## CS 140 Midterm 1 -- 8 February 2010

## Name <br> Perm\#

Problem 1 [20 points total] In Lake Wobegon, all the women are strong, all the men are goodlooking, and all the children are above average. Well, everyone can't be above average--but here we'll count how many are.

We have $\mathbf{n}$ kids and $\mathbf{p}$ processors. Each processor starts out with $\mathbf{n} / \mathbf{p}$ elements of a vector IQ of the $\mathbf{n}$ kids' IQ values. Your goal is to compute the average of all $\mathbf{n}$ IQs (that is, their sum divided by $\mathbf{n}$ ), and also to figure out how many of the $\mathbf{n}$ IQs are larger than average. The results, called averageIQ and numHighIQ, should end up on processor 0 . For example, if the entries in IQ[] are $110,90,120$, and 100 , then the averageIQ is $420 / 4=105$, and the numHighIQ is 2 (since two values, 110 and 120, are larger than average). A sequential algorithm to do this on one processor would be as follows. Note that IQ[] and averageIQ are doubles, not integers.

```
double sum = 0;
for (int i = 0; i < n; i++)
        sum += IQ[i];
double averageIQ = sum / n;
int numHighIQ = 0;
for (int i = 0; i < n; i++)
    if (IQ[i] > averageIQ) numHighIQ ++;
```

For this problem only, you don't have to worry about the efficiency of your code.
(1a) [10 points] Using pseudo-code, show how to do this in MPI using send and recv.
(1b) [10 points] Using pseudo-code, show how to do this in MPI using broadcast and reduce.

Problem 2 [20 points total] This problem compares two different data layouts for matrix-vector multiplication on a message-passing machine. All $\mathbf{n}$ elements of a vector $\mathbf{x}$ are on processor 0 . The elements of an $\mathbf{n}$-by-n array $\mathbf{A}$ are divided evenly among $\mathbf{p}$ processors, with $\mathbf{n}^{2} / \mathbf{p}$ elements per processor. The goal is to have all $\mathbf{n}$ elements of the product $\mathbf{A}^{*} \mathbf{x}$ end up on processor 0 . For Algorithm 1, each processor has $\mathbf{n} / \mathbf{p}$ rows of A. For Algorithm 2, each processor has a square block of $\mathbf{A}$ with $\mathbf{n} / \mathbf{s q r t}(\mathbf{p})$ rows and $\mathbf{n} / \mathbf{s q r t}(\mathbf{p})$ columns. Assume that $\mathbf{n}$ is divisible by $\mathbf{p}$, and that $\mathbf{p}$ is a perfect square. You don't have to show the code for the two algorithms; just answer these questions.
(2a) [2 $1 / 2$ points] Draw a clearly labeled diagram of the data layout for Algorithm 1.
(2b) [2½ points] Draw a clearly labeled diagram of the data layout for Algorithm 2.
(2c) [15 points] Complete the following table with the parallel time $\mathbf{t}_{\mathrm{p}}$, the span $\mathbf{t}_{\infty}$, and the total communication volume $\mathbf{v}$ for each algorithm. For $\mathbf{t}$, we count only multiplication operations (which is why the work $\mathbf{t}_{\mathbf{1}}$ is $\mathbf{n}^{2}$ ). You can omit lower-order terms in your answer, for example by writing $\mathbf{n}^{2}$ instead of something like $\mathbf{n}^{2}-\mathbf{n}+\mathbf{1}$.

|  | Work <br> $\mathbf{t}_{\mathbf{1}}$ | Parallel time <br> $\mathbf{t}_{\mathbf{p}}$ | Span <br> $\mathbf{t}_{\infty}$ | Comm volume <br> $\mathbf{v}$ |
| :--- | :---: | :---: | :---: | :---: |
| Algorithm 1 | $\mathbf{n}^{2}$ | $\mathbf{n}^{2} / \mathbf{p}$ |  |  |
| Algorithm 2 | $\mathbf{n}^{2}$ |  |  |  |

Problem 3 [20 points] Suppose you have $\mathbf{p}$ processors, $\mathbf{P}(\mathbf{0})$ through $\mathbf{P}(\mathbf{p} \mathbf{- 1})$, each with local (double) variables $\mathbf{x}, \mathbf{y}$, and $\mathbf{d}$ (plus any other local variables you need). Each ( $\mathbf{x}, \mathbf{y}$ ) represents a point in the plane, so each processor has one point. The goal is for each processor to set its own $\mathbf{d}$ to the shortest distance between its point and any other processor's point. For example, if there are three processors with points

$$
P(0): x=1, y=1 \quad P(1): x=-1, y=0 \quad P(2): x=0, y=0
$$

then $(\mathbf{1}, \mathbf{1})$ 's closest point is $(\mathbf{0}, \mathbf{0})$, and $(\mathbf{- 1 , 0})$ 's closest point is $(\mathbf{0}, \mathbf{0})$, and $(\mathbf{0}, \mathbf{0})$ 's closest point is $(\mathbf{- 1 , 0})$, so the result should be

$$
P(0): d=\operatorname{sqrt}(2) \quad P(1): d=1 \quad P(2): d=1
$$



Write message-passing code (pseudocode is fine) to achieve this. Rules:

- Use *blocking* send and receive calls for all communication.
- Each processor $\mathbf{P}(\mathbf{i})$ should only send to / receive from its neighbors $\mathbf{P ( i - 1 )}$ and $\mathbf{P ( i + 1 )}$, where we also include $\mathbf{P}(\mathbf{p}-\mathbf{1})$ and $\mathbf{P}(\mathbf{0})$ as neighbors of each other.
- For full credit, your algorithm should use no more than $2 \mathbf{p}$ rounds of message-passing, and should have parallel computation time $\mathbf{t}_{\mathbf{p}}=\mathbf{O}(\mathbf{p})$.
Hint: Send copies of all the processors' $(\mathbf{x}, \mathbf{y})$ values around a merry-go-round ring.
(Note: It's an interesting problem in computational geometry to do this in *less* than $\mathbf{O}(\mathbf{p})$ parallel time; but you don't have to do that for the exam problem.)

Problem 4 [20 points total] You have a function called findmax that computes the largest element in an array of size $\mathbf{n}=2^{\mathbf{k}}$. The serial version of your code looks like the following:

```
double findmax(double * array, int n) {
    double max = array[0];
    for (int i = 1; i < n; i++)
            if (array[i] > max) max = array[i];
    return max;
}
```

(4a) [10 points] Explain briefly why you can't parallelize this function in cilk++ by just changing the for loop to a cilk_for. Give a small example (say $\mathbf{n}=\mathbf{3}$ or $\mathbf{4}$ ) of what can go wrong.
(4b) [10 points] Describe a way to parallelize this function using cilk_spawn. (You don't have to write syntactically correct cilk++ code, just be sure your description is clear.)

Problem 5 [20 points total] Short answer questions.
(5a) [10 points] What is an embarrassingly parallel problem? Give an example.
(5b) [10 points] A sequential program spends $99 \%$ of its time on a computation that could be done efficiently in parallel, and the other $1 \%$ on a computation that can't be parallelized at all. What can you say about maximum speedup for a parallel version of this program?

