Mastermath midterm examination Parallel Algorithms. Solution.

Teacher: Rob H. Bisseling, Utrecht University

October 25, 2017

Each of the four questions is worth 10 points. The solutions also contain the subdivision of the points for the different parts of the questions.

1. The four parameters of the BSP model are:

   - [1 pt] p, the number of processors of the parallel computer.
   - [1 pt] r, the computing rate in floating-point operations (flops) per second of a single processor.
   - [2 pt] g, the communication cost per data word in flop time units incurred for sending a word into the communication network, or receiving a word.
   - [2 pt] l, the global synchronisation cost in flop time units.

   [4 pnt] The parameters p, g, l are relevant for the design of a BSP algorithm because their ratio influences the relative fraction of computation, communication, and synchronisation time. The parameter r is merely a normalisation constant that does not influence the algorithm design (but it influences how long you have to wait for your program to finish!).

2. [7 pt] The following algorithm for processor $P(s)$ computes the value r of the runner-up, i.e., the second-largest numerical value. We assume that as a result we want the value r replicated over all processors. We use the cyclic distribution, but a block distribution also works.

Input: $\mathbf{x}$: vector of length n, distr$(\mathbf{x}) = \phi$, with $\phi(i) = i \bmod p$, for $0 \le i < n$.
Output: r = runner-up value of $\mathbf{x}$, repl$(r) = P(*)$.

$m_s := \max\ \{x_i\ :\ 0 \le i < n \wedge \phi(i) = s\};$ ▷ Superstep (0)
$r_s := \max\ \{x_i\ :\ 0 \le i < n \wedge \phi(i) = s \wedge x_i < m_s\};$

for $t := 0$ **to** $p-1$ **do** ▷ Superstep (1)
  put m_s, r_s in $P(t)$;

$m := \max\ \{m_t\ :\ 0 \le t < p\};$ ▷ Superstep (2)
$t_m := \text{argmax}\ \{m_t\ :\ 0 \le t < p\};$
$r := \max\ \{m_t\ :\ 0 \le t < p \wedge m_t < m\} \cup \{r_{t_m}\};$

[3 pt] The cost analysis of the algorithm is as follows. In an implementation, superstep (0) is done by one loop, which keeps the local maximum and the local runner-up encountered so far. Each array value needs at most two comparisons, so the cost is $2\lceil n/p\rceil + l$. Superstep (1) is a $2(p-1)$-relation with cost $2(p-1)g + l$. Superstep (2) is carried out by all processors redundantly and it costs $2p + l$. The total cost of the algorithm is $2\lceil n/p\rceil + 2p + 2(p-1)g + 3l$.

Since we did not specify which processors need the answer, an alternative solution is possible, which is even faster: just put m_s in superstep (1), which saves $(p-1)g$ in the cost, then compute m redundantly and compute r only on processor $P(t_m)$. This processor can then print r, if needed.

3. [5 pt] The parallel algorithm starts with a sequential mergesort on the local blocks $x(0 : n/2-1)$ and $x(n/2 : n-1)$ of processors $P(0)$ and $P(1)$, respectively. Then each processor sends all its values to the other. After that, $P(0)$ performs a merge of its own values with those of $P(1)$ starting at the lowest value, until it has reached $n/2$ output values. $P(1)$ does the same, but starting at the highest value.

   [3 pt] The sequential mergesort costs $(n/2)\log_2(n/2)+l$. The communication superstep costs $(n/2)g+l$. The final superstep costs $n/2+l$. The total cost is thus $(n/2)(\log_2 n + g) + 3l$. In terms of memory, we double the local size required, since we store the values of both processors.

   We can reduce the communication cost by having $P(0)$ send its k largest values to $P(1)$, and $P(1)$ its k smallest values to $P(0)$ and then perform the merge of the final superstep. If we choose a sufficiently large k, this will provide all information needed to the two processors. Otherwise, we need to send more values and perform another superstep. A good value of k (for random input) would be a value slightly larger than $n/4$.

[2 pt] A possible extension of the algorithm to four procesors starts by running the algorithm for two processors on the processor pairs $P(0:1)$ and $P(2:3)$, in parallel. This costs $(n/4)(\log_2 n - 1 + g) + 3l$. Then a communication superstep is performed, where $P(0)$ sends all its data to $P(2)$, and vice versa, and the same for $P(1)$ and $P(3)$. This costs $(n/4)g + l$. $P(0)$ and $P(3)$ can then obtain their final $n/4$ values, with cost $n/4 + l$. In the same superstep, $P(1)$ and $P(2)$ can determine their remaining $n/4$ values (those not ending up in $P(3)$ and $P(0)$, respectively), at no extra cost. Finally, $P(1)$ and $P(2)$ perform a two-processor merge, with cost $(n/4)g + n/4 + 2l$. The total cost is thus $(n/4)(\log_2 n + 1 + 3g) + 7l$. In terms of memory, we still only double the local size required.

4. [3 pt] First, we determine a suitable distribution ϕ of the vector $\mathbf{x}$ of length $N = kn$. Since the matrix $B \otimes A$ consists of blocks of size $n \times n$ which are either A, $-A$, or 0, it is convenient to have complete blocks of $\mathbf{x}$ of length n assigned to a processor. There are k such blocks. Furthermore, blocks at block distance $k/2$ are combined by the multiplication operation, so it is convenient if these are also on the same processor. A cyclic distribution of blocks achieves this, because $k/2$ is a multiple of p. The resulting distribution is

$$\phi(i) = (i \text{ div } n) \bmod p, \text{ for } 0 \le i < N.$$

This is called the *block-cyclic distribution* with block size n. Any block size b with $n \le b \le N/(2p)$ will work.

[4 pt] The algorithm for processor $P(s)$ is:

Input: $\mathbf{x}$: vector of length n, $\mathbf{x} = \mathbf{x}_0$, $\text{distr}(\mathbf{x}) = \phi$.
Output: $\mathbf{x} = (B \otimes A)\mathbf{x}_0$.

for $r := s$ **to** $k/2 - 1$ **step** p **do** ▷ Superstep (0)
  $\mathbf{X} := Ax(rn : (r+1)n - 1)$;
  $\mathbf{Y} := Ax(rn + N/2 : (r+1)n - 1 + N/2)$;
  $x(rn : (r+1)n - 1) := \mathbf{X} + \mathbf{Y}$;
  $x(rn + N/2 : (r+1)n - 1 + N/2) := \mathbf{X} - \mathbf{Y}$;

Here, r is the block number and $\mathbf{X}, \mathbf{Y}$ are auxiliary vectors of length n; their use saves some computations.

[3 pt] The cost analysis is as follows. There is only one superstep, namely a computation superstep, and no communication, because of the choice of ϕ. The computation consists of $k/(2p)$ iterations of the main loop, where each iteration performs two matrix-vector multiplications of cost $2n^2$ flops, one vector subtraction of cost n, and one vector addition of cost n. Thus, the total cost is $(k/(2p)) \cdot (4n^2 + 2n) + l = kn(2n+1)/p + l$.