CPTS 411 Programming Project #2: *Parallel Reduction*

Due October 3, 2019 (11:59pm PDT)

Assignment type: Team of size up to 2 encouraged
Where to submit? Blackboard dropbox for Programming Project #2

The goal of this project is to implement, test and evaluate the Parallel Reduction operation.

Notation:

Let $\otimes$ be any binary associative operator (e.g., addition, multiplication, min, max).
Let n denote the input array size, and p denote the number of processes.

For this assignment, you can assume that:
i) n is a multiple of p, and n is much larger than p; and
ii) p is always a power of two.

Problem Statement:

The input is an array of n numbers (you can assume positive integers). We will denote this array as $A[0..n-1]$, and the element $A[i]$ as a_i.

The output is a single value x, where $x = a_0 \otimes a_1 \otimes a_2 \otimes ....\otimes a_{n-1}$.

Precondition: Initially, every process has a distinct block of n/p elements from array A.
Postcondition: At the end, the final global reduced value x is available on *all* processes (i.e., All-Reduce).

Project description:
Given the above problem statement:

(Coding)

1. Write a function called *MyReduce(…)* that implements the parallel reduction algorithm we discussed in class (using hypercubic permutation).
2. Write another function called *NaiveReduce(…)* that implements the simple way to perform reduction using the Array/Bus permutation.
3. Write a third function called *MPILibraryReduce(…)* that internally simply calls the *MPI_Allreduce(…)* function to implement the main reduction operation.

(Testing)

4. Write a function called *GenerateArray(…)* to generate a random array ***A*** of integers of size *n*, which is specified by the user as a program argument. You can use any of your

favorite random number generating functions. If you are not familiar with random functions in C, you can refer to help on functions like drand48 or rand.

5. Correctness testing: Pass the newly generated array ***A*** as input to all three functions to perform reduction. For testing purposes, you can use *addition* and *maximum* as two different binary associative operators to check your correctness. Note that for correctness, regardless of the number of processes (*p*) used, the answer should be the same for all three functions.
6. Performance comparison: Next, compare the runtime performance of all three reduce functions for varying number of processes, from *p = 1, 2, 4, 8,…,64*, and for varying input sizes, from *n=1K, 2K, 4K, …,* <you can go as high as 1M or more if you wish, as long as the runtimes are under 5-10 minutes>. For performance testing I suggest you use either addition or maximum operator. No need to do for both. Also, I suggest you test for all combinations of <n,p> (think of a table, where *n* represents rows and *p* represents columns).

(Report)

7. Compile a report to present your observations and your justification/explanation of those observations. I leave the format up to you, but the things that matter for the report are:
   a) Concise presentation of the results (in a way the performance results are easier to understand). Results should include three components: Parallel Runtime, Speedup, and Efficiency. It is best to tabulate these calculations in an excel or similar spreadsheet, and then plot them.

   b) Clear statement of what you observed vs. what you were expecting in terms of performance scaling (as a function of *p* and as a function *n*); and

   c) Statement of any experimental setup issues or assumptions (if any) – for instance, for some input size if you decided to not run because it was taking too long (say >15 mins) then you should state that.

Deliverables (zipped into one zip file - with your names on it):

Note, for those of you who worked in teams of size 2, both of you should submit, but only one of you should submit the full assignment along with the report and cover page stating who your other partner was, and the other person simply submits the cover page.

If you worked alone, cover page is optional.

i) Source code (along with any scripts you wrote to automate testing);
ii) Report in PDF