Lab 1

Stampede Orientation

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.tar file (in ~train00) into your directory:

  ```
  $ tar xvf ~train00/lab1.tar
  ```

- Move into the newly created lab1 directory:

  ```
  $ cd lab1 # first char is lower case "L"; last is a one
  $ pwd
  $ ls
  ```

Part 1 – 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 lab1batch     # or vi, or emacs, or just cat lab1batch
  ```
  << you shouldn't need any changes >>

- Launch the batch job

  ```
  $ sbatch lab1batch
  ```

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

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

- When job completes, take a look at results:

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

TACC

THE UNIVERSITY OF
TEXAS
AT AUSTIN

Part 2 – An Interactive Session (srun)

- Launch a one-node interactive session in the development queue

  ```
  $ srun -n 16 -t 00:15:00 -p development --pty /bin/bash -l
        # last char is lower case "el"
  ```

     if system asks for a project code, modify the call by adding the following flag

     (it must occur before /bin/bash):

  ```
  -A 20131204MIC
  ```

     same code to be used in job script

- When session begins, compile hello.F90* from compute node:

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

- Run the code:

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

- Set OpenMP threads and try again

  ```
  $ export OMP_NUM_THREADS=4
  $ ./hello
  ```

*Note: the capital "F" in the suffix allows the compiler to interpret correctly the macros in the source code. If the suffix were "f90" the compilation would require a "-cpp" flag.

Part 3 – Run MIC App from the Host

- While on the compute node, recompile to produce "native MIC" code (compilers are not visible from the MIC):

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

- Launch the MIC code from the host:

  ```
  $ ./helloMIC
  ```

    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_OMP_NUM_THREADS=60
  $ export MIC_ENV_PREFIX=MIC
  $ ./helloMIC
  ```

Part 4 – 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 lab1 directory with explicit cd (alias and env variable not avail):

  ```
  $ cd /work/01875/djames     # replace with your own path
  $ cd lab1
  ```

- Run your MIC code:

  ```
  $ ./helloMIC
  ```

- Change the MIC's thread count and run code again (don't use "MIC" prefix):

  ```
  $ export OMP_NUM_THREADS=25
  $ ./helloMIC
  ```

- Return to host, then end srun session as desired:

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

THE UNIVERSITY OF TEXAS AT AUSTIN