DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 428/6281: Parallel Programming
Fall 2014
ASSIGNMENT 2
Due date and time: Friday, October 24th before midnight.

Programming Questions (60 marks):

Q.1. *Parallel Quicksort on a hypercube topology*: There are different formulations for parallel quicksort on distributed memory machines. One such algorithm for a d-dimensional hypercube is discussed in problem 9.17 of the textbook (page 421). Algorithm 9.9 and Figure 9.21 (pages 422 and 423) in the book further elaborate it. In this problem, you will implement this specific quicksort algorithm on the cluster. For doing so, you will assume a virtual hypercube topology on the cluster nodes.

Compare its performance with the best sequential sorting algorithm (e.g., quicksort) to sort a large input sequence (you can generate the numbers randomly). As in assignment 1, you should plot speed-up versus number of processes graphs for different input data sizes and explain your findings.

Written Questions (40 marks):

Q.2. Now consider a naïve implementation of quick sort on a distributed memory machine. The sketch of the algorithm is as follows:

At depth i of the divide-and-conquer tree, sequence S is input to process P_j, which first checks if the base condition is met based on a predefined threshold on input size. If base condition is met, P_i sequentially sorts S using sequential quick sort, and if i > 1 then sends the result back to its parent at depth (i-1). If base condition is not met, process P_j partitions its input sequence S into two parts based on a pivot x: sequence S_1 is less than or equal to the pivot and sequence S_2 is greater than the pivot. Then P_j spawns a process P_{j+1}, assigns S_1 to P_{j+1}, and keeps S_2 to be sorted by itself. Finally, P_j receives the sorted sequence S_1 from P_{j+1}, merges it with its own sorted sequence S_2 to create the sorted sequence S, and if i > 1 then sends S back to its parent at depth (i-1).

How does this algorithm compare with the algorithm in Q1 in terms of (i) load balancing, and (ii) efficiency? What is(are) their major difference(s)? Explain your answers either intuitively or analytically.

Q.3. Consider the parallel formulation of quick sort for a d-dimensional hypercube that you implemented in Q.1. How does this algorithm compare with Bitonic sort using a d-dimensional hypercube in terms of (i) load balancing, and (ii) efficiency? Does pivot selection play any role? Does input data play any role? Explain your answer either intuitively or analytically.

Submit all your answers, including well documented source code for the programs, in pdf and/or text formats only. All files should be archived into a single file (e.g., a single .zip file), and submitted through EAS.