LAB : Introduction to MPI

Ritu Arora

Email: rauta@tacc.utexas.edu

February 6th, 2012

Introduction

- You will learn
  - How to write MPI code
  - How to compile and execute MPI code on Ranger & Lonestar

- What will you do
  - Compile and execute the code for the programs discussed in the lecture and exercises
  - Modify the code for the exercises to embed MPI routines

Assumptions

- You have access to TACC resources
  - If not, please create a TACC portal account and submit a request for allocation

    https://portal.tacc.utexas.edu/
  - If you are using your PC or laptop and do need or have a TACC portal account, please check the documentation of your MPI installation for appropriate MPI commands to compile (*e.g.*, `mpicc`) and run the code (*e.g.,* `mpirun,` and `mpiexec`)
  - If you have access to some other cluster and do need or have a TACC portal account, please check the user-guide of your cluster for instructions to compile and run the MPI jobs . You might have to use a different job script than what was discussed in the lecture

- In case you are using TACC resources, you are familiar with basic Linux commands

Accessing Lab Files
(if you DO NOT have access to TACC Resources)

- Download the file **`trainingMPI.tgz`** from the following link:

  http://www.tacc.utexas.edu/user-services/training/course-materials

- Uncompress the tar file, **`trainingMPI.tgz`**

- Switch to the directory **`trainingMPI`**

Accessing Lab Files (if you have access to TACC Resources)

- Log on to Ranger or Lonestar using **`your_login_name`**

This is your existing login, or the portal login (and password) you recently created.

- Uncompress the tar file, **`trainingMPI.tgz`** , that is located in the **`~train00`** directory into your HOME directory.

```
ssh <your_login_name>@ranger.tacc.utexas.edu

tar -xvzf ~train00/trainingMPI.tgz

cd trainingMPI
```

Exercise 1

- **Objective**: Learn to compile and run MPI code

- Compile the sample code `mpiExample4.c`

```
login3$ mpicc -o mpiExample4 mpiExample4.c
```

- Modify the job script, `myJob.sh`, to provide the name of the executable to the `ibrun` command

- Submit the job script to the SGE queue and check it's status

`login3$ qsub myJob.sh` (you will get a `job id`)

`login3$ qstat` (check the status of your job)

- When your job has finished executing, check the output in the file `myMPI.o<job id>`

Exercise 2

- **Objective**: Learn to use the MPI wall-clock timer

- Switch to the subdirectory **`exercise`** within the directory **`trainingMPI`**

  ```
  login3$ cd exercise
  ```

- Modify the code in the file `mpiExample5.c`
  - Read the comments in the file for modifying the code
  - You have to insert two calls to the MPI timer routine , extend the variable declaration section, and modify the print statement
  - By measuring the difference in the value of time retuned by the two calls to the MPI timer routine, you can find the total time taken to execute a section of the code
- Compile the code and execute it via job script (see Exercise 1 for details)
- Notice the output (time taken) that gets printed in the output file

Exercise 3

- **Objective**: Notice the difference between `MPI_Allreduce` and `MPI_Reduce` routines

- Assumption: you are in the subdirectory **`exercise`** within the directory **`trainingMPI`**
- Modify the code in the file `mpiExample8.c`
  - Read the comments in the file for modifying the code
  - You have to replace the MPI_Reduce routine with the MPI_Allreduce routine
- Compile the code and execute it via job script (see Exercise 1 for details)
- Compile and execute `mpiExample4.c`
- Compare the output of `mpiExample8.c` with the output of `mpiExample4.c`
  - Notice the value of the variable `sumTotal`