Mastermath midterm examination Parallel Algorithms. Solution.

Teacher: Rob H. Bisseling, Utrecht University

October 23, 2019

Each of the four questions is worth 10 points.

1. (a) A BSP algorithm consists of a sequence of supersteps. A superstep contains either a number of computation steps or a number of communication steps, or in certain cases both, followed by a global barrier synchronization.

   In a computation superstep, each processor performs a sequence of operations on local data. In a communication superstep, each processor sends and receives a number of messages. In a mixed superstep, both computation and communication take place.

   At the end of a superstep, all processors synchronize. Each processor checks whether it has finished all its obligations of that superstep. In the case of a computation superstep, it checks whether the computations are finished. Processors wait until all others have finished. When this happens, they all proceed to the next superstep.

   (b) An example of a balanced 4-relation is the situation where processor $P(0)$ sends four data words to $P(1)$, $P(1)$ sends four data words to $P(0)$, $P(2)$ sends four data words to $P(3)$, $P(3)$ sends four data words to $P(2)$. All processors send and receive four data words. Hence the h-relation is a 4-relation.

   An example of an unbalanced 4-relation is when $P(0)$ sends four data words to $P(1)$ and the other two processors are idle. The maximum number sent or received by a processor is four. Hence the h-relation is also a 4-relation.

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

   (b) Superstep (0) takes n/p operations since we flip a bit at most n/p times. The cost is hence $n/p + l$. Superstep (1) costs $(p-1)g + l$ because processor $P(0)$ receives $p-1$ data words (here: bits), and the other processors send one data word. Superstep (2) is similar to Superstep (0), and takes p operations. Its cost is $p + l$. The total cost of the algorithm is

$$\frac{n}{p} + p + (p-1)g + 3l.$$

   (c) $P(s)$ actually needs to put a parity only if it is 1 (odd). So if we initialise $parity_t = 0$ for all t on $P(0)$, we do not have to put even parities. For long vectors (large n), the probabilty of both parities is about equal, except for very small or very large d. In that common case, we save half the communication cost. For $d \approx 0$, the probability of even parity becomes larger than half, so we save even more. For $d \approx 1$, we can reverse the roles of 0 and 1.

Algorithm 1 Parity computation for processor $P(s)$.

```
$b := n/p$;                                    ▷ Superstep (0)
$parity_s := 0$
for $i := sb$ to $(s+1)b - 1$ do
    if $x_i = 1$ then
        $parity_s := 1 - parity_s$;

put $parity_s$ in $P(0)$;                      ▷ Superstep (1)

if $s = 0$ then                                ▷ Superstep (2)
    $parity := 0$
    for $t := 0$ to $p - 1$ do
        if $parity_t = 1$ then
            $parity := 1 - parity$;
```

3. (a) The matrix P_σ is defined by

$$(P_\sigma)_{ij} = \begin{cases} 1 & \text{if } i = \sigma(j) \\ 0 & \text{otherwise,} \end{cases} \quad \text{for } 0 \leq i, j < n.$$

Here, the matrix P_σ is the same as the identity matrix, except for the columns k and r. We have $a_{kr} = a_{rk} = 1$, and all other elements in columns k and r are zero. Note that P_σ is its own transpose, and hence its own inverse.

(b) Multiplying A from the left by P_σ swaps its rows k and r. Multiplying A from the right swaps its columns k and r.

(c) The row and column swap are given as Algorithm 2. For brevity, we write $\phi(i) = i \bmod M$. Since the synchronization cost is high, the row and column swaps must be carried out in the same superstep. This is possible, but it requires special attention to the matrix elements a_{kk} and a_{rr}. These are treated separately.

(d) Assume that $\phi(r) \neq \phi(k)$, which is the worst case for communication. (If $\phi(r) = \phi(k)$, no communication is needed.) A processor that owns part of row k sends n/M data words for the row swap and receives the same number of data words. The same holds for column k. The diagonal processor $(\phi(k), \phi(k))$ needs to send both a part for row k and a part for column k, including a single element a_{kk} sent separately (once). This is also the maximum number of data words any processor has to send or receive. Therefore, the cost of Superstep (1) is $(2n/M - 1)g + l$. The computation in Superstep (1) is for free in our benign cost model, and the cost of this superstep is l. The total BSP cost of the algorithm is

$$(2\frac{n}{M} - 1)g + 2l.$$

Algorithm 2 Row and column swaps for $P(s,t)$.

```
if $\phi(k) = s$ then                                                    ▷ Superstep (0)
    for all $j : 0 \leq j < n \wedge \phi(j) = t \wedge j \neq k \wedge j \neq r$ do
        put $a_{kj}$ as $\hat{a}_{kj}$ in $P(\phi(r), t)$;
if $\phi(r) = s$ then
    for all $j : 0 \leq j < n \wedge \phi(j) = t \wedge j \neq k \wedge j \neq r$ do
        put $a_{rj}$ as $\hat{a}_{rj}$ in $P(\phi(k), t)$;

if $\phi(k) = t$ then
    for all $i : 0 \leq i < n \wedge \phi(i) = s \wedge i \neq k \wedge i \neq r$ do
        put $a_{ik}$ as $\hat{a}_{ik}$ in $P(s, \phi(r))$;
if $\phi(r) = t$ then
    for all $i : 0 \leq i < n \wedge \phi(i) = s \wedge i \neq k \wedge i \neq r$ do
        put $a_{ir}$ as $\hat{a}_{ir}$ in $P(s, \phi(k))$;

if $\phi(k) = s \wedge \phi(k) = t$ then
    put $a_{kk}$ as $\hat{a}_{kk}$ in $P(\phi(r), \phi(r))$;
if $\phi(r) = s \wedge \phi(r) = t$ then
    put $a_{rr}$ as $\hat{a}_{rr}$ in $P(\phi(k), \phi(k))$;
if $\phi(k) = s \wedge \phi(r) = t$ then
    put $a_{kr}$ as $\hat{a}_{kr}$ in $P(\phi(r), \phi(k))$;
if $\phi(r) = s \wedge \phi(k) = t$ then
    put $a_{rk}$ as $\hat{a}_{rk}$ in $P(\phi(k), \phi(r))$;

if $\phi(k) = s$ then                                                    ▷ Superstep (1)
    for all $j : 0 \leq j < n \wedge \phi(j) = t \wedge j \neq k \wedge j \neq r$ do
        $a_{kj} := \hat{a}_{rj}$;
if $\phi(r) = s$ then
    for all $j : 0 \leq j < n \wedge \phi(j) = t \wedge j \neq k \wedge j \neq r$ do
        $a_{rj} := \hat{a}_{kj}$;
if $\phi(k) = t$ then
    for all $i : 0 \leq i < n \wedge \phi(i) = s \wedge i \neq k \wedge i \neq r$ do
        $a_{ik} := \hat{a}_{ir}$;
if $\phi(r) = t$ then
    for all $i : 0 \leq i < n \wedge \phi(i) = s \wedge i \neq k \wedge i \neq r$ do
        $a_{ir} := \hat{a}_{ik}$;
if $\phi(k) = s \wedge \phi(k) = t$ then
    $a_{kk} := \hat{a}_{rr}$;
if $\phi(r) = s \wedge \phi(r) = t$ then
    $a_{rr} := \hat{a}_{kk}$;
if $\phi(k) = s \wedge \phi(r) = t$ then
    $a_{kr} := \hat{a}_{rk}$;
if $\phi(r) = s \wedge \phi(k) = t$ then
    $a_{rk} := \hat{a}_{kr}$;
```

4. (a) We choose an $M \times M$ block distribution for the $n \times n$ matrix C. The element c_{ij} is computed at processor $P(i \text{ div } b, j \text{ div } b)$, where $b = n/M$ is the block size. To compute the value

$$c_{ij} = \sum_{r=0}^{k-1} a_{ir} b_{rj},$$

we need the values of row i of A and of column j of B. These values are available at $P(i \text{ div } b, 0)$ and $P(0, j \text{ div } b)$, respectively.

At the start of the algorithm, we obtain all the rows needed. Processor $P(s, 0)$ thus has to broadcast its block of n/M rows of A to all the other processors in its processor row $P(s, *)$. Each row has k elements.

The broadcast is most efficiently carried out as a two-phase broadcast, by first sending a set of n/p rows to each processor $P(s,t)$ and then broadcasting this set to the others in processor row $P(s, *)$. Note that we must spread the corresponding $n/M \times k$ block of A by spreading the rows, and not the columns, because there are too few columns (the matrix being too skinny). The broadcast of the columns of B is carried out in a similar way, and its phases are merged with those of A to save two synchronisations. The computation is then done in the next (final) superstep for all the n^2/p local elements c_{ij}.

(b) In Superstep (0) of the algorithm, we perform phase 0 of the broadcast for both A and B. Processor $(0,0)$ has most work, as it sends out a block of size $n/M \times k$ of A, except the part it keeps (n/p rows of size k), and similarly for B. The total cost is thus $2(nk/M - nk/p)g + l = (2nk/p)(M-1)g + l$. In Superstep (1), each received part of size $n/p \times k$ is broadcast, costing $2(nk/p)(M-1)g + l$ for A and B together. In Superstep (2), a perfectly balanced computation is carried out, each processor performing $2k$ flops for each of its n^2/p elements, at a cost of $2n^2k/p + l$. The total cost is

$$\frac{2n^2k}{p} + \frac{4nk(M-1)}{p} g + 3l.$$