Mastermath retake midterm examination Parallel Algorithms.

Teacher: Rob H. Bisseling, Utrecht University

December 20, 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. [10 pt] A BSP algorithm is an algorithm running on one or more processors, and it consists of a sequence of phases, called supersteps. A superstep can be either a computation superstep in which the processors carry out a number of computations in parallel, or a communication superstep, in which they communicate with each other, i.e., they send and/or receive data. It is also possible to have a mixed superstep in which the processors do both. Each superstep is followed by a synchronisation, in which the processors wait for each other until they all have finished the operations of that superstep.

2. (a) [5pt] The following algorithm for processor $P(s)$ computes the required norms and gives every processor a copy of them.

Input: $\mathbf{x}^0, \ldots, \mathbf{x}^{p-1}$: vectors of length n, all distributed by the block distribution over p processors.
Output: $\alpha^j = ||\mathbf{x}^j||$ for $j = 0, \ldots, p-1$, replicated over all processors.

$b := \frac{n}{p}$; ▷ Superstep (0)
for $j := 0$ **to** $p-1$ **do**
  $\alpha_s^j := 0$;
for $i := sb$ **to** $(s+1)b-1$ **do**
  for $j := 0$ **to** $p-1$ **do**
    $\alpha_s^j := \alpha_s^j + ((\mathbf{x}^j)_i)^2$;

for $j := 0$ **to** $p-1$ **do** ▷ Superstep (1)
  put α_s^j in $P(j)$;

$\alpha^s := 0$; ▷ Superstep (2)
for $t := 0$ **to** $p-1$ **do**
  $\alpha^s := \alpha^s + \alpha_t^s$;
$\alpha^s := \sqrt{\alpha^s}$;

for $t := 0$ **to** $p-1$ **do** ▷ Superstep (3)
  put α^s in $P(t)$;

(b) [3 pt] The cost of superstep (0) is $2p \cdot n/p + l = 2n + l$; this superstep computes the local contribution α_s^j to α^j for all j. The cost of (1) is $(p-1)g + l$; this superstep sends the local contribution to a single processor responsible for the computation of α^j. Note that this communication step is well-balanced, so there is no point in sending all results to everybody. The cost of (2) is $p + 1 + l$, counting 1 flop for computing the square root. The cost of (3) is $(p-1)g + l$; this superstep takes care that every processor has a copy of the computed norms. The total cost of the algorithm is

$$2n + p + 1 + 2(p-1)g + 4l.$$

(c) If l is very high, we can save a superstep by modifying supersteps (1)–(3) into

```
for j := 0 to p − 1 do                                  ▷ Superstep (1')
    for t := 0 to p − 1 do
        put α_s^j in P(t);

for j := 0 to p − 1 do                                  ▷ Superstep (2')
    α^j := 0;
    for t := 0 to p − 1 do
        α^j := α^j + α_t^j;
    α^j := √α^j;
```

The total cost of the algorithm now is

$$2n + p(p+1) + p(p-1)g + 3l.$$

Looking at the communication and synchronisation terms only, we see that the modified algorithm is more efficient if $(p-1)(p-2)g < l$, so the break-even point is $l \approx p^2 g$.

3. There are a many different ways to solve this problem, giving you an opportunity to show your creativity.

   Solution 1 First perform a (parallel) regular samplesort for the total set of integers A, each processor $P(s)$ starting with an input set A_s, then locally remove the duplicates from the sorted output and for $s < p-1$ send the highest local value to $P(s+1)$, which checks whether its lowest value is a duplicate of the value received. Then the local count of unique values is sent to $P(0)$ which sums the counts, to obtain $|A|$.

   Cost analysis: Samplesort for the total array of size pn costs about $n \log_2 n + 2ng + 5l$. Removing duplicates for a sorted block costs $2n + l$ (assuming that the processor with the largest received block from the samplesort has $2n$ values.) The following steps cost $g + l$, $1 + l$, $(p-1)g + l$, $p + l$. The total cost of the algorithm now is about

$$n(2 + \log_2 n) + 2ng + 10l.$$

   Solution 2 First perform a local quicksort, so we have A_s sorted. Then rotate the sets in $p-1$ supersteps around the processors. In each superstep, the local set A_s is compared at cost $2n$ with a passing set

A_t. If a duplicate integer is detected in this comparison, and $s > t$, the local integer is deleted from A_s. Thus the first copy of an integer survives. At the end, the sets A_s contain only unique integers.

The cost is $n \log_2 n + l$ for the quicksort, and $(p-1)(2n + ng + l)$ for the rotation, and then $(p-1)g + l$, $p + l$, as before.

The total cost of the algorithm now is about

$$n(2(p-1) + \log_2 n) + (p-1)ng + (p+2)l.$$

Solution 1 is better than solution 2.

4. (a) [3 pt] We can use a 1D row distribution by blocks for the input and output matrix, where element x_{ij} is assigned to $P(i \text{ div } n/p)$, numbering the processors in 1D fashion. (A 1D cyclic row distribution would also work.) The FFT2D then performs 1D FFTs on the n/p local rows, in time $(n/p) \cdot 5n \log_2 n + l = (5n^2 \log_2 n)/p + l$. The matrix X is then transposed, which is a communication superstep of cost $(2n^2/p)g$, the factor 2 coming from the fact that X is a complex matrix. (Actually, if we count more precisely, this cost is slightly less by a term $(2n^2/p^2)g$.) Then another set of 1D FFTs on local rows is performed, followed by another transposition back to the original distribution. The total cost of the algorithm is

$$\frac{10n^2 \log_2 n}{p} + \frac{4n^2}{p}g + 4l.$$

[1pt] This algorithm works for $p \leq n$.

(b) [3 pt] We can use an $M \times N$ 2D cyclic distribution of the matrix, where $p = MN$, with M, N also powers of 2. We first perform FFT1D$(X_{i,*}, n, N)$ on every row i with $i \bmod M = s$, where we number the processors in 2D fashion with numbers $P(s,t)$. In this 1D FFT, we first perform the superstep S_0 for all the local rows, then S_1 for all the local rows, and then S_2 for all the local rows. After that, we do the same for the columns, using FFT1D$(X_{*,j}, n, M)$, for all local columns j.

The computation cost of this algorithm is the same as in (a). The communication cost is the cost of twice redistributing all the data, and hence $\frac{4n^2}{p}g$. The number of supersteps is only three, since computation superstep S_2 for rows is immediately followed by computation superstep S_0 for columns. The total cost of this algorithm is

$$\frac{10n^2 \log_2 n}{p} + \frac{4n^2}{p}g + 3l.$$

[1pt] This algorithm works for $M, N \leq \sqrt{n}$. If we assume that p is an even power of 2, we can choose $M = N = \sqrt{p}$, and we need to satisfy $p \leq n$.

(c) [2pt] The second approach has lower synchronisation cost, saving l.