CS 635 - Spring 2012
Project 1
This homework assignment is due by **10 pm on March 9, 2012.**

Implementing Shearsort in MPI

Background

Shearsort [1] is a well known parallel sorting algorithm for sorting n data values on a $\sqrt{n} \times \sqrt{n}$ array of processing nodes that communicate as a two-dimensional mesh. The algorithm sorts the data (one value stored per processing node) by alternately sorting rows and columns of the mesh. On completion, the numbers appear in *snakelike order* after the $2 \log \sqrt{n} + 1 = \log n + 1$ phases of Shearsort.

Phases $1, 3, \ldots, \log \sqrt{n} + 1$ sort all the rows in parallel. The odd rows are sorted into increasing order, while the even rows are sorted into decreasing order in these odd phases. In phases $2, 4, \ldots, \log \sqrt{n}$, the columns are sorted in parallel, each into increasing order.

To sort each row (or column) of processing nodes, the parallel Odd-Even Transposition Sort (in which pairs of compare-exchanges are done in parallel) is used. See our textbook or class lecture notes for a review of this sort.

Block Decomposition of Data

Most often, the number of items to be sorted, n, and the number of processors available for executing the Shearsort algorithm, p, are not the same. If $n = p$, then Shearsort can be applied directly with one integer stored per processor. However, if $n \gg p$, then the list of integers must be partitioned onto the p processors.

In this situation, the n integers are typically distributed to the p processing nodes using either a *strip* or a *block* decomposition. For a standard block decomposition, each block would contain $\frac{\sqrt{n}}{\sqrt{p}} \times \frac{\sqrt{n}}{\sqrt{p}}$ integers so each node in the array of processors would manage $\frac{n}{p}$ numbers. (See Chapter 3 for additional discussion.)

Within each block, Shearsort's *original* odd-even computations would be executed serially by the respective processing node. However, communication between the processing nodes is still required (why?) if block decomposition is used.

Computation and Communication Times

A simple theoretical model in terms of n and p can be developed to determine the computation time needed by each processor to perform its local odd-even sorting when block decomposition is used. Furthermore, the time to send data between processing nodes can be theoretically estimated by using a common measure for node-to-node communication in parallel computation:

$$t = \alpha + \beta w$$

where α denotes the time needed to initiate message transfer, β is the per word transfer time, and w is the number of words (or integers in this case) in the message. We refer to α as the *communications latency* while β^{-1} is the *communications bandwidth*.

Your Tasks

1. Estimate the α and β constants for sending a set of integers between nodes on the Medusa cluster- to do this, you can modify the `pingpong.c` program from Homework 2. Repeat these experiments until you feel you have measured these constants correctly.

2. Determine a theoretical estimate for the total execution time used by the block decomposition method for sorting n integers using the Shearsort algorithm on p processors. Then substitute the α and β values you determined above into your equations.

3. Using the SPMD style of MPI programming,

   - Implement the Shearsort algorithm for sorting $\sqrt{n} \times \sqrt{n}$ integers on an array of $\sqrt{p} \times \sqrt{p}$ processors, applying block decomposition when $n \gg p$.
   - Your program should use one process to read in the data from a file, after which it distributes the data to the nodes in blocks.
   - The first line of the input file should indicate the number of remaining integers in the file; i.e. the first integer is the number n of items to be sorted and the remainder of the file contains the actual integers to be sorted.

4. Perform quantitative experiments to measure the total time used by your program for the following values of n and p. (You can also use larger data sets.)

n	p
16	4 & 16
64	4 & 16
256	4 & 9 & 16
1024	4 & 9 & 16
4096	4 & 9 & 16
16384	4 & 9 & 16

   Make sure you do not include the I/O times within your measurements; only the time to execute your MPI Shearsort algorithm!!

5. Display your quantitative results in table form: for example,

   - For each fixed n, list the measured and estimated total execution time as a function of p.
   - For each fixed p, list the measured and estimated total execution time as a function of n.

6. Interpret your results and compare them to your theoretical analyses:

   - Are your experimental results consistent with your theoretical estimates?
   - What appears to be the best choices for n and p?

Turn In:

- A Project Report that includes:
  - a brief overview of your implementation that includes a description of how you implemented the Shearsort algorithm with block decomposition (3 page maximum)
  - your results for the above tasks (5 page maximum)
  - complete source listings of your programs (see below)

  - the output results from two executions of your program: $n = 16$ and $n = 64$, both with $p = 16$ to verify that your MPI Shearsort code correctly sorts the data.

    The initial input array should contain an input list sorted in reverse order; e.g. for $n = 16$ and $p = 16$, the data distribution before Shearsort is applied would look like:

$$\begin{bmatrix} 16 & 15 & 14 & 13 \\ 12 & 11 & 10 & 9 \\ 8 & 7 & 6 & 5 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

    Your program output should also include the measured times for communication and computation.

- MPI source code:
  - A copy of your MPI code for determining α and β.
  - A copy of your MPI code for Shearsort.
  - Both listings should have short comment sections delineating the major computation and communication steps of your implementations. Include any special instructions needed for compiling and/or executing your code on Medusa.

  Make sure your code runs on the Medusa cluster if you developed it somewhere else. It will be tested.

- Electronic submission: Your Project Report and your source code should be uploaded to the Moodle website *before* the due date/time.

Words of Wisdom

Please do *not* leave this project until the last week before it is due. When many users are on the nodes, measurements will not be accurate and program executions could take much longer than normal.

We also know from experience that hardware and network failures frequently occur when something is due so don't wait until the last moment to do this project.

Incomplete projects will likely be assigned low grades. Please consult the Project Score Sheet (available on Moodle) to see how points will be allocated.

References

[1] I.D. Scherson and S. Sen, "Parallel Sorting in Two-Dimensional VLSI Models of Computation," *IEEE Transactions on Computers*, Vol. 38, No. 2 (1989), pp. 238-249.

A PDF copy of the paper can be obtained from the IEEE Digital Library through the online Research Databases at GMU Library: go to *http://library.gmu.edu* and select the engineering database.