Addressing Practical Aspects of Container-native Workflows with Popper

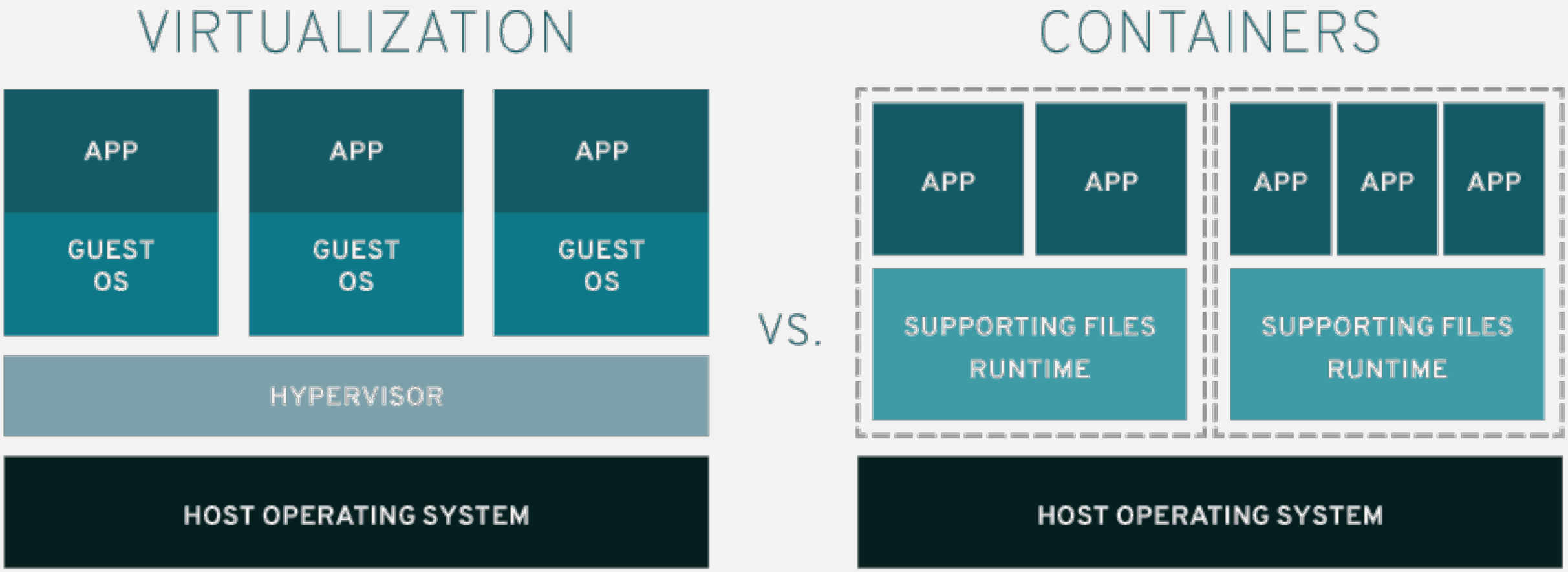
Ivo Jimenez

Research Scientist and CROSS Incubator Fellow

UC Santa Cruz

What is a container?

What is a container?



Main benefit of using containers

Bring Your Own Environment (BYOE) to shared infrastructure

Main benefit of using containers

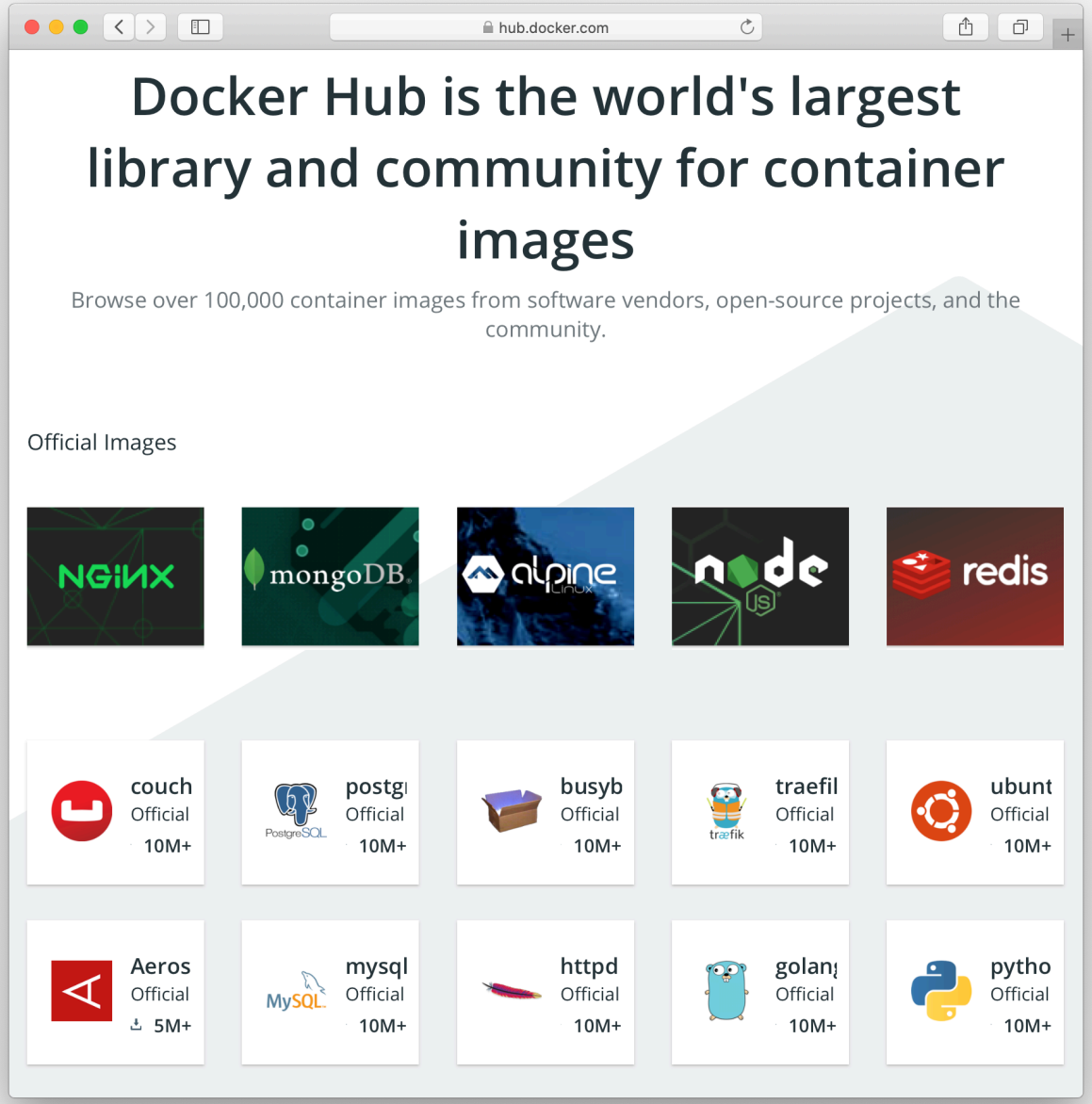
Bring Your Own Environment (BYOE) to shared infrastructure

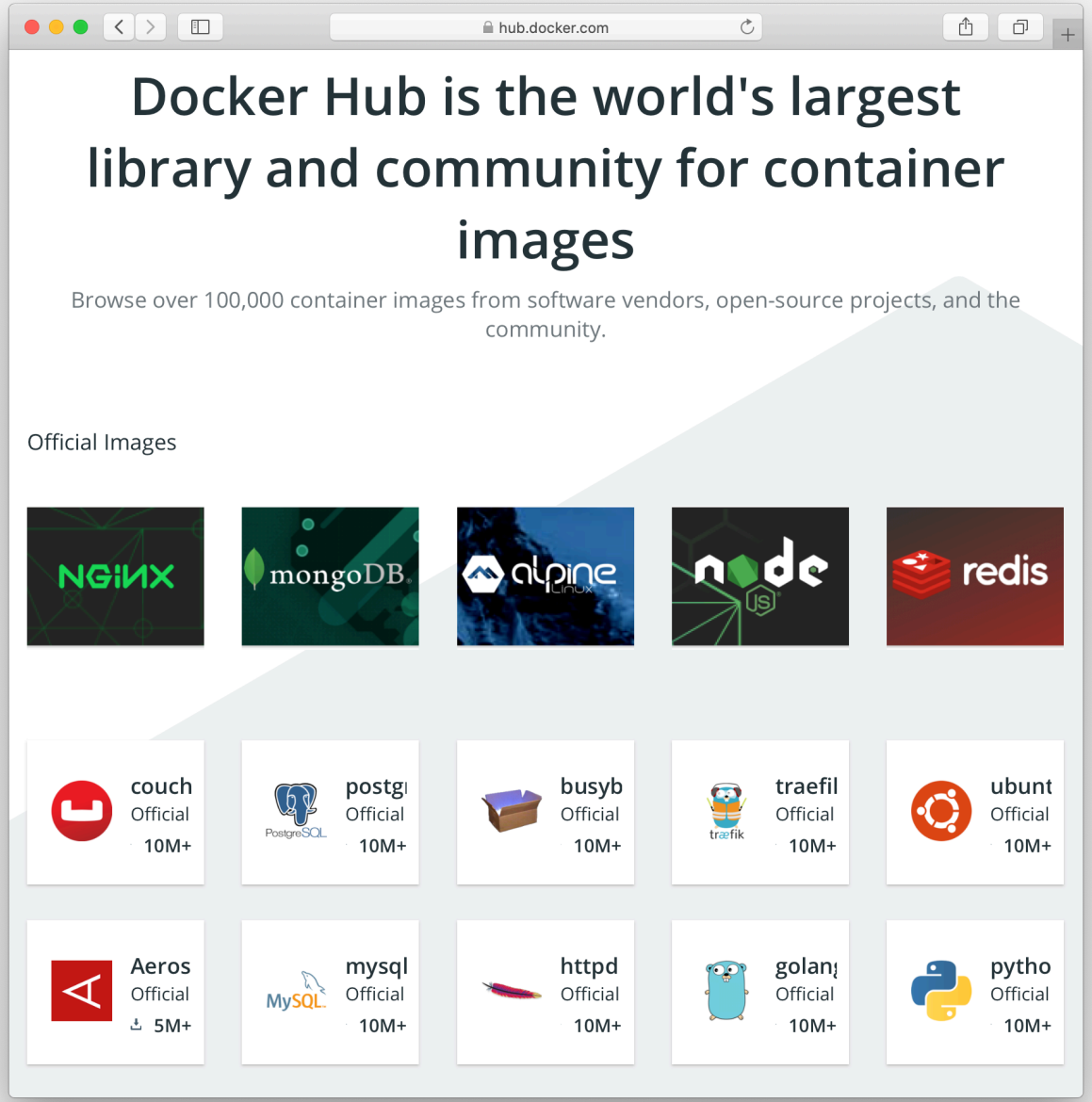
```
docker run mattrayner/lamp:latest-1804
```

Main benefit of using containers

Bring Your Own Environment (BYOE) to shared infrastructure

```
docker run tensorflow/tensorflow:2.1.1-gpu-jupyter
```


hub.docker.com
Docker Hub is the world's largest library and community for container images
Browse over 100,000 container images from software vendors, open-source projects, and the community.
Official Images
NGINX
mongoDB.
alpine Linux
node JS
redis
couch
Official
10M+
postgr
Official
10M+
busyb
Official
10M+
traefil
Official
10M+
ubunt
Official
10M+
Aeros
Official
5M+
mysql
Official
10M+
httpd
Official
10M+
golan
Official
10M+
pytho
Official
10M+


hub.docker.com
Docker Hub is the world's largest library and community for container images
Browse over 100,000 container images from software vendors, open-source projects, and the community.
Official Images
NGINX
mongoDB.
alpine Linux
node JS
redis
couch Official 10M+
postgr Official 10M+
busyb Official 10M+
traefil Official 10M+
ubunt Official 10M+
Aeros Official 5M+
mysql Official 10M+
httpd Official 10M+
golan Official 10M+
pytho Official 10M+


cloud.sylabs.io
Home
Singularity Library
Remote Builder
Keystore
Help
Sign In
Singularity Library
Are you looking for additional features, having a problem or would like to provide feedback? Please email us at support@sylabs.io
Share your awesome containers!
Sign In
Signed With: CE1B8D0C1A2E9D236F718717F24A65F0C9E01833
PYCHUANG/DEFAULT/OLDARCH
Download 1.65 GB
torchgpu20160801
Description: See https://hub.c
AMD64
STARS: 0
DOWNLOADS: 10
AARONTHOLT/DEFAULT/MFIX-EX
Download 363.56 MB
phase2-develop_bfa2e5f
Description: Containers to bu
AMD64
STARS: 0
DOWNLOADS: 195
Signed With: 418F830D41583C1C8F0CAA32C0970D077A0E1661
NUCCI/DEFAULT/RGDAL-SF
Download 1.02 GB
latest
AMD64
STARS: 0
DOWNLOADS: 2
HUSI253/DEFAULT/STAIN_NORM
Download 572.43 MB
py3-lhx-2020.06.24
Description: run python library s
AMD64
STARS: 0
DOWNLOADS: 2
RPOLICASTRO/DEFAULT/ZENT_TOC
Download 1.18 GB
MOST DOWNLOADED
library/default/alpine 284741
library/default/busybox 142065
sylabs-bot/smoke-test-collection/testimage1 118546
sylabs-bot/smoke-test-collection/testimage0 118536
sylabs/tests/not-default 99416
sylabs/tests/unsigned 46973
sylabs/tests/signed 21219
library/default/ubuntu 15126
library/default/centos 7544
sylabs/tests/busybox 7040
MOST STARRED
library/default/ubuntu 26
library/default/alpine 17
library/default/centos 13
sylabsed/examples/lolcow 11

What is the container-native paradigm?

What is the container-native paradigm?

Use containers for everything:

- Build software, pre-process data, run experiments, analyze results, generate a manuscript, etc.

What is the container-native paradigm?

Use containers for everything:

- Build software, pre-process data, run experiments, analyze results, generate a manuscript, etc.

If you install software directly on your machine, then you are not following a container-native approach*

*other than personal productivity tools such as a text editor, web browser, email reader, calendar app, etc.

Practical problems that arise when working under the container-native paradigm

Practical problems that arise when working under the container-native paradigm

- Dealing with multi-container workflows
    - Complex application testing and prototyping becomes difficult to reproduce if done by hand

Practical problems that arise when working under the container-native paradigm

- Dealing with multi-container workflows
    - Complex application testing and prototyping becomes difficult to reproduce if done by hand
- Myriad of container runtimes
    - Docker, Podman, LXD, Singularity, Charliecloud, …

Practical problems that arise when working under the container-native paradigm

- Dealing with multi-container workflows
    - Complex application testing and prototyping becomes difficult to reproduce if done by hand
- Myriad of container runtimes
    - Docker, Podman, LXD, Singularity, Charliecloud, …
- Lack of common orchestration platform support
    - SLURM, Kubernetes, …

Practical problems that arise when working under the container-native paradigm

- Dealing with multi-container workflows
    - Complex application testing and prototyping becomes difficult to reproduce if done by hand
- Myriad of container runtimes
    - Docker, Podman, LXD, Singularity, Charliecloud,
- Lack of common orchestration platform support
    - SLURM, Kubernetes, …

Popper

Practical problems that arise when working under the container-native paradigm

- Dealing with multi-container workflows
  - Complex application testing and prototyping becomes difficult to reproduce if done by hand
- Myriad of container runtimes
  - Docker, Podman, LXD, Singularity, Charliecloud,
- Lack of common orchestration platform support
  - SLURM, Kubernetes, …

Popper

Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake

Carsten Uphoff
Sebastian Rettenberger
Michael Bader
uphoff@in.tum.de
rettenbs@in.tum.de
bader@in.tum.de
Technical University of Munich
Boltzmannstr. 3
85748 Garching, Germany

Elizabeth H. Madden
Thomas Ulrich
Stephanie Wollherr
Alice-Agnes Gabriel
madden@geophysik.uni-muenchen.de
ulrich@geophysik.uni-muenchen.de
wollherr@geophysik.uni-muenchen.de
gabriel@geophysik.uni-muenchen.de
Ludwig-Maximilians-Universität München
Theresienstr. 41
80333 Munich, Germany

ABSTRACT

We present a high-resolution simulation of the 2004 Sumatra-Andaman earthquake, including non-linear frictional failure on a megathrust-splay fault system. Our method exploits unstructured meshes capturing the complicated geometries in subduction zones that are crucial to understand large earthquakes and tsunami generation. These up-to-date largest and longest dynamic rupture simulations enable analysis of dynamic source effects on the seafloor displacements.

To tackle the extreme size of this scenario an end-to-end optimization of the simulation code SeisSol was necessary. We implemented a new cache-aware wave propagation scheme and optimized the dynamic rupture kernels using code generation. We established a novel clustered local-time-stepping scheme for dynamic rupture. In total, we achieved a speed-up of 13.6 compared to the previous implementation. For the Sumatra scenario with 221 million elements this reduced the time-to-solution to 13.9 hours on 86,016 Haswell cores. Furthermore, we used asynchronous output to overlap I/O and compute time.

CCS CONCEPTS

• **Applied computing → Earth and atmospheric sciences**;

KEYWORDS

ADER-DG; earthquake simulation; tsunami coupling; dynamic rupture; petascale performance; hybrid parallelization; local time stepping; asynchronous output

ACM Reference format:
Carsten Uphoff, Sebastian Rettenberger, Michael Bader, Elizabeth H. Madden, Thomas Ulrich, Stephanie Wollherr, and Alice-Agnes Gabriel. 2017. Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake. In *Proceedings of SC17, Denver, CO, USA, November 12–17, 2017,* 16 pages.
DOI: 10.1145/3126908.3126948

1 INTRODUCTION

"Megathrust" earthquakes in subduction zones, caused by oceanic lithosphere sliding into the mantle beneath an overriding tectonic plate, have released most of the seismic energy over the last century. No other type of tectonic activity is known to produce earthquakes that exceed moment magnitudes M_w of 9 and trigger teletsunamis traveling whole oceans.

A particularly devastating example is the M_w 9.1 2004 Sumatra-Andaman Earthquake and Indian Ocean Tsunami. The Sumatra earthquake ruptured the greatest fault length of any recorded earthquake and triggered a series of tsunamis, killing up to 280,000 people in 14 countries, and even displaced Earth's North Pole by 2.5 cm.

For the coming decades, there is no realistic hope to reliably predict earthquakes, such that the method of choice to mitigate earthquake- and tsunami-related damage to our societies is to forecast seafloor displacement and strong ground motions for likely earthquake scenarios. Dynamic rupture simulations combine non-linear frictional failure and seismic wave propagation in a multi-physics manner to produce physics-based forecasts [e.g. 5, 20, 30, 38, 65] and provide insight into the poorly understood fundamental processes of earthquake faulting [e.g. 2, 4, 29, 32, 33, 68].

A correct representation of subduction zone complexity is crucial, but, in concurrence with the giant spatial extent and temporal duration to be resolved, extremely challenging from a numerical point of view. Simulations must capture slip and seismic waves occurring for hundreds of seconds along hundreds of kilometers but at the same time resolve the meter-scale physical processes taking place at the rupture tip of the earthquake source.

A recent rise of 3D HPC earthquake simulation software [e.g. 18, 43, 48, 49, 72, 75, 76] allows for the simulation of various aspects of earthquake scenarios and enables researchers to answer geophysical questions complicated by the lack of sufficiently dense

A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

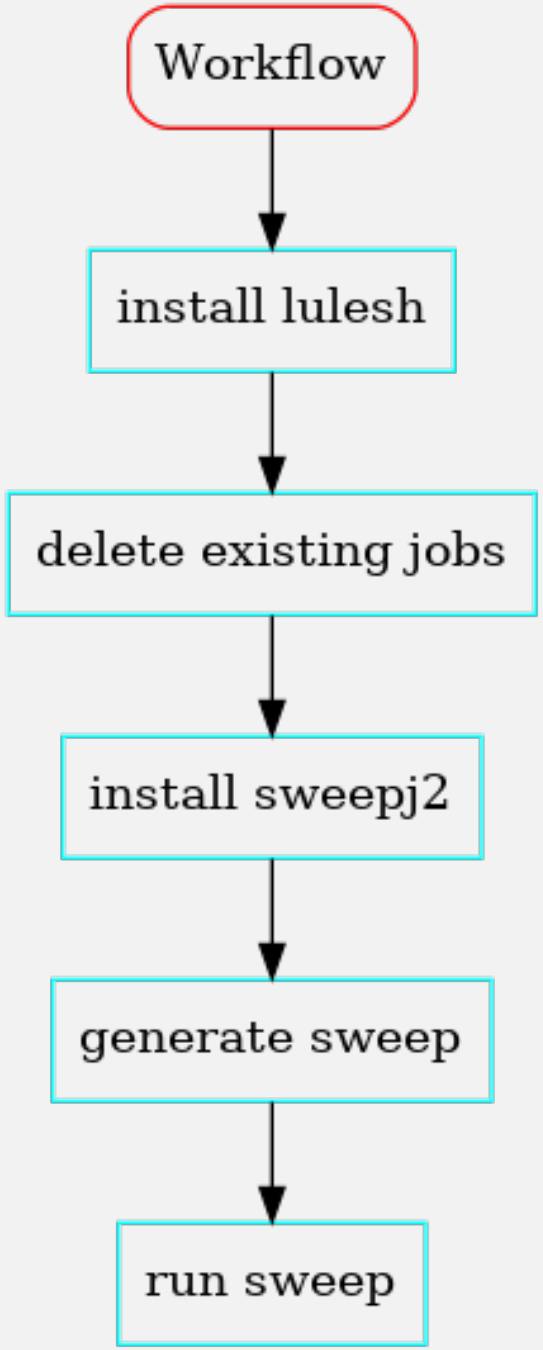
A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```




```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```


```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

https://github.com/getpopper/popper


```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

https://github.com/getpopper/popper

Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake

Carsten Uphoff
Sebastian Rettenberger
Michael Bader
uphoff@in.tum.de
rettenbs@in.tum.de
bader@in.tum.de
Technical University of Munich
Boltzmannstr. 3
85748 Garching, Germany

Elizabeth H. Madden
Thomas Ulrich
Stephanie Wollherr
Alice-Agnes Gabriel
madden@geophysik.uni-muenchen.de
ulrich@geophysik.uni-muenchen.de
wollherr@geophysik.uni-muenchen.de
gabriel@geophysik.uni-muenchen.de
Ludwig-Maximilians-Universität München
Theresienstr. 41
80333 Munich, Germany

ABSTRACT

We present a high-resolution simulation of the 2004 Sumatra-Andaman earthquake, including non-linear frictional failure on a megathrust-splay fault system. Our method exploits unstructured meshes capturing the complicated geometries in subduction zones that are crucial to understand large earthquakes and tsunami generation. These up-to-date largest and longest dynamic rupture simulations enable analysis of dynamic source effects on the seafloor displacements.

To tackle the extreme size of this scenario an end-to-end optimization of the simulation code SeisSol was necessary. We implemented a new cache-aware wave propagation scheme and optimized the dynamic rupture kernels using code generation. We established a novel clustered local-time-stepping scheme for dynamic rupture. In total, we achieved a speed-up of 13.6 compared to the previous implementation. For the Sumatra scenario with 221 million elements this reduced the time-to-solution to 13.9 hours on 86,016 Haswell cores. Furthermore, we used asynchronous output to overlap I/O and compute time.

CCS CONCEPTS

• **Applied computing → Earth and atmospheric sciences**;

KEYWORDS

ADER-DG; earthquake simulation; tsunami coupling; dynamic rupture; petascale performance; hybrid parallelization; local time stepping; asynchronous output

ACM Reference format:
Carsten Uphoff, Sebastian Rettenberger, Michael Bader, Elizabeth H. Madden, Thomas Ulrich, Stephanie Wollherr, and Alice-Agnes Gabriel. 2017. Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake. In *Proceedings of SC17, Denver, CO, USA, November 12–17, 2017,* 16 pages.
DOI: 10.1145/3126908.3126948

1 INTRODUCTION

"Megathrust" earthquakes in subduction zones, caused by oceanic lithosphere sliding into the mantle beneath an overriding tectonic plate, have released most of the seismic energy over the last century. No other type of tectonic activity is known to produce earthquakes that exceed moment magnitudes M_w of 9 and trigger teletsunamis traveling whole oceans.

A particularly devastating example is the M_w 9.1 2004 Sumatra-Andaman Earthquake and Indian Ocean Tsunami. The Sumatra earthquake ruptured the greatest fault length of any recorded earthquake and triggered a series of tsunamis, killing up to 280,000 people in 14 countries, and even displaced Earth's North Pole by 2.5 cm.

For the coming decades, there is no realistic hope to reliably predict earthquakes, such that the method of choice to mitigate earthquake- and tsunami-related damage to our societies is to forecast seafloor displacement and strong ground motions for likely earthquake scenarios. Dynamic rupture simulations combine non-linear frictional failure and seismic wave propagation in a multi-physics manner to produce physics-based forecasts [e.g. 5, 20, 30, 38, 65] and provide insight into the poorly understood fundamental processes of earthquake faulting [e.g. 2, 4, 29, 32, 33, 68].

A correct representation of subduction zone complexity is crucial, but, in concurrence with the giant spatial extent and temporal duration to be resolved, extremely challenging from a numerical point of view. Simulations must capture slip and seismic waves occurring for hundreds of seconds along hundreds of kilometers but at the same time resolve the meter-scale physical processes taking place at the rupture tip of the earthquake source.

A recent rise of 3D HPC earthquake simulation software [e.g. 18, 43, 48, 49, 72, 75, 76] allows for the simulation of various aspects of earthquake scenarios and enables researchers to answer geophysical questions complicated by the lack of sufficiently dense

A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/

A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

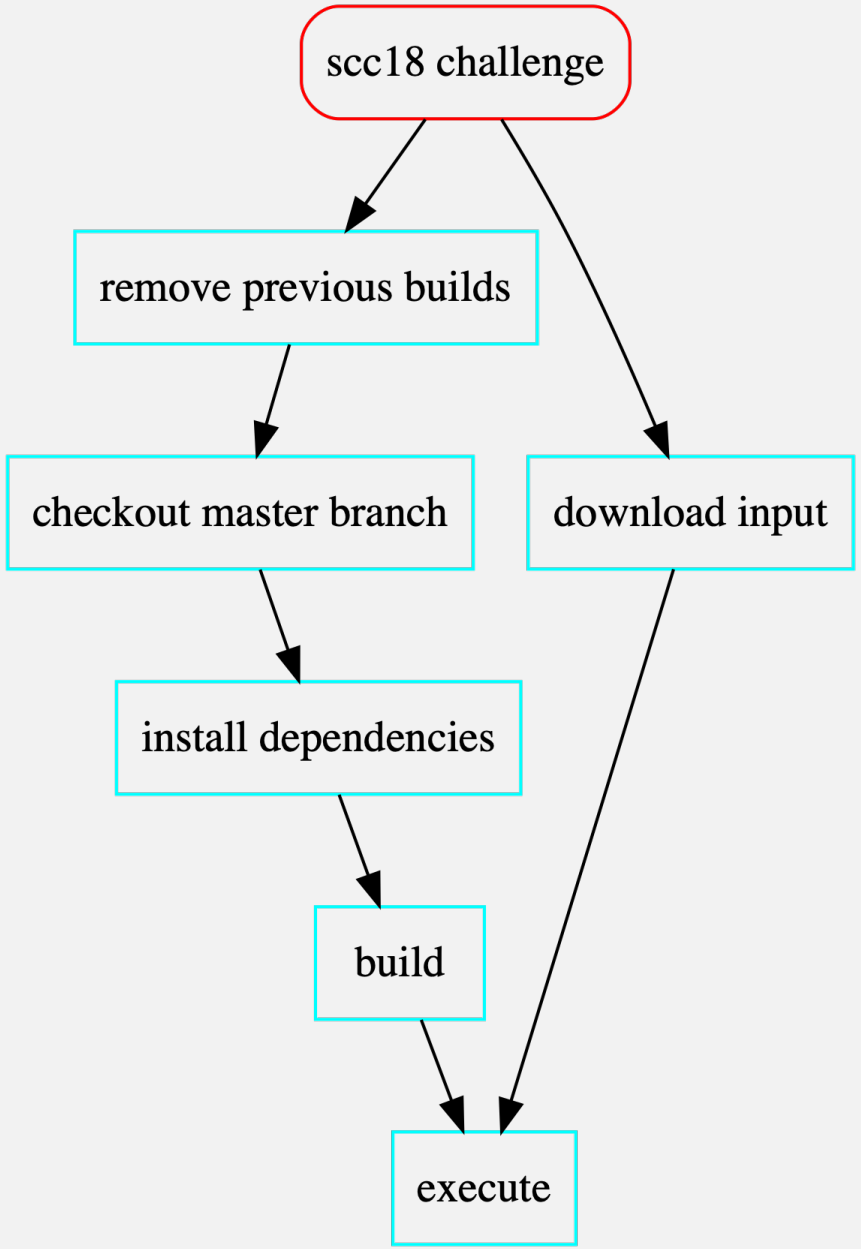
For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/



A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

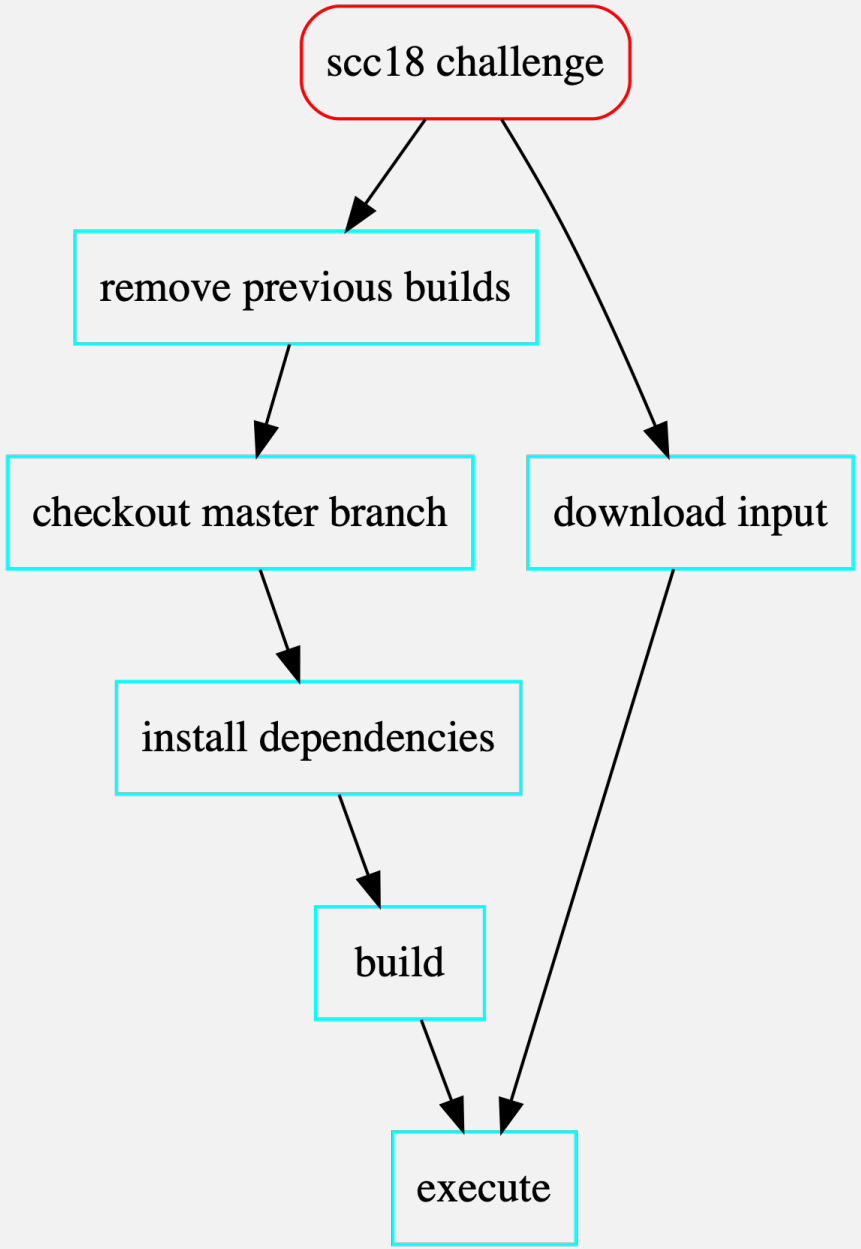
For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/



A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

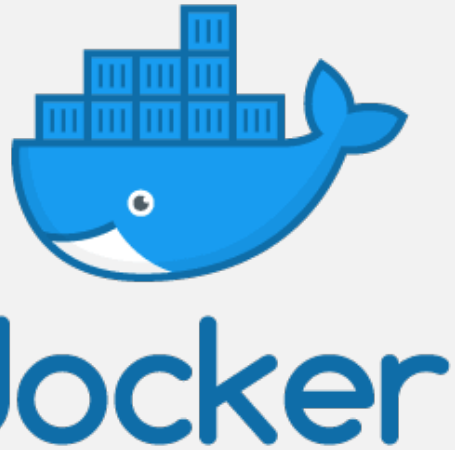
The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

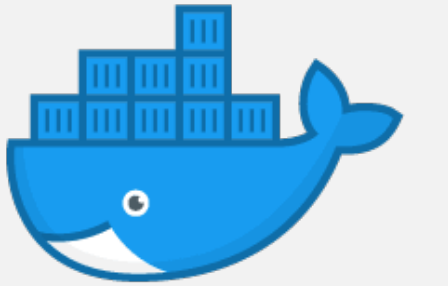
Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/

SCC18: https://github.com/getpopper/seissol-workflows
SCC19: https://github.com/getpopper/normalmodes-workflows

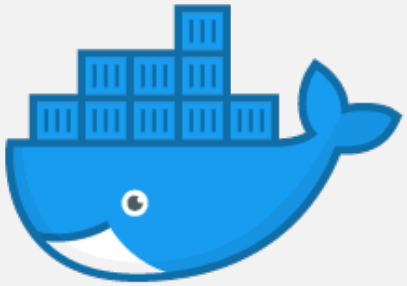
Practical problems that arise when working under the container-native paradigm

- Dealing with multi-container workflows
    - Complex application testing and prototyping becomes difficult to reproduce if done by hand
- Myriad of container runtimes
    - Docker, Podman, LXD, Singularity, Charliecloud, …
- Lack of common orchestration platform support
    - SLURM, Kubernetes, …


docker

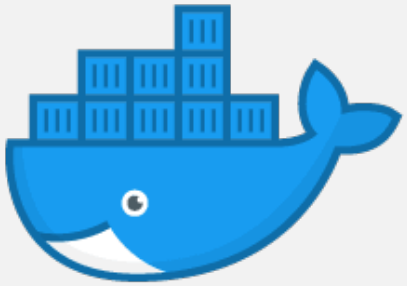


docker



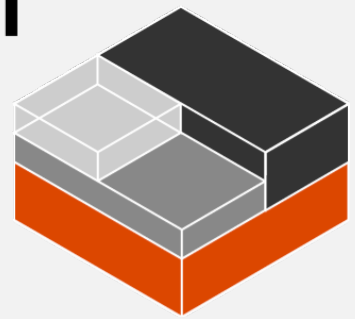
docker

podman

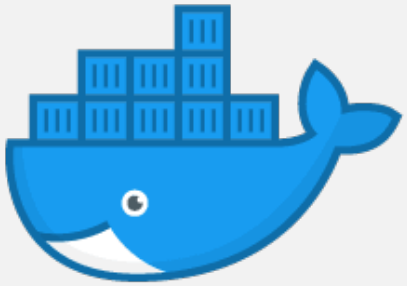


docker

podman



lxd

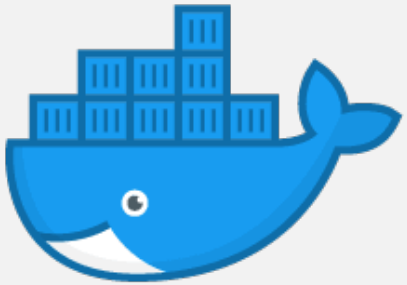


docker

podman

Charliecloud

lxd

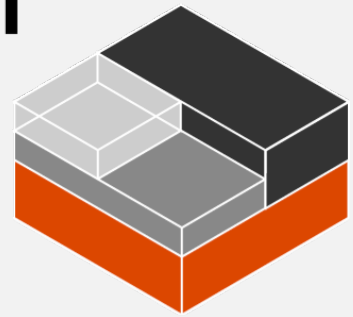


docker

kata
containers

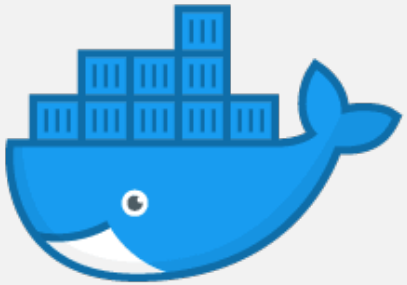
podman

Charliecloud



lxd

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```



docker

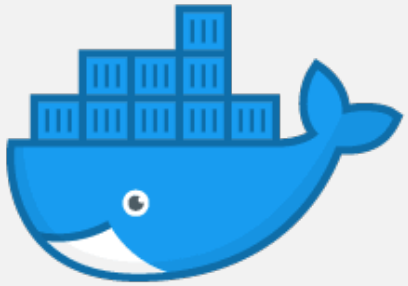
kata containers

podman

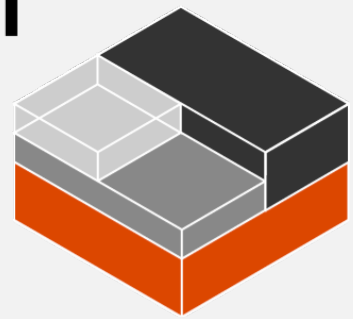
Charliecloud

lxd

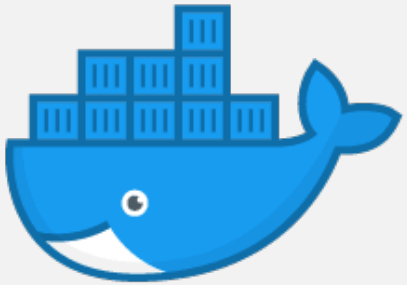
```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```



podman



```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```



docker

kata containers

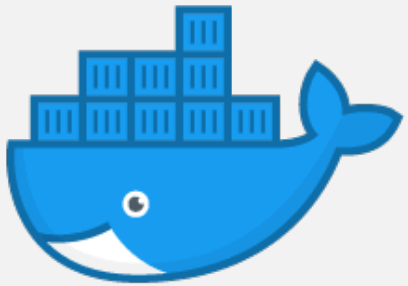
podman

Charliecloud

lxd

Popper

```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

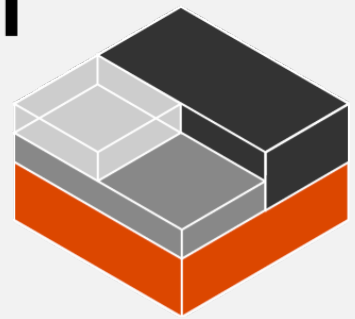


docker

kata containers

podman

Charliecloud

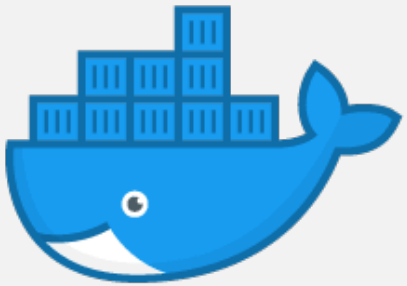


lxd

Popper

```
$> popper run
```

```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```



docker

kata containers

podman

Charliecloud

lxd

Popper

```
$> popper run --engine podman
```

```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

docker

kata containers

podman

lxd

Charliecloud

Practical problems that arise when working under the container-native paradigm

- Dealing with multi-container workflows
    - Complex application testing and prototyping becomes difficult to reproduce if done by hand
- Myriad of container runtimes
    - Docker, Podman, LXD, Singularity, Charliecloud, …
- Lack of common orchestration platform support
    - SLURM, Kubernetes, …

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

kubernetes

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

HTCondor

kubernetes

slurm
workload manager

```yaml
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

HTCondor

kubernetes

slurm
workload manager

GRID ENGINE

Popper

```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

HTCondor

kubernetes

slurm
workload manager

GRID ENGINE

Popper

```
$> popper run
```

```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

HTCondor

kubernetes

slurm
workload manager

GRID ENGINE

Popper

```
$> popper run --resman slurm
```

```
steps:
- id: install lulesh
  uses: popperized/spack@master
  args: [spack, install, -j8, lulesh+mpi]

- id: delete existing jobs
  uses: popperized/bin/sh@master
  args: [rm, -fr, sweep/jobs]

- id: install sweepj2
  uses: popperized/python-actions@master
  args: [pip, install, sweepj2]

- id: generate sweep
  uses: jefftriplett/python-actions@master
  args: [
    "sweepj2",
    "--template", "./sweep/script.j2",
    "--space", "./sweep/space.yml",
    "--output", "./sweep/jobs/",
    "--make-executable"
  ]

- id: run sweep
  uses: popperized/spack@master
  args: [run-parts, ./sweep/jobs]
```

HTCondor

kubernetes

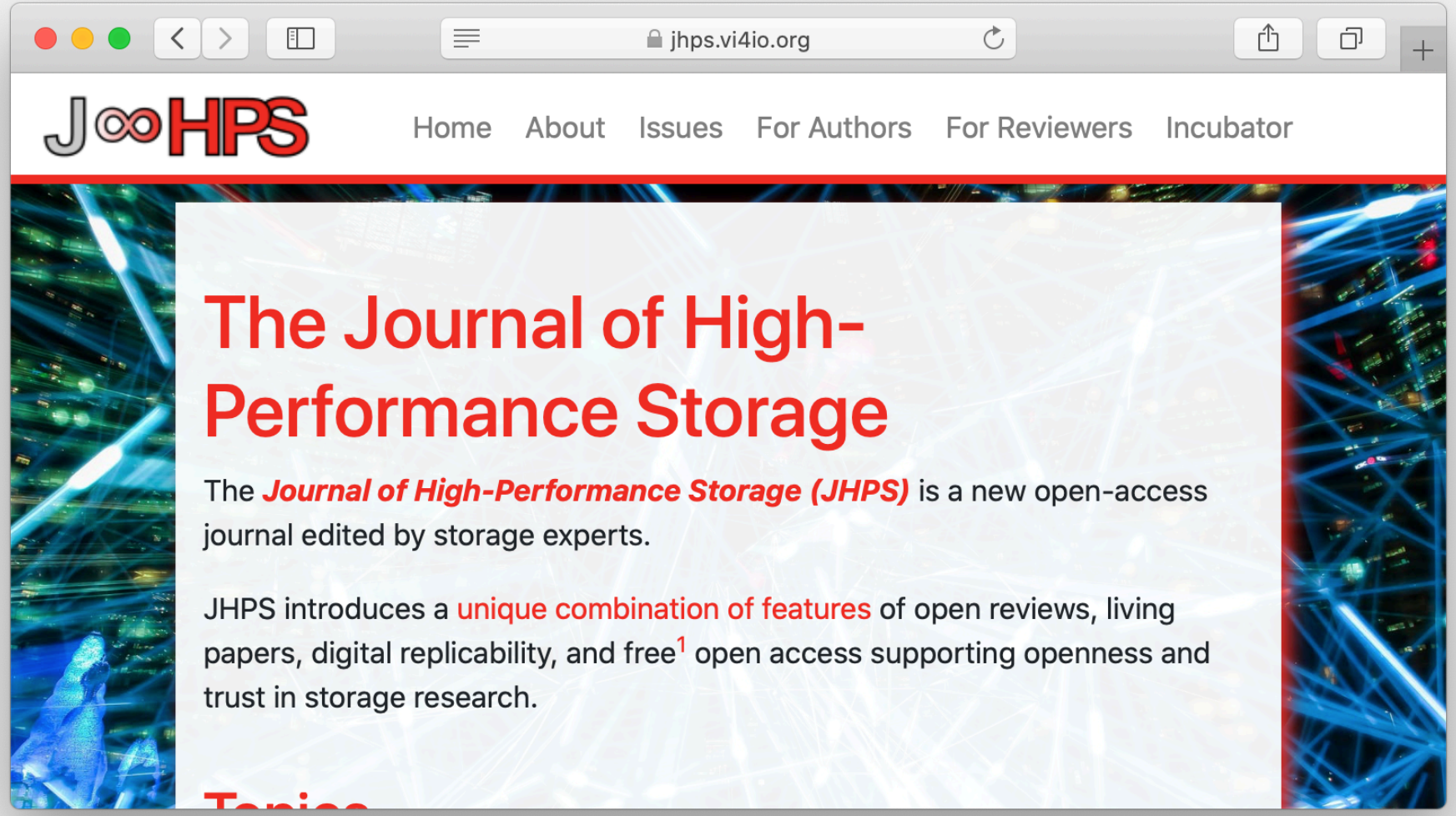
slurm
workload manager

GRID ENGINE

Take away

- The container-native paradigm improves reproducibility aspects of experimentation workflows.
- But, as always, there's no free lunch: working under this new paradigm comes with its associated problems
- Popper addresses these in a holistic and user-friendly way

Popper as an alternative for minimizing overheads in the peer-review process of journal and conference submissions



reproducibility.acm.org

ACM Emerging Interest Group on

Reproducibility and Replicability

Fostering a diverse and inclusive community around the issues of reproducibility and replicability of computational research.

Join Now!

Mission

Foster a broad and inclusive intellectual community around the issues of reproducibility of computational research. Attain SIG status within ACM.

Activities

The EIG proposes two working groups, one to focus on conference planning and execution activities, and the other to coordinate discussions and efforts regarding reproducibility (best/better) practices.

Events

Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake

Carsten Uphoff
Sebastian Rettenberger
Michael Bader
uphoff@in.tum.de
rettenbs@in.tum.de
bader@in.tum.de
Technical University of Munich
Boltzmannstr. 3
85748 Garching, Germany

Elizabeth H. Madden
Thomas Ulrich
Stephanie Wollherr
Alice-Agnes Gabriel
madden@geophysik.uni-muenchen.de
ulrich@geophysik.uni-muenchen.de
wollherr@geophysik.uni-muenchen.de
gabriel@geophysik.uni-muenchen.de
Ludwig-Maximilians-Universität München
Theresienstr. 41
80333 Munich, Germany

ABSTRACT

We present a high-resolution simulation of the 2004 Sumatra-Andaman earthquake, including non-linear frictional failure on a megathrust-splay fault system. Our method exploits unstructured meshes capturing the complicated geometries in subduction zones that are crucial to understand large earthquakes and tsunami generation. These up-to-date largest and longest dynamic rupture simulations enable analysis of dynamic source effects on the seafloor displacements.

To tackle the extreme size of this scenario an end-to-end optimization of the simulation code SeisSol was necessary. We implemented a new cache-aware wave propagation scheme and optimized the dynamic rupture kernels using code generation. We established a novel clustered local-time-stepping scheme for dynamic rupture. In total, we achieved a speed-up of 13.6 compared to the previous implementation. For the Sumatra scenario with 221 million elements this reduced the time-to-solution to 13.9 hours on 86,016 Haswell cores. Furthermore, we used asynchronous output to overlap I/O and compute time.

CCS CONCEPTS

• **Applied computing → Earth and atmospheric sciences**;

KEYWORDS

ADER-DG; earthquake simulation; tsunami coupling; dynamic rupture; petascale performance; hybrid parallelization; local time stepping; asynchronous output

ACM Reference format:
Carsten Uphoff, Sebastian Rettenberger, Michael Bader, Elizabeth H. Madden, Thomas Ulrich, Stephanie Wollherr, and Alice-Agnes Gabriel. 2017. Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake. In *Proceedings of SC17, Denver, CO, USA, November 12–17, 2017,* 16 pages.
DOI: 10.1145/3126908.3126948

1 INTRODUCTION

"Megathrust" earthquakes in subduction zones, caused by oceanic lithosphere sliding into the mantle beneath an overriding tectonic plate, have released most of the seismic energy over the last century. No other type of tectonic activity is known to produce earthquakes that exceed moment magnitudes M_w of 9 and trigger teletsunamis traveling whole oceans.

A particularly devastating example is the M_w 9.1 2004 Sumatra-Andaman Earthquake and Indian Ocean Tsunami. The Sumatra earthquake ruptured the greatest fault length of any recorded earthquake and triggered a series of tsunamis, killing up to 280,000 people in 14 countries, and even displaced Earth's North Pole by 2.5 cm.

For the coming decades, there is no realistic hope to reliably predict earthquakes, such that the method of choice to mitigate earthquake- and tsunami-related damage to our societies is to forecast seafloor displacement and strong ground motions for likely earthquake scenarios. Dynamic rupture simulations combine non-linear frictional failure and seismic wave propagation in a multi-physics manner to produce physics-based forecasts [e.g. 5, 20, 30, 38, 65] and provide insight into the poorly understood fundamental processes of earthquake faulting [e.g. 2, 4, 29, 32, 33, 68].

A correct representation of subduction zone complexity is crucial, but, in concurrence with the giant spatial extent and temporal duration to be resolved, extremely challenging from a numerical point of view. Simulations must capture slip and seismic waves occurring for hundreds of seconds along hundreds of kilometers but at the same time resolve the meter-scale physical processes taking place at the rupture tip of the earthquake source.

A recent rise of 3D HPC earthquake simulation software [e.g. 18, 43, 48, 49, 72, 75, 76] allows for the simulation of various aspects of earthquake scenarios and enables researchers to answer geophysical questions complicated by the lack of sufficiently dense

A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/

A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

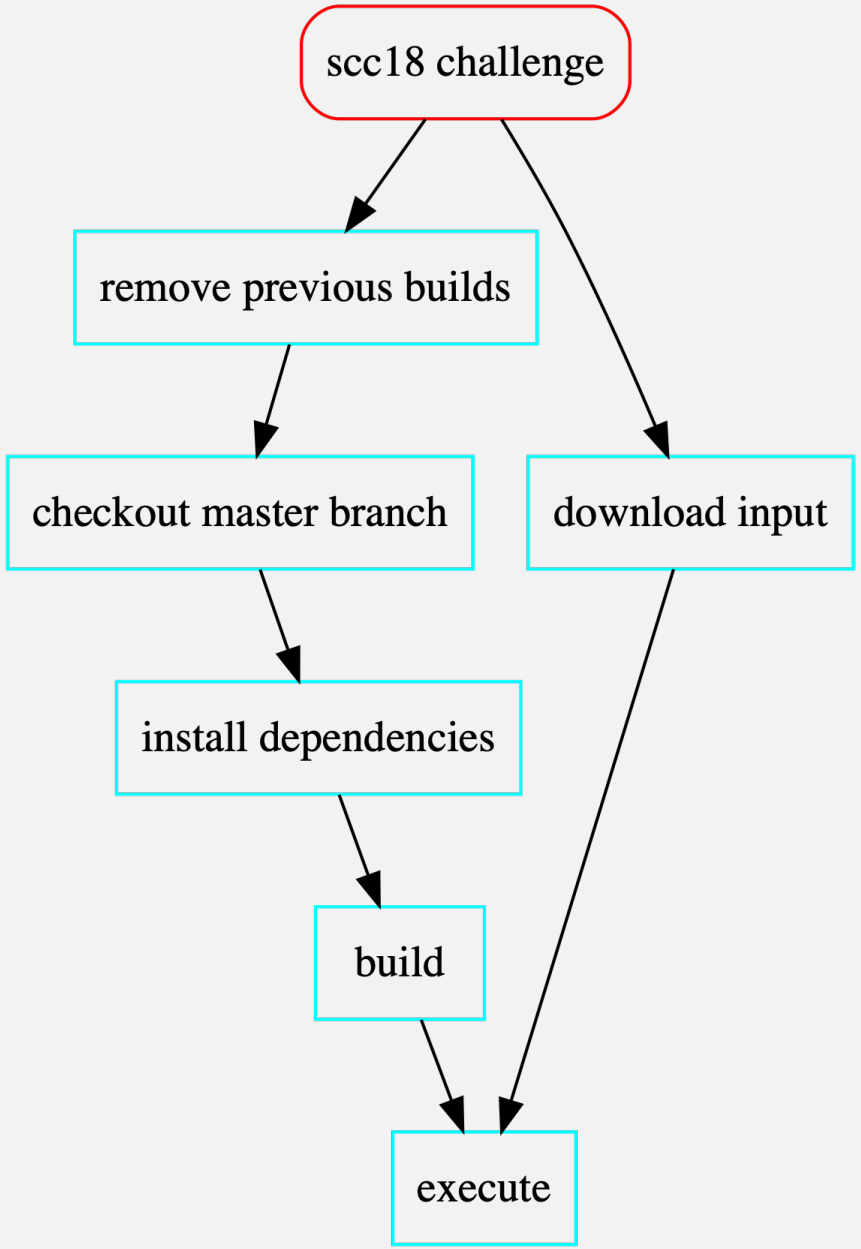
For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/



A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

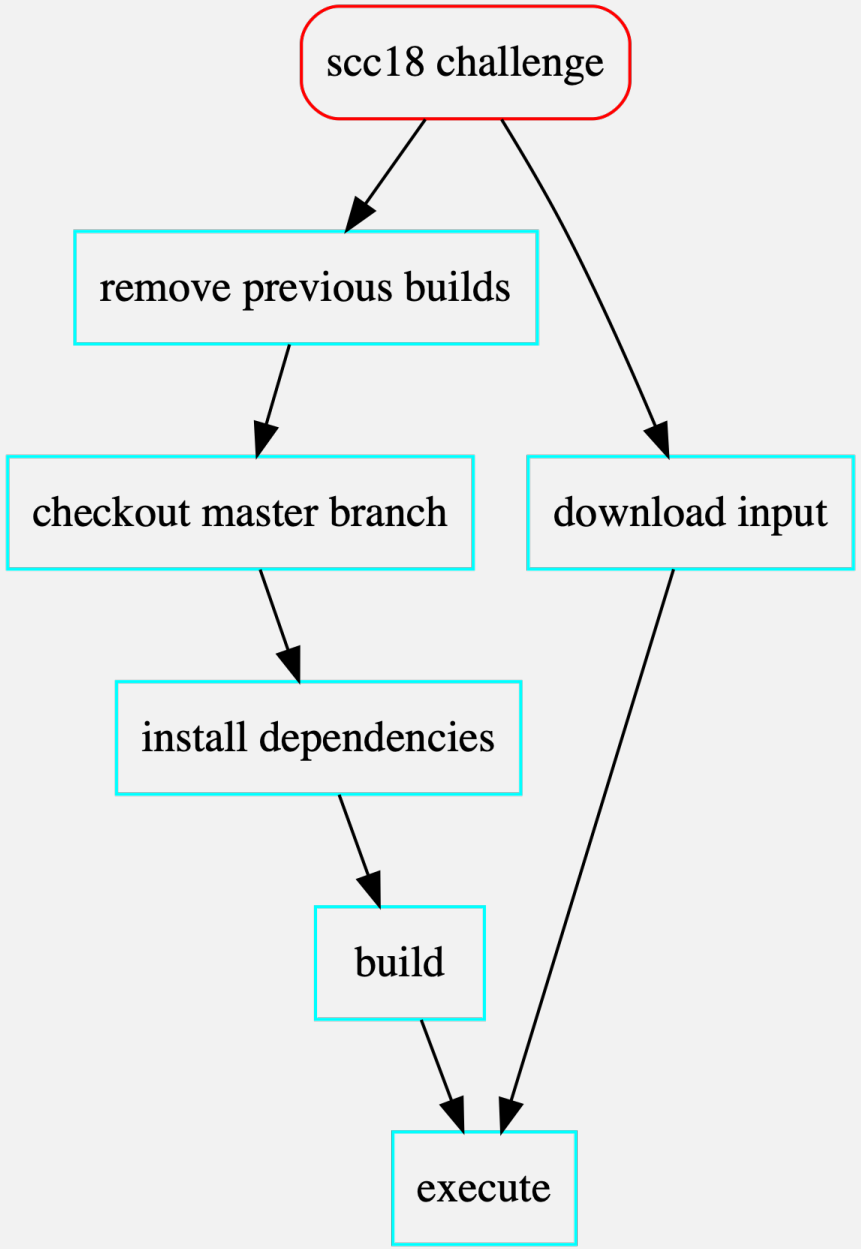
For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/



A ARTIFACT DESCRIPTION: EXTREME SCALE MULTI-PHYSICS SIMULATIONS OF THE 2004 SUMATRA MEGATHRUST EARTHQUAKE

A.1 Abstract

This artifact description contains information about the complete workflow required to set up simulations with the Shaking Corals version of SeisSol. We describe how the software can be obtained and the build process as well as necessary preprocessing steps to generate the input dataset for the node level performance measurements. Input datasets for the scaling and production runs are not publicly available due to their size. In addition, the artifact description outlines the complete workflow from the raw input data to the final visualization of the output.

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** Arbitrary high-order DERivative Discontinuous Galerkin (ADER-DG) with clustered local time stepping.
- **Program:** SeisSol (www.seissol.org); version: *Shaking Corals*.
- **Compilation:** Intel C/C++ and Fortran Compiler.
- **Binary:** -
- **Data set:** CAD model of Sumatra subduction zone and Bay of Bengal assembled with GoCAD; Mesh generated with the Simulation Modeling Suite from Simmetrix (http://simmetrix.com/); see Section 5.1 for input data sets concerning topography, etc.
- **Run-time environment:** Lenovo NeXtScale nx360M5 WCT with IBM MPI (SuperMUC), Cray XC40 with Cray MPI (Shaheen II and Cori)
- **Hardware:** Optimized code is available for Intel Haswell, Intel Knights Landing and other Intel architectures. An unoptimized fallback version is also available.
- **Output:** Timings from the log file; optional receiver (seismic stations) and visualization output.
- **Experiment workflow:** See below.
- **Publicly available?** Code and example datasets are publicly available. The original input datasets for this paper are available

A.2.5 Datasets. A setup including a mesh with over 3 million elements for the 2004 Sumatra-Andaman earthquake can be obtained from Zenodo https://dx.doi.org/10.5281/zenodo.439946. Due to the large size of the production-run meshes, these are only available upon request.

A.3 Installation

Shaking Corals uses SCons for compilation and supports various options to customize the binary. The following command compiles the release version using optimized Intel Haswell kernels for convergence order 6. The resulting binary will support a hybrid MPI+OpenMP parallelization and netCDF for mesh initialization. Note that netCDF is optional but recommended for large runs.

```
$ scons order=6 compileMode=release \
      generatedKernels=yes arch=dhsw \
      parallelization=hybrid commThread=yes \
      netcdf=yes
```

To get a full list of all available options, run:

```
$ scons --help
```

For a more detailed description, see https://github.com/SeisSol/SeisSol/wiki.

A.4 Experiment workflow

SeisSol requires at least three input files: The file `DGPATH`, a parameter file, and a mesh file. `DGPATH` should contain a single text line with the full path to the Maple folder inside the repository.

The parameter file is in Fortran namelist format and contains all parameters required to setup the simulation. Parameters can reference other files, e.g. to specify a list of receiver stations or more complex material parameters. The parameter file for the Sumatra earthquake can be used as a starting point for simulations.

Meshes are usually constructed from CAD models using either the Simulation Modeling Suite from Simmetrix or Gmsh. To convert meshes to the custom netCDF format, the preprocessing tool PUMGen is available on Github https://github.com/TUM-I5/PUML/

SCC18: https://github.com/getpopper/seissol-workflows
SCC19: https://github.com/getpopper/normalmodes-workflows

Thanks!

github.com/getpopper/popper