CSE 6220 Introduction to High Performance Computing
Spring 2015
Programming Assignment 3
Due April 17th, 2015

Parallel Quicksort

In this assignment, you'll implement a program to sort integers in parallel by doing a straightforward parallelization of the sequential quicksort algorithm. The algorithm is described here at a high-level and you should fill in the remaining implementation details.

The input to the algorithm is a file containing the number of integers to be sorted followed by the integers themselves. The output of the program is the sorted list of the integers. Input and output of the data is already taken care of for your convenience in the programming framework.

Let n denote the number of integers and p denote the number of processors. The integers are equally distributed to the processors: Each processor has an array A holding either $\lceil \frac{n}{p} \rceil$ or $\lfloor \frac{n}{p} \rfloor$ integers. The algorithm is recursive: At some stage during the algorithm, suppose there is a communicator of size q on which m integers should be sorted and that they are distributed in the usual manner. If $q = 1$, then any serial sorting algorithm can be used to sort the input. You may either write your code, copy from a book or web site if available, or even use the Unix sort program.

If $q > 1$, all processors in the communicator use the same random number generator to generate a random number between 0 and $m - 1$, say k. The processor that has the k^{th} integer (called pivot from here on) broadcasts it to all processors in the communicator. Each processor goes through its integer array and partitions it into two subarrays − one containing integers less than or equal to the pivot and another containing integers greater than the pivot. Using an *Allgather* operation, the number of integers in each of the two subarrays on all the processors are gathered on every processor. Compute the total number of integers less than or equal to the pivot (say m') and the total number of integers greater than the pivot (say m''). Partition the q processors to the two subproblems of sorting m' integers and sorting m'' integers respectively, by allocating processors in proportion to the problem sizes (make sure you do not allocate zero processors to a problem). Using the gathered information on the sizes of the subarrays on all processors, each processor can compute where to send its data and from where to receive its data. Use an *Alltoall* communication to perform the data transfer. Create two new communicators corresponding to the two partitions. Recursively sort within each partition.

Code Framework

We provide a prepared code framework for this assignment. The framework is hosted on Georgia Tech's Enterprise GitHub page at: `https://github.gatech.edu/pflick3/cse6220-prog3`. The README file gives an overview of the code and of the files you'll have to modify. In case that we have to make updates to the code framework, we will publish the updates in this repository.

Report

Write a short report in which you evaluate the performance of your implementation for different inputs and parameters. For instance, fix a problem size, vary the number of processors, and plot the run-time as a function of the number of processors. Fix the number of processors and see what happens as the problem size is varied. Try to come up with some important observations on your program. E.g., what is the minimum problem size per processor to get good parallel efficiency? We are not asking that specific questions be answered but you are expected to think on your own, experiment and show the conclusions along with the evidence that led you to reach the conclusions.

Grading

The program will be graded on correctness and usage of the required parallel programming features. You should also use good programming style and comment your program so that it is readable and understandable.

Submission

The programming assignment is to be done by groups of two students each. No matter how you decide to share the work amongst yourselves, each student is expected to have full knowledge of the submitted program. To submit the programming assignment, turn in a zipped or tarred file containing the following files:

1. A text file containing the name of people on the team and their contributions.
2. Source code of your implementation
3. Your report in PDF format.