Mastermath midterm examination Parallel Algorithms. Solution.

Teacher: Rob H. Bisseling, Utrecht University

October 24, 2018

Each of the four questions is worth 10 points.

1. (a) An h-relation is a communication superstep where each processor sends at most h data words to other processors and receives at most h data words, and where at least one processor sends or receives h words.

   (b) The cost of an h-relation is

$$T_{\text{comm}}(h) = hg + l,$$

   where g and l are machine-dependent parameters and the time unit is the time of one flop.

2. (a) The parallel algorithm is given as Algorithm 1.

   (b) Superstep (0) costs $2g + l$ because processors $P(s)$ with $0 < s < p - 1$ send and receive two data words. This holds for the general case $p > 2$. For $p = 2$, the cost is $g + l$. Superstep (1) costs $7n/p$ flops, since we have 3 multiplications, 2 additions, 1 subtraction, and 1 comparison for each of the n/p local vector components y_i. Therefore, the total cost of the algorithm is

$$T = \frac{7n}{p} + 2g + 2l.$$

Algorithm 1 Neural network computation for processor $P(s)$.

```
$b := n/p$;                                    ▷ Superstep (0)
if $s > 0$ then
    get $x_{sb-1}$ from $P(s-1)$;
else
    $x_{-1} := 0$;
if $s < p-1$ then
    get $x_{(s+1)b}$ from $P(s+1)$;
else
    $x_n := 0$;
                                               ▷ Superstep (1)
for $i := sb$ to $(s+1)b-1$ do
    $y_i := \max(a_i x_{i-1} + b_i x_i + c_i x_{i+1} - d_i, 0)$;
```

3. (a) In stage k, with $k = 0, 1, \ldots, n-1$, the sequential matrix update performs $2(n-k-1)^2$ flops. The total number of flops in the updates is

$$\sum_{k=0}^{n-1} 2(n-k-1)^2 = 2\sum_{k=0}^{n-1} k^2 = \frac{2(n-1)n(2n-1)}{6} = \frac{2n^3}{3} - n^2 + \frac{n}{3}.$$

(b) The parallel matrix update is given as Algorithm 2. Note that we do not have to ensure that $i, j < n$, because the $b \times b$ blocks fit nicely in the matrix, where $b = n/M = n/\sqrt{p}$.

Algorithm 2 Parallel matrix update in stage k for $P(s,t)$.

```
$b := n/M$;
for $i = \max(k+1, sb)$ to $(s+1)b-1$ do
    for $j = \max(k+1, tb)$ to $(t+1)b-1$ do
        $a_{ij} := a_{ij} - a_{ik}a_{kj}$;
```

(c) In stages $0 \le k < n-b$, processor $P(M-1, M-1)$ has to update its complete $b \times b$ block, which takes $2b^2$ flops. Others perform the same amount of work, or less. The total time for this part of the algorithm is $2(n-b)b^2$. In stages $n-b \le k < n$, processor $P(M-1, M-1)$ is the only processor working and it can run the sequential algorithm. (In fact, it need not communicate or synchronise anymore.) Using the result of (a) with b instead of n, gives a time of $\frac{2b^3}{3} - b^2 + \frac{b}{3}$ for the matrix updates. The total time for the matrix updates is

$$2nb^2 - \frac{4}{3}b^3 - b^2 + \frac{b}{3} = \frac{2n^3}{p} - \frac{4n^3}{3p\sqrt{p}} - \frac{n^2}{p} + \frac{n}{3\sqrt{p}}.$$

(d) Comparing the main term $2n^3/p$ of the matrix updates of the parallel algorithm to the main term $2n^3/3$ of the sequential algorithm, we see that the speedup is about $p/3$. This is the maximum attainable, since we did not yet take the other part of the computation (the divisions in column k) and the communication and synchronisation into account, which will lower the speedup.

4. In the case of partial pivoting, we perform a FindMax(k) operation, which consists of finding a local element in column k with maximum absolute value, sending this value to all processors in the processor column that owns matrix column k, and finally redundantly determining the global winner among the M candidates. The cost of FindMax(0) (at the start) is $n/M + M + (M-1)g + 3l$.

   (a) For rook pivoting, we need FindMax operations on rows as well as columns. To find a rook pivot faster, we can set all M processor columns to work, searching columns $0, 1, \ldots, M-1$ simultaneously for their maximum. This costs the same as searching for one maximum. For each maximum element a_{rt} found, the corresponding row r must be searched for the maximum a_{rc}. If $c = t$, we have found a rook pivot. Otherwise, we can search column c. We alternate between searching columns and searching rows. During the search the absolute values encountered increase.

   A subtle point is that we want to execute exactly one FindMax operation per processor column or processor row. If there are more candidate columns or rows, one is chosen, and the others are delayed. If there are no candidates, a new column or row is chosen. It is expected that after a few rounds, a rook pivot is found.

   Processors need to inform all others in case of success, to prevent further rounds to occur. This can be achieved by using a boolean variable **done**, which is set to true at the end of the algorithm at an expected extra cost of $(p-1)g$ (assuming only one processor finds a rook pivot in the same round).

(b) Only a few iterations are expected to be performed, each with a cost of $n/M + M + (M-1)g + 3l$. Therefore, the total cost is a small constant times $n/M + M + (M-1)g + 3l$.