OpenSBLI setup guide

David J. Lusher: d.lusher@soton.ac.uk

May 2, 2018

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, and hasn't been tested on OSX/Windows. 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.

The guide is for installation on a local machine with the GNU C compiler (gcc) with openMPI, on HPC clusters you will have to modify the steps/load the necessary modules to the available compilers, MPI distribution and HDF5 libraries. It is important that the HDF5 installation was built with the same compiler/MPI distrubution that you are using to build the code. For generating CUDA, OpenCL, or OpenACC code, these libraries should be installed and their location exported in step 4. OPS and OpenSBLI are installed to your home directory, if you wish to have them elsewhere then modify the locations accordingly.

1. Install OpenMPI and HDF5 libraries, git and pip/tk (skip this step if using a machine with pre-configured software modules):

   ```
   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:

   ```
   pip install --user sympy scipy numpy h5py matplotlib flake8
   ```

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

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

   ```
   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/
   ```

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

5. Build the OPS libaries:

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

   If you are using CUDA you will also have to build:

   ```
   make cuda
   make mpi_cuda
   ```

6. Clone the latest OpenSBLI with your username and change branch:

   ```
   git clone https://<your-username>@bitbucket.org/spjammy/opensbliweno.git
   cd opensbliweno
   git fetch && git checkout feature/curvilinear
   ```

7. Export OpenSBLI to your PYTHONPATH in `~/.bashrc`:

   ```
   export PYTHONPATH=$PYTHONPATH:/home/<username>/opensbliweno/
   ```

8. Generate a code in OpenSBLI:

   ```
   cd ~/opensbliweno/apps/restructured/TGV/
   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.

9. 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.

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

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

11. 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.

Things to note for HPC clusters:

1. The main difference when using HPC clusters is the locations specified in step 4 will be different. Depending on your machine you will have to load modules for a compiler, MPI distribution, CUDA and HDF5. You will need to find out where on the machine these libaries are located and export them as required.

2. For different compilers the `OPS_COMPILER` variable 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.

3. If there is no HDF5 you will have to build it yourself:

```
Obtain the source at 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
```

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

5. Job submission scripts will often need module load commands in them, and `LD_LIBRARY_PATH` needs to be exported with the HDF5/CUDA library locations.