sam(oa)² Documentation

The sam(oa)² Team

Dec 04, 2020

sam(oa)^2 (https://gitlab.lrz.de/samoa/samoa) is a software package for a dynamically adaptive, parallel solution of 2D partial differential equations on triangular grids. Application include multiphase flow in heterogeneous porous media and tsunami wave propagation (see sam(oa)^2-flash).

CHAPTER 1

Introduction

sam(oa)^2 (https://gitlab.lrz.de/samoa/samoa) is a software package for a dynamically adaptive, parallel solution of 2D partial differential equations on triangular grids. It implements a memory efficient algorithm for grid generation, refinement, and traversal, base on space-filling curves. sam(oa)^2 features efficient adaptive mesh refinement for tree-structured triangular meshes and provides parallelization in shared (using OpenMP) and distributed (via MPI) memory. It has been shown to scale up to thousands of compute cores, with problem sizes that exceed one billion grid cells with dynamic adaptive refinement and coarsening of cells (Meister et al., 2016). Application include multiphase flow in heterogeneous porous media and tsunami wave propagation (see SamoaFlash).

CHAPTER 2

Related publication

Meister, O., Rahnema, K., & Bader, M. (2016). Parallel memory-efficient adaptive mesh refinement on structured triangular meshes with billions of grid cells. ACM Transactions on Mathematical Software (TOMS), 43(3), 1-27, URL, http://delivery.acm.org/10.1145/2950000/2947668/a19-meister.pdf.

CHAPTER 3

Installation

Here we detail some general guidelines for installing sam(oa)^2 and its dependencies. For installing sam(oa)^2-flash, please visit *sam(oa)^2-flash installation*.

3.1 Dependency list

The following prerequisites are necessary for installing and running sam(oa)^2:

- git
- scons
- gfortran 4.7 or higher OR Intel Fortran Compiler 13.0 or higher
- (Optional) ASAGI v0.5.0 or higher for external geodata
- (Optional) Netcdf data files for ASAGI: For porous media flow, download the SPE10 data files from SPE10. A script is included in the data directory that converts them to netcdf files.
- (Optional) curl

3.2 Installing the dependencies

Here we detail some basic guidelines for installing the dependencies.

3.2.1 Linux system with admin privilege

In this case, packages can be straightforwardly installed using a package handling utility, e.g. for debian like systems:

```
sudo apt-get install curl
```

3.2.2 HPC Cluster

Typically, some libaries are preinstalled on HPC clusters through enviroment modules. They can be loaded with the module command, e.g.:

```
module load netcdf
```

It is also possible that some of the required environement variables are not properly set up by the module command. One way to look at what exactly the module command is initializing is to run:

```
module display netcdf
```

For module not already installed on the HPC cluser, a manual installation is required. The same procedure is typically followed. Libraries are installed by cloning a git repository, running cmake, and then make.

Once installed, some environement variables may need to be updated, for the system to be able to find the library. $CPATH and $LIBRARY_PATH may have to be updated for a sucessful compilation and linking of samoa. For example, if the dependency was installed in install-dir:

```
export CPATH=install-dir/include/:$CPATH
export LIBRARY_PATH=install-dir/lib:$LIBRARY_PATH
```

and $LD_LIBRARY_PATH should be updated for samoa to find dynamic libraries at run time:

```
export LD_LIBRARY_PATH=install-dir/lib:$LD_LIBRARY_PATH
```

3.2.3 Accessing github behind a firewall

Some HPC servers restricts access to outside sources and thus does not allow connections to https servers. Here we detail the case of the SuperMUC sever, but this can be adapted to other HPC servers. There are two methods to clone sam(oa)^2 from github on the SuperMUC:

- By accessing the SuperMUC file system as a remote directory. git can then be executed locally:

```
nohup sshfs <login>@supermuc.lrz.de:<samoa_dir> <local_dir>
cd <local_dir>
git clone https://gitlab.lrz.de/samoa/samoa .
```

- By login with remote port forwarding. In this case an alternative URL must be used to clone the git repository:

```
ssh -X <login>@supermuc.lrz.de -R <port>:github.com:9418
cd <samoa_dir>
git clone git://localhost:<port>/meistero/Samoa .
```

This will download the source files for samoa into samoa_dir. This page provide further guidelines for setting a port forwarding.

3.3 Installing samoa

First clone samoa:

```
git clone https://gitlab.lrz.de/samoa/samoa .
```

In order to view all the compilation options sam(oa)^2 provides, you can execute the following command now:

```
cd samoa
scons --help
```

Typical settings are:

```
scons asagi_dir=<asagi_dir> compiler=gnu scenario=darcy -j<threads>
scons asagi_dir=<asagi_dir> compiler=intel target=debug scenario=swe -j<threads>
```

If you wish to simulate simple scenarios that do not require data files you can also disable asagi with the flag

```
scons asagi=No ...
```

Executables will be created in the bin directory.

3.4 SuperMUC-NG

Curl, ASAGI, ImpalaJIT and yaml-cpp have to be manually installed, following the procedure described above. Once done, samoa can be build using:

```
module load numactl/2.0.11-intel hdf5/1.8.20-intel-impi netcdf/4.6.1-intel-impi-
↪hdf5v1.8-parallel
export myLibs=path to installed libraries
export CPATH=$myLibs/include/:$CPATH
export LIBRARY_PATH=$myLibs/lib:$LIBRARY_PATH
export LIBRARY_PATH=$NETCDF_BASE/lib/:$HDF5_BASE/lib/:$LIBRARY_PATH
scons config=my_conf.py
```

3.5 Linux Cluster and MAC Cluster

The following modules should be loaded before compiling ASAGI and sam(oa)^2 on the Linux and MAC clusters

```
module unload gcc python
module load git cmake scons netcdf gcc/4.7
module load gnuplot
```

sam(oa)^2 supports both multithreaded and single-threaded MPI. Both ASAGI and sam(oa)^2 must link to the same respective libraries, thus it is necessary to compile ASAGI twice: once without MT support and once with MT support. Rename the single-threaded library to “libasagi_nomt.so” and the multi-threaded library to “libasagi.so”.

At this point, you should be able to compile ASAGI and sam(oa)^2.

CHAPTER 4

Introduction

sam(oa)^2-flash models tsunami propagation and inundation by solving the depth-integrated (hydrostatic) shallow water equations using adaptive mesh refinement on tree-structured triangular grids. It implements the limiter-based well-balanced second-order Runge-Kutta discontinuous Galerkin method developed by Vater et al. (2019), allowing wave propagation with high accuracy. The scheme is mass-conservative, preserves positivity of the fluid depth and accurately computes small perturbations from the still water state at rest (e.g., tsunami waves). The influence of bathymetry and bottom friction is parameterized through source terms. In particular, bottom friction is parameterized through Manning friction by a split-implicit discretization. It features an accurate and robust wetting and drying scheme for the simulation of flooding and drying events at the coast (Vater and Behrens (2014), Vater et al. (2015, 2019)). Adaptive mesh refinement allows for efficiently tsunami modeling over large domains, while simulatenously allowing for high local resolution.

CHAPTER 5

References

Vater, S., & Behrens, J. (2014). Well-balanced inundation modeling for shallow-water flows with discontinuous Galerkin schemes. In Finite volumes for complex applications VII-elliptic, parabolic and hyperbolic problems (pp. 965-973). Springer, Cham.

Vater, S., Beisiegel, N., & Behrens, J. (2015). A limiter-based well-balanced discontinuous Galerkin method for shallow-water flows with wetting and drying: One-dimensional case. Advances in water resources, 85, 1-13.

Vater, S., Beisiegel, N., & Behrens, J. (2019). A limiter-based well-balanced discontinuous Galerkin method for shallow-water flows with wetting and drying: Triangular grids. International Journal for Numerical Methods in Fluids, 91(8), 395-418

CHAPTER 6

Converting SeisSol output files

SeisSol is a scientific software for the numerical simulation of seismic wave phenomena and earthquake dynamics. We here detail how to convert output files from SeisSol to sam(oa)^2-flash compatible files.

sam(oa)^2-flash reads the ground displacement data in a netcdf file format. SeisSol outputs the ground surface displacement time-histories as an unstructured grid (surface output), and support several file types (Hdf5, posix) and data format (float or double).

To convert such output data to a netcdf file, the displacement-converter can be used. The procedure to compile the `displacement-converter` is detailed in the README file. The `displacement-converter` is fully functional, but some refactoring may be required to merge functionalities on different branches.

The `master` branch allows simply converting a SeisSol surface output to netcdf. It only supports Hdf5 files, in double precision.

The `thomas/tanioka_reconstruction` adds the contribution of the horizontal displacements (Tanioka and Satake, 1996) to the vertical displacements. This branch support any possible output file from SeisSol. The `--fault_x1 --fault_x2 --fault_y1 --fault_y2` options allow defining a rectangular area over which the earthquake displacements are not interpolated. Else, the discontinuity from surface faulting may be smoothed out.

6.1 Spatio-temporal extent of the displacement file

The spatial extent of the displacement file (and of the earthquake scenario) should be large enough to capture the coseismic displacement area. Also, the simulated duration of the earthquake scenario has to be long enough to ensure that all significant seismic waves, in particular, the surface waves, have left the domain. Else, such transient waves may translate into unphysical permanent surface offsets at the end of the surface displacement phase.

CHAPTER 7

sam(oa)^2-flash installation

sam(oa)^2-flash comes in pair with a set of `bash` shell installation scripts, which automatize its installation. The scripts are able to download and install the `hdf5`, `fox`, `netcdf_c`, `netcdf_cpp` and `asagi` dependencies.

First clone samoa and checkout the flash-testing-xdmf branch:

```
git clone https://gitlab.lrz.de/samoa/samoa .
git checkout flash-testing-xdmf
```

The installation scripts are to be found in `script/XDMF`.

7.1 Prerequisites

Important! The default and recommended behavior is to install all libraries into a directory outside of the default system search path. This is to prevent library conflicts with the existing system. However, if you already have a version of HDF5 and/or FoX, ASAGI or NetCDF in your search path (either by installing from the distribution package sources or by loading a module), this might cause trouble. Check your link path and order carefully.

The scripts require `git`, `autotools` and `cmake` in addition to the GNU compiler or intel compiler toolchain.

7.2 Scripts configuration

The scripts can be run in 4 modes of configuration:

- GNU compiler without MPI `gnu nompi`
- GNU compiler with MPI `gnu mpi`
- Intel compiler without MPI `intel nompi`
- Intel compiler with MPI `intel mpi`

The following environment variables may be configured:

- `GIT_PROTOCOL`. May be either `https` (default) or `ssh`. Use this if you have to tunnel your outgoing internet connections (e.g. on SuperMUC-NG, see *Accessing github behind a firewall*).
- `ASAGI_NUMA`. Controls ASAGI NUMA support. Default: `-DNONUMA=1` (disabled NUMA). Set to `-DNONUMA=0` to enable NUMA support.
- `HDF5_URL`. May be used to override the HDF5 git repository URL. Use this if you have no SSH access to the original HDF5 BitBucket repository.

7.3 Installation scripts

- `./install_all_hpc.sh [library directory path]`: Install all required libraries into the specified directory, using the intel compiler with MPI support. If no directory is specified, the default location `~/local` will be used. It is advised to provide an absolute path, however testing whether this is strictly necessary is yet inconclusive. Use this script to setup your HPC environment.
- `./install_all.sh <mpi|nompi> <intel|gnu> [library directory path]`: This script is called by the `./install_all_hpc.sh` script with the parameters `mpi intel`. It can be used to specify the alternative GNU compiler or to specify the MPI support e.g. for testing.
- `./install_lib.sh <name> <mpi|nompi> <intel|gnu> [library directory path]`: This script is repeatedly called by the `./install_all.sh` script. It downloads and installs a specific library in the given configuration.
- `./asagi.sh`, `./fox.sh`, `./hdf5.sh`, `./netcdf_c.sh`, `./netcdf_cxx.sh`: These scripts are not meant to be called directly. Instead, use on of the wrapper scripts above. These scripts contain download and build instructions for the specific libraries. You may edit these in order to adjust the download sources or compilation flags.

After a successful compilation, the specified directory will contain the required libraries in `<directory>/<compiler>/<serial|parallel>/lib` and the header files in `<directory>/<compiler>/<serial|parallel>/include`. Feed this to your linker. For example, if you were to install the libraries to `/opt/samoa_xdmf_libs` using the intel compiler with MPI support, then the libraries would be located at `/opt/samoa_xdmf_libs/intel/parallel/lib`. Further information are available at this link.

7.4 Custom built vs. pre-installed packages

Some Linux distributions do not yet package the recent required versions of HDF5 and FoX. In such cases the compilation from source is recommended.

Using the the pre-installed modules on CoolMUC and SuperMUC-NG is possible. The recommended modules are: - hdf5/1.10.2-intel-impi-threadsafe

For ASAGI: - netcdf/4.6.1-intel-impi-hdf5v1.10-parallel - netcdf-cxx4/4.3.0-intel-impi-hdf5v1.10 - netcdf-fortran/4.4.4-intel-impi-hdf5v1.10

7.5 Examples

- Local testing, using GCC and no MPI:

```
./install_all.sh nompi gnu /opt/samoa_xdmf_libs
```

- Setup on SuperMUC-NG:

```
GIT_PROTOCOL=ssh ASAGI_NUMA='-DNONUMA=0' HDF5_URL='ssh://gitlab.lrz.de/ChristophHonal/
↪hdf5.git' ./install_all_hpc.sh /dss/dsshome1/0D/ga63yos3/samoa_xdmf_libs
```

This command requires a working SSH tunnel to GitHub and LRZ GitLab (see *Accessing github behind a firewall*).

This command overrides the HDF5 repository URL, because the author has no SSH access to the HDF5 bitbucket, and SuperMUC-NG requires all outgoing connections to be via a SSH tunnel. The URL given is accessible to anyone logged in into LRZ GitLab.

This command installs the libraries to `/dss/dsshome1/0D/ga63yos3/samoa_xdmf_libs`, which is in the home directory of the author. Please adjust this to your own.

7.6 sam(oa)^2-flash installation

Once the dependencies have been installed, sam(oa)^2-flash can be installed with scons. In order to view all the compilation options sam(oa)^2 provides, you can execute the following command now:

```
scons --help
```

The compilation can be configurated either inline, e.g.:

```
scons asagi_dir=<asagi_dir> compiler=gnu scenario=asagi -j<threads>
```

or through a configuration python file (here conf.py):

```
scons config=conf.py
```

Provided you are using a branch with proper XDMF support, the Samoa SCons configuration takes the following arguments, for example if the libraries were installed to `/opt/samoa_xdmf_libs`, using the GNU compiler with MPI support:

```
xdmf='true'
xdmf_fox_dir='/opt/samoa_xdmf_libs/gnu/parallel'
xdmf_hdf5_dir='/opt/samoa_xdmf_libs/gnu/parallel'
```

When using ASAGI (*asagi='true'*), you must also configure the ASAGI and NetCDF linker paths:

```
asagi_dir='/opt/samoa_xdmf_libs/gnu/parallel'
netcdf_dir='/opt/samoa_xdmf_libs/gnu/parallel'
```

CHAPTER 8

Parallel run configuration

sam(oa)^2 provides parallelization in shared (using OpenMP) and distributed (via MPI) memory. The number of thread (openMP) is to be set through the environement variable `OMP_NUM_THREADS`.

```
export OMP_NUM_THREADS=nthreads
```

The number of rank is then set with mpirun when running the code:

```
mpirun -n nranks ./samoa_flash options
```

This leads to sam(oa)^2 being executed on nthreads*nranks cpus.

CHAPTER 9

Execution

For execution parameters refer to the online help by calling the executable with '-h' or '–help'.

CHAPTER 10

sam(oa)^2-flash parameters

10.1 Properly resolving islands

Upon initialization, sam(oa)^2 starts with a minimum grid depth (`d = dmin`). The coast cells are refined to the maximum grid depth. A cell is considered coast cell if `abs(bathymetry) < refined_bathymetry` for one of its vertex. Tiny islands may not be properly captured by the vertices of the initial grid if `dmin` is not large enough. In this case, upon refinement (e.g. when the first waves reach the island), such islands would be suddenly resolved. This sudden spurious offset of the bathymetry will source some unphysical waves.

10.2 displacement_height

During the earthquake phase, a stepwise refinement strategy is enforced. Large displacements are highly refined, while small displacements are less refined. The `displacement_height` parameter offers a reference value for evaluating the magnitude of the displacement. If set smaller than the actual maximum displacement, a larger region of the displacement will be refined.

10.3 dry_tolerance

`dry_tolerance` is a threshold below which cells are interpreted as dry and not considered in the timestep computation anymore. Decreasing `dry_tolerance` allows more accurate results at the coast (inundation), but leads to smaller time step size and then longer time to solution. Note also that a too low value of `dry_tolerance` can lead to stability issue: In shallow water equations, the eigenvalues are `p/h + sqrt(gh)` and `p/h - sqrt(gh)`. The dry case is `h->0`. In the analytic case, when `h->0`, so does the momentum and `p/h` is constant. In the numeric case `h->0` does not necessarily come in pair with `p->0`, and `p/h` can diverge. The dry tolerance gives us a maximal eigenvalue `p/dry_tolerance`. In short, `dry_tolerance` constrains the trade-off between accuracy, stability, and simulation time.