DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 428: Parallel Programming
Winter 2012

ASSIGNMENT 1

Due date: Wednesday, February 1st before 23:59.

Programming Questions (50 marks):

In this assignment you will be solving an embarrassingly parallel problem, with minimal communication overhead and minimal I/O requirement. The main objectives are to make yourself familiar with MPI programming on the cluster, and also make you familiar with performance measurement.

Q.1. *PI calculation using the Master-Worker paradigm*: The value of π (PI) can be calculated in different ways. An approximate algorithm for calculating PI and its parallel version are provided towards the end of the tutorial that we discussed in the first two classes. Here is a link to the tutorial:

https://computing.llnl.gov/tutorials/parallel_comp/

The pseudo-code and C code that uses MPI are also provided in the tutorial. Here is what you are required to do:

a) Write a sequential version of the algorithm and execute it on a single node of the cluster. Measure the execution time.
b) Execute the parallel version of the given program with varying number of workers (e.g. 2, 4, 6, etc) and measure the parallel execution time in each case. Ideally workers should be mapped to distinct nodes of the cluster.
c) In the given program, the master and the worker processes are spawned simultaneously. Another way to implement the program is to first create the Master process, which subsequently spawns the workers through explicitly calling MPI_Comm_Spawn. Modify the given program accordingly to use this dynamic spawning mechanism available in the current versions of MPI (MPI 2.0 onwards). Repeat the same experiments as in (b) above.
d) Plot a *speedup* versus *number of workers* curve based on your experiments in (a) and (b) or (c) above. Explain any unusual behavior, e.g., slow down, sub-linear speedup, etc.

Written Questions (50 marks):

Q.2. Referring to the programming question before, both solutions (b) and (c) are based on static decompositions of the problem. There is also a way to dynamically solve the problem where the workload of a worker is only fixed dynamically (i.e. at run time). Dynamic solutions are sometimes better because they facilitate balancing load more easily. Provide the pseudo-code for such a dynamic solution to the previous problem of PI calculation (Hint: one way is to think about a task-pool/task-farm based solution strategy).

Q.3. Answer the following questions:

a) Is it possible to attain 100% efficiency in a direct/naive parallelization strategy of a divide-and-conquer algorithm using multiple processors? Explain your answer. If your answer is “No”, then explain what additionally could be done to increase the efficiency of a parallel divide-and-conquer based solution strategy.

b) “Increasing granularity could improve performance only if there is communication overhead in the fine-grained solution”. True or False? Explain your answer.

Q. 4. Let d be the maximum degree of concurrency in a task dependency graph with t tasks and critical path length l. Prove that $\lceil t/l \rceil \leq d \leq t - l + 1$.

Submit all your answers, including well documented source code for the parallel program, 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.