CITS3402 High Performance Computing
Laboratory Sheet 5

CITS3402 High Performance Computing

How to run MPI code on the cluster

- The head node is : `ecm-ubl-006.uniwa.uwa.edu.au`
- There are home directories set up for you at the head node. There is no storage in the other nodes. It is possible to synchronize your files in the other nodes, that I will explain later.
- The other nodes are : `node-1` to `node-29`

First `ssh` to the head node.

Then create a host file `host` and write the names of the nodes in this file, one node in each line. For example:
```
node-1
node-2
```
will create a cluster consisting of two nodes. You can add up to `node-29`

Now write some MPI code. For example, here is a simple MPI code that computes Fibonacci numbers. There is no parallelism here, just each MPI process independently computes the task. You can experiment with different Fibonacci numbers, the code as it is computes `fib(42)`.

Compile the code by `mpicc myMPI.c` this will create an executable file `a.out`, or you can use some other name for your executable by `mpicc -o myMPI myMPI.c` for example.

You have to now set up your executable in each node of the cluster. Run `syncCluster`. It may ask your response when you run it first, say 'yes' to everything. It will synchronize seamlessly later.

Run your code at the head node `mpirun --hostfile host myMPI`. The code will run on the cluster you have created in the hostfile called `host`.

If you just execute `mpirun myMPI`, it will run your job only at the head node. You can experiment with different host file configurations. Since each node has 12 cores, MPI creates those many processes on each node.

How to run your code using the queue

- Create a shell script, you can give any name with a `.sh` extension. The name of my script is `runSch.sh` and its contents are:

  ```
  #PBS -l nodes=8:ppn=4
  source /etc/bash.bashrc
  mpirun a.out
  ```

  You can only change the `nodes=8:ppn=4` part, in particular the `8` and `4` values. The first value indicates how many nodes you want to use (the highest is `29`) and the second value indicates how many cores on each node you want to use (`12` is the highest). `a.out` is the name of my executable, your executable may have another name. Of course, you have to compile your code as mentioned above. You must not change anything else in the script.

- `qsub runSch.sh` (assuming the name of the shell script is `runSch.sh`) is the command to submit the job to the queue. You will get a message like `124.ECM-UBL-006.uniwa.uwa.edu.au` on your console, that indicates that the job is submitted to the queue. You will get a file in your directory called `runSch.sh.o123` or something similar when the execution is complete. This file will have all the results of your execution.
- There are a few other options you may use. For example, you can send yourself an email on start, abort or completion of your job:
```
#PBS -m abe
#PBS -M your-email
```

Amitava Datta
September 2018

School of Computer Science & Software Engineering
The University of Western Australia
Crawley, Western Australia, 6009.
Phone: +61 8 9380 2716 - Fax: +61 8 9380 1089.
CRICOS provider code 00126G