

Lab 11: Message Passing Interface (MPI)

Instructor: Vivek Sarkar

Resource Summary

Course wiki: <https://wiki.rice.edu/confluence/display/PARPROG/COMP322>

Staff Email: comp322-staff@mailman.rice.edu

Important tips and links:

edX site : <https://edge.edx.org/courses/RiceX/COMP322/1T2014R>

Piazza site : <https://piazza.com/rice/spring2014/comp322/home>

Java 8 Download : <https://jdk8.java.net/download.html>

IntelliJ IDEA : <http://www.jetbrains.com/idea/download/>

HJ-lib Jar File : <http://www.cs.rice.edu/~vs3/hjlib/habanero-java-lib.jar>

HJ-lib API Documentation : <https://wiki.rice.edu/confluence/display/PARPROG/API+Documentation>

HelloWorld Project : <https://wiki.rice.edu/confluence/display/PARPROG/Download+and+Set+Up>

Sugar Login: ssh *your-netid*@sugar.rice.edu and then login with your password

Linux Tutorial visit <http://www.rcsg.rice.edu/tutorials/>

As in past labs, create a text file named lab_11_written.txt in the lab_11 directory, and enter your timings and observations there.

1 MPI Environment Setup

1. Download the lab11.zip file provided on the course wiki, and unzip its contents in the lab_11/ directory.
2. Run the following command in the lab_11/ directory to set up the environment for executing mpiJava programs, “*source setup.txt*”.

2 Matrix Multiply using MPI-Java

Your assignment today is to fill in incomplete MPI calls in a matrix multiply example that uses mpiJava. You should complete all the necessary MPI calls in `MatrixMult.java`, to make it work correctly. There are comments (TODOs numbered 1 to 14) in the code that will help you with modifying these MPI calls. You can look at the slides for Lecture 30 for an overview of the mpiJava `Send()` and `Recv()` calls, and at <http://www.hpjava.org/mpiJava/doc/api> for the API details (click on the “Comm” link).

Though MPI is designed for execution on distributed-memory machines, we will create multiple sequential MPI Processes within a single SUGAR node for the purpose of this lab. Thus, all parallelism will stem from the use of multiple MPI processes within a single SUGAR node.

The steps to compile and run the updated `MatrixMult.java` file on the command line are as follows:

1. Compile the program with the Makefile provided: *make*
2. Run the program with the Makefile provided, using 8 processes: *make run8*
3. Repeat with 1, 2 and 4 processes:
make run1
make run2
make run4

What performance differences do you see for different numbers of processes?