Lab: Introduction to MPI (For C Programmers)

Ritu Arora

Texas Advanced Computing Center

April 18th, 2013

Email: rauta@tacc.utexas.edu

Introduction

- You will learn
  - How to write MPI code
  - How to compile and execute MPI code on Stampede

- 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

Accessing Lab Files

- Log on to Stampede 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>@stampede.tacc.utexas.edu

tar -xvzf ~train00/trainingMPI.tgz

cd trainingMPI
```

Please Note

- For the students who registered through the **TACC** User Portal, the project is **20130418HPC**

- For the students who registered through the **XSEDE** User Portal, the project is **TG-TRA120007**

- In the job submission script, provide the project number (replace “A-xxxxx” in “-A A-xxxxx”) that is relevant to your case.

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 and the project allocation
- Submit the job script to the SLURM queue and check its status

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

  `login3$` **`squeue -u <your-user-name>`** (check job status)
- 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 returned 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**: **Learn to use MPI collective calls :** `MPI_Allreduce`
- Modify the code in the file `mpiExample8.c` in the subdirectory **`exercise`** within the directory **`trainingMPI`**
  - Read the comment in the file for modifying the code
  - You have to replace the `MPI_Reduce` routine with `MPI_Allreduce`
  - See the lecture slides for details on the `MPI_Allreduce` routine
- Compile the code and execute it via job script (see Exercise 1 for details)

Exercise 4

- **Objective**: **Learn to use MPI collective calls :** `MPI_Bcast`
- Modify the code in the file `mpiExample11.c` in the subdirectory **`exercise`** within the directory **`trainingMPI`**
  - Read the comment in the file for modifying the code
  - You have to call the `MPI_Bcast` routine as instructed
  - See the lecture slides for details on the `MPI_Bcast` routine
- Compile the code and execute it via job script (see Exercise 1 for details)