Parallel Computing 90 (2019) 102568

Contents lists available at ScienceDirect

Parallel Computing

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

Review

Student cluster competition 2018, team northeastern university: Reproducing performance of a multi-physics simulations of the Tsunamigenic 2004 Sumatra Megathrust earthquake on the AMD EPYC 7551 architecture

C. Bunn*, H. Barclay, A. Lazarev, F. Yusuf, J. Fitch, J. Booth, K. Shivdikar, D. Kaeli

Department of Electrical and Computer Engineering Northeastern University Boston, MA 02115, United States

ARTICLE INFO

Article history:
Received 12 April 2019
Revised 16 September 2019
Accepted 1 October 2019
Available online 18 October 2019

Keywords:
Reproducibility
Earthquake simulation
AMD EPYC
LIBXSMM

ABSTRACT

This paper evaluates the reproducibility of a Supercomputing 17 paper titled Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake. We evaluate reproducibility on a significantly smaller computer system than used in the original work. We found that we able to demonstrate reproducibility of the multi-physics simulations on a single-node system, as well as confirm multi-node scaling. However, reproducibility of the visual and geophysical simulation results were inconclusive due to issues related to input parameters provided to our model. The SC 17 paper provided results for both CPU-based simulations as well as Xeon Phi based simulations. Since our cluster uses NVIDIA V100s for acceleration, we are only able to assess the CPU-based results in terms of reproducibility.

1. Introduction

In the SC17 paper titled Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake [1], SeisSol [2] is used to simulate a dynamic rupture scenario of the 2004 Sumatra earthquake. To simulate an earthquake of this magnitude, SeisSol required end-to-end optimization in order to scale well on a highly parallel computer architecture. The SC17 paper evaluates the performance of the resulting implementation, called Shaking Corals.

The performance enhancements provided by Shaking Corals over previous versions of the simulation can be attributed to a number of optimizations applied to the ADER-DG scheme used in the baseline SeisSol version for solving the seismic wave equation. Uphoff et al. [1] modify the scheme by expanding large matrices appearing in wave-propagation terms into chain products of smaller matrices, which avoids cache evictions and reduces the total required number of operations. The Shaking Corals implementation also adds support for local time stepping (LTS) during simulations with dynamic rupture, allowing further speed-up over the improvements to wave-propagation kernels already mentioned.

This reproducibility report evaluates SeisSol's performance on a smaller computer system using two different performance metrics: 1) single node performance, and 2) performance scaling. In addition, the geophysical output data from our simulation is compared to the results presented by Uphoff et al. [1]. We evaluate SeisSol based on these performance metrics and accuracy dimensions to assess reproducibility of the results appearing in the original SC17 paper.

2. Background

Faced with the challenge of predicting earthquakes reliably, the capacity to model earthquake dynamics with large-scale simulations is highly desirable in order to identify likely earthquake zones. With the recent growing interest in earthquake simulation software, the need for rigorous optimization and petascale performance in realistic, large-scale simulations has become increasingly apparent.

Uphoff et al. [1] present a simulation of the 2004 Sumatra-Andaman earthquake. This scale of simulation was only made possible through end-to-end optimization of the SeisSol simulator, called Shaking Corals (SC). Shaking Corals uses the ADER-DG scheme to perform numerical simulation of seismic wave propagation and dynamic rupture, characterized by a compute-bound

* Corresponding author.
E-mail addresses: bunn.c@husky.neu.edu (C. Bunn), barclay.h@husky.neu.edu (H. Barclay), lazarev.a@husky.neu.edu (A. Lazarev), yusuf.f@husky.neu.edu (F. Yusuf), fitch.j@husky.neu.edu (J. Fitch), booth.j@husky.neu.edu (J. Booth), shivdikar.k@husky.neu.edu (K. Shivdikar), kaeli@ece.neu.edu (D. Kaeli).

https://doi.org/10.1016/j.parco.2019.102568
0167-8191/

Table 1
Cluster specifications.

Hardware	Description
Number of Nodes	4 Nodes
Node Model	PowerEdge R725
Vendor	Dell EMC
CPU Model	AMD EPYC 7551
Cores per CPU	32
CPUs per Node	2
Memory per Node	512 GB of DDR4 RAM
Storage per Node	240 GB SATA SSD
Interconnect	Mellanox Infiniband EDR
Operating System	Ubuntu 18.04 LTS

process that is well-suited to massively-parallel implementations. Uphoff et al. [1] detail several optimization strategies used in their implementation and achieve an overall speed-up of 13.6x over the base version.

Uphoff et al. [1] implement a clustered local time stepping (LTS) scheme for both wave propagation and dynamic rupture elements, sacrificing theoretical single-node performance for scalability and performance on supercomputers. Without LTS, the costliest element bounds the performance of other elements, since they all share a global time step; LTS ensures that elements only share time steps with nearby or adjacent elements. They also target increased per-node performance by optimizing matrix multiplication kernels in both the wave-propagation and dynamic rupture models, decomposing large matrices into products of smaller ones to reduce the total number of operations and reduce cache evictions. This optimization also reduced the size of intermediate values, reducing the overall space complexity as a side effect. The Shaking Corals version of SeisSol achieves a significant single-node performance increase relative to the base version and scales well across many nodes.

In order to evaluate the performance of this application, we utilize a 4-node cluster with specifications provided in Table 1. Each node contains a Dell EMC PowerEdge R725 equipped with two 32-core AMD EPYC 7551 processors, 512GB DDR4 RAM, and a 240 GB SATA SSD. Each AMD EPYC 7551 processor has 2 MB of L1I cache, 1MB of L1D cache, 16MB of L2 cache, and 64 MB of L3 cache. Compared to the Intel Xeon E5-2697 v3 used by Uphoff et al. [1], the AMD EPYC processors used have a much larger cache size at all levels. While the large number of cores provided by the AMD processors is advantageous, the platform choice also increased the complexity of compiling the application, as the original application targeted Intel-based platforms. A Mellanox Infiniband EDR interconnect provides node-to-node connectivity, allowing efficient MPI communication between nodes. Each node in the cluster hosts a minimal installation of Ubuntu 18.04 LTS. SeisSol was compiled and run within Docker containers, ensuring a high degree of consistency in runtime environment. The Dockerfile used to build the container is available on GitHub[1]. In order to run, the command `docker build.` was run in the same directory as the Dockerfile. In the Dockerfile, the build process is executed inside of the container. Once SeisSol is built inside of the container, it runs interactively by using the command `docker run -v /host/directory/:/container/directory --network=''host'' /bin/bash/`. From here, the generated binary runs across multiple nodes using `mpiexec`.

3. Compilation and runtime environment

As discussed in Section 2, the cluster used to execute this application is based on an AMD CPU platform with NVIDIA GPU accelerators, but the application originally targeted Intel-based CPU platforms and Intel-based Xeon Phi accelerators. This difference introduced some potential challenges in terms of reproducibility, as much of SeisSol is built upon Intel-specific libraries. SeisSol was compiled using settings for Intel's Haswell architecture on our AMD EPYC-based cluster.

Table 2
Cluster specifications.

Dependency	Version
SCons	2.2.0
HDF5	1.8.11
NetCDF	4.4.1.1
LIBXSMM	1.9
Metis	5.1.0
ParMetis	4.0.3

This reproducibility study focuses on the evaluation of the Shaking Corals [3] release of SeisSol. In order to accurately reproduce the results report by Uphoff et al. [1], the same libraries and build tools listed in the project documentation at the time of publication were used to compile on this system. The same version of each library or tool listed in the SeisSol documentation [4] was used during compilation given in Table 2.

To compile SeisSol on our system, SCons 2.2.0 [5] was used to manage the build process. Because Uphoff et al. [1] did not specify a specific version of the compiler to use, version 7.3.0 of g++ was used. As SeisSol employs the HDF5 format for checkpointing and writing results, HDF5 1.8.11 [6] was linked to SeisSol during the build process. NetCDF 4.4.1.1 [7] was linked to initialize large unstructured meshes. LIBXSMM 1.9 [8] was used to generate optimized assembly kernels for matrix-matrix multiplication. Finally, Metis 5.1.0 [9] and ParMetis 4.0.3 [10] were used to enable support for the PUML mesh format across multiple nodes.

An example of a reproducibility challenge was the use of the LIBXSMM library [8] to perform a majority of the small matrix multiplications in the application. LIBXSMM is described as an Intel-specific library [8], whereas our cluster had only AMD CPUs. However, LIBXSMM utilizes common SIMD extenetsions (e.g., AVX2 support) which are common on both the AMD EPYC and Intel Haswell architectures. Given this commonality between the two instruction set architectures (ISA), the generated executable was able to take advantage of the hardware-based SIMD support for AVX vector operations that were previously only available on the Intel ISA. While OpenBLAS [11] performed better on our platform, LIBXSMM was used to more closely match the runtime environment used in the original paper.

We did not evaluate the scalability of the KNL implementation as part of this reproducibility exercise. Much of the optimizations for the Intel Xeon Phi Knight's Landing (KNL) implementation of SeisSol is specific to Intel hardware and software. This would require substantial porting efforts of the existing KNL implementation to run on our competition cluster comprised of AMD CPUs and NVIDIA GPUs.

4. Shaking corals proxy performance

Uphoff et al. [1] reported the single node performance of the generated flux kernels using a proxy program. This program generates a random initial dataset and uses the same kernels generated in SeisSol to measure performance. In this work the Shaking Corals version of the proxy is utilized to collect performance results. The Baseline version of this proxy program was not run during the competition due to the limited time available to run all of the programs. Instead, the proxy program for Shaking Corals was

[1] https://github.com/christopherbunn/SeisSol-SC18.

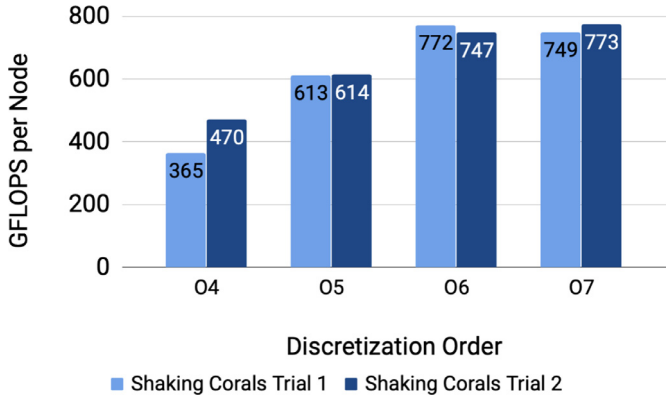

Fig. 1. Shaking Corals proxy application performance across two trials and four discretization orders (GFLOPS/node).

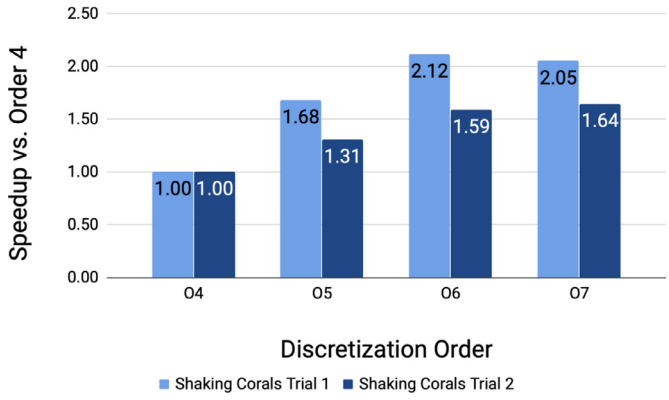

Fig. 2. Shaking Corals proxy application performance comparison (Speedup).

run twice to observe variability across runs. Double precision is used, and the Haswell memory layout and flags are used. As the experimental setup does not include Xeon Phi accelerators, only the CPU implementation is measured.

An SCons build script was used to create multiple binaries with different discretization orders. A total of 4 separate binaries were created with orders 4, 5, 6, and 7. Fig. 1 shows the resulting performance per node for the 4 different orders across the two different runs of each binary. The command `./seissol_proxy 100000 100 all` was used to run this application, where `./seissol_proxy` indicates the name of the proxy binary, 100000 refers to the number of elements in the test mesh, and 100 refers to the number of meshes. The results for discretization order 4 show there was a large performance variability between runs. This is due to the additional time it takes to load the data into the CPU caches on the first run, as well as the overhead associated with initializing related libraries.

Fig. 2 shows the speedup at each order, using the performance of Order 4 as a baseline. Similar to the results obtained by Uphoff et al. [1], there is a slight drop off in GFLOPS observed between orders 6 and 7. On our cluster, the results obtained show that Order 6 achieves the highest performance, with a speedup of 2.12 × as compared to the performance of Order 4.

Order 6 seems to provide the best balance of workload size, which corresponds with the results observed by Uphoff et al. [1] Our cluster was able to achieve higher performance in terms of GFLOPS/node than the Haswell-based cluster used in the original study, which is likely due to the higher core count and higher clock speeds of our AMD EPYC-based cluster nodes.

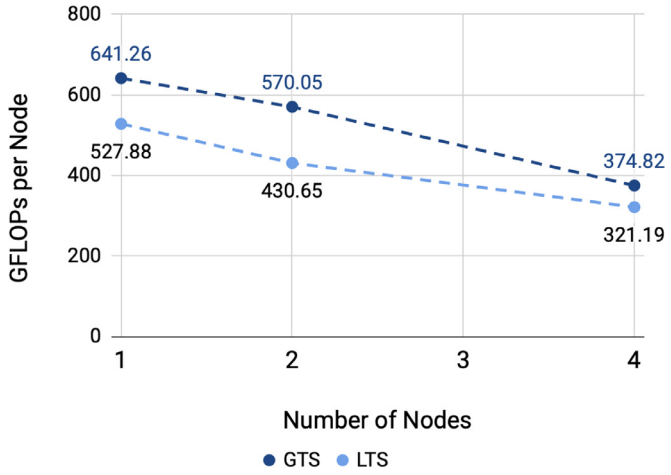

Fig. 3. A Comparison of Local Time Stepping and Global Time Stepping Performance Using GFLOPs per Node.

5. Local vs. global time stepping

In the Shaking Corals [3] release of SeisSol, local time stepping (LTS) is implemented for dynamic rupture to enable increased performance and scalability on multi-node clusters. Compared to the previous global time stepping (GTS) implementation, local time stepping allows for more efficient load balancing across nodes at the expense of lower single-node performance. A smaller input mesh was used to obtain performance results for LTS and GTS.

In our tests, we ran both local and global time stepping implementations on 1, 2, and 4 nodes presented in Fig. 3. Global time stepping resulted in better performance than local time stepping, with the performance gains diminishing as the number of nodes is increased. This result is consistent with the results observed by Uphoff et al. [1]. In Fig. 3, the performance drops off as the number of nodes increases due to the increased amount of communication overhead required to complete the simulation. The scaling properties of the application are much more apparent on a small number of nodes due to the reduction in the size of the test dataset. The extrapolated run times for global time stepping decrease as more nodes are added, but the rate of decrease is reduced. The time-to-solution is not compared in this reproducibility paper due to the scaled-down nature of the of the model used in the competition. Because the model is modified to be fully simulated within the span of the competition, directly comparing the time-to-solution would be inaccurate.

In order to efficiently distribute the simulation workload across all available nodes, the PUMgen utility [12] is used to repartition the mesh according to the number of nodes that are currently being tested. For example, when moving from a 4 node setup to a 2 node setup, PUMGen produces an efficient set of mesh partitions for each node. This helps balance workload effectively across nodes, resulting in linear speedup.

6. Geophysical results

A typical SeisSol run produces three xdmf files, which can be visualized using ParaView [13]. Due to unclear instructions in the documentation and in Uphoff et al.'s paper [1], the exact color scheme used in the paper cannot be reproduced in Paraview. The beige arrows coming from the surface represent the vertical displacement vectors calculated from the results. Green arrows are used to represent the horizontal displacement of the fault. The image shown in Figs. 4 and 5 represents 500 seconds of simulation time.

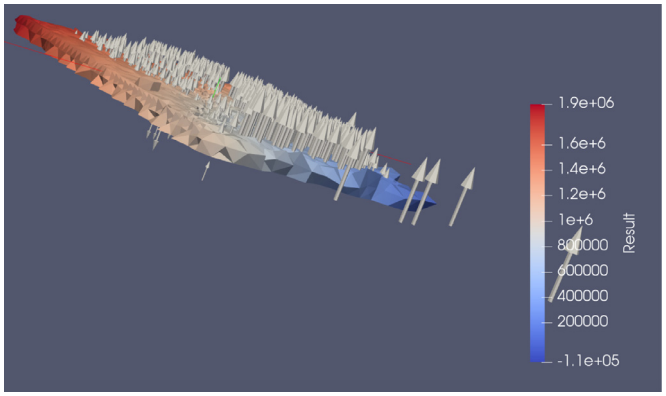

Fig. 4. Vertical Seafloor Displacement. The large magnitude of the arrows is similar to that obtained by Uphoff et al. [1].

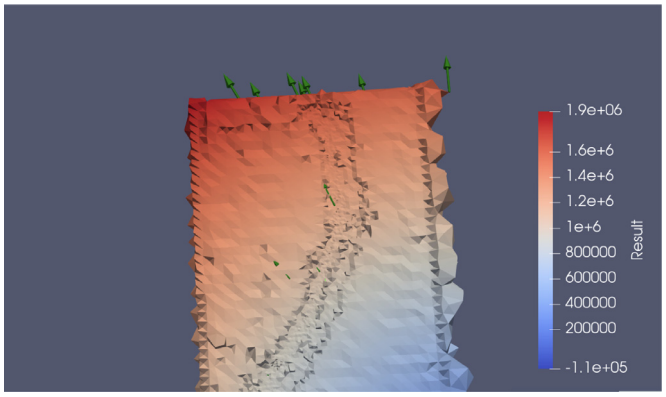

Fig. 5. Horizontal Seafloor Displacement. The upward direction of the arrows is similar to that obtained by Uphoff et al. [1].

Unfortunately, the simulation generated by the datasets provided at the competition was unstable. After approximately 40 seconds, the simulation produces uninterpretable results. The movement of the model is extremely unrealistic after this point and the model view is obscured by incorrect data. This can be resolved by changing the CFL parameter from 0.5 to 0.4. However, this issue was identified after all of the simulation data had already been collected. Because only a few hours remained in the competition when this issue was discovered, it was infeasible to rerun the simulation with the new CFL parameter. While the output correctness of SeisSol was impacted when using these faulty parameters, the performance and timing data discussed in the previous section is unaffected. The following interpretation is based off the first 40 seconds of this simulation.

Based on the results obtained during the first 40 seconds, it appears that the seafloor is significantly displaced, both vertically and horizontally. In comparison with the results presented by Uphoff et al. [1], the plates on both sides of the fault are shifted up vertically, as well as shifted towards the right horizontally. These shifts are likely due to the dataset resolution, so it is not possible to display the faults in this simulation. However, the direction of the generated vectors show the presence of multiple faults. Because of the instability exhibited by the simulation due to parameters specified during the competition, the accuracy of the results produced is in question. As such, we are not able to comment on the reproducibility of accurate simulation output.

7. Conclusion

In this paper we reported on our effort to reproduce prior work by Uphoff et al. [1] performing earthquake simulation. Our results show that the performance on a single node and the scalability of multi-node implementations using local and global time stepping are all reproducible. The general scaling properties of the application are consistent, with additional nodes reducing the runtime of the application and decreasing overall GFLOPs per node. However, because of the instability found in the simulation, the geophysical results obtained are inconclusive. This can be resolved by changing the CFL parameter to a lower value.

Even though the original implementation did not target an AMD-based platform, the CPU performance results still remain competitive and the LIBXSMM library was compatible with our system. The single node performance, as well as the scaling rate across multiple nodes shown by Uphoff et al. [1], are reproducible. In addition, the limited geophysical results collected during the competition also generally align with those seen in the prior work. The results suggest that the performance of the CPU-based version of SeisSol is reproducible on an AMD-based cluster.

Declaration of Competing Interest

The authors have no conflicts of interests to report.

References

[1] 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, in: SC '17, ACM, New York, NY, USA, 2017, pp. 21:1–21:16, doi:10.1145/3126908.3126948.
[2] Ludwigs-Maximilians-Universitt, the Technical University of Munich, Welcome to the seissol homepage!, 2019.
[3] Ludwigs-Maximilians-Universitt, the Technical University of Munich, Release shaking corals seissol/seissol, 2017.
[4] Ludwigs-Maximilians-Universitt, the Technical University of Munich, Seissol seissol documentation, 2018.
[5] S. Foundation, Scons: A software construction tool, 2019.
[6] T.H. Group, The hdf5 library and file format - the hdf group, 2019.
[7] UCAR, Network common data form (netcdf), 2019.
[8] H. Pabst, hfp/libxsmm/ library targeting intel architecture for specialized dense and sparse matrix operations, 2019.
[9] G. Karypis, Metis - serial graph partitioning and fill-reducing matrix ordering — karypis lab, 2015.
[10] G. Karypis, Parmetis - parallel graph partitioning and fill-reducing matrix ordering — karypis lab, 2015.
[11] Z. Xianyi, Openblas / an optimized blas library, 2019.
[12] Ludwigs-Maximilians-Universitt, the Technical University of Munich, Seissol/pumgen/ mesh generation for seissol, 2018.
[13] Kitware, Paraview, 2019.