CSE Ph.D. Qualifying Exam, Spring 2022
High Performance Computing

Instructions:

Please answer three of the following four questions. All questions are graded on a scale of 10. If you answer all four, all answers will be graded and the three lowest scores will be used in computing your total.

Questions:

1. Suppose an array E contains the elevations at mileposts along the Appalachian trail, in order. The *isolation* of a milepost is the minimum distance you must travel, forward or backward, to a find a milepost of greater elevation. (The isolation of the highest point(s) on the trail is infinite.)

   (a) Design a parallel algorithm to compute the array I of isolations of each milepost.

   (b) Analyze the runtime of your algorithm as a function of the number of mileposts m and the number of processors p.

   (c) Determine how many processors can be used to run your algorithm with $\Theta(1)$ efficiency.

2. (a) Let A and B be two arrays of size n distributed across p processors such that each processor has $\frac{n}{p}$ consecutive entries. Suppose array B contains a permutation of $0, 1, 2, \ldots, n-1$. We need to compute array C of size n such that $C[i] = A[B[i]]$. Which MPI communication primitives should be used here and what is the estimated run-time?

   (b) An undirected graph G consists of n vertices and m edges. The graph is described by an unordered list of m edge tuples – each of the form (u, v) to describe the edge connecting u and v. Assume the edge tuples are distributed evenly among p processors. Design a parallel algorithm to compute the degree of each vertex in the graph.

3. Let L denote a $n \times n$ triangular matrix, i.e., the (i, j) entry of L is $L_{ij} = 0$ if $j > i$. Let x and b denote two $n \times 1$ vectors. You wish to solve the system of equations $Lx = b$ on a very large distributed memory parallel computer with P compute nodes.

   Design an efficient and *load balanced* algorithm for solving $Lx = b$ by forward substitution.

   (a) How is the matrix L and the vector b distributed across the P nodes?

   (b) How is the solution x distributed across the P nodes?

   (c) Describe your algorithm in pseudocode and in words.

(d) What is the load balance ratio, i.e., the amount of work performed by the node with the most work divided by the amount of work performed by the node with the least work?

(e) Write an expression that estimates the total time required by the computation. Be sure to define any symbols you use.

You may assume that n is large and, for simplicity of the presentation of your algorithm, that P satisfies some simple constraints (e.g., P is even) that possibly depends on n.

For simplicity, you may assume there are no parallel resources within a compute node. You may also assume that you will not encounter any "numerical" problems, i.e., overflow, divide by zero, etc.

4. **Disaggregated memory.** An emerging concept in the design of distributed-memory systems, which is driven by datacenter workloads, is *disaggregated memory.* This question asks you to think about what implications such systems have for parallel algorithms.

   In the traditional model of a distributed-memory parallel computer, there are P compute nodes, each with M words of main memory and Z words of cache. (Assume one level of cache for simplicity.) These nodes are connected by a network of some topology, like a 3-D torus or Dragonfly, characterized by attributes like bisection width, diameter, and number of links. Let the local cache-to-main-memory bandwidth be β_M and suppose the time to send a message between any two nodes follows the usual latency-link bandwidth model, $\alpha + \beta w$ where w is the message size in words.

   In the simplest model of a disaggregated-memory system, each compute node no longer has any main memory. That is, each of the P nodes only has a local cache of size Z. Instead, the memory is distributed among Q memory-only nodes, each with $\hat{M}$ words of memory such that $PM = Q\hat{M}$. All nodes, whether compute-only or memory-only, are again connected by a network. Messages may be sent between compute-only nodes, between memory-only nodes, or between compute- and memory-only pairs. Suppose the message cost in this system follows the same $\alpha + \beta w$ model.

   What are the characteristics of a parallel algorithm that will be faster or more scalable on a disaggregated system than a traditional system (assuming the number of compute nodes, P, is the same)? To get full credit, in addition to making qualitative comments about the design differences between these systems, try to give an analytical response, for example, by considering the system parameters as given above and show an analytical tradeoff.