# 11.3 Activity 1

## 11.3.0 Access to a High-Performance Computing Cluster

The activities in this module can be performed on a desktop machine running any Linux/UNIX platform using open-source compilers and software. That said, supercomputer resources are now available to almost anyone in the U.S. academic/research community. We encourage you to explore these resources and try to run some of these applications on an HPC cluster. Click on a link to learn about these resources. Your instructor may have an allocation you can use on one or more of these systems. Or you can request an XSEDE startup allocation.

### Example Resources:

- National Center for Supercomputing Applications (NCSA): Blue Waters
    - About: https://bluewaters.ncsa.illinois.edu/about-blue-waters
    - Hardware: https://bluewaters.ncsa.illinois.edu/hardware-summary
    - Getting Started: https://bluewaters.ncsa.illinois.edu/documentation
- The eXtreme Science and Engineering Discovery Environment (XSEDE)
    - Resources: https://portal.xsede.org/allocations/resource-info
    - Create an XSEDE account: https://portal.xsede.org/#/guest
    - Getting Started: https://portal.xsede.org/documentation-overview

Once you have an account on a HPC cluster, you will need to read through the online documentation to figure how to use ssh dual authentication to login, and learn about the basics of how to run jobs on a supercomputer.

### Detailed Examples:

- Blue Waters: Submitting Jobs and Running Programs on Blue Waters
- Stampede2 (XSEDE): Stampede2 User Guide - TACC User Portal

## 11.3.1 Download and Install PLUTO

You may complete this activity on a local Linux machine, or on a remote server. For the installation on a remote cluster, use your login node. In this example, we use Blue Waters. This will work on XSEDE systems, and many other HPC systems. Substitute username for your Blue Waters account name. You will need DUO authentication.

```
local> ssh username@bw.ncsa.illinois.edu
```

```
username@h2ologin2: git clone https://github.com/JohannesBuchner/PLUTO.git
```

The default shell on Blue Waters and XSEDE is bash. You will need to add an environment variable to use PLUTO. You will need to edit text files. Blue Waters has the following text

editors: vi (vim), nano and emacs. Learn one. Many programmers use vi and nano. In place of vi in the instructions, substitute your favorite editor.

```
username@h2ologin2: vi .bashrc
```

Add this line to the end of file:
`export PLUTO_DIR=$HOME/PLUTO`

```
username@h2ologin2: source .bashrc
username@h2ologin2: cd $PLUTO_DIR
username@h2ologin2: ll
```

This will list the PLUTO directory structure (ls is a short list, ll is a long list). For now we will work directly in the Test_Problems directory. Later you will copy files into your scratch directory, and run compute jobs there.

### 11.3.1 Run a PLUTO Test Problem

The PLUTO 4.3 User Guide can be found here: http://plutocode.ph.unito.it/userguide.pdf
We will follow Chapter 0 in the User Guide.

```
username@h2ologin2: cd $PLUTO_DIR/Test_Problems/HD/Sod
username@h2ologin2: cp pluto_01.ini pluto.ini
username@h2ologin2: cp definitions_01.h definitions.h
username@h2ologin2: python $PLUTO_DIR/setup.py
```

Choose `Auto-Update`, then select `Linux.gcc.defs`, and enter. This will create a makefile. Make will compile the code for this test problem. We will run the pluto executable on the login node. Later we will run the test code with MPI on the compute nodes.

```
username@h2ologin2: make
```

If the compilation worked (no errors, warnings are ok), it will produce a bunch of object .o files, and an executable pluto. You may delete .o files. Run pluto.

```
username@h2ologin2: make clean
username@h2ologin2: ./pluto
username@h2ologin2: ll
```

Check the output. This should produce two data files: `data.0000.dbl` and `data.0001.dbl`

The default output format in PLUTO is not readable by VisIt, so we will change the output format to VTK, which is. Edit the `pluto.ini` file so that the `[Static Grid Output]` looks like:

```
[Static Grid Output]

uservar    0
dbl        -1.0  -1   single_file
flt        -1.0  -1   single_file
vtk        90.5  -1   single_file
tab        -1.0  -1
ppm        -1.0  -1
png        -1.0  -1
log         100
analysis   -1.0  -1
```

Then remake pluto and rerun pluto.

```
username@h2ologin2: python $PLUTO_DIR/setup.py
username@h2ologin2: make
username@h2ologin2: rm *.o
username@h2ologin2: ./pluto
username@h2ologin2: ll
```

Check the output. This should produce two data files:

```
data.0000.vtk
data.0001.vtk
```

## 11.3.2 Compile PLUTO with MPI

Each cluster has different rules about how to compile parallel code. See the instructions for choosing and optimizing compilers on your cluster, and go to the appropriate subsection below.

### 11.3.2.1 MPI on Blue Waters

On Blue Waters, there is a general purpose wrapper code to compile serial and parallel code using a variety of compilers: Cray, GNU, PGI and Intel. If you are using Blue Waters, then complete this section. First we need to tell PLUTO how to compile MPI code.

```
username@h2ologin2: vi $PLUTO_DIR/Config/Linux.mpicc.defs
```

Change the `CC` flag to look like this:

```
#############################################################
#
#
#     Configuration file for mpicc (parallel)
#
```

```
#############################################################################
###
CC       = cc
CFLAGS   = -c -O3
LDFLAGS  = -lm

PARALLEL = TRUE
USE_HDF5 = FALSE
USE_PNG  = FALSE
```

Still in the PLUTO/Test_Problems/HD/Sod directory:

```
username@h2ologin2: python $PLUTO_DIR/setup.py
```

Select: `Change makefile`
Select: `Linux.mpicc.defs`
Select: `Auto-update`

```
username@h2ologin2: make
username@h2ologin2: make clean
```

Note that compiling with MPI takes longer, and may generate more warnings than gcc. Most systems will not let you run parallel executables from the login node. So now we have to learn how to run our code on the compute nodes in batch mode.

#### 11.3.2.2 mpicc

Many clusters like Stampede 2 use mpicc to compile MPI code.

```
login4.stampede2(1020)$ which mpicc
/opt/apps/intel18/impi/18.0.2/bin/mpicc
```

However, if your installation returns:

```
/usr/bin/which: no mpicc in …
```

Then you will need to load an MPI compiler. This is often done using the module command.

```
login4.stampede2(1020)$ module avail mpi
login4.stampede2(1020)$ module load impi
login4.stampede2(1020)$ which mpicc
```

Assuming mpicc is available on your system, you can proceed to compile your code. Still in the PLUTO/Test_Problems/HD/Sod directory:

```
username@h2ologin2: python $PLUTO_DIR/setup.py
```

Select: `Change makefile`
Select: `Linux.mpicc.defs`

Select: `Auto-update`

```
username@h2ologin2: make
username@h2ologin2: make clean
```

Note that compiling with MPI takes longer, and may generate more warnings than gcc. Most systems will not let you run parallel executables from the login node. So now we have to learn how to run our code on the compute nodes in batch mode.

## 11.4 Run PLUTO with MPI on compute nodes

Each cluster has its own way of scheduling and managing jobs on the compute nodes. There are two widely used workload managers: Slurm and the Portable Batch System (PBS). Both are functionally similar and easy to use. Blue Waters uses PBS. Stampede2 for example uses Slurm.

### 11.4.1 Submit a job with the Portable Batch System (PBS)

The command for submitting a job is `qsub`. The command for running the job on the compute nodes on Blue Waters is called `aprun`. On other systems, the command is called `mpirun`. Check the user guide on your system. You will create a simple `qsub` script to setup and run your job. Find out how many physical cores exist on each node on your cluster. On Blue Waters, each node contains two sockets (CPUs) and each socket has 16 physical cores. So `ppn=32`. The minimum `walltime` on Blue Waters is 5 minutes. Edit the script accordingly:

```
username@h2ologin2: vi pluto.pbs
```

```
#!/bin/bash
#PBS -l nodes=1:ppn=32:xe
#PBS -l walltime=00:05:00
#PBS -N pluto

cd $PBS_O_WORKDIR

aprun -n 32 ./pluto
```

Now submit the job with `qsub`. Copy and paste the job name. Check the status of the job using `qstat`. The Status column can read Q (queued), R (running), and C (complete).

```
username@h2ologin2: qsub pluto.pbs
username@h2ologin2: qstat <job>.bw
```

### 11.4.2 Submit a job with Slurm

The command for submitting a job is `sbatch`. The command for running the job on the compute nodes on Stampede2 is called `aprun`. On other systems, the command is called `mpirun`.

Check the user guide on your system. You will create a simple `sbatch` script to setup and run your job. Find out how many physical cores exist on each node on your cluster. On the SKX nodes on Stampede2 (including `skx-normal` queue), each node has 2 sockets (CPUs), and each socket has 24 processors (cores). So, `-n 48` and `-N 1`.

On the KNL nodes (including the `normal` queue), each node can run up to 64 MPI tasks per node. Start with 32, so `-n 32`. Edit the script accordingly:

```
username@h2ologin2: vi pluto.slurm
```

```
#!/bin/bash
#SBATCH -p normal    #chooses the queue type normal
#SBATCH -n 32        #number of cores
#SBATCH -N 1         #number of nodes
#SBATCH -t 00:05:00  #max time to run

ibrun ./pluto
```

Now submit the job with `sbatch`. Check the status of all your jobs using `squeue` or `showq`.

```
username@h2ologin2: sbatch pluto.slurm
username@h2ologin2: showq -u
```

## 11.5 Check your output

Congratulations, you have run your first MPI simulation. PLUTO should have created two output .vtk files, and may have created a log file for each processor, depending on your system. Look for the .vtk files in the Sod directory (see screenshot below).

```
mgagne@h2ologin1:~> cd PLUTO/Test_Problems/HD/Sod/
mgagne@h2ologin1:~/PLUTO/Test_Problems/HD/Sod> ll
total 10796
-rw------- 1 mgagne EOT_bbct     9600 Jun 22 08:27 data.0000.dbl
-rw------- 1 mgagne EOT_bbct     6714 Jun 22 10:44 data.0000.vtk
-rw------- 1 mgagne EOT_bbct     9600 Jun 22 08:27 data.0001.dbl
-rw------- 1 mgagne EOT_bbct     6714 Jun 22 10:44 data.0001.vtk
-rw------- 1 mgagne EOT_bbct      126 Jun 22 08:27 dbl.out
-rw------- 1 mgagne EOT_bbct     1002 Jun 22 07:35 definitions_01.h
-rw------- 1 mgagne EOT_bbct     1002 Jun 22 07:35 definitions_02.h
-rw------- 1 mgagne EOT_bbct     1024 Jun 22 07:35 definitions_03.h
-rw------- 1 mgagne EOT_bbct     1024 Jun 22 07:35 definitions_04.h
-rw------- 1 mgagne EOT_bbct     1064 Jun 22 07:35 definitions_05.h
-rw------- 1 mgagne EOT_bbct     1112 Jun 22 07:35 definitions_06.h
-rw------- 1 mgagne EOT_bbct     1119 Jun 22 07:35 definitions_07.h
-rw------- 1 mgagne EOT_bbct     1176 Jun 22 07:35 definitions_08.h
-rw------- 1 mgagne EOT_bbct     1180 Jun 22 07:35 definitions_09.h
-rw------- 1 mgagne EOT_bbct     1002 Jun 22 10:26 definitions.h
-rw------- 1 mgagne EOT_bbct    19468 Jun 22 10:44 grid.out
-rw------- 1 mgagne EOT_bbct     2762 Jun 22 07:35 init.c
-rwx------ 1 mgagne EOT_bbct     3180 Jun 22 10:26 makefile
-rwx------ 1 mgagne EOT_bbct 10613408 Jun 22 10:27 pluto
-rw------- 1 mgagne EOT_bbct      911 Jun 22 07:35 pluto_01.ini
-rw------- 1 mgagne EOT_bbct      918 Jun 22 07:35 pluto_02.ini
-rw------- 1 mgagne EOT_bbct      912 Jun 22 07:35 pluto_03.ini
-rw------- 1 mgagne EOT_bbct      918 Jun 22 07:35 pluto_04.ini
-rw------- 1 mgagne EOT_bbct      920 Jun 22 07:35 pluto_05.ini
-rw------- 1 mgagne EOT_bbct      913 Jun 22 07:35 pluto_06.ini
-rw------- 1 mgagne EOT_bbct      912 Jun 22 07:35 pluto_07.ini
-rw------- 1 mgagne EOT_bbct      888 Jun 22 07:35 pluto_08.ini
-rw------- 1 mgagne EOT_bbct      888 Jun 22 07:35 pluto_09.ini
-rw------- 1 mgagne EOT_bbct     3991 Jun 22 10:44 pluto.0.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.10.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.11.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.12.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.13.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.14.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.15.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.16.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.17.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.18.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.19.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.1.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.20.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.21.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.22.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.23.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.24.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.25.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.26.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.27.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.28.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.29.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.2.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.30.log
-rw------- 1 mgagne EOT_bbct     3968 Jun 22 10:44 pluto.31.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.3.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.4.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.5.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.6.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.7.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.8.log
-rw------- 1 mgagne EOT_bbct     3967 Jun 22 10:44 pluto.9.log
-rw------- 1 mgagne EOT_bbct        0 Jun 22 10:44 pluto.e11316026
-rw------- 1 mgagne EOT_bbct      911 Jun 22 10:26 pluto.ini
-rw------- 1 mgagne EOT_bbct      474 Jun 22 10:44 pluto.o11316026
-rw------- 1 mgagne EOT_bbct      119 Jun 22 10:40 pluto.pbs
-rw------- 1 mgagne EOT_bbct      256 Jun 22 08:27 restart.out
-rw------- 1 mgagne EOT_bbct      407 Jun 22 08:09 sysconf.out
-rw------- 1 mgagne EOT_bbct       72 Jun 22 08:27 test.e11315944
-rw------- 1 mgagne EOT_bbct   126420 Jun 22 08:27 test.o11315944
-rw------- 1 mgagne EOT_bbct      126 Jun 22 10:44 vtk.out
mgagne@h2ologin1:~/PLUTO/Test_Problems/HD/Sod>
```