Lab 1: Stampede User Environment

Allocations:

TACC->TRAINING-HPC

XSEDE->TG-TRA140011

May, 2015

Antia Lamas-Linares

TACC

THE UNIVERSITY OF TEXAS AT AUSTIN

Part 0 – Grab the Lab Files

- Login to Stampede

```
$ ssh <username>@stampede.tacc.utexas.edu
```

- Change to your $WORK directory:

```
$ cdw
$ pwd
$ module list
```

- Untar the file lab1_env.tar file (in ~train00) into your directory:

```
$ tar xvf ~train00/stampede_user_env_lab.tar
```

- Move into the newly created lab1_env directory:

```
$ cd stampede_user_env_lab
$ pwd
$ ls
```

Part 1 – Run a sanitycheck

- Load the “sanitytool” module

  $ module load sanitytool

- Run a sanitycheck in silent mode

  $ sanitycheck

- Run a sanitycheck in verbose mode

  $ sanitycheck -v

  You expect to see some thing like:

  sanitycheck Version: 1.0.1 @git@ 2014-05-09 20:00

  …..

  All tests passed

Part 2 – Using Modules

- List current loaded modules

  $ module list
- List available modules

  $ module av
- List the help information of one module

  $ module help hdf5 #or any modules you are interested
- Load extra modules

  $ module load fftw2 matlab #or any other modules you need
- Save to your default modules

  $ module sd

Use the command "**module reset**" to restore to system defaults.

Part 3 – Run an MPI Batch Job (sbatch)

- Compile the mpipi program:

```
$ mpicc mpipi.c -o mpipi
```

- Open the batch script in an editor to see if you need to change it:

```
$ nano mybatch     # or vi, or emacs
```

<< Change the project "-A" if necessary>>

- Launch the batch job

```
$ sbatch mybatch
```

- Monitor the job's status (when done, command will return nothing):

```
$ squeue -u <username>
$ showq | less     # hit space bar to advance
$ squeue | less    # hit space bar to advance
```

- When job completes, take a look at results:

```
$ ls                      # Note presence/names of output files
$ less mpipi.xxxxx.out    # "xxxxx" is your job's jobid
$ less mpipi.xxxxx.err    # "xxxxx" is your job's jobid
```

Part 4– An Interactive Session (idev)

- Compile hello.F90 with and without Open MP.

```
$ ifort -openmp hello.F90 -o hello1
$ ifort hello.F90 -o hello2
```

- Use idev:

```
$ idev -A TRAINING-HPC
```

- Run the code:

```
$ ./hello1      # you're on a compute node, not a login node
$ ./hello2
```

- Set OpenMP threads and try again:

```
$ export OMP_NUM_THREADS=4
$ ./hello1
$ ./hello2
```

- Clear the thread count and try again:

```
$ unset OMP_NUM_THREADS
$ ./hello1
$ ./hello2
```

Part 5 (advanced) – Run MIC App from the Host

- Open a new terminal and recompile to produce "native MIC" code (compilers are not visible from the MIC):

```
$ ifort -mmic -openmp hello.F90 -o hello.mic
```

- Back to the previous terminal, launch the MIC code from the host:

  On the Computenode:

```
$ ./hello.mic
```

  Note: the program reports 244 “processors” because each MIC core has four hardware threads. It may not be efficient to run this many threads.

- From the host, modify the MIC thread count and try again:

```
$ export MIC_ENV_PREFIX=MIC
$ ./hello.mic
```

Part 6 (advanced) – Visit the MIC

- First note the full path to your working directory:

```
$ echo $WORK  # you'll need this info when you get to the MIC
```

- Go the MIC using ssh:

```
$ ssh mic0    # the "zero" identifies the MIC card
```

- Move into the stampede_user_env directory with explicit cd:

```
$ cd /work/12345/yourname #replace with your own path
$ cd stampede_user_env_lab
```

- Run your MIC code:

```
$ ./hello.mic
```

- It fails because it cannot find libiomp5.so.

```
$ newpath="/opt/apps/intel/13/composer_xe_2013.2.146/compiler/
  lib/mic"
$ export LD_LIBRARY_PATH=$newpath:${LD_LIBRARY_PATH}
$ ./hello.mic
```

- End session:

```
$ exit    # to return to host
$ exit    # to end idev session
```