text{m})$ with $M_{xy} = M_{yz} = M_0 = 10^{18}\,\text{Nm}$ being the only non-zero entries in the moment tensor. The moment time history is given by $M_0(1-(1+\frac{t}{T})exp(-\frac{t}{T}))$ with $T = 0.1\,\text{s}$. The LOH.1 benchmark compares a total of nine receivers at the surface. The material parameters are $\rho = 2600\,\frac{\text{kg}}{\text{m}^3}$, $\lambda = 20.8\,\text{GPa}$, and $\mu = 10.4\,\text{GPa}$ up to a depth of 1 km. In the remainder of the domain the parameters are given by $\rho = 2700\,\frac{\text{kg}}{\text{m}^3}$, $\lambda = 32.4\,\text{GPa}$, and $\mu = 32.4\,\text{GPa}$.

We used the software gmsh [12] to generate a problem-adapted tetrahedral mesh. The material interface was integrated into the surface mesh, which resulted in interface-aligned faces of our tetrahedral elements. We specified a characteristic length of 100 m in $[-5\,\text{km}, 13.67\,\text{km}] \times [-5\,\text{km} \times 15.392\,\text{km}] \times [0\,\text{km}, 1\,\text{km}]$, 257 m in $[-5\,\text{km}, 13.67\,\text{km}] \times [-5\,\text{km} \times 15.392\,\text{km}] \times [1\,\text{km}, 7\,\text{km}]$, and 771 m everywhere else. To ensure smooth mesh coarsening in the 1 km thick layer, we additionally defined an attractor and used the overall minimum characteristic length for meshing. Further, we used gmsh's built-in optimizer and Netgen-interface to improve mesh quality. The final mesh consisted of 11,060,982 tetrahedral elements. We used used fourth order in space and time and 256 nodes of Cori-II to simulate the 9 s of the benchmark. To ensure correctness of EDGE's full capabilities, we fused eight simulations. However, we simply used identical input for all fused simulations and therefore obtained eight identical solutions.

Fig. 3 examplary compares EDGE's obtained particle velocity in x-direction u to the reference solution. We see that the solutions match very well, which is confirmed by Tab. 1, showing the single-valued envelope misfit EM and single-valued phase misfit PM [24, 23] for all nine seismic receivers and three particle velocities u, v, and w. Here, the misfits stay well below the treshold of 5%, referring to the highest accuracy level of the benchmark.

5.2 Single Node Performance

In this section we discuss EDGE's single node performance when running the LOH.1 benchmark (see Sec. 5.1), discretized with a total of 350,264 tetrahedral elements. Additionally, as in all following performance studies, we greatly limited the number of time steps to avoid unnecessary computations. All runs in this section were carried out on a single node of Cori-II in flat-quadrant mode and with all memory allocated in HBM through numactl's membind-feature. We used a setting identical to our per-node layout in distributed memory runs by utilizing

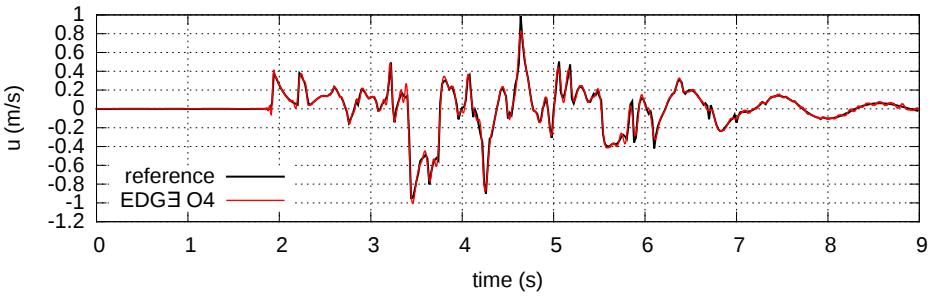

Fig. 3. Synthetic seismogram of EDGE for quantity u at the nith seismic receiver located at $(8647\,\mathrm{m}, 5764\,\mathrm{m}, 0)$ in red. The reference solution is shown in black.

	location (m)			u (%)		v (%)		w (%)		max (%)	
	x	y	z	EM	PM	EM	PM	EM	PM	EM	PM
1	0	693	0	0.75	0.29					0.75	0.29
2	0	5543	0	1.20	0.16					1.20	0.16
3	0	10392	0	1.17	0.17					1.17	0.17
4	490	490	0	0.80	0.31	0.74	0.34	1.05	0.23	1.05	0.34
5	3919	3919	0	1.06	0.15	1.10	0.15	0.97	0.19	1.10	0.19
6	7348	7348	0	1.12	0.17	1.13	0.17	0.96	0.19	1.13	0.19
7	577	384	0	0.84	0.32	0.73	0.33	1.09	0.23	1.09	0.33
8	4612	3075	0	0.94	0.15	1.37	0.17	0.98	0.19	1.37	0.19
9	8647	5764	0	1.01	0.18	1.33	0.18	0.96	0.19	1.33	0.19

Table 1. Single-valued envelope misfit EM and single-valued phase misfit PM in percent for the nine receivers in the LOH.1 benchmark. The misfits are given for non-zero seismograms of the reference solution in a frequency range between 0.13 Hz and 5 Hz.

only 66 cores for computations in EDGE. The first of the two remaining cores was left empty for the OS, the other core hosted the communication and management thread. We compare EDGE's performance to the software package SeisSol in the version *201511* [13] using global time stepping and support for AVX512. Here, we left the first tile idle and pinned the communication thread to the last core, as required by SeisSol for highest performance. Fig. 4 compares the required time-to-solution of both codes for 500 time steps. First, we ran traditional, non-fused simulations with both codes for orders $\mathcal{O} = 2, \ldots, 6$, abbreviated with O2C1,...,O6C1. Additionally, Fig. 4 shows EDGE's relative performance when fusing eight forward simulations for orders $\mathcal{O}2$, $\mathcal{O}3$, and $\mathcal{O}4$, abbreviated with O2C8, O3C8 and O4C8. We see that EDGE, despite targeting at fused simulations, is able to maintain a high fraction of SeisSol's performance when running single, non-fused forward simulations. In the case of O2C1, EDGE even outperforms SeisSol since SeisSol's zero-padding introduces a significant overhead overturning improvements of alignment to cache lines. However, for orders higher than $\mathcal{O}2$ these optimizations pay off, leading to a higher performance of SeisSol. For the sixth order configuration O6C1, EDGE reaches 96% of SeisSol's perfor-

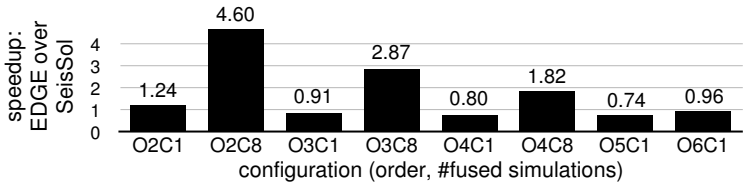

Fig. 4. Speedup of EDGE over SeisSol. For convergence rates $\mathcal{O}2 - \mathcal{O}6$ results for a single non-fused forward simulations (O2C1-O6C1) are presented. Additionally, respective per-simulation speedups for orders $\mathcal{O}2 - \mathcal{O}4$ are presented when using EDGE's full capabilities by fusing eight simulations (O2C8-O4C8).

mance. The reason for this relatively higher performance, compared to O4C1 and O5C1, is given by the $\mathcal{B}(6) = 56$ basis function of this setting. $\mathcal{B}(6)$ is a multiple of 8 and naturally leads to 64-byte aligned DOFs Q_k^n and time integrated DOFs $\mathcal{I}_k^n$ in EDGE since the base pointers of all our heap data structures are aligned 4,096 byte boundaries.

Comparing EDGE's performance on a simulation-by-simulation basis to SeisSol, when running eight fused simulations, we observe a factor 1.8-4.6 improvement in time-to-solution. This result confirms our theoretical discussion in Sec. 3, where we identified higher arithmetic intensities and increased regularity of fused simulations as key advantages. As shown in Fig. 1, the potential speedup offered by the higher arithmetic intensities is largest in the memory-bound, low order regime. Moving to the compute-bound high order simulations, the increased regularity becomes more important, leading to a substantial, but relatively smaller, $1.8\times$ speedup over SeisSol for O4C8.

5.3 Weak Scaling

The setup of our weak scaling study follows the convergence analysis in Sec. 5.1. However, to further avoid unnecessary computations, we replaced the initial value computation of the DOFs, requiring an L2-projection, with zero values and disabled the error-norm computation. Instead, we added a total of 8 seismic sources, where only one of the sources was active in a single forward simulation. Further, in comparison to other work [14], we left the more demanding periodic boundary conditions intact, which is supported by EDGE for regular meshes and enables convergence studies in distributed memory setups. We used a total of 276,480 tetrahedral elements per node and studied the performance of fourth and sixth order convergence. In the case of the sixth order runs we present results for a single, non-fused simulation (O6C1). For the fourth order runs we present performance for a single forward simulation (O4C1) and eight fused simulations (O4C8). Considering the memory consumption of the heavy data structures touched in the time marching loop – $Q_k^n, \mathcal{I}_k^n, A_{k,c}^*, A_{k,i}^{\pm}$ in Eq. (3), Eq. (4), and Eq. (5) – our weak scaling setup only has a moderate size, underlining the relevance of this scaling study. O4C1's matrices consume 2.2 GiB per node, O4C8's matrices 7.4 GiB and O6C1's matrices 3.6 GiB per node

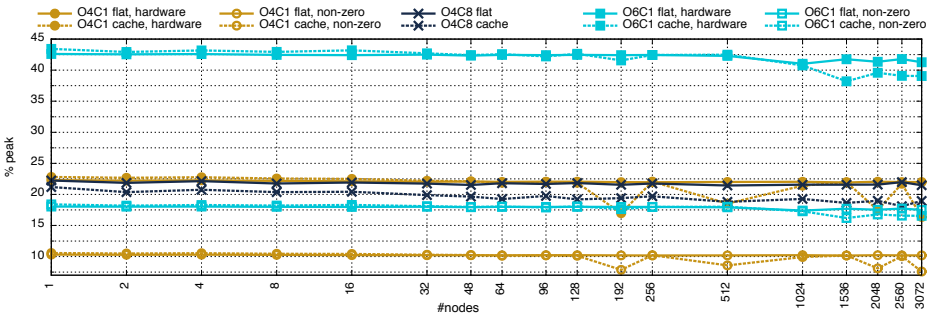

Fig. 5. Weak scaling study on Theta. Shown are hardware and non-zero hardware peak efficiencies of all configurations in cache and flat mode. O denotes the order and C the number of fused simulations.

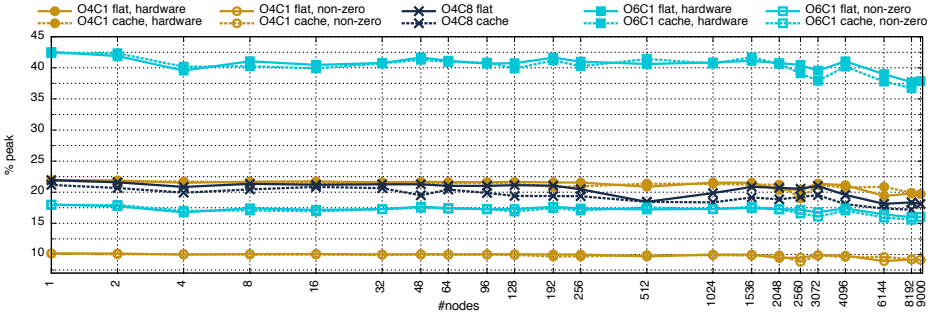

Fig. 6. Weak scaling study on Cori-II. Shown are hardware and non-zero hardware peak efficiencies of all configurations in cache and flat mode. O denotes the order and C the number of fused simulations.

Fig. 5 shows the hardware and non-zero peak efficiencies of our weak scaling on 1 to 3,072 nodes of Theta. Here, the hardware peak efficiency counts every of the double-precision floating point operation performed in hardware, while the non-zero peak efficiency only considers those of non-zero entries in our kernel's matrices (see Sec. 2). We see that EDGE obtains more than 38% of hardware peak efficiency in cache mode and more than 41% in flat mode for all O6C1 runs on Theta. The highest sustained hardware performance on Theta was obtained in flat mode and is 3.4 PFLOPS, which corresponds to a non-zero performance of 1.4 PFLOPS and a parallel efficiency of 97 %. Moving to the fourth order configurations O4C1 and O4C8, EDGE is able to maintain the single node speedup (see Fig. 4) offered by its fusion capabilities at scale. In fact O4C8 outperforms O4C1 in per-simulation time-to-solution by 2.1 × when running in flat mode at scale. Due to O4C8's sparse matrix-operators, this corresponds to a hardware and nonzero peak efficiency of 21.5 % on 3,072 nodes, which is equivalent to a sustained performance of 1.8 PFLOPS and a parallel efficiency over 96 %.

Fig. 6 shows our weak scaling study on 1 to 9,000 nodes of Cori-II. The

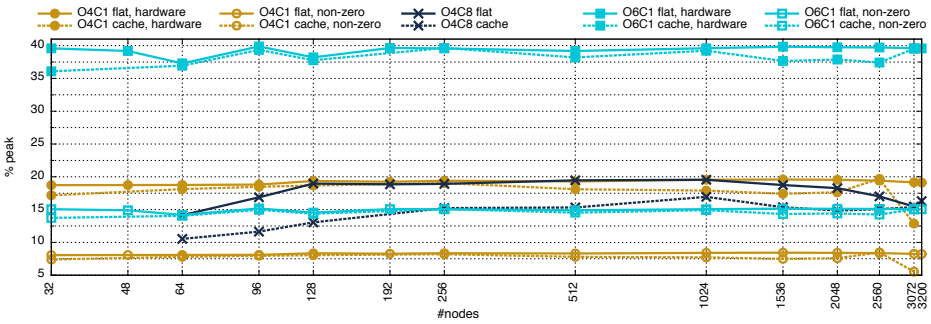

Fig. 7. Strong scaling study on Theta. Shown are hardware and non-zero peak efficiencies of all configurations in cache and flat mode. O denotes the order and C the number of fused simulations.

obtained peak efficiencies are almost identical to Theta and, once again, show EDGE's high hardware and non-zero peak efficiencies. On Cori-II, we sustained 10.4 PFLOPS in hardware when running O6C1 in flat mode on 9,000 nodes. To the best of our knowledge, this is the highest obtained peak performance for seismic simulations with ADER-DG, outperforming 24,576 cards of Tianhe-2, reaching 8.6 PFLOPS [14]. Further, O4C8 in flat mode has a 2.0 × higher single simulation throughput than O4C1 on 9,000 nodes with a sustained hardware and non-zero peak efficiency of 18.1 %, corresponding to 5.0 PFLOPS.

5.4 Strong Scaling

Our final performance study consists of two strong scaling setups of the LOH.1 benchmark (see Sec. 5.1). Here, we used a total of 172,386,915 tetrahedral elements on 32 to 3,072 nodes of Theta and a total of 340,727,199 tetrahedrons on 128 to 8,192 nodes of Cori-II. EDGE required a total of 1.7 TiB of memory for O4C1, 4.7 TiB for O4C8, and 2.6 TiB for O6C1 on Theta. Analogue, the setup consumed a total of 3.5 TiB for O4C1, 9.4 TiB for O4C8, and 5.2 TiB on Cori-II.

Fig. 7 shows the hardware and non-zero peak efficiencies of the cache and flat mode runs on Theta. We observe that the efficiencies are close to the weak scaling depicted in Fig. 5. Here, we have to remember that the weak scaling study relied on a perfectly balanced, artificial setup, while our strong scaling's mesh is fully unstructured and partitioned by Metis. When analyzing the performance of the O4C8-runs in detail, we see a plateau between 192 and 2048 nodes with performance dropping below and afterwards. The reason for the lowered performance below 192 nodes is the total memory requirements of the computational data structures exceeding Xeon Phi's 16 GB of HBM, required for optimal performance of Eq. (7). For high node counts, we see a degradation due to the extreme layout of the strong scaling, reaching a 50 × increase of O4C8 and 100 × of O4C1 and O6C1 at 3,200 nodes. Comparing the stable flat mode performance of O4C1 to that of O4C8, we see that the parallel efficiency of O4C8 drops sooner.

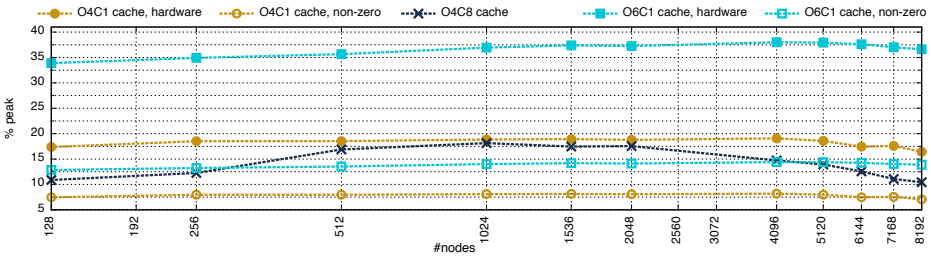

Fig. 8. Strong scaling study on Cori-II. Shown are hardware and non-zero hardware peak efficiencies of all configurations in cache mode. O denotes the order and C the number of fused simulations.

Recalling Sec. 3, this is property of the fused simulations, since the communication volume simply grows one-to-one with the number of fused simulations. However, the simulation throughput of O4C8 is greatly increased, which leaves less time spent in computations for hiding communication. The highest hardware performances were obtained in flat mode on 3,200 nodes: 1.6 PFLOPS (19.1 %) for O4C1, 1.4 PFLOPS (16.3 %) for O4C8, and 3.4 PFLOPS (39.6 %) for O6C1. With respect to non-zero peak performances, this corresponds to 0.7 PFLOPS (8.2 %) for O4C1, 1.4 PFLOPS (16.3 %) for O4C8 due to sparse matrix-matrix operators, and 1.3 PFLOPS (15.1 %) for O6C1.

Fig. 8 takes our strong scaling one step further, with a cache mode node-range of 128 to 8,192 on Cori-II. As already observed in the weak scaling in Fig. 6, the relative performance of all runs is slightly lower on Cori-II than on Theta, due to the higher per-socket performance. Again we observe an plateau for O4C8 due to HBM, but dropping performance for all runs at higher node counts. These drops are most severe for the O4C8 runs since our high single-node speedup (see Fig. 4) significantly decreases the time per simulation and time step, but keep the communication volume constant, exposing communication in the strong scaling. On 8,192 nodes every node only handles a total of 41,593 elements in average, facing an increase of 64 × in potential computer power from 128 nodes. Since cache mode is very sensitive to large compute we can observe a drop in performance at scale. Here, we reach a hardware performance of 4.1 PFLOPS (16.4 %) for O4C1, 2.6 PFLOPS (10.4 %) for O4C8 and 9.1 PFLOPS (36.6 %) for O6C1. The corresponding non-zero performances are 1.8 PFLOPS (7.1 %) for O4C1, 2.6 PFLOPS (10.4 %) for O4C8 and 3.5 PFLOPS (13.9 %) for O6C1.

6 Conclusions

This article has introduced EDGE[3], a novel solver for fused seismic simulations which aims at increasing the throughput of extreme scale seismic ensemble simulations. For highest accuracy, EDGE utilizes the Discontinuous Galerkin (DG)

[3] EDGE is available under the 3-clause BSD license at `http://dial3343.org`.

method for spatial and the Arbitrary high order DERivatives (ADER) scheme for time discretization, implemented for unstructured tetrahedral meshes. The occurring kernel routines, small sparse and dense matrix-matrix multiplications, are accelerated by a sophisticated runtime code generation approach. This technique allows for hardware efficiencies of more than 40% for single runs (10-20% non-zero efficiency) and more than 20% of non-zero efficiency when conducting fused simulations. Depending on the chosen order, fused simulations can offer an increased throughput of 1.8× to 4.6×. With respect to achieved raw performance EDGE weak-scaled to 9,000 nodes of the Cori-II supercomputer, while running at 10.4 PFLOPS at order six. For a fused fourth order run EDGE achieved 5.0 PFLOPS of non-zero/non-padded performance using small sparse matrix kernels. In addition to these excellent weak-scaling results, EDGE also exhibits nearly the same performance in case of strong scaling. This is achieved by a carefully designed parallel implementation, which minimizes threading overhead and maximizes MPI message progression. When strong scaling by 100 × on Theta and 64 × on Cori-II, EDGE sustained a performance of 3.4 PFLOPS and 9.1 PFLOPS, respectively.

Acknowledgements

Only the great support of experts at NERSC and ALCF made our extreme-scale results possible. In particular, we thank J. Deslippe, S. Dosanjh, R. Gerber, and K. Kumaran. This work was supported by the Southern California Earthquake Center (SCEC) through contribution #16247. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575.

References

1. Ch. 4 & Ch. 6. In J. Reinders, J. Jeffers, and A. Sodani, editors, *Intel Xeon Phi Processor High Performance Programming Knights Landing Edition*. 2016.
2. S Aoi and H Fujiwara. 3D finite-difference method using discontinuous grids. *Bulletin of the Seismological Society of America*, 1999.
3. D Appelö and NA Petersson. A stable finite difference method for the elastic wave equation on complex geometries with free surfaces. *Communications in Computational Physics*, 2009.
4. P Bastian, C Engwer, J Fahlke, M Geveler, D Göddeke, O Iliev, O Ippisch, R Milk, J Mohring, S Müthing, M Ohlberger, D Ribbrock, and S Turek. *Hardware-Based Efficiency Advances in the EXA-DUNE Project*. 2016.

5. M Benjemaa, N Glinsky-Olivier, VM Cruz-Atienza, and J Virieux. 3D dynamic rupture simulations by a finite volume method. *Geophysical Journal International*, 2009.
6. A Breuer, A Heinecke, S Rettenberger, M Bader, A-A Gabriel, and C Pelties. Sustained petascale performance of seismic simulations with SeisSol on SuperMUC. In *International Supercomputing Conference*, 2014.
7. E Chaljub, D Komatitsch, J-P Vilotte, Y Capdeville, B Valette, and G Festa. *Spectral-element analysis in seismology*. 2007.
8. E Chaljub, E Maufroy, P Moczo, J Kristek, F Hollender, P-Y Bard, E Priolo, P Klin, F de Martin, Z Zhang, W Zhang, and X Chen. 3-D numerical simulations of earthquake ground motion in sedimentary basins: testing accuracy through stringent models. *Geophysical Journal International*, 2015.
9. M Dumbser and M Käser. An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes – ii. the threedimensional isotropic case. *Geophysical Journal International*, 2006.
10. K Duru and EM Dunham. Dynamic earthquake rupture simulations on nonplanar faults embedded in 3D geometrically complex, heterogeneous elastic solids. *Journal of Computational Physics*, 2016.
11. V Etienne, E Chaljub, and J Virieux. An hp-adaptive discontinuous galerkin finite-element method for 3-D elastic wave modelling. *Geophysical Journal International*, 2010.
12. C Geuzaine and J-F Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.
13. A Heinecke, A Breuer, M Bader, and P Dubey. High order seismic simulations on the intel xeon phi processor (knights landing). In *International Conference on High Performance Computing*, 2016.
14. A Heinecke, A Breuer, S Rettenberger, M Bader, A-A Gabriel, C Pelties, A Bode, W Barth, X-K Liao, K Vaidyanathan, et al. Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2014.
15. A Heinecke, G. Henry, M Hutchinson, and H. Pabst. LIBXSMM: Accelerating Small Matrix Multiplications by Runtime Code Generation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2016.
16. T Ichimura, K Fujita, P Quinay, L Maddegedara, M Hori, S Tanaka, Y Shizawa, H Kobayashi, and K Minami. Implicit nonlinear wave simulation with 1.08T DOF and 0.270T unstructured finite elements to enhance comprehensive earthquake simulation. 2015.
17. T Ichimura, M Hori, and J Bielak. A hybrid multiresolution meshing technique for finite element threedimensional earthquake ground motion modelling in basins including topography. *Geophysical Journal International*, 2009.
18. George K and Vipin K. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0, 2009.
19. T-S Kang and C-E Baag. An efficient Finite-Difference method for simulating 3D seismic response of localized basin structures. *Bulletin of the Seismological Society of America*, 2004.
20. M Käser, M Dumbser, J Puente, and H Igel. An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes – iii. viscoelastic attenuation. *Geophysical Journal International*, 2007.

21. D Komatitsch and J Tromp. Spectral-element simulations of global seismic wave propagation—II. three-dimensional models, oceans, rotation and self-gravitation. *Geophysical Journal International*, 2002.
22. J Kristek, P Pazak, M Galis, and H Igel. Comparison of numerical methods for seismic wave propagation and source dynamics-the spice code validation. 2006.
23. M Kristeková, J Kristek, and P Moczo. Time-frequency misfit and goodness-of-fit criteria for quantitative comparison of time signals. *Geophysical Journal International*, 2009.
24. M Kristeková, J Kristek, P Moczo, and SM Day. Misfit criteria for quantitative comparison of seismograms. *Bulletin of the seismological Society of America*, 2006.
25. J Liu, B Chandrasekaran, J Wu, W Jiang, S Kini, W Yu, D Buntinas, Peter Wyckoff, and DK Panda. Performance comparison of mpi implementations over infiniband, myrinet and quadrics. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*.
26. S Ma and P Liu. Modeling of the perfectly matched layer absorbing boundaries and intrinsic attenuation in explicit Finite-Element methods. *Bulletin of the Seismological Society of America*, 2006.
27. T Malas, T Kurth, and J Deslippe. *Optimization of the Sparse Matrix-Vector Products of an IDR Krylov Iterative Solver in EMGeo for the Intel KNL Manycore Processor*. 2016.
28. P Moczo, J Kristek, and V Vavryčuk. 3D heterogeneous staggered-grid finite-difference modeling of seismic motion with volume harmonic and arithmetic averaging of elastic moduli and densities. *Bulletin of the Seismological Society of America*, 2002.
29. P Moczo, J Robertsson, and L Eisner. *The Finite-Difference Time-Domain Method for Modeling of Seismic Wave Propagation*. 2007.
30. A Modave, St-Cyr, A, and T Warburton. GPU performance analysis of a nodal discontinuous galerkin method for acoustic and elastic models. *Computers & Geosciences*, 2016.
31. D Peter, D Komatitsch, Y Luo, R Martin, N Goff, E Casarotti, P Loher, F Magnoni, Q Liu, C Blitz, NissenMeyer, T, P Basini, and J Tromp. Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes. *Geophysical Journal International*, 2011.
32. A Pitarka. 3D elastic finite-difference modeling of seismic motion using staggered grids with nonuniform spacing. *Bulletin of the Seismological Society of America*, 1999.
33. JF Shepherd and CR Johnson. Hexahedral mesh generation constraints. *Engineering with Computers*, 2008.
34. WW Symes and T Vdovina. Interface error analysis for numerical wave propagation. *Computational Geosciences*, 2009.
35. R Taborda and J Bielak. Large-Scale earthquake simulation: Computational seismology and complex engineering systems. *Computing in Science & Engineering*, 2011.
36. Cruz-Atienza, VM, J Virieux, and H Aochi. 3D finite-difference dynamic-rupture modeling along nonplanar faults. *Geophysics*, 2007.