CSE Qualifying Exam: High-Performance Computing

Fall 2020

Instructions

- Please answer three of the following four questions. All questions are graded on a scale of 0-10 points. If you answer all four, all answers will be graded and the three lowest scores will be used in computing your total.
- Please write clearly and concisely, explain your reasoning, and show all work. Points will be awarded for clarity as well as correctness.
- Unless otherwise specified, assume a distributed memory model of computation where a message of m words can be sent in $\mathcal{O}(\tau + \mu m)$ time, where τ denotes the latency and μ denotes the per word transfer time. You may assume each processor can send and receive one message within the same parallel communication step.

Problem 1

Consider p distributed processes. Each process knows which processes it needs to send data to. However, processes do not know which other processes will send it data. Each process only needs to send data to a small number of other processes.

Design a "scalable" algorithm that processes can use to determine which processes will send it data. By "scalable," we mean that processes cannot use all-to-all communication, or use any communication involving $O(p)$ steps.

Write pseudocode to help explain your algorithm. As a guide, your algorithm should be implementable in MPI. Write clearly. Points will be deducted for unnecessary comments.

Problem 2

Consider a perfect binary tree[1] with $N = 2^k - 1$ nodes. Suppose the nodes are indexed in the usual "packed" way:

1. The nodes are numbered (indexed) from 0 to $N - 1$.
2. The root's index is the integer 0.
3. Let i be the integer index of any non-leaf node. Its left and right children have indexes $2i + 1$ and $2i + 2$, respectively.
4. Let i be the index of any non-root node. Its parent has the index $j = \lceil \frac{i-1}{2} \rceil$.

[1] A perfect binary tree is one where every non-leaf node has exactly two children, and all leaves are at the same level.

Furthermore, each node i is associated with a real-valued weight, $A[i]$. (Weights could be negative.) Given any two nodes in the tree, there is a unique acyclic path that connects them. The *length* of this path is the sum of the weights of all nodes along the path, including the endpoints.

Give an efficient shared-memory parallel algorithm to compute the tree's *maximum path length*. That is, for all pairs of nodes (i, j), return the weight of the pair whose path-length is maximum. (You only need to return the weight, not the path itself.)

Problem 3

You have p processes and each process r has a sparse vector $v^r \in \mathbb{R}^n$, represented as an array of pairs $\{(i, v_i^r) : v_i^r \neq 0\}$. Each component v_i^r is nonzero with independent probability $\phi \in (0, 1)$.

You want every process to have a complete, dense copy of v^{total}, where $v_i^{\text{total}} = \sum_r v_i^r$. You could probably write your own parallel algorithm for this, but you're short on time, so two friends give you suggestions for using existing MPI routines.

(A) Use MPI_Allgatherv to gather all nonzeros onto every process, then have each process add them into v^{total} for itself.

(B) Turn each sparse vector into a dense vector and use MPI_Allreduce to combine them.

1. (4 points) For both (A) and (B), model the computation time T_{comp} and communication time T_{comm}.
2. (6 points) You want to your program to always use the best algorithm with the information available. Develop a criterion based on p, n, and ϕ for when you will use (A) and when you will use (B).

Problem 4

The edge list of an *undirected* graph $G = (V, E)$ is randomly shuffled and partitioned into p processors such that each process has $\frac{2|E|}{p}$ *directed* edges. That is, in the edge list, each undirected edge $\{u, v\}$ is represented with two directed edges: (u, v) and (v, u). Design two efficient algorithms to calculate the following, and analyze their runtime.

1. (4 points) The degrees for each of k given vertices $< v_1, v_2, \ldots, v_k >$, where k is a small constant.
2. (6 points) The degrees of all vertices. Discuss design decisions and limits of your algorithm, if any.