CISC 372: INTRODUCTION TO PARALLEL PROGRAMMING
Fall 2003
Individual Programming Assignment 2
Due Date: start of class, Thursday, October 9, 2003

1 Objectives

The objective of this assignment is to write some simple MPI programs to learn how to accomplish communicaton between processes, and to time MPI programs. The side effect of this assignment is that you will be implementing several versions of a global sum reduction operation.

2 Procedure

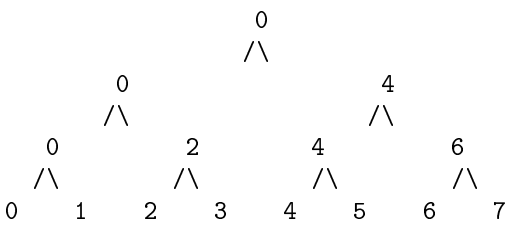
1. **Centralized Summation:** Write an MPI program that computes the global sum of results from computations on the data local to each process in the following way. First, for ease of creating local data, use the rank of each process as its local data. To have some computation at each node, have each node locally compute the summation 1 to 10000 of the local rank (myrank+myrank+...myrank), performing the additions explicitly, not by multiplying 10000 times myrank! We're just creating some fair amount of local computation! The overall centralized summation is the sum of these local sums. Assuming that there are N processes, the global sum should be computed by having processes 1 through N-1 send their local result to process 0, and having process 0 receive their local results and compute the global sum of the local results. The processes should not have to send their local results in any particular order, and process 0 should not have to receive them in any particular order. That is, process 0 should not explicitly try to receive the results in order 1 through N-1. This causes undue synchronization. Your centralized summation program should be written in a general manner in order to work for any number of processes N > 1, without recompilation of the program code.

   Take 3 timings each for 4 processes, 8 processes, and 16 processes. The goal of the timings are to compare the three algorithms. So, in order to compare them, we need to time all of each algorithms starting after the initial MPI_init, comm_rank and comm_size commands, and before the printing of the results. The printing takes up too much time. So, you want to place your timing statements such that process 0 does the calls, and does the first call before it does any work (including its own computation of its local sum), and after it gets all the results from everyone and does the global summation. Tyipcally, process 0 starts on the machine you are logged into and submitting the mpirun command, so it will get started before anyone else, so this will include the complete computation time of all local sums and the global sum and all communication.

2. **Circular Shifting Summation:** Write an MPI program that computes the global sum of the local results of each process in the following way. Again, assume that the rank of the process is the local data and that the same local summation is computed as above. Assuming that there are N processes, process N-1 should send its result to the next lower numbered process N-2, which adds its local results to the running sum (i.e., we call this a partial sum), and sends the current partial sum to the next lower ranked process. This continues until process 0 receives the partial sum from process 1. Process 0 then adds its local sum (I know, it is zero, but we are mimicking summation of local data), and prints out the final global sum. You need to time from the start of process N-1 sending its data, until just before process 0 prints. Take 3 timings for 4 processes, 8 processes, and then for 16 processes. Again, your code should be written in a general manner, able to work for any N > 1 number of processes without recompilation.

3. **Tree Summation:** Write an MPI program that computes the global sum of the local results of each process in the following way. Again, assume that the local data is the rank of the process and that the local result is computed as above. Assume that there is a number of processes, N > 1, that is a power of 2 (i.e., 2, 4, 8, 16). The communication and summation should be performed in a tree-like pattern. The following diagram shows the communication for 8 processes, where the numbers indicate

the process that is doing the summation at that point in the tree computation. So, process 1 sends its local result to process 0, which computes a partial sum, and waits for process 2 to send its partial sum of 2 and 3. Then, process 0 computes the partial sum of its current sum with the partial sum from process 2. Then, process 0 waits for the partial sum from process 4, and computes the final global sum. Each interior node in this logical tree is a partial sum computation. The original local results are at each process at the bottom of this tree of computation. You should time from the start of computing local sums at the bottom of the tree computation until just before process 0 prints the final result. Take 3 timings for 4, 8, and 16 processes.



4. Create a single graph that displays the averages of your timings for 4, 8, and 16 processes, for each method. The horizontal axis should be the number of processes involved in the summation. The vertical axis should be the timings. The graph should include only your average timings from 3 runs, so there should be 9 points on the graph. A bar graph might be a nice depiction of this data. A different bar could be used for each of the three methods, for a given number of processes. You should use a tool that automatically creates the graph based on the data inputs.

5. **Modified Versions of Summation Programs:** Now, copy each of your 3 summation programs, and modify each copied version such that a given node does not start to compute its local sum until it has received data from all processes who are supposed to be sending to it. So, some nodes can begin summing immediately (have no expected received data), while others need to wait until they receive all expected data before they perform their local summation for their partial sum.

   Perform the same timings as before for these three modified programs. Create a single graph as before for the original versions of these programs.

3 Hints for Getting Good Timing Data

In order to get reliable timing data, you need to make sure that no one else is using the cluster. To do this, you should run the `spyall` command just prior to making a performance run. Also, from past experience, leaving the timings until last evenings before the assignment due date makes getting reliable timing runs very difficult to obtain due to the many other folks trying to get access to the cluster for timings.

4 Experimental Report

Your experimental report should consist of the following sections in this order. It is strongly recommended that you type your report using a word processor rather than handing in a hand-written report.

1. **Cover Page:** Title, author, course number and semester, date.
2. **Project Summary:** In one paragraph, summarize the activities of the lab.
3. **Hypothesis:** First, state and justify your hypothesis. Which method for summation do you believe should take the longest time and which should take the shortest time? Why? Explain your hypothesis in the context of a large number of processes and a small number of processes, and a large amount of computation at each node versus a small amount of computation at each node.
4. **Analysis:**

(a) **Collected Timing Data.** This section includes both a chart of the raw collected timings, and the graph depicting the averages pictorially.

(b) **Analysis.** In English, explain your timing results. Describe your observations concerning the different methods, and how it compares to your hypothesis. Explain why you believe the timings are different or similar between the methods. How does it compare to your hypothesis?

5. **Conclusions:** This section consists of a discussion of what you learned from the various aspects of the lab. Discuss possible reasons for inconsistencies or discrepancies in your data versus what you would have expected to happen.
6. **General Remarks:** Any comments on the lab itself that you would like to make in terms of suggestions for improvement or problems you experienced.
7. **Appendix:** Your code for the three versions of summation.

Please staple all parts of your lab together, and label each piece. Be prepared to discuss your results in class.

5 Criteria for Evaluation

Your lab will be evaluated according to the following criteria:

1. 10 pts: Centralized summation
2. 3 pts: Timings of centralized summation
3. 10 pts: Shifting summation
4. 3 pts: Timings of shifting summation
5. 25 pts: Tree summation
6. 3 pts: Timings for tree summation
7. 10 pts: Modified versions of each program above
8. 6 pts: Timings for the modified versions
9. 5 pts: Graph of averages
10. 25 pts: Experimental report
    (a) 2 pts: cover page
    (b) 5 pts: project summary
    (c) 3 pts: hypothesis
    (d) 10 pts: analysis of timing results (includes points for code in appendix)
    (e) 5 pts: conclusions

6 Extra Credit

5 pts: Develop a different communication pattern that you believe might work differently than these three patterns, perform the same timings as above, and report and justify your findings.