HIGH PERFORMANCE EARTHQUAKE SIMULATIONS

ALEXANDER NIKOLAS BREUER

A RESEARCH STATEMENT

The understanding of earthquake dynamics is greatly supported by highly resolved coupled simulations of the rupture process and seismic wave propagation. This grand challenge of seismic modeling requires an immense amount of supercomputing resources, thus optimal utilization by software is imperative. Driven by recent hardware developments, the increasing demand for parallelism and data locality often requires replacing major software parts to bring efficient numerics and machine utilization closely together.

B CONTRIBUTIONS

In my doctoral research project I develop a new computational core for the seismic simulation package SeisSol\(^1\). For minimal time-to-solution the new core is designed to maximize value and throughput of the floating point operations performed in the underlying ADER discontinuous Galerkin discretization method. My computational core covers the entire memory layout of the computational data structures, auto-tuned sparse and dense matrix kernels, hybrid parallelization from many-core nodes up to machine-size and a novel high performance clustered local time stepping scheme.

SeisSol uses static, adaptive, unstructured tetrahedral meshes. For that reasons the memory access patterns are known at initialization. My computational core performs a strict sorting of all global and element-local data structures and thus embeds the structure of my clustered local time stepping scheme already in the memory layout. This maximizes linear data access in the computational main loops and thus greatly supports hardware prefetching. For vector-loads my memory layout supports arbitrary vector-alignment by using zero-padding for all relevant data structures in single- and double-precision arithmetic. I am featuring near-memory accesses in multi- and many-core parallelism by replicating the computational shared memory parallelization in my NUMA-aware initialization. Further my memory layout introduced a strict splitting of interior, copy and ghost cells into SeisSol. Due to my clustered local time stepping scheme, this layout not only features sending and receiving data to a single neighboring partition, but also enables the crucial communication to multiple neighboring, local time stepping clusters in the same partition. The duplication of corner-case elements in the copy and ghost layer, having more than one communication neighbor, allows me to send and receive the data as is without any additional communication buffers.

On a single-core my computational core uses highly specialized matrix-kernels for the limited number of low-rank discontinuous Galerkin matrix-matrix operators. The generation of matrix kernels is developed in collaboration with A. Heinecke at Intel Labs and was recently adopted into the libxsmm-library\(^3\). Here I am feeding all discretization- and architecture-specific information, such as the target vector instruction set, leading dimen-

\(^1\)https://github.com/SeisSol/SeisSol
\(^2\)https://github.com/TUM-I5/seissol_kernels/
\(^3\)https://github.com/hfp/libxsmm
sions or matrix dimensions, to the library and generate respective sparse or dense matrix-matrix multiplication kernels. Additionally, I performed auto-tuning via dry runs for WSM-EP, SNB-EP, HSW-EP and KNC to decide whether sparse- or dense-computations are beneficial in terms of time-to-solution. The result is a set of matrix kernels for all relevant execution modes of SeisSol, which I hardwire via function pointers to my ADER-DG kernels.

For multi- and many-core parallelism I am using OpenMP in my computational loops. Due to the careful setup of the data structures the implementation overhead of OpenMP is low, but scales perfectly.

Distributed memory parallelism in my computational core is realized via the Message Passing Interface (MPI). Here, I am using asynchronous communication, which is attached to a single core for MPI-progression. Information on the current status of the communication between the dedicated communication core and the remaining cores is shared via volatile signaling variables.

My clustered local time stepping scheme summarizes tetrahedrons with similar time steps together and processes them together in a single cluster. The spatial distribution of a single cluster’s elements is allowed to be arbitrary, thus no connectivity of elements in a cluster is required. This design is crucial for unstructured meshes, which tend to have very heterogeneous time step distributions. In my time management scheme I am resolving dependencies of the clusters in time and schedule the individual work-items for processing by the computational cores. The time management prioritizes critical work, which is communication-related data and small time step clusters. Additionally, whenever a work-item in the copy layer is not able to proceed, because of pending communication requests, my core immediately continues with independent work. In summary this scheme maximizes the overlap of computation and communication.

C RESULTS

My computational core reduces time-to-solution of SeisSol by several factors and scales beyond 1 million cores. At machine-size the new core enabled a landmark simulation of the Landers 1992 earthquake. For the first time this simulation allowed the analysis of the complex rupture behavior resulting from the non-linear interaction of frictional sliding and seismic wave propagation at high geometric complexity.

Fig. 4 shows the single-node performance of my computational core on a 12-core WSM-EP-node, a 16-core SNB-EP-node, a 36-core HSW-EP-node and a 60-core KNC-card. The diagram shows the double-precision GFLOPS in hardware (light bars) and in terms of non-zero operations (dark bars) for convergence rates 2-7 of the ADER-DG scheme. All results were measured simulating the LOH.1 benchmark and by using a single constant time step in my clustered local time stepping scheme. The key finding of the results
Figure 2: Illustration of two local time stepping clusters in a Mount Merapi simulation. Colors denote different MPI-partitions. The first cluster covers the time step interval \([2\Delta t, 4\Delta t)\) and resolves regions with heavy adaptive mesh refinement. The second cluster covers the time step interval \([4\Delta t, 8\Delta t)\) and almost exclusively covers the interior of the volcano.

Figure 3: Seismic wave-propagation of the large-scale Mount Merapi simulation on SuperMUC-2 after two seconds of simulated time. Warmer colors denote higher wave field energy.

in Fig.\[1\] is the almost optimal node-utilization (\(\approx 60\%\) of theoretical peak) for high-order simulations, which holds although my clustered local time stepping scheme is used.

Fig.\[2\] shows two example clusters of my LTS clustering applied to a simulation of seismic wave propagation in Mount Merapi. Especially the first cluster contains splitter elements, which are not connected to the remaining elements and thus underline the flexibility of my local time stepping scheme.

In Breuer et al. [2014] we presented SeisSol’s first petascale simulations of seismic wave propagation in Mount Merapi using all 147,456 SNB-EP cores of SuperMUC-1. These runs used sixth order of convergence, a discretization with 99 million elements and utilized an early global time stepping version of my computational core. Due to this version of the core we were able to speedup SeisSol by a factor of \(5 \times - 10 \times\) depending on the core counts, due to the less-optimal scaling of the original implementation. The SuperMUC system\[4\] was recently extended with a 86,016 core HSW-EP machine. SuperMUC-2 delivers approximately the same HPL-performance (2.8 PFLOPS) as the SNB-EP system. As part of the early access period of SuperMUC’s second phase, I repeated our Mount Merapi simulation. However this time I used all of my computational core’s features, including local time stepping, the newly tuned HSW-EP matrix-kernels and my new asynchronous communication scheme. In contrast to the 2014 simulation, this run also featured wave-field output (work of S. Rettenberger, TUM).

\[4\]https://www.lrz.de/services/compute/supermuc/systemdescription/
The simulation sustained petascale performance for 63 minutes on all cores of SuperMUC-2 and achieved a total time of 10 simulated seconds. The results of this simulation are remarkable and show that my new core is not only able to sustain the petascale performance of the 2014 global time stepping scheme, but also increases the value per FLOP greatly. Comparing the time-to-solution of both simulations, my full-blown computational core achieves an additional speedup of $5.6 \times$ compared to the version of 2014. This result translates to a total speedup of more than $25 \times$ compared to the original implementation and summarizes the scientific outcome of my doctoral project.

REFERENCES


