Assignment 5 – MPI Matrix Multiplication on Ring and CUDA Vector Sum

CS4823/CS6643 Parallel Computing

1. Readings

Read Chapter 6 (Message Passing Paradigm) and CUDA programming material.

Problems

1. MPI Programming

a. (20 points) Write a MPI program on Fox server to multiply two n-by-n matrices using p processors organized in a ring network with $1 \le p \le 12$. Use the algorithm laid out here precisely: P_0 will generate and send ith row-band of A and ith row-band of BT - the transpose of matrix B- to P_i, $1 \le i \le p-1$ (thus, space allocated should only be to accommodate row-bands of A, BT and C). To compute their row-bands of product matrix C, the processors will use p steps of computation (multiplying row-bands of A and BT producing a block of row-band of C) and (p-1) steps of communication (sending the row-band of BT to the right neighbor and receiving from the left neighbor). You must only use send/receive primitives. P_0 finally receives the ith row-bands of product matrix C from P_i, $1 \le i \le p-1$, and prints it out.

b. (5 point) Prepare a speedup plot (Ts/Tp) or a table with varying n and vary number of processes in the available range. Use pure sequential time with three nested loops for Ts.

Hint: A 2D array (matrix) is essentially a 1D array with row-major order. You may implement the sequential code in the same program and time it, followed by parallel code and time that, and calculate the speedup. Experiment and choose sufficiently large n for a reasonable speedup. Larger n may result in better speedup. Suppress the printing when timing.

Submission: Submit your source code and plot/table.

2. CUDA programming

(10 points) Experiment with the demo CUDA program for vector sum of length n on DGX server with (i) one block and (ii) multiple blocks. Submit (i) the speedup plot/table, and (ii) the source code file, adequately documented.

Vary n as 2^{10}, 2^{15}, and 2^{20} (or more if desired, and possible). Use p, the number of threads, at maximum number you can employ (up to n/2). Experiment to get best performances. Obtain the speedup relative to the sequential timings on CPU without any printing overheads. Produce a speedup plot with n on x-axis and S_p on y-axis, one curve for each case.