MOOSE and Ferret Compile Notes

Lukasz Kuna and John Mangeri

February 14, 2019

1 MOOSE Compile on Hornet Cluster

To install your own version of MOOSE on the Hornet Cluster start by creating a projects directory in the location that you would like to run your calculations from:

```
mkdir projects
```

Next enter that directory and clone the newest version of MOOSE:

```
cd projects/
git clone https://github.com/idaholab/moose.git
```

Considering that the Hornet cluster currently has issues with git, a few additional commands and adjustments must be made. First, recreate the libmesh directory inside of MOOSE:

```
cd projects/moose/
rm -r libmesh
git clone https://github.com/libMesh/libmesh.git
```

Next, recreate the MetaPhysicL library inside of libmesh:

```
cd projects/moose/libmesh/contrib
rm -r metaphysicl
git clone https://github.com/roystgnr/MetaPhysicL.git
```

And finally return to the moose directory and execute the following commands:

```
cd projects/moose/
git submodule update --init
git submodule update --init --recursive
```

At this point in time you can return to the projects directory and add the moose compile script provided in the Appendix. Be sure to adjust the path appropriately as well as the SBATCH options.

2 MOOSE Compile on Blues Cluster

First start with the PETSc instructions at
https://mooseframework.org/getting_started/installation/manual_installation_gcc.html
and install to the home directory. Then ensure your /.moose-profile is

```
export PACKAGES_DIR=/home/mangerij/blues/moose-compilers

export CC=mpicc
export CXX=mpicxx
export F90=mpif90
export F77=mpif77
export FC=mpif90

export ARCH=gcc
export PETSC_DIR=$PACKAGES_DIR/petsc-3.9.3/
```

assuming your username is mangerij. Next, follow the above prescription (Hornet) for manually pulling and updating the libmesh submodule with git. Then run the libmesh build script and compile/test *directly* on the node. No other build or compile scripts are needed.

3 Appendix

3.1 Moose Compile Script

```
#!/bin/bash -x

#SBATCH --partition=SandyBridgePriority
#SBATCH --ntasks=16
#SBATCH -o moose-compile.out
#SBATCH -e moose-compile.out
#SBATCH --exclude=cn[01-64,105-320]


echo > moose-compile.out

source /apps2/Modules/default/init/bash


module purge
module load zlib/1.2.11 openssl/1.0.2o libcurl/7.60.0 gcc/5.4.0-alt git/2.7.2 \
            mpi/openmpi/1.10.3 cuda/7.5 python/2.7.6 libxml2/2.9.3-gcc \
            boost/1.61.0-gcc-mpi hdf5/1.8.12 petsc/3.8.4-gcc-mpi hwloc/5.6.1

export \
    CPPFLAGS=$(echo "-I${INCLUDE//:/ -I}") -g \
    LDFLAGS=$(echo "-L${LD_LIBRARY_PATH//:/ -L}") \
#     MOOSE_JOBS=1 \
    MOOSE_JOBS=$SLURM_CPUS_ON_NODE \
    BOOST_ROOT=/apps2/boost/1.61.0-gcc-mpi \
    HDF5_DIR=/apps2/hdf5/1.8.12 \
    PETSC_DIR=/apps2/petsc/3.8.4-gcc-mpi

export \
    CC=mpicc \
    CXX=mpicxx \
    FC=mpifort \
    F90=mpif90

/shared/sergelab/testprojects/moose/scripts/update_and_rebuild_libmesh.sh

cd /shared/sergelab/testprojects/moose/test
make --jobs=16
./run_tests -j 16
```

3.2 Ferret Compile Script

```
#!/bin/bash -x

#SBATCH --partition=SandyBridgePriority
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH -o ferret-compile.out
#SBATCH -e ferret-compile.out
#SBATCH --exclude=cn[01-64,105-320]
#SBATCH --time=00:30:00


echo > ferret-compile.out

source /apps2/Modules/default/init/bash
module purge
module load zlib/1.2.11 openssl/1.0.2o gcc/5.4.0-alt libcurl/7.60.0 mpi/openmpi/1.10.3 \
            cuda/7.5 python/2.7.6 petsc/3.8.4-gcc-mpi libxml2/2.9.3-gcc \
            boost/1.61.0-gcc-mpi hdf5/1.8.12 hwloc/5.6.1 git/2.7.2

export \
    CPPFLAGS=$(echo "-I${INCLUDE//:/ -I}") -g \
    LDFLAGS=$(echo "-L${LD_LIBRARY_PATH//:/ -L}") \
#    MOOSE_JOBS=1 \
    MOOSE_JOBS=$SLURM_CPUS_ON_NODE \
    BOOST_ROOT=/apps2/boost/1.61.0-gcc-mpi \
    HDF5_DIR=/apps2/hdf5/1.8.12 \
    PETSC_DIR=/apps2/petsc/3.8.4-gcc-mpi

export \
    CC=mpicc \
    CXX=mpicxx \
    FC=mpifort \
    F90=mpif90

cd /shared/sergelab/testprojects/ferret
./configure --prefix=/shared/sergelab/testprojects/ferret
make -j 4  MOOSE_DIR=/shared/sergelab/testprojects/moose/
./run_tests -j 4
```

3.3 Batch Script

```
#!/bin/bash
#SBATCH --partition=SandyBridgePriority        # Name of partition
#SBATCH --ntasks=8                            # Request 48 CPU cores
#SBATCH --time=6:59:00                         # Job should run for up to 2 hours (for example)
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --exclude=cn65,cn77,cn78,cn79,cn80,cn103,cn267,cn266
#SBATCH -o Poly.out
#SBATCH -e Poly.out

echo > Poly.out


source /apps2/Modules/default/init/bash
module purge
module load zlib/1.2.11 openssl/1.0.2o gcc/5.4.0-alt libcurl/7.60.0 mpi/openmpi/1.10.3 \
            cuda/7.5 python/2.7.6 petsc/3.7.3-gcc-mpi libxml2/2.9.3-gcc boost/1.61.0-gcc-mpi \
            hdf5/1.8.12 hwloc/5.6.1 git/2.7.2

export \
    CPPFLAGS=$(echo "-I${INCLUDE//:/ -I}") -g \
    LDFLAGS=$(echo "-L${LD_LIBRARY_PATH//:/ -L}") \
#    MOOSE_JOBS=1 \
    MOOSE_JOBS=$SLURM_CPUS_ON_NODE \
    BOOST_ROOT=/apps2/boost/1.61.0-gcc-mpi \
    HDF5_DIR=/apps2/hdf5/1.8.12 \
    PETSC_DIR=/apps2/petsc/3.8.4-gcc-mpi

export \
    CC=mpicc \
    CXX=mpicxx \
    FC=mpifort \
    F90=mpif90

cd /shared/sergelab/testprojects/ferret

for i in {0..1};
do
mpiexec -n 8 ./ferret-opt -i BTO_mono_dispersion_A1_def.i
done
```