GIGAIO

WHITE PAPER

In-situ Analysis and Visualization of Earthquake Simulation

PEARC19

In-situ Analysis and Visualization of Earthquake Simulation

Dawei Mu
National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign
Urbana, IL
dmu@illinois.edu

Jon Moran
GigaIO, Inc
Carlsbad, CA
jmoran@gigaio.com

Hui Zhou
The Second Monitoring and Application Center, CEA
Xi'an, Shaanxi
hui@smac.ac.cn

Yifeng Cui
San Diego Supercomputer Center, University of California, San Diego
La Jolla, CA
yfcui@sdsc.edu

Ronald Hawkins
San Diego Supercomputer Center, University of California, San Diego
La Jolla, CA
rhawkins@sdsc.edu

Mahidhar Tatineni
San Diego Supercomputer Center, University of California, San Diego
La Jolla, CA
mahidhar@sdsc.edu

Steve Campbell
GigaIO, Inc
Carlsbad, CA
scampbell@gigaio.com

ABSTRACT

Nowadays, numerical simulation has played a vital role in analyzing and assessing earthquakes and their affects. Storage I/O performance and network bandwidth have not kept pace with the growth of computing power; as a result, post-processing has become a bottleneck to end-to-end simulation performance. One approach to solving this performance imbalance is to reduce the amount of output data by implementing in-situ visualization, which constructs the visualization concurrent with the simulation. In this paper, we propose a new software utility named "awp-odc-insitu" that is based on the well-known open-source seismic simulation software "awp-odc-os" and capable of performing in-situ visualization functionality by employing the open-source data analysis and visualization application "ParaView" and its in-situ library "ParaView Catalyst". Moreover, the paper discusses the implementation of in-situ functionality and analyzes the performance and efficiency of the "awp-odc-insitu" code to demonstrate the code is of potential use in practical seismic research.

CCS CONCEPTS

• **Human-centered computing → Geographic visualization**;
• **Applied computing → Earth and atmospheric sciences**.

PEARC '19, July 28 – August 1, 2019, Chicago, IL

ACM ISBN 978-1-4503-7227-5…$15.00
https://doi.org/10.1145/1122445.1122456

KEYWORDS

in-situ, visualization, earthquake simulation

ACM Reference Format:
Dawei Mu, Jon Moran, Hui Zhou, Yifeng Cui, Ronald Hawkins, Mahidhar Tatineni, and Steve Campbell. 2018. In-situ Analysis and Visualization of Earthquake Simulation. In *PEARC '19: Practice and Experience in Advanced Research Computing, July 28 – August 1, 2019, Chicago, IL.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Nowadays, numerical simulation plays an increasingly critical role in analyzing and assessing earthquakes and their effects. Compared to the improvement of computational performance evident in the leadership-class high-performance computing architectures, the improvement of Storage I/O performance and network bandwidth is slow-paced; as a result, post-processing has become a bottleneck to end-to-end simulation performance. One approach to solving this performance imbalance is to reduce the amount of output data by implementing in-situ visualization, which constructs the visualization concurrent with the simulation [7].

In this paper, we propose a software utility named "awp-odc-insitu" that is based on the open-source seismic simulation software "awp-odc-os" [5]. Over the years, the "awp-odc-os" team has implemented numerous optimizations for this software [6]. The "awp-odc-os" software is well-known for its high performance and efficiency; this paper describes the objective of implementing in-situ visualization functionality within "awp-odc-os" by employing the open-source data analysis and visualization application "ParaView" [2] and its in-situ library, "ParaView Catalyst" [3]. Moreover, the paper discusses the functionality and analyzes the performance and efficiency of the "awp-odc-insitu" code to demonstrate the code is of potential use in practical seismic research.

The paper is organized as follows: In Section 2, we review some related work. The methodology will be introduced in section 3. We

present the workflow of the "awp-odc-insitu" software, its implementation and key features in Section 4. In Section 5, we show some performance benchmarks. Finally, concluding remarks are given in section 6.

2 RELATED WORK

Given the motivation mentioned in the Introduction, there have been successful attempts to implement in-situ visualization in other scientific domains. Among these previous works, we introduce four of them which also implement their software by utilizing the ParaView ecosystem. Dr. Ahrens and his team successfully implemented the novel ParaView Cinema framework into the MPAS-Ocean simulation which can provide highly interactive, image-based in situ visualization and analysis that promotes exploration [1]. A research team from Los Alamos National Laboratory lead by Dr. Patchett has performed a large-scale in-situ visualization simulation for an asteroid-generated tsunami, which results of these simulations will support NASA's Office of Planetary Defense in deciding how to best track near-Earth objects (NEOs) [10]. Lorendeau from University of Bordeaux has successfully integrated the ParaView Catalyst into code Saturne, which is a computational fluid dynamics code used by the largest electricity producers in Europe [8]. Hong Yi and his colleagues demonstrated their approach for in-situ visualization using ParaView Catalyst with a fully resolved turbulent flow through a 2x2 reactor sub-channel with complex geometry. They were able to perform the simulation steering along with in-situ visualization to adjust the pressure gradient which drove the flow through the periodic domain until a desired flow rate was achieved [12].

3 METHODOLOGY

Our goal in this work is to integrate the ParaView with "awp-odc-os" software and use ParaView Catalyst to implement real-time seismic wave visualization. ParaView is an open-source, multi-platform data analysis and visualization application developed and maintained by Kitware, Inc., which adapts the Visualization Toolkit (VTK) as the data processing and rendering engine and has a user interface written using Qt application development framework. The "awp-odc-os" is a CUDA C based open source scientific modeling software developed and maintained by HPGeoC Lab at San Diego Supercomputer Center, which is capable of high performance and large-scale earthquake wave propagation simulations. Our effort within this work can be concluded as three main phases. Firstly, we located the code used to perform memory transfer between CPU and GPU in "awp-odc-os" and inserted our C based code to re-direct the data downloaded from GPU memory into a VTK grid data structure located in CPU memory. Note that we discarded the data buffer design used in "awp-odc-os" code because our "awp-odc-insitu" code is aiming at real-time visualization. Secondly, we developed an adaptor code with ParaView C++ API to pass the VTK grid data to Paraview GUI utilizing the Catalyst Co-Processing library. Within this phase, we also implemented a multiple streams scheme to overlap the communication with computation. Thirdly, We prepared Python scripts with ParaView Python API alongside with our C/C++ code to perform tasks like in-situ visualization, movie rendering and data output.

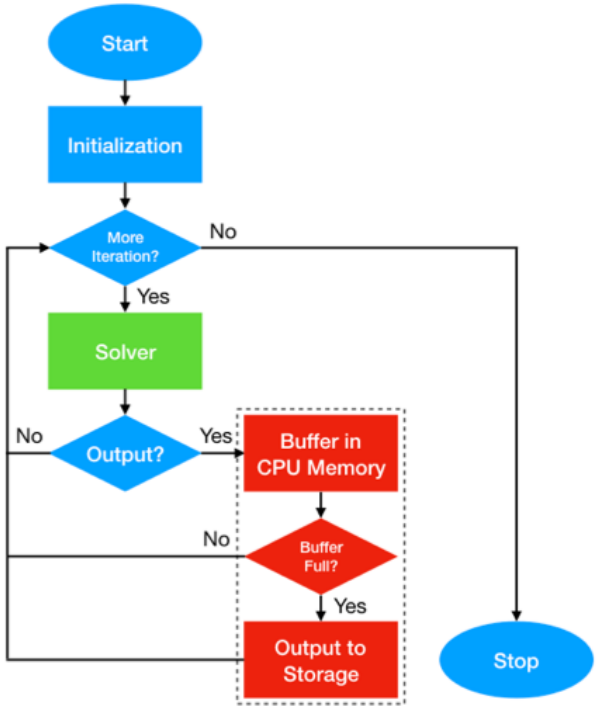

Figure 1: The workflow for "awp-odc-os" software.

4 IMPLEMENTATION

In the following section, we will describe the implementation of "awp-odc-insitu", as well as the design of the "awp-odc-insitu" workflow that makes "awp-odc-insitu" different from the original "awp-odc-os" implementation.

The visualization software package, "ParaView", can be used to integrate post-processing and/or visualization along with distributed seismic simulation [2] - the original workflow of the "awp-odc-os" software needs to be changed in order to add in-situ visualization functionality using ParaView. As shown in Figure 1, before a simulation starts, the "awp-odc-os" software will create an I/O buffer in main memory, for storing data from multiple output iterations. For each output iteration, the CPU downloads the output data from the GPU and saves it in the I/O buffer. Whenever the I/O buffer is full, the CPU outputs the data to storage in binary format with MPI-IO and empties the I/O buffer for subsequent output iterations.

As shown in Figure 2, the computing process of "awp-odc-os" and "awp-odc-insitu" is about the same. As for handling I/O tasks, the "awp-odc-insitu" implementation discards the I/O buffer in CPU memory, because the result will be visualized as soon as it is available rather than buffering it in memory. For each output iteration, the CPU still downloads the output data from the GPU memory; However, the output data is sent to a ParaView co-processing adaptor. The adaptor is created to call the ParaView co-processing library, aka the Paraview Catalyst library, which maps the output data into VTK data structures and passes the dataset to ParaView to perform the following I/O procedures [2].

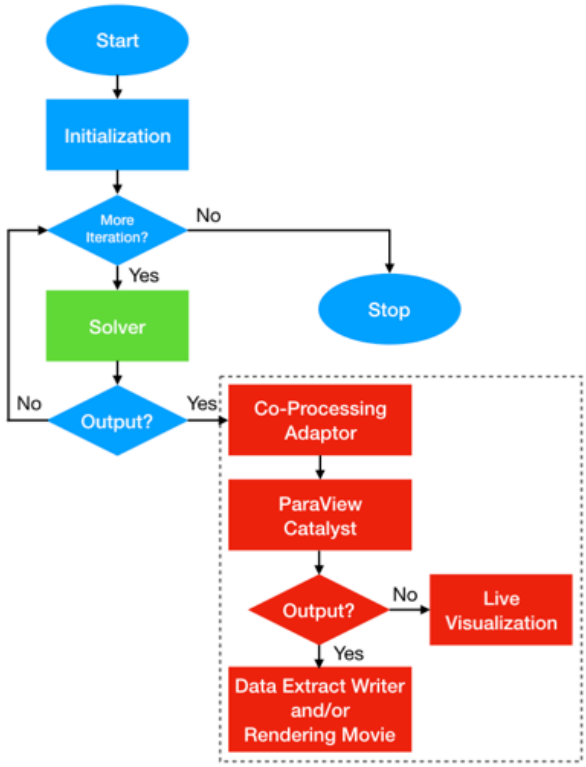

Figure 2: The workflow for "awp-odc-insitu" software.

The ParaView Catalyst library provides three main output options: data extract writers, movie rendering, and live visualization. We prepared generalized Catalyst co-processing python scripts along with our "awp-odc-insitu" software for all three options so users can select the proper script to implement its corresponding functionality. The data extract writer script will save the output data in the form of a series of "Parallel VTK Unstructured Grid" (pvtu) files, which can be opened and viewed directly by the ParaView GUI. The movie rendering script generates one frame of image from the top-view of the output data for every output iteration and these frames can be later rendered as a movie. As shown in Figure 3, the live visualization script sends the output data directly to the ParaView "Catalyst" pipeline so the user can view the data in real-time with the ParaView GUI. Since our "awp-odc-insitu" has packed values along x-axis(Vx), y-axis(Vy) and z-axis(Vz) together, the user can switch among Vx, Vy and Vz components seamlessly. Moreover, the user can also attach a terrain satellite map as the texture map of the output grid to improve the visualization experience.

5 BENCHMARKING AND ANALYSIS

5.1 Graphical Display with Remote Desktop

The "awp-odc-os" software can run on a wide range of different systems from a single notebook computer with NVIDIA GPU to peta-scale supercomputers like "Blue Waters" and "Titan" [4] [11]. Our "awp-odc-insitu" software inherits most of the feature variety from "awp-odc-os", however, since we desired to provide in-situ visualization, our implementation requires a local or remote desktop for the ParaView GUI display. The solution we chose for remote vieiwing is VirtualGL and TurboVNC to provide the remote X-Windows environment. VirtualGL is an open source toolkit that gives any Unix or Linux remote display software the ability to run OpenGL applications with full 3D hardware acceleration and the TurboVNC is a derivative of VNC that is tuned to provide peak performance for 3D and video workloads [9]. The "awp-odc-insitu" software requires installation of both VirtualGL and TurboVNC on the remote GPU server. Before the simulation starts, the user needs to open a VNC server session on the remote machine and access the remote desktop from the local machine with a VNC viewer.

5.2 Benchmark Environment

We ran our benchmark tests of the "awp-odc-insitu" code on the "rincon" machine which is a local system utilizing the GigaIO™ FabreX™ solution. The connections between compute, storage and application accelerator resources in the GigaIO FabreX network are implemented with the robust, packetized communication protocol of industry-standard PCI Express (PCIe). The specific configuration comprises a Supermicro 5018R-M 1U server with a Xeon E5-2680 v4 @ 2.4GHz (16 cores) with 128GB of main memory. There are three PCIe Gen 3 x16 connections between the FabreX AIC resources chassis; each connecting 4x NVIDIA GTX 1080 Ti GPUs for a total of 12 GPUs. The system also contains 8 Samsung 800GB NVMe drives in a RAID0 configuration, which is also connected to the FabreX switch with a PCIe Gen 3 x16 connection. The code was compiled on the Ubuntu-1804 operating system with GCC-7.3.0 and CUDA-10.0 along with MPI package OpenMPI-2.1.1.

5.3 Performance Analysis

We performed a weak-scaling test to compare the performance and efficiency of the "awp-odc-insitu" software and the "awp-odc-os" software. The workload on each GPU is 512x512x256 grid points for a total of 2000 iterations. Both codes output the ground velocity vectors (Vx, Vy, Vz) every 100 iterations. The performance and efficiency benchmark results are shown in Tables 1 and 2, respectively. For the "awp-odc-insitu" software, as mentioned above, the "grid" mode outputs the data in "pvtu" format, the "movie" mode outputs the data as one frame of top-view image in "png" format, and the "live" mode outputs the data directly to the ParaView GUI instead of saving the data to storage. As for the "awp-odc-os" software, the "binary" mode outputs the data in the binary format with MPI-IO and the "no-output" mode means the code won't save the output data at all.

We choose to measure the performance by counting floating point operations per second (FLOP/s). Since the I/O operation is one part of the end-to-end simulation process which consumes run-time without contributing floating point operations, the less time used by I/O operation the higher FLOP/s the software can achieve within a given time frame. For accessing parallel efficiency, we choose to measure as a percentage the ratio of the achieved performance to the ideal performance scaled from 1x GPU up to 12x GPUs.

From the results shown in Table 1, we can see that "awp-odc-insitu" achieved comparable parallel performance and efficiency

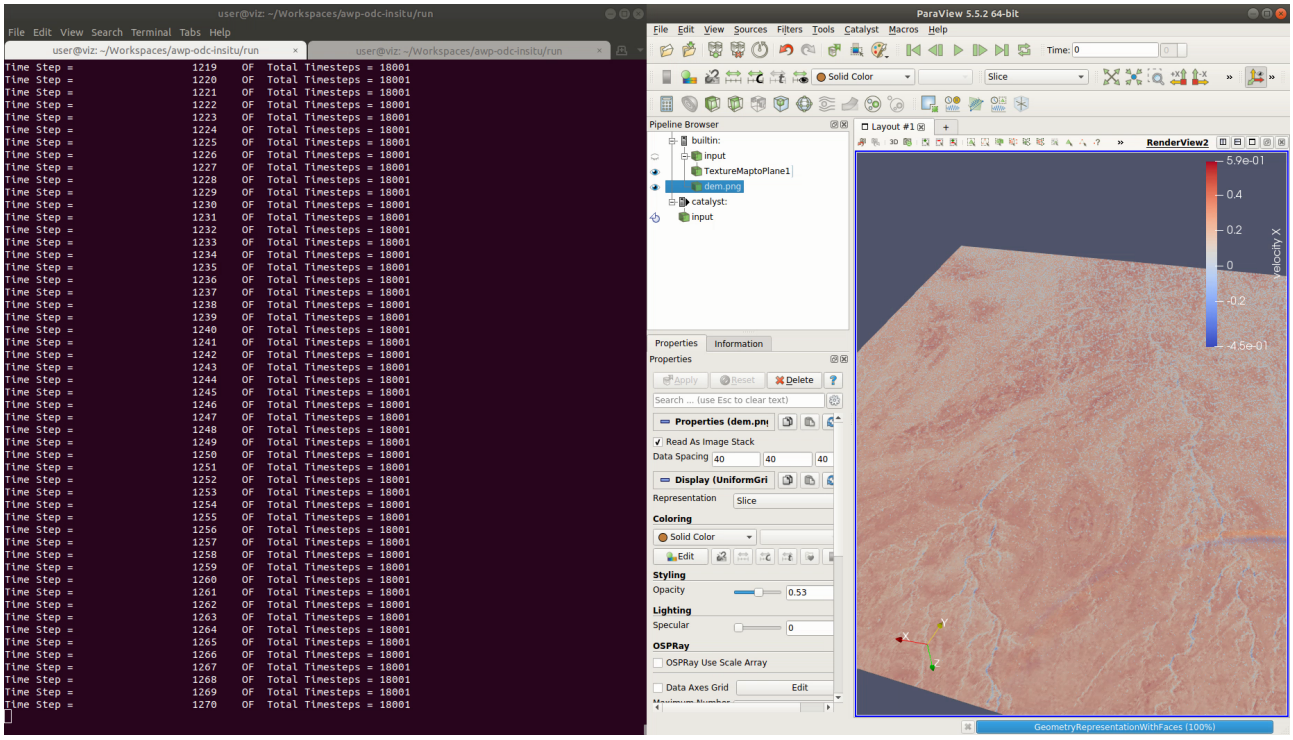

Figure 3: Screenshot captured while "awp-odc-insitu" was performing live-visualization with 4x NVIDIA GTX1080Ti GPUs; the simulated seismic wave is propagating through the surface of the earth.

to the "awp-odc-os" implementation. Taking the largest 12x GPU tests as an example, the "grid" mode of "awp-odc-insitu" obtains a performance of 2362.78 GFLOP/s which is very close to the "binary" mode of "awp-odc-os" code at 2393.02 GFLOP/s. The "movie" mode of our "awp-odc-insitu" only achieved 2101.01 GFLOP/s. However, this performance is also acceptable if we consider the output file size of this mode is only 145 KB, which is 444x smaller than the output file size in the "binary" mode of "awp-odc-os" code. When we compare the "live" mode with the "no-output" mode, we found that we implemented the live visualization with a very small overhead, from 2673.61 GFLOP/s to 2611.42 GFLOP/s. We note that our "awp-odc-insitu" grid mode out-performed "awp-odc-os" binary mode, even the "no-output" mode with 1-2 GPUs, which is due to the fact that our "awp-odc-insitu" uses the "multiple CUDA streams" scheme which overlaps the I/O processing with GPU computing. This "multiple CUDA streams" feature will be included in a future open-source "awp-odc-os" release.

The parallel efficiency results in Table 2 show that our "awp-odc-insitu" code achieves very close scalability compared to the "awp-odc-os" software. Both the "awp-odc-os" software and "awp-odc-insitu" software can achieve 90% parallel efficiency up to 8x GPUs. For the 12x GPU test, "awp-odc-os" binary mode only out-performed "awp-odc-insitu" grid mode by 3.5% and the "awp-odc-insitu" software in movie and live modes all achieved higher parallel efficiency, which shows the "awp-odc-insitu" has the potential to be deployed on large distributed systems.

6 CONCLUSIONS AND FUTURE WORK

Over the course of this work, we developed the "awp-odc-insitu" code to provide in-situ visualization functionality for the "awp-odc-os" seismic simulation software. With in-situ visualization, researchers can both reduce the usage of storage and improve the post-processing efficiency. To verify the parallel performance and efficiency of our code, we also ran some benchmarks using this software on a multiple-GPU machine and the results show that our in-situ implementation achieves expected performance and efficiency.

The current implementation of "awp-odc-insitu" is just the beginning of our research project. In addition to continuing optimization of the I/O operations and tuning performance, we plan to benchmark our code on a larger distributed system like the Comet system at the San Diego Supercomputer Center or the Blue Waters system at the National Center for Supercomputing Applications. Another goal for the near future is data compression - this feature will allow the "awp-odc-insitu" code to further reduce I/O data volume, which could improve the I/O operations efficiency.

ACKNOWLEDGMENTS

This work was conducted under the auspices of the SDSC Advanced Technology Lab (ATL) with generous support from SDSC and GigaIO, Inc. Access to a testbed at GigaIO's facility was also provided. Author Dawei Mu was supported by the NCSA Innovative Systems Laboratory. Author Yifeng Cui's contribution was

Table 1: Parallel performance comparison between "awp-odc-insitu" and "awp-odc-os". (unit: GFLOP/s)

software	output mode	1x GPU (01x01)	2x GPUs (02x01)	4x GPUs (02x02)	8x GPUs (04x02)	12x GPUs (04x03)
awp-odc-insitu	grid	252.31	497.68	965.93	1834.51	2362.78
	movie	203.63	400.31	811.74	1475.61	2101.01
	live	257.61	505.71	978.46	1867.11	2611.42
awp-odc-os	binary	244.58	485.64	942.47	1815.55	2393.02
	no-output	247.68	493.05	978.13	1940.51	2673.61

Table 2: Parallel efficiency comparison between "awp-odc-insitu" and "awp-odc-os". (unit: %)

software	output mode	1x GPU (01x01)	2x GPUs (02x01)	4x GPUs (02x02)	8x GPUs (04x02)	12x GPUs (04x03)
awp-odc-insitu	grid	100.00	98.62	95.71	90.89	78.04
	movie	100.00	98.29	99.66	90.58	85.98
	live	100.00	98.15	94.96	90.60	84.48
awp-odc-os	binary	100.00	99.28	96.34	92.79	81.54
	no-output	100.00	99.54	98.73	97.94	89.96

made possible through the XSEDE's Extended Collaborative Support Service (ECSS) program. Author Hui Zhou in this research was supported by the Science for Earthquake Resilience project (XH18070).

REFERENCES

[1] James Ahrens, Sébastien Jourdain, Patrick O'Leary, John Patchett, David H Rogers, Patricia Fasel, Andrew Bauer, Mark Petersen, Francesca Samsel, and Benjamin Boeckel. 2014. In situ mpas-ocean image-based visualization. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Visualization & Data Analytics Showcase.*

[2] Utkarsh Ayachit. 2015. *The paraview guide: a parallel visualization application.* Kitware, Inc.

[3] Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. 2015. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization.* ACM, 25–29.

[4] Yifeng Cui, Kim B Olsen, Thomas H Jordan, Kwangyoon Lee, Jun Zhou, Patrick Small, Daniel Roten, Geoffrey Ely, Dhabaleswar K Panda, Amit Chourasia, et al. 2010. Scalable earthquake simulation on petascale supercomputers. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis.* IEEE, 1–20.

[5] Yifeng Cui, Efecan Poyraz, Kim B Olsen, Jun Zhou, Kyle Withers, Scott Callaghan, Jeff Larkin, C Guest, D Choi, Amit Chourasia, et al. 2013. Physics-based seismic hazard analysis on petascale heterogeneous supercomputers. In *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE, 1–12.

[6] Yifeng Cui, E Poyraz, J Zhou, S Callaghan, P Maechling, TH Jordan, L Shih, and P Chen. 2013. Accelerating cybershake calculations on the xe6/xk7 platform of blue waters. In *2013 Extreme Scaling Workshop (XSW 2013).* IEEE, 8–17.

[7] Akira Kageyama and Tomoki Yamada. 2014. An approach to exascale visualization: Interactive viewing of in-situ visualization. *Computer Physics Communications* 185, 1 (2014), 79–85.

[8] Benjamin Lorendeau, Yvan Fournier, and Alejandro Ribes. 2013. In-situ visualization in fluid mechanics using catalyst: A case study for code saturne. In *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV).* IEEE, 53–57.

[9] Karin Meier-Fleischer, Niklas Röber, and Michael Böttinger. 2014. Visualization in a Climate Computing Centre. In *EGU General Assembly Conference Abstracts*, Vol. 16.

[10] John Patchett, F Samsel, K Tsai, Galen R Gisler, David H Rogers, Greg D Abram, and Terece L Turton. 2016. Visualization and analysis of threats from asteroid ocean impacts. In *Supercomputing.*

[11] Daniel Roten, Yifeng Cui, Kim B Olsen, Steven M Day, Kyle Withers, William H Savran, Peng Wang, and Dawei Mu. 2016. High-frequency nonlinear earthquake simulations on petascale heterogeneous supercomputers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* IEEE Press, 82.

[12] Hong Yi, Michel Rasquin, Jun Fang, and Igor A Bolotnov. 2014. In-situ visualization and computational steering for large-scale simulation of turbulent flows in complex geometries. In *2014 IEEE International Conference on Big Data (Big Data).* IEEE, 567–572.