Petaflop Seismic Simulations in the Public Cloud

Alexander Breuer[1] and Yifeng Cui[1]

UC San Diego, La Jolla CA 92093, USA
{anbreuer,yfcui}@ucsd.edu

Abstract. During the last decade cloud services and infrastructure as a service became a popular solution for diverse applications. Additionally, hardware support for virtualization closed performance gaps, compared to on-premises, bare-metal systems. This development is driven by offloaded hypervisors and full CPU virtualization. Today's cloud service providers, such as Amazon or Google, offer the ability to assemble application-tailored clusters to maximize performance. However, from an interconnect point of view, one has to tackle a 4-5$\times$ slow-down in terms of bandwidth and 25$\times$ in terms of latency, compared to latest high-speed and low-latency interconnects. Taking into account the high per-node and accelerator-driven performance of latest supercomputers, we observe that the network-bandwidth performance of recent cloud offerings is within 2$\times$ of large supercomputers. In order to address these challenges, we present a comprehensive application-centric approach for high-order seismic simulations utilizing the ADER discontinuous Galerkin finite element method, which exhibits excellent communication characteristics. This covers the tuning of the operating system, normally not possible on supercomputers, micro-benchmarking, and finally, the efficient execution of our solver in the public cloud. Due to this performance-oriented end-to-end workflow, we were able to achieve 1.09 PFLOPS on 768 AWS c5.18xlarge instances, offering 27,648 cores with 5 PFLOPS of theoretical computational power. This correlates to an achieved peak efficiency of over 20% and a close-to 90% parallel efficiency in a weak scaling setup. In terms of strong scalability, we were able to strong-scale a science scenario from 2 to 64 instances with 60% parallel efficiency. This work is, to the best of our knowledge, the first of its kind at such a large scale.

Keywords: high-order DG · seismic simulations · earthquake simulations · cloud computing · petascale computing

1 Introduction and Related Work

About 10 years after the introduction of cloud services, their 2018 worldwide revenue is estimated above 175 billion U.S. dollars, with a projected growth of over 17% in 2019[1]. Further, recent enhancements of Cloud Service Providers

[1] Source: https://www.gartner.com/en/newsroom/press-releases/2018-09-12-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2019.

(CSPs), e.g., the introduction of lightweight virtualizations and high-bandwidth networks, led to competitive solutions for the High Performance Computing (HPC) market. Yet, federal and institutional machines dominate the November 2018 Top500 List. This dominance is accompanied by an intense discussion of the HPC community, often questioning the feasibility of clusters, operating in the cloud [7, 14, 18, 17]. Therefore, virtualized Infiniband solutions [15] or loosely coupled applications were proposed [8].

This work studies the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) and the Google Compute Engine (GCE) of the Google Cloud Platform (GCP) in the context of large-scale HPC. First, we present thorough general-purpose performance benchmarking, explaining crucial HPC implications of the cloud providers' hardware settings. Next, we present a comprehensive study of high-order seismic simulations with the ADER discontinuous Galerkin finite element method. The method has been continuously and extensively optimized for extreme-scale performance (more than 10 PFLOPS) in the last five years [19, 4, 22, 2, 12, 3]. However, respective advancements are limited to on-premises bare-metal machines. By exploiting the public cloud for the setup of tailored elastic supercomputers, we obtain a true end-to-end approach, starting at the machine setup, covering HPC optimizations, and reaching the full spectrum of modeling and simulation. Our contributions in this work are as follows:

a) Sec. 2 motivates the need for fused forward simulations in earthquake science and summarizes the application EDGE, short for Extreme-scale Discontinuous Galerkin Environment. This section also introduces a new open source surface meshing tool and a new dynamic load balancing scheme for the solver's shared memory parallelization in noisier execution environments.

b) Sec. 3 illustrates, that the open-source HPC ecosystem is well-prepared to operate high performance cloud computing solutions with latest hardware enhancements. Here, we describe the optimization of the CentOS7 Linux operating system for our cloud clusters, the preparation of custom machine images through system-wide setups of dependencies, and the use of the batch scheduling tools AWS ParallelCluster and Slurm GCP for elastic scalability.

c) Sec. 4 assesses the theoretical performance of AWS EC2 and GCE through rigorous micro-benchmarking and shows that recent cloud-offerings are performance-comparable to bare-metal, on-premises systems.

d) Sec. 5 analyzes the performance and scalability of the software EDGE in the cloud. We demonstrate that it is possible to achieve petascale performance for tightly coupled high-order DG simulations. This includes nearly matching the performance of an entire 2013 Top10 supercomputer (SuperMUC) for the same scientific workload, when using an elastic petascale cluster in the public cloud.

We conclude our presentation by summarizing transferable observations and discussing implications for the future of HPC in Sec. 6.

2 Earthquake Simulations

High-dimensional challenges in earthquake science are common and have an inherently parallel inter-problem component. Important examples are Probabilistic Seismic Hazard Analysis (PSHA), the derivation of seismic velocity models through tomographic inversion, or seismic source inversions. Common approaches exploit the linearity of the used seismic wave propagation models. This enables reciprocity in the Strain Green's Tensors (SGTs) [5, 24], which, in simple words, allows us to exchange seismic sources with seismic receivers.

For example, CyberShake [11], the approach of the Southern California Earthquake Center to PSHA, discretizes the study area into hazard sites. Each site is a point of interest at the surface, where we quantify the seismic hazard, originating from potential fault ruptures in the vicinity of the site. We have two options to compute the ground shaking from the discretized high-dimensional space of uncertain ruptures: a) Run one forward simulation for every fault rupture and sample the seismic wave field at each of the hazard sites, or b) exploit reciprocity by running two (horizontal ground motion components only) or three forward simulations for every hazard site, and sample the seismic wave field at the surrounding faults. The latter case is preferable, if the number of hazard sites is much smaller than the number of considered ruptures, as in the case of PSHA.

In either case, the simulation setup of close-by ruptures in a), or close-by hazard sites in b) is, except for the used source discretization, typically identical. Shared parameters include the seismic velocity model, the mesh, the simulations' end time, and the output sampling of the wave field. From a computational perspective, this allows us to exploit inter-problem parallelism by fusing multiple forward simulations within one execution of the solver. The Extreme-scale Discontinuous Galerkin Environment (EDGE) is the first seismic solver, which integrates the idea of fused simulations into the entire modeling and simulation pipeline [4]. The remainder of this section describes a model setup, covering the San Andreas Fault's Parkfield section using EDGE. This fused setup is also used as the setting for our strong scaling study in Sec. 5.

2.1 Fused Forward Simulations

We use the Discontinuous Galerkin (DG) method in space and the ADER scheme in time to solve the elastic wave equations in velocity-stress formulation. The elastic wave equations are a linear system of hyperbolic partial differential equations:

$$q_t + Aq_x + Bq_y + Cq_z = S. \tag{1}$$

$\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3$ is the vector of Cartesian coordinates and $t \in \mathbb{R}^+$ time. Subscripts denote partial derivatives. The three normal stresses σ_{xx}, σ_{yy} and σ_{zz}, the three shear stresses σ_{xy}, σ_{xz} and σ_{yz}, and the three particle velocities in x-, y- and z-direction, given as u, v and w, are summarized in the nine-dimensional vector of quantities $q(\boldsymbol{x}, t) = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}, u, v, w) \in \mathbb{R}^9$. The three space-dependent Jacobians $A(\boldsymbol{x}), B(\boldsymbol{x}), C(\boldsymbol{x}) \in \mathbb{R}^{9\times 9}$ depend on the seismic velocity model. The right-hand-side term, $S(x, t)$, accounts for seismic sources.

Application of the ADER-DG machinery leads to the discrete formulation. We use unstructured tetrahedral meshes for the spatial discretization of the computational domain. The discrete formulation consist of a series of small and sparse matrix-tensor products, which drive our computational single core performance. EDGE's fused approach allows us to execute these products as fully-vectorized sparse operators on cache-line-aligned degrees of freedom without artificial zero-padding [4]. More precisely, the LIBXSMM-library[2] is used to run-time generate and compile vectorized kernels, targeting Intel's AVX512 instruction set extensions. In the following, we use a fifth order ADER-DG scheme in space and time, and refer to [4] for further details on EDGE's discretization.

Previous versions of EDGE implemented the Standard Rupture Format[3] (SRF) for the source terms $S(x,t)$. The SRF discretizes kinematic ruptures as a collection of rupturing planar sub-faults, which act on the stress tensor. We replaced this implementation by a new and generic point source discretization in this work. Our new HDF5-based source format allows us to modify the particle velocities (not only the stresses) in the source terms, as required for the implementation of point forces at the surface. Surface point forces are, for example, used for forward simulations in PSHA. Additionally, EDGE's new source input reduces the modeling burden by projecting specified sources, outside of the computational domain, to the surface of the mesh.

2.2 Model Setup

Mesh: In the first step of our setup, we derived a surface triangulation from the 1/3rd arc-second Digital Elevation Models (DEMs) of the USGS National Map 3DEP Downloadable Data Collection[4] in the area of interest. For this purpose, we introduce the tool EDGEcut, based on the open-source library CGAL [1, 23]. EDGEcut is able to automatically triangulate a projected DEM and to compute feature-preserving intersections of the discretized mountain topography with specified outflow boundaries. We used the transverse Mercator projection with center at 35.817°N, 120.365°W to project the DEM to a plane. The projection center coincides with the epicenter of the 2004 Parkfield event in [6]. Further, we introduced outflow boundaries at an 80 km epicentral distance in every cardinal direction and 40 km below sea level. EDGEcut supports problem-adapted surface meshing by following the attractor concept of the volume mesher Gmsh [9]. Here, we defined an attractor at (-6 km, 6 km, 0) and linearly coarsened the surface mesh by eight times in an attractor-distance from 10 km to 50 km. We used a minimal edge length of 200 m and identical refinement specifications for the final volume meshing through Gmsh.

Velocity Model: We used a homogeneous velocity model with a density of $\rho = 2.8\,\mathrm{g/cm}^3$, an s-wave velocity of $v_s = 1.2\,\mathrm{km/s}$, and a p-wave velocity of $v_p =$

[2] LIBXSMM is available from: https://github.com/hfp/libxsmm.
[3] http://equake-rc.info/static/publish/paper/SRF-Description-Graves_2.0.pdf.
[4] https://catalog.data.gov/dataset/national-elevation-dataset-ned-1-3-arc-second-downloadable-data-collection-national-geospatial

3 km/s. Note, that EDGE's tool EDGE-V supports data-based mesh annotations via the Unified Community Velocity Model [20] and velocity-aware mesh refinement. However, as outlined below, our focus in this work is topography support for high-dimensional earthquake science.

Sources, Receivers and SGTs: Currently, most approaches to high-dimensional earthquake science use flat topography. The reason is often originated in the use of finite difference forward solvers, relying on regular meshes. As we are reaching higher resolved frequencies through the use of more powerful supercomputers, this lack of modeling complexity is getting more severe. Thus, for example in PSHA, adding topography to the forward solves is one of the most urgent model extensions. In this work, we benchmark the accuracy of three-dimensional reciprocal computations through SGTs, when using mountain topography in EDGE. For this purpose we placed eight sources at the surface. Our setup uses two configurations for each of the sources, a point force in x-direction (West-East), and a point force in y-direction (South-North). The source-time function of the point forces is given through the following Gaussian:

$$S(t) = \mathrm{e}^{-60 \cdot t^2}. \tag{2}$$

In addition, we ran a single forward simulation with a single double-couple point source, located at $(0, 0, -7622.4\,\mathrm{m})$. The source-time function of this source is a Ricker wavelet:

$$S(t) = \left(\frac{1}{2} - (1.92\pi)^2 \cdot t^2 \right) \mathrm{e}^{-(1.92\pi)^2 \cdot t^2}. \tag{3}$$

We obtained all simulation results in this section by using EDGE's fifth order ADER-DG scheme, 32-bit floating point precision, and by running in the Google Cloud Platform. Fig. 1 illustrates our model setup, where the visualized wave fields correspond to the eight forward simulations with the point force in South-North direction. Further, similar to [24], Fig. 2 compares the synthetics of the single forward simulation to the SGT-derived synthetics of the point forces. Here, each of the signals was convolved with the Ricker wavelet as the new source-time function. We observe an almost perfect fit of the seismograms, which confirms the applicability of EDGE's reciprocal SGT pipeline within our modeling constraints. This procedure could now, for example, be extended to PSHA, where the insertion of rupture uncertainties reduces to a data-processing step w.r.t. source convolutions, once the forward simulations are completed.

2.3 Shared Memory Dynamic Load Balancing

EDGE is exposed to two sources of load imbalances in the shared memory domain: a) Possibly runtime-dependent performance variations of the worker threads, and b) diverse memory access patterns, caused by the unstructured mesh, when reading face-adjacent data in EDGE's neighboring kernel. Tasks of the operating system can contribute background noise to the first case. Here, our core-specialization in Sec. 3 isolates most of the operating system's tasks to

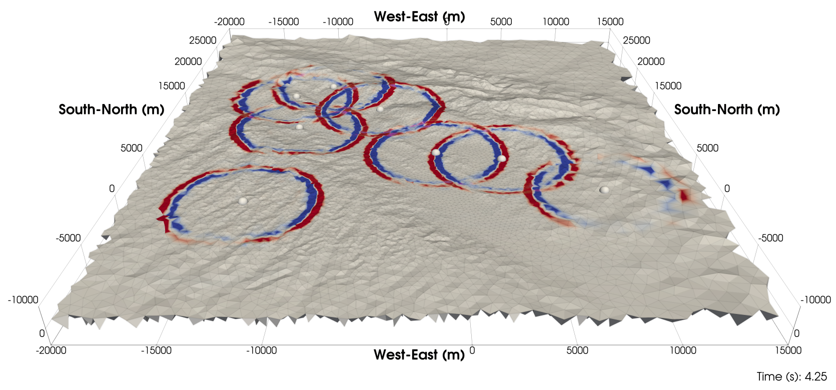

Fig. 1. Visualization of the surface point force forward simulations. The output requirements were greatly reduced by only writing tetrahedrons at the surface and by limiting the output to the first and constant of the 35 modes of every fused simulation. The gray spheres indicate the locations of the surface point sources. Colors denote the South-North particle velocities of the eight South-North point forces after 4.25 simulated seconds. Warm colors denote positive velocities, cold colors negative ones. The entire run covered 16 fused settings and was executed in GCP.

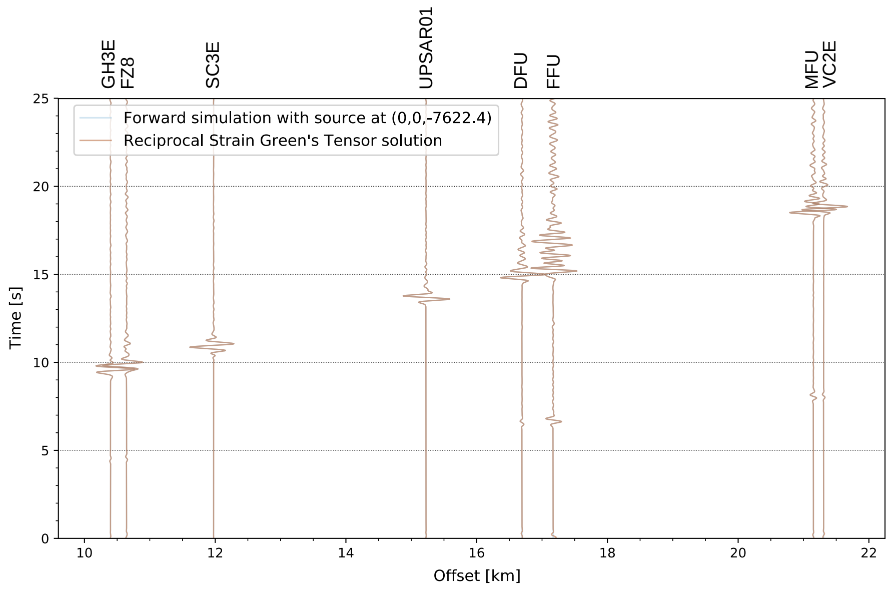

Fig. 2. Comparison of the post-processed point force simulations with the double-couple reference. Shown are the seismograms of the particle velocity in South-North direction for the eight stations at the surface. The x-axis reflects hypocentral distance. The convolved SGTs are largely indistinguishable from the reference. At the very beginning of each seismogram, a small and expected offset is visible, since we processed the raw signals without tapering.

reserved cores, not used by any of EDGE's threads. However, within the preparation of this work, we additionally observed rare ($\approx$1% probability per node and run) performance drops of isolated cores in our program execution. These drops appear to be independent of the Skylake-processor (bare-metal or virtualized) and the affected core seems to be random. They cannot be circumvented by using 1 GiB huge pages, as done in [16]. The impact on the overall runtime of EDGE without dynamic load balancing is severe, since we observed up to 20% slowdowns in our ADER-DG kernels.

We introduce a dynamic load balancing scheme to account for possible imbalances in EDGE's shared memory domain. Our load balancer is called, whenever EDGE reaches a synchronization point and the threads are joined. Synchronization automatically happens for wave field output at the free-surface or in the volume, whereas our point-wise sampling of the wave field at seismic receivers is entirely asynchronous. Thus, in settings using receivers only, e.g., in an SGT-only configuration, we enforce artificial synchronization for the purpose of load balancing. This synchronization interval is a runtime parameter, where all our scaling runs in this work used 5% of the simulation time, resulting in a total of 19 synchronization points for the duration of each run. Whenever we reach a synchronization point, we determine if re-balancing of any of our work regions is required. EDGE's seismic solver has four significant work regions: 1) the ADER time-prediction and local update of the send-elements, which computes data, required by other ranks, 2) the ADER time-prediction and local update of inner-elements, independent of communication within a time step, 3) the neighboring update of the send-elements, requiring data from other ranks, and 4) the neighboring update of inner-elements, not requiring any data of other ranks within a time step [4].

Let us assume a single work region, where worker w is responsible for N_w elements in a time step. This worker spent a total of t_w seconds in respective work packages from the previous synchronization point to the current one, where the load balancing is executed. Further, N_{all} is the number of all elements in the work region, $\max(t_w)$ is the maximum invested time of any worker, $\min(t_w)$ is the minimum spent time of any worker, and $\text{ave}(t_w)$ is the average time, spent by the workers. We define the element throughput T_w of a worker w, the element throughput T_{all} of all workers, the imbalance I of the work region, and the rebalancing R_w of worker w as follows:

$$T_w = \frac{N_w}{t_w}, \quad T_{\text{all}} = \sum_{w=1}^{W} T_w, \quad I = \frac{\max(t_w) - \min(t_w)}{\text{ave}(t_w)} \quad R_w = \frac{T_w \cdot N_{\text{all}}}{T_{\text{all}}}. \qquad (4)$$

Now, whenever the imbalance I exceeds a given threshold, e.g., $I > 2.5\%$ in our case, we re-balance our work region by assigning R_w elements to each of the workers. If this does not lead to a worker for every element, we increase the number of elements per worker round-robin, until the N_{all} elements are distributed.

3 Cloud Setup

The cloud offers Infrastructure as a Service (IaaS). This allows us to customize the entire setup of our high-performance compute clusters to match EDGE's demands. Included are not only the choice of the underlying hardware, e.g., the CPU architecture, but also the entire software stack, for example, the operating system and its boot-options. This section describes the setup of our customized cloud-based compute clusters using the Amazon Web Services (AWS) and the Google Cloud Platform (GCP). The obtained single-application clusters are highly specialized to maximize the performance of EDGE (see Sec. 2). All used software and tools are freely available and open source. Thus, in contrast to commercial high performance cloud computing solutions, respective charges of our clusters solely originate from the used AWS and GCP resources.

The first software-related step of our cloud setup is the generation of customized cloud images, which are used for our login and compute instances. We base our AWS cloud image on the AWS ParallelCluster[5]-variant of CentOS7. Our GCP CentOS7 cloud image customizes the cloud-offered images of the GCP centos-7 family. Both cloud image setups share the same set of scripts, which first install all tools and libraries, required for building and executing our solver. For example, we install a recent version of the GNU compilers, OpenMPI, the libraries HDF5 and MOAB, or the performance monitoring tools Score-P and Scalasca. Once all software is installed system-wide, we customize the configuration of the operating system to maximize the instances' performance and to minimize possible interference with our solver. Using dual-socket instances, we reserve the first core of every socket for the operating system and instruct it through the GRUB2 bootloader to exclusively use these two cores. We complement this configuration in the job executions by pinning our applications' threads to all but the two set-aside cores. Upon completion of the setups, we store the images permanently in the cloud. We open-sourced the scripts, tuning our cluster, such that our findings can be transferred to other software.

We use the two tools AWS ParallelCluster and Slurm GCP[6] to generate our high performance computing clusters. AWS ParallelCluster and Slurm GCP use the clouds' APIs for this step, for example, by generating respective virtual private networks, or by using our machine images for the compute instances. Both tools offer a variety of configuration-options, where the most important ones are the used instance types, the used cloud images, and the instance placing. For maximum network performance, we use a dynamic AWS placement group and a single GCP zone for all of the clusters' instances. Further, since we are generating single-application clusters, we either generate clusters, exactly matching our instance requirements, or allocate no initial compute instances at all. In the latter case, if not used for computations, AWS ParallelCluster runs a single master instance, and our Slurm GCP configuration a single controller instance. The submission and monitoring of jobs on the generated clusters is similar to ev-

[5] AWS ParallelCluster is available from: https://aws-parallelcluster.readthedocs.io.
[6] Slurm GCP is available from: https://github.com/SchedMD/slurm-gcp.

ery other Slurm-based[7] on-premises solution. However, after job submission, the change in Slurm's node types triggers respective resume-scripts in background. These scripts elastically allocate instances from the cloud for use as Slurm nodes. Analogue, after completion of the job and a pre-defined idle time, the compute instances are released back to the cloud through suspend-scripts. Because our AWS and GCP charges are dominated by the number of allocated instances, the process of elastically allocating and releasing infrastructure minimizes costs.

4 Benchmarking the Cloud

This section summarizes important Key Performance Indicators (KPIs) of various cloud instance types on a per-node basis. We limit ourselves to Intel Xeon instances featuring Skylake-SP CPUs, as our application makes heavy use of 512-bit vector instructions, while maintaining a small memory footprint. CSPs use special versions of these processors and do not publish their processor-specifications. The same applies to the specifications of the physical memory population or network details. We mitigate this lack of documentation by studying a set of micro-benchmarks and a single-node setup of EDGE on various instance types. The obtained performance is then compared to runs on an on-premises bare-metal dual-socket Intel Xeon Platinum 8180 machine. The Xeon 8180 is the top-of-the-line processor, that is generally available and fully documented[8]. Apart from illustrating the actual performance of each instance type, we also set the micro-benchmarks' performance in relation to respective charges. This allows us to pick the best cloud solution for our application in terms of U.S. dollars ($) per simulation.

Tab. 1 summarizes all KPIs, we were able to gather from online documentation for our considered instance types. In the text of this section, we shorten notation by only using the lower-case names of the instance families, when referring to the considered instance models: **n1** for n1-highcpu-96, **c5** for c5.18xlarge, **c5n** for c5n.18xlarge, and **m5** for m5.24xlarge.

As instances are only described by their number of vCPUs (which are hardware threads) and the amount of available memory, it is hard to conclude how the actual underlying dual-socket platform is comprised. Let us take the vCPU count as an example. Here, the number of physical CPU cores could be higher and remaining "empty" cores could run the hypervisor. We found an indication in the AWS News Blog, that c5, c5n and m5 use the so-called Nitro Hypervisor, which provides nearly full hardware performance[9]. This indicates that no cores are set aside for additional management tasks. For Google's n1 we were not able to find a hint supporting one or the other assumption. A micro-benchmark could determine this detail by trying to determine Skylake's last level cache size, which

[7] AWS ParallelCluster supports further submission systems, e.g., AWS Batch or SGE.
[8] https://ark.intel.com/products/120496/Intel-Xeon-Platinum-8180-Processor-38-5M-Cache-2-50-GHz
[9] AWS News Blog post: https://aws.amazon.com/blogs/aws/amazon-ec2-update-additional-instance-types-nitro-system-and-cpu-options/.

KPI	n1-highcpu-96	c5.18xlarge	c5n.18xlarge	m5.24xlarge	on-premises
CSP	Google	Amazon	Amazon	Amazon	N/A
CPU name	N/A	8124M*	8124M*	8175M*	8180
#vCPU (incl. SMT)	2x48	2x36	2x36	2x48	2x56
#physical cores	N/A	2x18**	2x18**	2x24**	2x28
AVX512 Frequency	$\leq$2.0GHz	$\leq$3.0GHz	$\leq$3.0GHz	$\leq$2.5GHz	2.3GHz
DRAM [GB]	86.4	144	192	384	192
#DIMMs	N/A	2x10?	2x12?	2x12/24?	2x12
preemptive $/h	$0.72	$0.7	$0.7	$0.96	N/A
on-demand $/h	$3.4	$3.1	$3.9	$4.6	N/A
interconnect [Gbps]	16(eth)	25***(eth)	25***(eth)	25***(eth)	100(OPA)

Table 1. Publicly available KPIs for various cloud instances of interest to our workload. Pricing is for US East at non-discount hours on Monday mornings (obtained on 3/25/19). *AWS CPU core name strings were retrieved using the "lscpu" command; **AWS physical cores are assumed from AWS's documentation, indicating that all cores are available to the user due to the Nitro Hypervisor; ***supported in multi-flow scenarios (means multiple communicating processes per host), each process is limited to 10 Gbps.

is built as an aggregated cache of Cache-Home-Agent (CHA) slices. Normally, the number of cores matches the number of active CHAs. However, this is not important to our application EDGE, hence we did not perform such a test.

As the specifications of the cloud CPUs are not publicly available, also their frequencies are largely unknown, especially when running AVX(512) instructions. Therefore, the AVX512 Turbo frequencies are unknown, but given that they are normally lower than the regular base frequency, we can take the frequencies in the CSPs' online documentation[10][11] as an upper limit.

Similar educated guessing is needed, when studying the instances' memory configurations. In n1's case, 0.9 GB per core are offered, resulting in 86.4 GB for a two-socket machine. If the machine would be fully populated with 6 DIMMs per socket, this would mean 7.2 GB DIMMs. Therefore, we assume that the physical memory is (much) bigger, but still do not know if all DIMMs are plugged in. For the AWS instances the amount of available memory is at least matching with 16GB populations, allowing the thesis, that c5 instances have 10 out of 12 slots in use, while c5n and m5 are fully populated. This theory is supported by AWS's recent announcement, that c5n instances can offer up to 19% higher memory bandwidth than c5 instances[12].

To shed more light on the instance types, we present micro-benchmarks with the goal to fill and/or refine some of the vague entries in Tab. 1. In particular, we test the floating point throughput, the memory throughput, the interconnects' capabilities and the full-application performance of a single instance.

[10] https://aws.amazon.com/ec2/instance-types/
[11] https://cloud.google.com/compute/docs/cpu-platforms
[12] https://aws.amazon.com/blogs/aws/new-c5n-instances-with-100-gbps-networking/

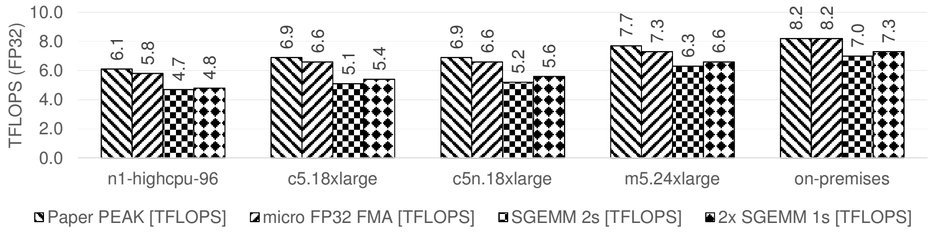

Fig. 3. Sustained FP32-TFLOPS of various instance types: a) simple FMA instruction from register (micro FP32 FMA), b) an MKL-SGEMM call, spanning both sockets (SGEMM 2s), and c) two MKL-SGEMM calls, one per socket (SGEMM 1s). All numbers are compared to the expected AVX512 turbo performance (Paper PEAK).

4.1 Floating Point Throughput

Our first test studies the raw floating point performance. This is a key performance metric for EDGE, since the solver's local kernel is heavily flop-bound in order five. This kernel executes small FP32 sparse matrix-tensor operators. The results of our performance tests are shown in Fig. 3. Here, the first bar for each instance type is theoretical peak performance, derived from documented values (CSPs' websites, data sheet for bare-metal [13]). We see that our microbenchmark, simply running FMA instructions through a sequence of 32 independent vfmadd231ps instructions, is able to reach close to the expected peak performance. While the bare-metal runs match our expectations, we observe about 5% lower values for the virtualized cloud instances. This can have several reasons: a) the virtualization is adding a slight overhead, or b) the AVX512 all-core turbo-frequencies are about 100 Mhz lower than the CSP-specified frequencies in Tab. 1. In summary, we see that n1 is able to get 71% of the bare-metal system. The AWS instance models c5/c5n reach 80%, while m5 is at 90%. These numbers are aligned with the difference in peak performance, meaning that the cloud configurations are within 95% of the efficiency of the bare-metal system.

Further, we ran SGEMM across both sockets and two SGEMMs per socket. In the latter case, we obtained, compared to the bare-metal system, 66% for n1, 74% for c5, 77% for c5n, and 90% for m5. This indicates a difference in the memory subsystem between c5 and c5n, and a weaker subsystem for n1. Also, c5n seems to throttle the AVX512 frequency by 100-200 MHz, as the ratios drop to 77% from 80%. We observed a 50:50 frequency split on the bare-metal machine between 2.2 and 2.3 GHz, due to the TDP limit of the CPUs. Apart from small performance losses ($\leq$10%), compared to our FMA benchmark, we conclude, that all instance types offer a solid performance relative to their peak. Thus, the instance type should be chosen by the pricing for flop-bound codes.

4.2 Memory

After analyzing the floating performance of heavily compute-bound kernels, we switch to the other extreme and investigate the offered memory bandwidth.

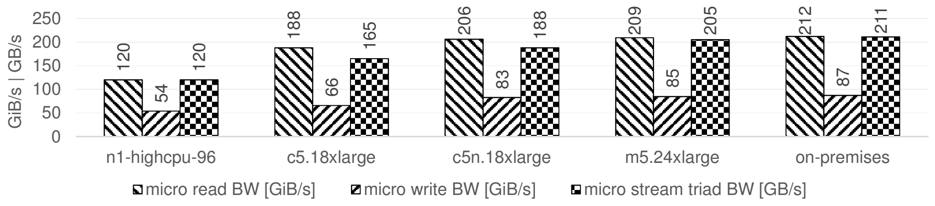

Fig. 4. Sustained bandwidth of various instance types: a) a pure read-bandwidth benchmark (read BW), b) a pure write-bandwidth benchmark (write BW), and c) the classic STREAM triad with a 2:1 read-to-write mix (stream triad BW).

EDGE's unstructured accesses to data of the four faced-adjacent tetrahedrons in the neighboring update kernel ($4 \cdot 35 \cdot 9 \cdot 16 \cdot 4$ byte/float = 80,640 bytes) generate a significant pressure on the memory subsystem. Given Skylake-SP's FLOPS/bandwidth ratios, this operation runs at close-to full read bandwidth for order five. As shown in Fig. 4, the indication of the SGEMM tests in Sec. 4.1 is confirmed, since the n1 and c5 instances do not reach Skylake-SP's maximum memory performance. Since Goto's algorithm [10] has a high write bandwidth demand, due to blocking of the inner product, we can now explain the SGEMM performance drop. In the case of n1, the measured read bandwidth is very low, indicating either a multiplexed system among several VMs, a low physical DIMM population, and/or issues with the virtualization of the NUMA domains of the host system. For the STREAM triad, c5n is 14% faster than c5. m5 is an additionally 9% faster than c5n and very close to the bare-metal solution. This is aligned with Amazon's statement that c5n can provide up to 19% more memory performance over c5. Taking the memory sizes into account, this hints that c5 instances have only 10/12 DIMM sockets populated, whereas c5n and m5 should use all 6 memory channels per socket. A reduction of the populated memory channels for compute-optimized instances is explainable, since DRAM is a huge cost factor in a datacenter. In summary, the c5n and m5 instance models behave similar to the bare-metal machine. While c5 has an additional degradation, n1's read bandwidth is considerably lower. Sec. 4.4 studies how the memory bandwidth influences EDGE's full application performance.

4.3 Interconnect

We close our micro-benchmarking section of the cloud by examining the interconnect, when using AWS's c5 and GCP's n1 instance models. Here, we ran a subset of the latest OSU MPI micro-benchmarks[13]. Fig. 5 depicts the following micro-benchmarks: point-to-point one process pair unidirectional bandwidth (osu_bw), point-to-point multiple processes pair unidirectional bandwidth (osu_mbw_mr), point-to-point one process pair bidirectional bandwidth (osu_bibw), point-to-point one process pair latency (osu_latency). osu_bw confirms our expectations

[13] http://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-5.5.tar.gz

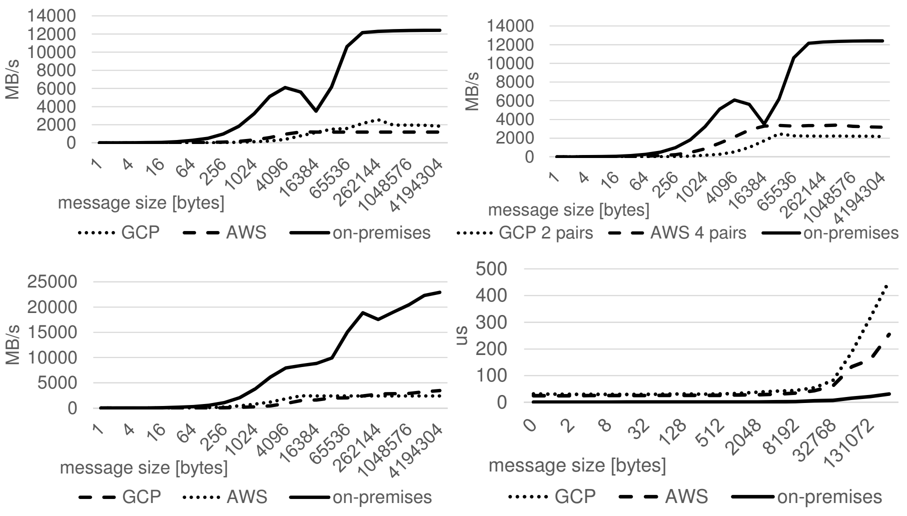

Fig. 5. Interconnect performance of n1-highcpu-96 (GCP), c5.18xlarge (AWS) and the on-premises, bare-metal system. Shown are results for the benchmarks osu_bw (top left), osu_mbw_mr (top right), osu_bibw (bottom left) and osu_latency (bottom right).

for all platforms: on the bare-metal system we get 12 GB/s for the Intel Omnipath 100 Gbps fabric, 1.8 GB/s (which is slightly short of 16 Gbps) for n1, and 1.2 GB/s for c5. As only one flow is active, the AWS performance is therefore limited to 10 Gbps. osu_bibw shows that all interconnects offer full-duplex transfers, however GCP's n1 instances need larger messages for full-duplex bandwidth. osu_mbw_mr, which runs multiple unidirectional channels through multiple ranks, does not offer an improvement when using Intel OPA. The n1 performance is now at full 16 Gbps for two processes. AWS's c5 peak is 25 Gbps with at most 10 GBps per process, meaning that we require at least 3 process pairs for full bandwidth. For two pairs (not shown), the c5 interconnect achieves exactly 20 Gbps. Finally, osu_latency demonstrates that for message sizes below 16 KiB, both n1 and c5 exhibit 25x higher latencies (25us vs 1us), compared to Intel OPA. However, recall, that we send #modes · #variables · #fused runs · 4 byte / float ≈ 20 KiB for every communicating face, when using order 5 and FP32. Thus, the size of the messages in EDGE, comprised of the communicating tetrahedral faces, are in the MiB range, which shadows the higher latency. Sec. 5 shows in a simple model of a structured setup, that our solver is robust w.r.t. a network, offering 5× less bandwidth than latest supercomputing fabrics. This is due to the face-only stencil of the used high-order DG method.

4.4 Single-Node Application Performance

Finally, after deriving the flop, bandwidth and network performance through micro-benchmarking, we execute a single node scenario of EDGE with FP32, 16

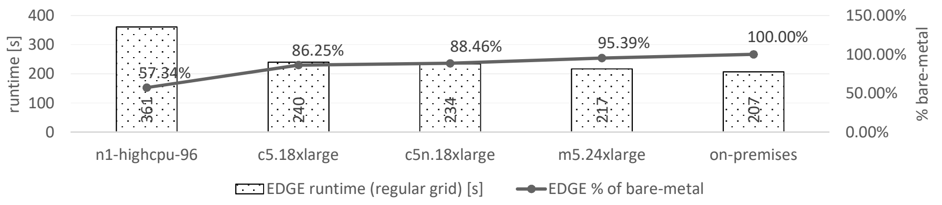

Fig. 6. Runtime of a regular setup of EDGE. As expected all cloud instances are slower than the top-bin bare-metal machine. AWS instances are within 85% of the bare-metal performance, the GCP instance achieves roughly 60% of the bare-metal performance.

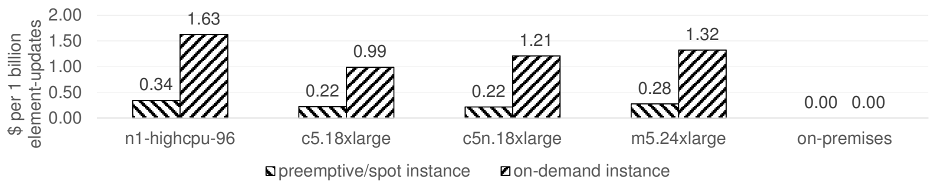

Fig. 7. US-Dollars ($) spent for one billion element-updates, when using preemptive/spot instances (interruptible by the CSP) and on-demand instances (uninterruptible).

fused runs and order 5 ($35 \cdot 9$ degrees of freedom per element and simulation). Based on our micro-benchmarks, we expect, that the c5n and m5 instances reach a performance close to the bare-metal machine. While c5's performance should only be slightly lower, the degradation of GCP's n1 is expected to be most severe. Fig. 6 confirms this estimate. Due to its higher flop performance, the m5 instance can get very close to the bare-metal machine (95%). c5 and c5n stay slightly under 90% of the bare-metal performance. n1 is still able to achieve 57% of the on-premises bare-metal solution with 70% of the flop and 57% of the STREAM performance.

However, when running in the cloud, one should also consider the instance pricing. Today's CSPs offer various types of instances, where we might utilize preemptive/spot instances for short-running or interruptible jobs. On-demand instances are best for uninterruptible long-running jobs. Tab. 1 provides the pricing for both types. We see, that the spot instances offer a huge discount. Fig. 7 sets the measured full-application performance in relation to the instance price. We use the price in dollars per 1 billion element updates as a metric. For a given mesh, number of time steps, and by considering EDGE's scalability (see Sec. 5), this allows us to derive the costs of an execution upfront. Additionally, we are able to derive the most efficient cloud configuration: c5n is most cost-efficient, when using spot instances, whereas c5 leads for on-demand settings. Note, that other factors, e.g., the amount of available memory, the interconnect performance, the storage costs, or the availability of instance types should be considered in a final decision as well.

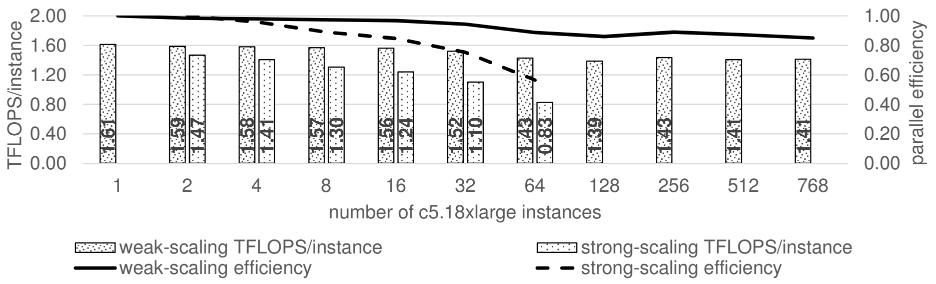

Fig. 8. Weak and strong scalability of EDGE in AWS EC2 on c5.18xlarge instances. We sustained 1.09 PFLOPS in weak-scaling on 768 instances. This elastic high performance cluster contained 27,648 Skylake-SP cores with a peak performance of 5 PFLOPS. The strong scaling setting on 64 instances had a performance of 53 TFLOPS.

5 Elastic Scalability

The previous section shows that, due latest hardware enhancements, single-instance public cloud executions can match the performance, provided by on-premises installations. In order to scale out, novel algorithms are needed to mitigate the 4-5x lower interconnect bandwidth. We address this by employing a high-order DG-solver with a high computation-to-communication ratio. As a side note, even latest on-premises systems can suffer from such an imbalance. A c5.18xlarge instance offers 6.6 TFLOPS with a bandwidth of 25 Gbps. In contrast to this, one node of the GPU-accelerated Summit supercomputer has 96 TFLOPS (also FP32) per node and offers 200 Gbps bandwidth[14]. Therefore, the cloud offers a FLOPS/bandwidth ratio, which is within 2x of Summit (of course the latency on Summit is still $\sim 25\times$ better than in the cloud). This means, in order to run efficiently in the cloud and on Summit, similar approaches for communication avoidance and hiding are needed. In the following, we scale our application to 768 c5.18xlarge instances, having 27,648 Skylake-SP cores with a peak performance of 5 PFLOPS. This is a high instance count, considering that one normally has to wait in queues on a dedicated supercomputer. In all our tests the cloud was able to serve our requests within 2.5 hours. That includes the largest setting, which required booting 768 instances with our custom machine image and registering them on the Slurm-controller.

We carried out two scaling tests, depicted in Fig. 8 (due to space limitations, we focussed on AWS for scaling and used GCP for the presented SGT runs). First, we executed a weak scaling scenario with two processes per instance to reach the benchmarked 20 Gbps bandwidth. Here, we took a regular five-fold subdivided hexahedral mesh with a total of 655,360 tetrahedral elements per instance for 1-512 instances. The 768 instance setting used $384 \cdot 512 \cdot 528 \cdot 5 = 675,840$ elements to provide a proper setting for our dimension-wise $8 \times 8 \times 24$ partitioning in this case. All boundary conditions were periodic, increasing the

[14] https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/

communication footprint. Further, we did not exploit any of the structure, when running our unstructured solver, e.g., stored all adjacency information explicitly.

Taking the 768 instance setting as an example, we obtain $(48 \cdot 64 \cdot 2 + 48 \cdot 22 \cdot 2 + 64 \cdot 22 \cdot 2) \cdot 2 = 22,144$ tetrahedral faces for the six rectangular sides of each partition. For each face, data of the face-adjacent tetrahedrons has to be communicated to neighboring ranks. One y-z-side ($64 \cdot 22 \cdot 2$ faces) neighbors the second rank on the other socket of each instance. Thus, we obtain an uni-directional network-only communication volume of $19.6875\,\text{KiB} \cdot 19,328 \approx 371.6\,\text{MiB}$, when also considering the number of required bytes for the elements' degrees of freedom. With a single-instance average time of 0.81 s per timestep, we can estimate a bandwidth requirement of ∼460 MiB/s on the fabric in one direction and 920 MiB/s full duplex. This is also confirmed by our extensive Scalasca measurements. As AWS EC2 delivers 1.2 GB/s per process, the headroom is sufficient to account for efficiency losses due to congestion with other cloud jobs, using the fabric. This is important, as we cannot influence the instances' placement within the datacenter's placement group, e.g., schedule instances, which are all connected to the same Ethernet switch. For up to 32 instances we measured perfect weak scalability of 95% and for larger cluster sizes a slight decrease, staying above 86%. As comparison, our in-house 32-node Xeon 8180 cluster, connected by an Intel OPA fabric, achieved a parallel efficiency of 98% on all 32 nodes.

Second, in addition to weak-scaling a structured mesh, we ran the SGT scenario of Sec. 2, which includes topography and has an unstructured mesh, comprised of 3,861,780 tetrahedral elements. This setting fits well into 288 GB of memory, provided by two c5.18xlarge instances. In this case, the volume-to-surface ratio shrinks with an increased instance count and we expect the fabric to be limiting at larger scale. Fig. 8 confirms this expectation. We can strong scale by 4× with close-to 90% efficiency. 60% are still possible, when using once again 8× more resources and strong scaling the original problem by 32×. While our elastic cloud cluster delivered 77% parallel efficiency on 32 instances, the 32 nodes bare-metal Intel OPA cluster achieved a parallel efficiency of 90%.

Last but not least, we want to highlight that for neither the weak- nor the strong-scaling case optimal placement in the datacenter or mesh-aware scheduling was exploited to keep results as generalizable as possible.

6 Discussion and Conclusion

This work demonstrates the efficient use of application-centric cloud clusters for modern and tightly-coupled scientific computing. Public cloud services offer elastic multi-petaflops machines, which were four years ago only available through on-premises supercomputing centers. In particular, we examined Google's and Amazon's cloud offerings. For a single instance, we observed a performance, matching that of a competitive on-premises bare-metal system. This is due to two recent hardware advancements: a) offloaded hypervisors and b) full virtualization support in CPUs for high performance VMs, allowing access to the underlying logical CPUs and NUMA domains. From an interconnect point of view, we observed a 4-5× slow-down in terms of bandwidth and ∼25× in terms of latency, compared to latest high-speed and low-latency interconnects. Taking into account the per-node performance of supercomputers, we see that recent cloud offerings are within ∼2× of latest accelerated supercomputers, when it comes to the per-node interconnect bandwidth. Therefore, no matter if one is running on an accelerated supercomputer or in the cloud, a bandwidth-efficient algorithm is needed. This work illustrates the cloud-specific end-to-end optimization of the high-order ADER-DG solver EDGE for seismic wave propagation problems, such as earthquake simulations. Due to EDGE's low communication volume, we were able to achieve 1.09 PFLOPS on 768 c5.18xlarge instances. These instances theoretically offer 5 PFLOPS, resulting in an application peak efficiency of more than 20% and a parallel efficiency of close-to 90% in a weak scaling setup.

This performance can be set into relation to previous work. In [3] a weak-scaling of another ADER-DG solver, SeisSol, is presented. The authors sustained 1.09 FP64 PFLOPS in hadware on the, at this time, Top500 #10-placed SuperMUC, an Intel Sandy Bridge system with 6.2 FP32 PFLOPS peak [21] (today this system is listed #64 in the Nov'18 edition of the Top500 list). The presented, application-relevant non-zero performance in [3] is 750 FP64-TFLOPS. In theory, this could double to 1.5 PFLOPS in a potential FP32 run[15]. Thus, five years later, public cloud clusters with 12 times less nodes and roughly five times less cores can replace a 2013 top 10 system.

This is due to three main factors: First, improvements in hardware (often associated with Moore's Law): instead of an 8-core CPU at 2.6 GHz, having two 256-bit VPUs without FMA, the current work uses an 18-core CPU at ≈3.0 GHz with two 512-bit FMA-VPUs per core. This means, that each Skylake core offers ∼4.5× higher capabilities than Sandy Bridge. Including the core-count, every socket in the cloud cluster is ∼10× more capable than a SuperMUC socket. Second, the efficient elimination of artificial zero-operations in the ADER-DG kernels through fused simulations, combined with runtime code-generation of sparse matrix-tensor kernels through the LIBXSMM library using single precision. And finally, third, an aggressive communication scheme, utilizing application-integrated MPI progression. Only the combination of all three aspects allows EDGE to reach high application-performance in the cloud.

In terms of strong scalability, we scaled a demanding setting from 2 to 64 instances with a parallel efficiency of 60%. This performance is a bit lower, compared to Intel OPA. For such scenarios, AWS announced the so-called EFA network, which provides lower latencies at up to 100 Gbps bandwidth.

Acknowledgements and References

EDGE, EDGEcut and the discussed cloud-related scripts are available under BSD-3 from the linked resources at: http://dial3343.org. We thank David Lenz for his contributions to EDGEcut. We thank the AWS Cloud Credits for Research and Aca-

[15] Verification of FP32 for ADER-DG seismic wave propagation is recent work (see http://doi.org/10.17605/OSF.IO/H9G5N and http://opt.dial3343.org).

demic Google Cloud program. At AWS we thank Walker Stemple, Linda Hedges, Aaron Bucher, Heather Matson, Randy Ridgley and Pierre-Yves Aquilanti for their patient and very helpful support. This work was supported by the Southern California Earthquake Center through award #18211.

1. Alliez, P., et al.: 3D mesh generation. In: CGAL User and Reference Manual (2018)
2. Breuer, A., et al.: Petascale local time stepping for the ader-dg finite element method. In: IPDPS'16
3. Breuer, A., et al.: Sustained petascale performance of seismic simulations with SeisSol on SuperMUC. In: ISC'14
4. Breuer, A., et al.: Edge: Extreme scale fused seismic simulations with the discontinuous galerkin method. In: ISC'17
5. Chen, P., et al.: Full-3D Seismic Waveform Inversion: Theory, Software and Practice. Springer Publishing Company, Incorporated, 1st edn. (2015)
6. Custódio, S., et al.: The 2004 mw6.0 parkfield, earthquake: Inversion of near-source ground motion using multiple data sets. Geophysical Research Letters **32**(23)
7. Deelman, E., et al.: The cost of doing science on the cloud: The montage example. In: SC '08
8. Evangelinos, C., et al.: Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-applications (2008)
9. Geuzaine, C., et al.: Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. Numerical Methods in Engineering **79**(11)
10. Goto, K., et al.: Anatomy of high-performance matrix multiplication. ACM Trans. Math. Softw. (2008)
11. Graves, R., et al.: Cybershake: A physics-based seismic hazard model for southern california. Pure and Applied Geophysics **168**(3), 367–381 (Mar 2011)
12. Heinecke, A., et al.: Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers. In: SC'14
13. Intel: Intel Xeon Processor Scalable Family Specification Update (2018)
14. Jackson, K.R., et al.: Performance analysis of high performance computing applications on the amazon web services cloud. In: CCCTS'10
15. Mauch, V., et al.: High performance cloud computing. Future Generation Computer Systems (2013)
16. McCalpin, J.D.: Hpl and dgemm performance variability on the xeon platinum 8160 processor. In: SC'18. pp. 18:1–18:13. IEEE Press, Piscataway, NJ, USA (2018)
17. Mohammadi, M., et al.: Comparative benchmarking of cloud computing vendors with high performance linpack. In: HPCCC'18
18. Napper, J., et al.: Can cloud computing reach the top500? In: UCHPC-MAW '09
19. Schoeder, S., et al.: Efficient Explicit Time Stepping of High Order Discontinuous Galerkin Schemes for Waves. arXiv e-prints arXiv:1805.03981 (May 2018)
20. Small, P., et al.: The scec unified community velocity model software framework. Seismological Research Letters **88**(6), 1539 (2017)
21. Top500 Authors: Top500 List, November 2013 (2013)
22. Uphoff, C., et al.: Extreme scale multi-physics simulations of the tsunamigenic 2004 sumatra megathrust earthquake. In: SC'17
23. Yvinec, M.: 2D triangulation. In: CGAL User and Reference Manual (2018)
24. Zhao, L., et al.: Strain green's tensors, reciprocity, and their applications to seismic source and structure studies. Bulletin of the Seismological Society of A. **96**(5)