Parallel Computing 90 (2019) 102564

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Review

Student Cluster Competition 2018, Team Nanyang Technological University: Reproducing performance of a Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake on the Intel Skylake architecture

Bian Wu[1,*], Weiliang Heng[1], Bu-Sung Lee

School of Computer Science and Engineering, Nanyang Technological University, Block N4, 50 Nanyang Avenue, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 12 April 2019
Revised 24 September 2019
Accepted 1 October 2019
Available online 14 October 2019

Keywords:
Reproducibility
SeisSol
Local time stepping
Earthquake simulation

ABSTRACT

Uphoff et al. (2017) presented a Multi-Physics Simulation of the Tsunamigenic 2004 Sumatra Megathrust Earthquake and described an optimization of the simulation code SeisSol Introduction. As a part of the Student Cluster Competition 2018, we review their paper and verify the claimed performance on our Intel Skylake architecture. In particular, we reproduce the single node performance and speedup of the Shaking Corals version of SeisSol compared with the baseline version, the strong scaling test of local and global time stepping, and the simulation of the 2004 Sumatra earthquake. Our reproduced results showed that the Shaking Corals version achieved a speedup of 1.31 and up to 51% of peak performance on the Intel Skylake architecture.

1. Introduction

SeisSol is one of the leading software packages for earthquake simulation, which solves seismic wave equations using Arbitrary high-order accurate DERivative Discontinuous Galerkin (ADER-DG) method [6]. In recent years, continuous efforts have been put on the optimization of SeisSol to achieve petascale performance on supercomputing architectures, which make it possible to conduct large scale seismic simulations with high resolution and capture small-scale features [3,4,6,9,10]. High-resolution seismic simulations not only help researchers to understand the complex earthquake faulting process but also help to forecast seafloor displacement and ground motions in possible earthquake scenarios. A SC17 paper entitled "Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake" (the SeisSol paper) presented a 500 s–1500 km physics based earthquake simulation, including the first ever dynamic rupture modeling of the 2004 Sumatra earthquake [10]. To deal with the extreme scale of this scenario, the SeisSol paper introduced a series of optimization on SeisSol, including a new cache-aware wave propagation scheme, an optimized code generation process of the dynamic rupture kernels, a clustered local-time-stepping scheme for petascale high performance, and an asynchronous output handling method for large checkpoint and visualization data. The optimized version of SeisSol is referred as Shaking Corals (SC) in the SeisSol paper and so in this paper.

There are three claims from the SeisSol paper that we aim to reproduce. First, the author claimed that for single node performance, the SC version gave a speed-up of 1.25 for discretization order 5 and 1.55 for order 7 with up to 56% of peak performance on HSW (Intel Xeon E5-2697 v3, at 2.2 GHz). Meanwhile, the SC version gave a speed-up of 1.14 for order 6 and 1.55 for order 7 with up to 50% of peak performance on KNL (Intel Xeon Phi 7250, at 1.2 GHz). Second, the author claimed that by adopting local time stepping (LST), the theoretical speed-up of SC version achieved 9.9 compared to global time stepping (GTS). In the test of 220,993,734 elements on Haswell, LTS yielded an extrapolated speedup of 6.8 compared to GTS. In the test of 51,020,237 elements on Knights Landing, LTS yielded an extrapolated speedup of 7.5 compared to GTS. Third, the author claimed that the simulation of the 2004 Sumatra earthquake reproduced the main observed characteristics of the real event, including its magnitude, fault slip distribution, and seafloor displacements.

We consider the SeisSol paper reproducible on the Intel Skylake architecture if we are able to repeat the experiments on our cluster and get results that match the claims mentioned above. All reproduced experiments were carried out during the Student Cluster

* Corresponding author.
E-mail addresses: bwu007@e.ntu.edu.sg (B. Wu), heng0181@e.ntu.edu.sg (W. Heng), ebslee@ntu.edu.sg (B.-S. Lee).
[1] These authors contributed equally.

https://doi.org/10.1016/j.parco.2019.102564
0167-8191/

Table 1
Hardware and software configuration.

CPU	Intel Xeon Gold 6148 (20 cores) * 2 Hyper-Threading disabled, 32 KB per core L1d/i, 1 MB per core L2, 1.375 MB per core L3
Memory	256 GB 2133 MHz DDR4
Storage	480 GB SSD
Vendor	Supermicro
OS	CentOS 7.3
Kernel	3.10.0-862-el7
Compiler	Intel Compiler 2017
Libraries	HDF5 1.8.11 [7], NetCDF 4.4.1.1 [8], zlib 1.2.11, CMake 3.9.6, metis 5.1.0 [1], parmetis 4.0.3 [2]

Competition at SC18 and the experiment datasets were provided by the competition committee.

2. Environment setup

Table 1 shows the hardware and software of our cluster, which are both different from the original experimental setup in the SeisSol paper [10].

The source code of SeisSol is cloned from https://github.com/SeisSol/SeisSol. We use the commit *fb97beff* as the SC version and the commit *8ff804b* as the baseline version (BL). The input dataset is provided by the competition committee where the mesh is originally partitioned into 4 sub-meshes for running on a 4-node architecture. Since we only have one node for the reproducibility experiments, we repartition the input file into one mesh using the following command:

```
pumgen -s netcdf input.4.nc 1 input.1.nc
```

Furthermore, we change the *EndTime* in the parameter file to 8.2551808 for the comparison between local and global time stepping in Section 4.2. The parameter file for the simulation in Section 4.3 is obtained from https://dx.doi.org/10.5281/zenodo.439946. To take advantage of non-uniform memory access (NUMA), we use the command line utility *numactl* supported by the Linux NUMA library to pin processes to cores that belong to the same socket when 20 cores or less are used.

3. Compilation/run description

Before compilation, we first migrate the source code of SeisSol to adapt to our Intel Skylake CPU architecture. To do this, there are multiple files that need to be modified. First, we edit the `arch.py` file under `site_scons` by adding the architecture `skx`, setting the alignment to 64, and adding the compilation flag `-xCORE-AVX512`. Next, we edit `Arch.py` under `auto_tuning/scripts/tune/gemmgen`, making sure that the alignment for skx is also 64 here and setting `enablePrefetch` to False. After that, we edit `precision.h` to make sure that both single precision skx and double precision skx are defined. For `buildVariablesFile`, below is what we set:

```
compileMode='release'
parallelization='hybrid'
commThread='yes'
arch='dskx'
generatedKernels='yes'
netcdf='yes'
netcdfDir='/path/to/netcdf'
hdf5='yes'
hdf5Dir='/path/to/hdf5'
logLevel='info'
```

The final command we use to build the executable file is presented below. Note that the best `memLayout` is chosen based on auto-tuning results.

```
scons -j 44 order=6
memLayout=skx/order6/tuned_layout.xml
buildVariablesFile=variable.py
```

For the SC version, we write a script to build, run, and analyze the tuning result. Once the tuning result is obtained, we take the optimal memory layout for a particular order. In this case, order equals 6. We then build the proxy application based on the previously obtained memory layout using the following command:

```
scons equations=elastic order=6 arch=dskx
memLayout=skx/order6/tuned_layout.xml
```

For the BL version, we use the proxy application from `libxsmm` and the SeisSol kernel for proxy is downloaded from https://github.com/hfp/libxsmm/tree/master/samples/seissol. Due to time constraints and limited computing resources we had during the competition, we did not run the auto-tuning scripts for the BL version, which take more than 12 h to run. Instead, we directly run `proxy_launch.sh`. In order to support Skylake architecture, we also edit `proxy_launcher.sh` such that alignment is set to 64 and compilation flag is set to `-xCORE-AVX512`. Also, alignment for Skylake need to be inserted into `Configuration.py` under `seissol_kernels/preprocessing/scripts/tools`. Moreover, the default matrices for Skylake named `all_dense`, `all_sparse`, and `sparse_dense` are added under `seissol_kernels/preprocessing`.

4. Experiments

4.1. Single node performance and speed-up

In the SeisSol paper, the wave propagation scheme was optimized by decomposing the flux matrix such that only 11 smaller matrices need to be stored, taking 165 KB of cache space [10]. In comparison, 48 full flux matrices needed to be stored taking 2.6 MB cache space before decomposition [10]. This improvement is especially friendly to Knights Landing (KNL) architecture, since its L2 cache is 1 MB, which is big enough to hold the 11 small matrices and can avoid the cache misses that will constantly happen when 48 full flux matrices are required. On the Intel Skylake (SKX) architecture, where both L1 and L2 cache are of the same size as KNL, a similar improvement is expected to be contributed by flux matrix decomposition.

The SeisSol paper used a proxy application to measure the single node performance and the speedup of the SC version compared to the BL version [10]. The original experiment was conducted for orders 4, 5, 6 and 7 using a mesh with 100,000 elements on both Haswell (HSW) and KNL architectures. The number of time steps that proxy ran was not mentioned in the paper. It was reported that SC achieved a speedup of 1.25 for order 6 and 1.55 for order 7 on HSW [10]. On KNL, SC achieved a speedup of 1.14 and 1.46, respectively [10]. Moreover, SC achieved up to 56% of peak performance on HSW (at 2.2 GHz) and up to 50% of peak performance on KNL (at 1.2 GHz) [10]. The method of calculating machine usage was not explicitly stated in the SeisSol paper. We speculate that the machine usage is the maximum GFLOPS of SC divided by the theoretical peak performance at stated frequency, which matches the claimed percentages [10]. We use the same method to calculate the machine usage in our experiment.

We reproduce the experiment on SKX using the same number of elements (100,000) and we set the time step to 100. Fig. 1

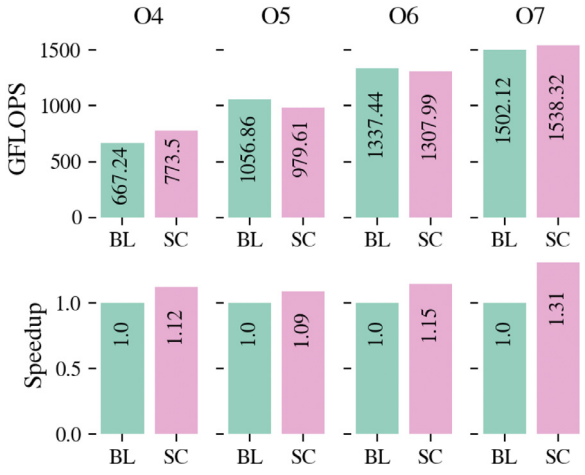

Fig. 1. Single node performance and speed-up of SC compared to BL for wave propagation on a mesh with 100,000 elements on the Skylake architecture.

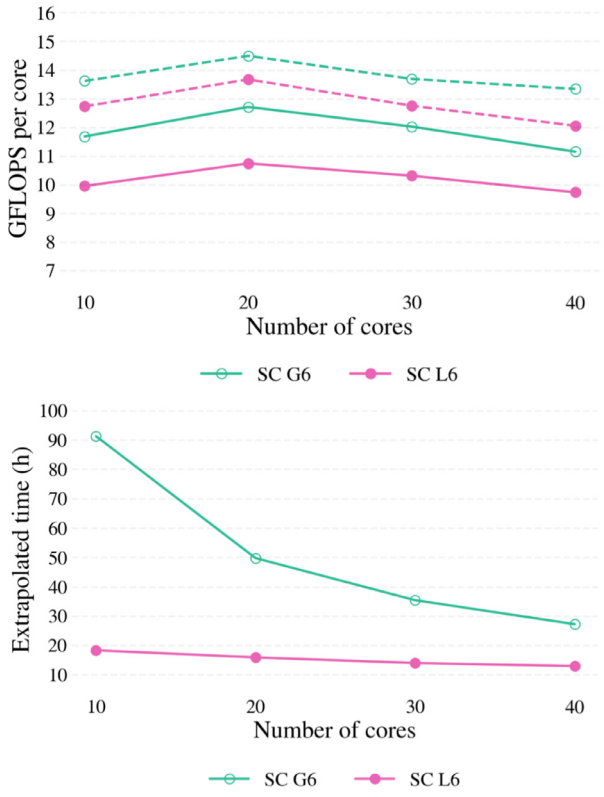

Fig. 2. Scalability test of the baseline (BL) version and the Shaking Corals (SC) version by core on the Skylake architecture. Test cases are local time stepping (L) and global time stepping (G) with order 6. In the top plot, the solid lines represent flops only in the wave propagation whereas the dotted lines includes flops in both wave propagation and dynamic rupture. The bottom plot shows the extrapolated time for a full simulation.

presents our reproduced single node performance and speedup of SC compared to BL. Note that the results of SC are based on auto-tuned memory layout, whereas the BL are based on default memory layout. According to our experience in preparing for the competition, the BL version only obtains negligible benefits from the auto-tuning of memory layout. Hence, we consider our results comparable with the results in the SeisSol paper. On SKX, SC is 1.15 times faster than BL for order 6 and 1.31 times faster for order 7. The speedup of SC on SKX matches that on HSW and KNL [10]. As for machine usages in our experiment, we calculate it by divide the maximum GFLOPS of SC with the theoretical peak performance of SKX (at 2.4 GHz). SC achieves up to 51% of peak performance on SKX, which in line with the single node performance tested on HSW and KNL [10].

4.2. Local v.s. global time stepping

Local time stepping (LTS) is an update scheme that allows each element to run its own optimal time step length and as a result computation time can be reduced. On the contrary, global time stepping (GTS) sets the time step length of all elements in the mesh to the same. To take advantage of hardware-oriented data structure and efficient load balancing methods, which are necessary for high performance and good scalability, a previously implemented clustered LTS scheme is used for wave propagation and a new clustered LTS scheme for dynamic rupture is introduced in the SeisSol paper [10]. Additionally, load-balancing is achieved by weighing each time cluster by its relative update rate [10]. In particular, time cluster c has a weight of $2^{c_{\max}-c}$, where $c_{\max}$ is the largest time cluster [10].

To show that the clustered LTS scheme is strongly scalable up to petascale, a scalability test is performed. The original experiment in the SeisSol paper simulates 220,993,734 elements on HSW and 51,020,237 elements KNL with 11 time clusters and in each cluster 5 to 5120 local time steps [10]. We reproduce the experiment on SKX using a much smaller mesh of 481,565 elements with the same number of time clusters and the same local time steps. The results of our reproduced results is represented in Fig. 2, where G6 represents global time stepping with order 6 and L6 represents local time stepping with order 6.

As the top plot of Fig. 2 shows, GFLOPS per code stays fairly constant as the number of cores increases, while a minor drop is observed when we scale from 20 cores to 30 cores. The performance drop can be explained by non-uniform memory access (NUMA), where a processor have its own local memory and it is faster to access the local memory than non-local memory [5]. Since we have two sockets with each socket consists of 20 cores, to benefit from the NUMA architecture, we bind process within one socket when 20 cores or less are used. When we scale to more than 20 cores, both sockets are used and GFLOPS decreasing is expected due to across-socket memory access. Performance drop continues as we further increase the number of cores, since more cross-socket memory access occurs. Although our reproduced results have a similar plot as the original scalability test in the SeisSol paper [10], the main reason for the performance drop in the original scalability test is the communication overheads when scaling across nodes.

The bottom plot of Fig. 2 shows the extrapolated time for a full simulation, which is obtained by multiplying the measured time for one time step with the number of time steps. In the original scalability test, the extrapolated time for LST and GST both decreased linearly with respect to the number of nodes. In our reproduced scalability test, the extrapolated time to solution for LST decreases linearly with respect to the number of cores involved in the computation. However, a sharp decrease is observed for GST when we scale from 10 cores to 20 cores. This might indicate that global time stepping does not perform well when we limit the number of cores available for computation.

4.3. Geophysical interpretation of simulation result

The original simulation of the 2004 Sumatra earthquake in the SeisSol paper was performed on a mesh with 221 million elements and order 6 accuracy in space and time. The local time step was up to 3.3 million, increasing by a factor of 1024 from the smallest

time cluster. The author claimed that the simulated result matches well with the main observational characteristics of the real events.

Due to the limited time of the Student Cluster Competition and constrained computing power of our single-node machine, the simulation is not reproducible on our cluster. The 500 s simulation takes 10 h to run on our cluster and hence we are not able to finish the simulation during the competition after our first run was unstable due to improper parameter setup.

5. Conclusion

In conclusion, the speedup, single node performance, scalability, and the difference in local time stepping and global time stepping of the SC version are successfully reproduced on the Intel Skylake architecture. We are able to reproduce the speedup and single node performance experiments because the Intel Skylake architecture has the same L1 and L2 cache size as KNL. The strong scalability test is reproduced on our machine because the SeisSol application is built and optimized for parallel HPC infrastructures and is scalable on various architectures including the Intel Skylake architecture. Due to the limitation of time and computing power, we are not able to reproduce the full simulation in the competition, hence cannot verify the SeisSol paper's claim on geophysical interpretation.

Declaration of Competing Interest

The authors would like to declare that we have no conflict of interest with other people or organizations.

References

[1] Karypis Lab, METIS – Serial Graph Partitioning and Fill-reducing Matrix Ordering, (http://glaros.dtc.umn.edu/gkhome/metis/metis/downloada). (2018).
[2] Karypis Lab, ParMETIS – Parallel Graph Partitioning and Fill-reducing Matrix Ordering, (http://glaros.dtc.umn.edu/gkhome/metis/parmetis/downloadb). (2018).
[3] A. Breuer, A. Heinecke, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties, Sustained petascale performance of seismic simulations with SeisSol on SuperMUC, in: Proceedings of the 2014 International Supercomputing Conference, Springer, 2014, pp. 1–18.
[4] 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, SC'14, IEEE, 2014, pp. 3–14.
[5] C. Lameter, Numa (non-uniform memory access): an overview, Queue 11 (7) (2013) 40.
[6] C. PELTIES, Accelerating SeisSol by generating vectorized code for sparse matrix operators, Parallel Comput. Accel. Comput. Sci. Eng. (CSE) 25 (2014) 347.
[7] The HDF Group, HDF5, (http://www.hdfgroup.org/HDF5). (2018).
[8] UCAR Unidata Program, Network Common Data Form (NetCDF), (http://doi.org/10.5065/D6H70CW6). (2018).
[9] C. Uphoff, M. Bader, Generating high performance matrix kernels for earthquake simulations with viscoelastic attenuation, in: Proceedings of the 2016 International Conference on High Performance Computing & Simulation (HPCS), IEEE, 2016, pp. 908–916.
[10] C. Uphoff, S. Rettenberger, M. Bader, E.H. Madden, T. Ulrich, S. Wollherr, A.-A. Gabriel, Extreme scale multi-physics simulations of the tsunamigenic 2004 sumatra megathrust earthquake, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC'17, ACM, New York, NY, USA, 2017, pp. 21:1–21:16, doi:10.1145/3126908.3126948. http://doi.acm.org/10.1145/3126908.3126948.