CSE Qualifying Exam: High-Performance Computing

Fall 2019

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.
- 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

To multiply a dense, $n \times n$ symmetric matrix by a dense vector, it is easy to think of an algorithm that only reads half of the entries in the matrix. However, it is also possible to design an algorithm that only requires half of the normal number of multiplications.

(a) Design an algorithm for multiplying a real, dense, $n \times n$ symmetric matrix A by a real, dense vector b, that asymptotically requires $n^2/2$ multiplications.

Hint: You might guess that you have no choice but to spend the $n^2/2$ multiplications on forming the intermediate quantities $z_{ij} = A_{ij}(b_i + b_j)$. You can then do any additions you like, as well as up to $\mathcal{O}(n)$ multiplications, to form the components of the result, c_i.

(b) Describe how to implement this algorithm so that it runs efficiently on a distributed memory computer, e.g., how to partition the matrix and vector, etc.

Problem 2

Load balancing.

1. *Identical Tasks:* Consider a p-processor parallel computer, in which each processor contains a certain number of independent tasks. Let n_i denote the number of tasks in P_i, and $n_{\max}$ denote the maximum number of tasks a processor has ($n_{\max} = \max_{i=0}^{p-1} n_i$). Assume each task is represented using $\mathcal{O}(1)$ space. We want to redistribute the tasks such that each processor has approximately the same number of tasks. Note that initially P_i only knows its assigned tasks and n_i. Give an algorithm to solve this problem and analyze its running time. Your algorithm should run in $\mathcal{O}(n_{\max} \log p)$ computation time and $\mathcal{O}((\tau + \mu n_{\max}) \log p)$ communication time.

2. *Heterogeneous Tasks:* Consider the modified problem where each task $t_{i,j}$ in P_i has a weight $w_{i,j}$ representing both its space and computational complexity, i.e., each task $t_{i,j}$'s space and computation complexity is $\mathcal{O}(w_{i,j})$. Describe how you would simply extend the algorithm you proposed to handle this case. Analyze its runtime, and discuss its optimality. If your algorithm, propose ways to improve its solution quality.

Problem 3

Pseudocode for the loop body of the standard conjugate gradient algorithm for solving the system $Ax = b$ is given below. x_i's, r_i's, p_i's, and s_i's are vectors; α_i's and β_i's are scalars.

CG1:

```
1. $x_i \leftarrow x_{i-1} + \alpha_{i-1} p_{i-1}$
2. $r_i \leftarrow r_{i-1} - \alpha_{i-1} s_{i-1}$
3. $\beta_i \leftarrow r_i^T r_i / r_{i-1}^T r_{i-1}$
4. $p_i \leftarrow r_i + \beta_i p_{i-1}$
5. $s_i \leftarrow A p_i$
6. $\alpha_i \leftarrow r_i^T r_i / p_i^T s_i$
```

Now we give pseudocode for the loop body of the Chronopolous-Gear variant of the conjugate gradient algorithm. Variables are the same as above, except w_i's are also vectors. If we ignore numerical considerations (a big if!), it is equivalent.

CG2:

```
1. $x_i \leftarrow x_{i-1} + \alpha_{i-1} p_{i-1}$
2. $r_i \leftarrow r_{i-1} - \alpha_{i-1} s_{i-1}$
3. $\beta_i \leftarrow r_i^T r_i / r_{i-1}^T r_{i-1}$
4. $w_i \leftarrow A r_i$
5. $\alpha_i \leftarrow r_i^T r_i / (w_i^T r_i - (\beta_i / \alpha_i) r_i^T r_i)$
6. $p_i \leftarrow r_i + \beta_i p_{i-1}$
7. $s_i \leftarrow w_i + \beta_i s_{i-1}$
```

Assume a simple distributed parallelism model with p processes equally distributing vectors of length n, where:

- The time to compute a fused mutiply-add ($\gamma \leftarrow \gamma + \delta\epsilon$) is 1.
- The time to send a message of m numbers is $\tau + \mu m$.
- The matrix-vector product Ab is known to take time $T_A(n, p, \mu, \tau)$.

a. What communication pattern is needed for the dot products $x^T y$ in these algorithms? What is a reasonable model for the runtime of a single dot product?

b. When (for what values of p, n, μ and τ) can **CG2** be faster than **CG1**? You are allowed to reorder the operations, introduce new variables, and/or introduce new communication patterns, as long as you do not change the outcome of each loop body or violate any data dependencies.

Problem 4

Let A and B be two matrices. Let $C \leftarrow A \times B$ denote the usual matrix-matrix product operation,

$$c_{i,j} = \sum_k a_{i,k} \cdot b_{k,j}, \tag{1}$$

that is, where each entry of the output $c_{i,j}$ is the dot-product between row i of A and column j of B. Next, let $C \leftarrow A \odot B$ denote the elementwise product,

$$c_{i,j} = a_{i,j} \cdot b_{i,j}. \tag{2}$$

Suppose A is a *sparse* matrix whose dimensions are $n \times n$. Now consider the operation

$$C = A \odot (A \times A). \tag{3}$$

Give a work-efficient parallel algorithm to compute C given A on a distributed memory machine. Analyze the running time of your algorithm assuming a communication model in which a single scalar arithmetic operation costs $O(1)$ time, and the time to send or receive a message using the cost model described in the **Instructions** for this exam. Explain why your proposed scheme should be regarded as efficient. If you need assumptions on the structure of the sparse matrix A to make your analysis tractable, state those; however, you may be penalized for making overly simplistic assumptions about that structure, so "assume wisely."