OpenSBLI setup guide

David J. Lusher, Satya P. Jammy, Neil D. Sandham

March 17, 2020

1 Installation guide

OpenSBLI requires a Python 2.7 install for code generation and the Oxford Parallel Structured software (OPS) library to build executables. The guide is for Linux platforms only, it hasn't been tested on OSX/Windows. Section 1.1 outlines the libraries that must be installed to use OpenSBLI + OPS, section 1.2 explains the build process of OPS, section 1.3 covers how to generate a code within OpenSBLI and section 1.4 gives the steps to compile and run the code. Section 1.5 has some additional information that may be applicable if running on an HPC cluster.

The installation of libraries will depend on whether you are using a local machine or an HPC cluster. If you are intending to use OpenSBLI on an HPC cluster you will need to load suitable system modules and point OPS to them via the environment variables outlined in section 1.2. MPI and HDF5 libraries must have been compiled with the same compiler or you will face errors during compilation of OPS executables. If you are not sure where the modules are located, please contact your system administrator. **Important note: SymPy version 1.1 is required, newer versions are currently not supported.** The requirements are:

OpenSBLI requirements (Python 2)

```
SymPy version 1.1 : Symbolic Python libary used by OpenSBLI
Matplotlib : Python plotting library
h5py : Python library to read in HDF5 output files for plotting
NumPy : Numerical Python library used in some plot scripts
SciPy: Scientific Python library used in some OpenSBLI classes
```

OPS requirements

```
git : Version control software used to clone the repository
A C compiler : (gcc, icc, Cray, ..)
An MPI distribution : (OpenMPI, MPICH, Intel MPI, ..)
HDF5 libraries : Used for I/O within OPS
CUDA/OpenCL libraries : Required only if you intend to run on GPUs
```

Important note: Codes can be generated with OpenSBLI on a local machine (laptop) and copied onto another machine to be translated, compiled and run with OPS. While you can install both OpenSBLI and OPS on an HPC cluster, it is not necessary to have both on the same machine. OPS however must be installed (and built) on the machine you wish to run simulations on.

1.1 Installation of libraries on an Ubuntu local machine

****Ignore this section if installing OpenSBLI on an HPC cluster****
This section was tested on a clean Ubuntu 18.04 LTS installation, similar packages can be found for other Linux distributions. Python packages are installed with pip, if pip is not already present on your system it can be installed with e.g. `sudo apt-get install python-pip`, on Ubuntu/Debian based systems. Alternatively, software distributions like Anaconda can be used to install the Python libraries. The guide is for installation on a local machine with the GNU C compiler (gcc) and OpenMPI.

1. Install OpenMPI and HDF5 libraries, git and pip/tk

   ```
   sudo apt-get install python-pip python-tk
   sudo apt-get install libhdf5-openmpi-dev
   sudo apt-get install git
   ```

2. Install the required Python packages for this user. Note that SymPy 1.1 is required, more recent versions will cause issues:

   ```
   pip install --user scipy numpy h5py matplotlib
   pip install --user sympy==1.1
   ```

1.2 OPS installation

In this section the installation of the OPS library is explained. **The paths that you put in step 2 will change depending on whether you are on a local machine or an HPC cluster.**

1. Clone the OPS library:
   `git clone https://github.com/OP-DSL/OPS.git`

2. Export OPS environment variables:
   Edit your `~/.bashrc` to include the lines:

   Local Ubuntu machine:

   ```
   export OPS_INSTALL_PATH=/home/<username>/OPS/ops/
   export OPS_COMPILER=gnu
   export OPS_TRANSLATOR=/home/<username>/OPS/ops_translator/c/
   export MPI_INSTALL_PATH=/usr/
   export HDF5_INSTALL_PATH=/usr/lib/x86_64-linux-gnu/hdf5/openmpi/
   export CUDA_INSTALL_PATH=/path/to/cuda-install/
   export USE_HDF5=1
   ```

   HPC cluster: (fill these in)

   ```
   export OPS_INSTALL_PATH=/home/<username>/OPS/ops/
   export OPS_COMPILER=gnu/cray/intel (pick one)
   export OPS_TRANSLATOR=/home/<username>/OPS/ops_translator/c/
   export MPI_INSTALL_PATH=/path/to/mpi-install/
   export HDF5_INSTALL_PATH=/path/to/hdf5-install/
   export CUDA_INSTALL_PATH=/path/to/cuda-install/
   export USE_HDF5=1
   ```

   To make these changes active in the current shell type: `source ~/.bashrc`

3. Build the OPS libraries:

   ```
   Change directory to ~/OPS/ops/c/
   make core seq mpi hdf5_seq hdf5_mpi
   ```

   If you are using CUDA/OpenCL for GPUs you will also have to build:

   ```
   make cuda mpi_cuda opencl mpi_opencl
   ```

   You may see some warnings that can be ignored, the ./lib/ folder should now contain some OPS libraries. If you encounter errors at this point it will be due to improper configuration of the modules/environment variables.

1.3 OpenSBLI installation and code generation

1. Clone the latest OpenSBLI with your username and switch branch:

   ```
   git clone https://github.com/opensbli/opensbli.git
   ```

2. Export the OpenSBLI folder to your PYTHONPATH in `~/.bashrc`:

   ```
   export PYTHONPATH=$PYTHONPATH:/home/<username>/opensbli/
   source ~/.bashrc
   ```

3. Generate a code in OpenSBLI (e.g. Taylor-Green vortex):

   ```
   cd ~/opensbli/apps/taylor_green_vortex/
   python taylor_green_vortex.py
   ```

   You should now have four new files:
   `opensbli.cpp`, `opensbliblock00_kernels.h`, `defdec_data_set.h`, `bc_exchanges.h`
   and some LaTeX files.

At this point the files may be copied onto another machine (if desired) where you have OPS installed. The OPS translation and compilation steps are described in the next section.

1.4 Compiling and running the code

1. Translate the code using OPS to generate parallel versions:

   ```
   python $OPS_TRANSLATOR/ops.py opensbli.cpp
   ```

   You should now have folders for CUDA, MPI and so on, plus an `opensbli_ops.cpp` file. You need to apply the translation step each time you make changes to the opensbli.cpp code generated by OpenSBLI. Ignore the clang-format warning, this is just an optional code indentation formatter and has no effect on the simulation.

2. Copy a Makefile into the directory and build the executable for the desired architecture (MPI, CUDA, OpenMP etc):

   ```
   cp ../Makefile ./
   make opensbli_mpi
   ```

3. Run the executable for number of processes 'np':

   ```
   mpirun -np 4 ./opensbli_mpi
   ```

   The simulation will now run, when finished the output data is written to an `opensbli_output.h5` HDF5 file for post-processing. If you wish to run the simulation again remove the HDF5 file from the directory before repeating the above steps as existing files won't be overwritten.

1.5 Things to note for HPC clusters:

1. For different compilers the `OPS_COMPILER` variable set in section 1.2 must be changed and you may need to modify some of the Makefiles. The easiest way is usually to export cc and CC to where your compiler is located, but you may need to explicitly edit the Makefiles to use icc/icpc for example. In the current OPS (02/2019) the Makefiles are all located in /OPS/makefiles/

2. If there is no suitable HDF5 available on the system you will have to build it yourself:

   ```
   Obtain the source: https://support.hdfgroup.org/HDF5/release/obtainsrc.html
   Extract the file, change to the directory and configure the build.
   For building parallel HDF5:
   ./configure --enable-parallel cc=/path/to/mpicc CC=/path/to/mpicc
   make
   make install
   ```

3. Depending on the machine you may need to add your HDF5/CUDA library locations to your `LD_LIBRARY_PATH` environment variable.

4. Job submission scripts will often need module load commands in them as the compute nodes are separate to login nodes. In addition `LD_LIBRARY_PATH` may need to be exported in the submission script with the HDF5/CUDA library locations.

5. If using gcc you may need to add:

   ```
   -fpermissive to CCFLAGS/CXXFLAGS in OPS/makefiles/Makefile.gnu
   ```