CSCI 3366 (Introduction to Parallel and Distributed Processing), Fall 2005

Homework 1

Assigned: September 7, 2005.

Due: September 16, 2005, at 11:59pm.

Credit: 20 points.

> *Note:* The HTML version of this document may contain hyperlinks. In this version, hyperlinks are represented by showing both the link text, formatted like this, and the full URL as a footnote.

1 Overview

The book briefly mentions that the time to send a message varies with the size of the message. For this assignment, your mission is to write an MPI program that demonstrates this relationship, as described below.

2 Details

2.1 The program

Your program should accept the following command-line arguments:

- The size of the message to send (M — for simplicity, let this be a count of `int`s).
- How many times to send the message (N).

When your program is started with P processes, it should do the following: Process 0 creates a message consisting of M `int`s (content of the message is not important) and sends it to process 1, which in turn sends it to process 2, and so forth, until finally process $P-1$ sends it back to process 0. The program repeats this N times, and then process 0 prints the number of processes, M, N, and the elapsed time for the message to make its N trips around the ring of processes.

2.2 Hints and tips

- Sample program `fancy-hello.c`[1] demonstrates how to access command-line arguments. Remember that you can use `atoi` to convert a string to an integer (read the man page for details), and that you can allocate space for `n` integers with `malloc(sizeof(int)*n)`.
- Sample program `time-msg.c`[2] demonstrates how to time something in an MPI program.
- Using MPI on the Linux Lab Machines[3] describes the commands you need to compile and run MPI programs.

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS3366_2005fall/SamplePrograms/MPI/fancy-hello.c`
[2] `http://www.cs.trinity.edu/~bmassing/Classes/CS3366_2005fall/SamplePrograms/MPI/time-msg.c`
[3] `http://www.cs.trinity.edu/~bmassing/Classes/CS3366_2005fall/Notes/mpi-howto/index.html`

- The course links page[4] contains links to sites where you can find more information about MPI. Notice also that there are man pages for most if not all MPI functions.

2.3 Running the program

Once you have confirmed that your program is operating correctly, use it to see how the time taken to send messages varies with any factor you think might affect it. Some things to try:

- Run the program for several different values of N. Is the time roughly proportional to N?
- Run the program for several different values of M. Is the time roughly proportional to M?
- Run the program with different numbers of processes. Is the time roughly proportional to the number of processes?
- Run the program on different collections of computers (e.g., on several of the Janus machines versus several of the Xena machines). Does it appear that one group of machines is faster than the other?
- Run the program under varying load conditions (e.g., when no one else is using the same computers versus when at least one is in use for other work — e.g., you are running some other application). Does this affect message-passing time?

Collect at least half a dozen measurements. For each measurement, record:

- Machines used.
- Day and time, and whether anyone else appeared to be using the same machines (you can determine this from the output of the `ruptime` command).
- Output of process 0 as described above.

Record these timing measurements in a text file. An easy way to do this is to redirect the output of the `mpirun` command into a text file, e.g.,

```
mpirun -np 10 myprogram 100 2 >> outfile
```

and then edit the output file. (The `>>` appends the output to `outfile`.)

3 What to turn in and how

Submit your program source and the text file with timing measurements by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 3366 homework 1"). You can develop your program on any system that provides the needed functionality, but I will test it on one of the department's Fedora Core 4 Linux machines, so you should probably make sure it works in that environment before turning it in.

[4] `http://www.cs.trinity.edu/~bmassing/Classes/CS3366_2005fall/HTML/links.html`