CSCI 3366 (Introduction to Parallel and Distributed Processing) Spring 2001

Homework 1

Assigned: January 23, 2001.

Due: January 30, 2001, at 5pm.

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.

Contents

1 Overview

We could use a message-passing program to compute the sum of the numbers from 0 to $N-1$ as follows:

- Each process determines its ID n (in MPI terminology, its rank).
- If the process has $n = 0$, it sends its ID to process 1 and waits to receive a message from process $N-1$. This message, when it arrives, will contain the desired sum; process 0 prints it.
- Otherwise, the process first waits to receive a message (containing the sum of the numbers from 0 through $n-1$) from process $n-1$. When it arrives, the process should add to this sum its own ID (n) and send the result to process $n+1$ (if $n < N-1$) or to process 0 (if $n = N-1$).

If we measure, in process 0, the time that elapses from the beginning of the calculation (before sending 0 to process 1) until the end of the calculation (after receiving the sum from process $N-1$), we will also have a rough estimate of how long it takes to send a message around a ring of N processes.

(Yes, this is a highly inefficient method of performing the calculation. It will, however, serve as a not-too-difficult first venture into message-passing programming and allow you to measure something interesting, namely the time required to send a sequence of messages).

2 Details

2.1 Writing the program

Write a program as described above, using MPI for message-passing. You can write the program in any language that allows calls to the MPI libraries — Fortran, C, or C++. When run on N processors, your program should print:

- The sum of the integers from 0 through $N - 1$.
- The time taken for the computation, measured from just before the first message is sent from process 0 to just after the final message is received in process 0.

Only process 0 should print anything.

You may find it helpful to look at some of the sample programs[1]. You may also find it helpful to review the Tips for Running MPI Programs[2].

2.2 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 and see whether the time taken to send N messages is proportional to N.
- Run the program on different collections of computers (e.g., on several of the Janus machines versus several of the Xena machines) and see whether this affects message-passing time.
- 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) and see whether this affects message-passing time.

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

- N (number of processes).
- Machines used (okay to just list the contents of the file you specified as MPI's `-machinefile` parameter).
- 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).
- Time taken for computation.

Record these timing measurements in a text file.

3 What to turn in

Submit the source code for your program, plus the text file containing your timing measurements, as described in the Guidelines for Programming Assignments[3], using a subject header of "cs3366 hw 1".

[1] `http://www.cs.trinity.edu/~bmassing/CS3366_2001springSamplePrograms/index.html`
[2] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/Notes/tips-mpi/index.html`
[3] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/Notes/pgmguidelines/index.html`