Compiling Trilinos + MILO

Tom Seidl

September 20, 2016

1 Mac OS X

Determine a directory where software will be installed and create an environment variable in your bashrc for it. For example ...

```
export INSTALL_DIR="/Users/${USER}/local"
```

I had to disable sip on El Capitan (see http://osxdaily.com/2015/10/05/disable-rootless-system-integrity-protection-mac-os-x/) before code would compile and link correctly. You may not have to do this.

1. Reboot your machine and hold down the Command + R keys when then machine is starting up to enter Recovery Mode.
2. Enter the Utilities menu and select Terminal.
3. Type "csrutil disable; reboot" into the command line (no quotes) and hit enter.
4. Restart the computer.

1.1 Required Software

1. CMake (install the dmg file) https://cmake.org
2. Xcode:
   - Install from the App Store
   - In a terminal, execute "xcode-select --install" (no quotes) to install command-line tools
   - Check and apply updates for OS X, XCode, and command-line tools
3. GCC 4.9: Download the binaries from http://hpc.sourceforge.net/. Create a folder in $INSTALL_DIR called gcc-4.9 and then place the bin, include, lib, libexec, and share folders from the download inside of it.

Add the following lines to your bashrc:

```
export PATH="${INSTALL_DIR}/gcc-4.9/bin:${PATH}"
export DYLD_LIBRARY_PATH="${INSTALL_DIR}/gcc-4.9/lib:${DYLD_LIBRARY_PATH}"
```

1.2 TPLs

- Openmpi (1.8.8) https://www.open-mpi.org/software/ompi/v1.8/
- Boost (1.61.0) https://sourceforge.net/projects/boost/files/boost/1.61.0/
- NetCDF (4.4.1) http://www.unidata.ucar.edu/downloads/netcdf/index.jsp

Execute the command below in a terminal

```
echo "using darwin : : ${INSTALL_DIR}/gcc-4.9/bin/g++ ;" >> ~/user-config.jam
```

and create environment variables for your boost, netcdf, and openmpi installations, e.g.

```
export OPENMPI_INSTALL_DIR=${INSTALL_DIR}/openmpi-1.8.8
export BOOST_INSTALL_DIR=${INSTALL_DIR}/boost-1.61
export NETCDF_INSTALL_DIR=${INSTALL_DIR}/netcdf-4.4.1
```

Add the following lines to your bashrc:

```
export PATH="${OPENMPI_INSTALL_DIR}/bin:${PATH}"
export DYLD_LIBRARY_PATH="${BOOST_INSTALL_DIR}/lib:${DYLD_LIBRARY_PATH}"
export DYLD_LIBRARY_PATH="${OPENMPI_INSTALL_DIR}/lib:${DYLD_LIBRARY_PATH}"
```

1.2.1 Building OpenMPI

Execute the following commands in the openmpi source directory

```
./configure --prefix=${OPENMPI_INSTALL_DIR}
make -j4
make install
```

1.2.2 Building Boost

Execute the following commands in the boost source directory

```
./bootstrap.sh --prefix=${BOOST_INSTALL_DIR}
./b2 -j4 install
```

1.2.3 Building NetCDF

Execute the following commands in the netcdf source directory

```
./configure --prefix=${NETCDF_INSTALL_DIR} --disable-netcdf-4 --disable-dap
make -j4
make install
```

1.3 Building Trilinos and MILO

Clone the Trilinos and MILO repos into wherever you keep your source code.

```
git clone https://github.com/trilinos/Trilinos.git trilinos
svn co svn+ssh://software.sandia.gov/svn/private/milo milo
```

Enter the Trilinos source directory and roll back the software to a version that is compatible with MILO

```
git checkout e4b74d
```

Create Trilinos and MILO build directories and copy the appropriate configuration scripts from milo/script into them.

Adjust the configure-trilinos-mac-opt script so that it points to the appropriate directories

In the Trilinos source directory, execute

```
./configure-trilinos-mac-opt
make -j2
make install
```

Enter the milo directory and adjust the configure-milo-mac-opt script so that it knows where the Trilinos installation is and execute

```
./configure-milo-mac-opt
make -j2
```

2 Skybridge

All compiling should be done in your home directory. Jobs are to be run in /gscratch/${USER} .

2.1 Getting an Account

1. Go to computing.sandia.gov and click on the WC Tool tab
2. Click Request or change WCID
3. Enter the BQ project/task (191144/03.01)
4. Select Born Qualified optimization and multiscale simulations (tmwilde is next to it) and hit Request

Your account will be ready to go in 15 minutes. To login to skybridge simply type

```
ssh skybridge
```

2.2 Required Modules

Execute the following commands upon logging into Skybridge (you may want to put these in your bashrc).

```
module rm intel/12.1 openmpi-intel/1.8
module load gnu/4.9.2 openmpi-1.8
```

2.3 TPLs

This build includes the hdf5-enabled version of netcdf and parallel netcdf. The libraries can be obtained using the links below

- CMake (3.6.2) https://cmake.org
- Boost (1.61.0) https://sourceforge.net/projects/boost/files/boost/1.61.0/
- zlib (1.2.8) http://www.zlib.net/
- HDF5 (1.8.16) https://support.hdfgroup.org/HDF5/
- Parallel-netcdf (1.6.1) https://trac.mcs.anl.gov/projects/parallel-netcdf/wiki/Download
- NetCDF (4.4.1) http://www.unidata.ucar.edu/downloads/netcdf/index.jsp

On skybridge, I placed all TPLs in the same install directory ($INSTALL_DIR). There are more of them, and it makes linking in the trilinos build script cleaner, but this set up is not necessary for a functional build.

2.3.1 Building CMake (optional)

If the "stock" CMake on skybridge works this step can be skipped. Execute the following commands in the cmake source directory

```
./bootstrap --prefix=${INSTALL_DIR}
make -j2
make install
```

2.3.2 Building Boost

Execute the following commands in the boost source directory

```
./bootstrap.sh --prefix=${INSTALL_DIR}
./b2 -j2 install
```

2.3.3 Building zlib (optional)

Execute the following commands in the zlib source directory

```
./configure --prefix=${INSTALL_DIR}
make -j2
make install
```

2.3.4 Building hdf5 (optional)

Execute the following commands in the hdf5 source directory

```
CC=mpicc ./configure --with-zlib=${INSTALL_DIR} --prefix=${INSTALL_DIR} --enable-shared \
--enable-production --enable-debug=no --enable-static-exec --enable-parallel
make -j2
make install
```

2.3.5 Building parallel netcdf (optional)

Execute the following commands in the pnetcdf source directory

```
CC=mpicc CFLAGS="-fPIC" ./configure --disable-fortran --prefix=${INSTALL_DIR}
make -j2
make install
```

2.3.6 Building netcdf

The instructions below assume you build all of the optional TPLs (except CMake). If you did not, then use the compilation instructions for netcdf from the Mac section of this document.

In the netcdf source directory, edit include/netcdf.h so that the variables are defined as below:

```
#define NC_MAX_DIMS 65536 /* max dimensions per file */
#define NC_MAX_VARS 524288 /* max variables per file */
```

Execute the following commands in the netcdf source directory

```
CC=mpicc CFLAGS="-I${INSTALL_DIR}/include" CPPFLAGS="-DNDEBUG" LDFLAGS="-L${INSTALL_DIR}/lib" \
./configure --enable-netcdf4 --enable-pnetcdf --disable-fsync --prefix=${INSTALL_DIR} \
--disable-dap --disable-v2
make -j2
make install
```

2.3.7 Building MatIO (optional)

MatIO enables the conversion from exodus files to mat files and vice versa. I have yet to explore this, but it sounds useful.

Execute the following commands wherever you keep source code

```
git clone https://github.com/tbeu/matio.git matio
cd matio
./autogen.sh
export LDFLAGS="-L${INSTALL_DIR}/lib"
 ./configure --with-hdf5=${INSTALL_DIR} --enable-mat73 --enable-shared --prefix=${INSTALL_DIR}
make -j2
make install
```

2.4 Building Trilinos and MILO

Clone the Trilinos and MILO repos into wherever you keep your source code.

```
git clone https://github.com/trilinos/Trilinos.git trilinos
svn co svn+ssh://software.sandia.gov/svn/private/milo milo
```

Enter the Trilinos source directory and roll back the software to a version that is compatible with MILO

```
git checkout e4b74d
```

Create Trilinos and MILO build directories and copy the appropriate configuration scripts from milo/script into them.

Adjust the configure-trilinos-skybridge-opt script so that it points to the appropriate directories

In the Trilinos source directory, execute

```
./configure-trilinos-skybridge-opt
make -j2
make install
```

Enter the milo directory and adjust the configure-milo-skybridge-opt script so that it knows where the Trilinos installation is and execute

```
./configure-milo-skybridge-opt
make -j2
```

2.5 Submitting Jobs

See https://computing.sandia.gov/platforms/sky_bridge/submitting_jobs for comprehensive instructions and tips.

A sample job submission script (skybridge_job_script_example) is located in milo/script .

To submit a job, create a directory somewhere in scratch, copy the job submission script and input files there (unless they are enormous, usually not the case for us), and then execute

```
sbatch skybridge_job_script_example
```

Execute the command below to monitor the status of your job

```
squeue -l | grep ${USER}
```

2.6 Visualization

You can visualize exodus files on skybridge remotely using paraview on your mac laptop. This requires a special build of paraview that is available at https://computing.sandia.gov/paraview/getting_paraview . Download version 5.1.2 . Before installing it, you must remove any paraview installation that currently lives on your mac. This can be done by entering the Applications folder in Finder and dragging paraview into the Trash.

1. Open paraview
2. Under the File tab, click connect
3. Select skybridge and hit the connect button
4. Select the number of nodes desired and time desired, and use FY160007 in the account field. Hit OK.