Mastermath midterm examination
Parallel Algorithms. Solution

Teacher: Rob H. Bisseling, Utrecht University

October 26, 2022

1. (a) A mixed superstep contains both computation and communication. Each processor performs a number of operations on local data and it sends data to other processors or it receives data from them.

   (b) The cost of a mixed superstep is an expression

   $$T = w + hg + l,$$

   where w is the maximum number of flops of a processor in the superstep, $h = \max(h_{\rm s}, h_{\rm r})$ with $h_{\rm s}$ the maximum number of data words a processor sends, $h_{\rm r}$ the maximum number of data words a processor receives, g the time per data word, l the global synchronisation time.

2. (a) The basic idea is that every processor $P(s)$ computes its local partial maxima, without regard for the others. $P(0)$ will then already have the correct result, but the others will have to perform a correction, which consists of comparing the maximum obtained with the maximum for all processors $P(t)$ with $t < s$. This can be done by each processor sending its maximum value to all higher-numbered processors. The maximum of the received maxima is then computed and in a final pass through the local data, the correction is carried ou. This is shown in the following algorithm:

input: $\mathbf{x}$ vector of length n, distr($\mathbf{x}$) = block.
output: $\mathbf{y}$ vector of length n, distr($\mathbf{y}$) = block,
$y_i = \max\{x_j \ : \ 0 \leq j \leq i\}$ for $0 \leq i < n$.

```
b = ⌈n/p⌉;                                          ▷ Superstep (0)
maxval_s := −∞;
for i := sb to (s + 1)b − 1 do
    if x_i > maxval_s then
        maxval_s := x_i;
    y_i := maxval_s;

for t := s + 1 to p − 1 do                          ▷ Superstep (1)
    put maxval_s in P(t);

maxval := −∞;                                       ▷ Superstep (2)
for t = 0 to s − 1 do
    if maxval_t > maxval then
        maxval := maxval_t;
for i := sb to (s + 1)b − 1 do
    y_i := max(y_i, maxval);
```

(b) The cost of superstep (0) is $\frac{n}{p}$ because 1 comparison is performed for every local element. The cost of superstep (1) is $(p-1)g$ because $P(0)$ sends $p-1$ data and other processors send less, and $P(p-1)$ receives $p-1$ data, and others receive less. The cost of superstep (2) is $p - 1 + \frac{n}{p}$ because of at most $p-1$ comparisons for computing *maxval*, and $\frac{n}{p}$ comparisons (taking a max) for the corrections. The total cost is thus

$$T = \frac{2n}{p} + p - 1 + (p-1)g + 3l.$$

3. (a) We choose the square cyclic distribution, because the four submatrices of A will have exactly the same distribution. Here we use the fact that n, M are powers of two, so $n \bmod M = 0$ and the submatrices fit with the distribution. Also $n > M$ so the case $n = M$ is excluded, where submatrices would be on different subsets of the processors. Now submatrices can be added or subtracted without any communication.

   (b) The parallel computation of B becomes:

```
for all i : 0 ≤ i < n/2 ∧ i mod M = s do
    for all j : 0 ≤ j < n/2 ∧ j mod M = t do
        b_ij := a_ij + a_{n/2+i,n/2+j};
        b_{i,n/2+j} := a_{n/2+1,j} − a_{i,n/2+j};
        b_{n/2+i,j} := a_{n/2+1,j} + a_{i,n/2+j};
        b_{n/2+i,n/2+j} := a_ij − a_{n/2+i,n/2+j};
```

(c) The BSP cost is $\frac{n^2}{p} + l$, because every processor computes $\frac{n^2}{p}$ elements of B, each requiring one flop. No communication is needed. The computation of B resembles part of the Strassen algorithm for matrix multiplication, which similarly benefits from a square cyclic distribution in the parallel case.

4. (a) The matrix update for $P(s,t)$ becomes:

```
for all j : k + 1 ≤ j < n ∧ j mod M = t do
    for all i : j ≤ i < n ∧ i mod M = s do
        a_ij := a_ij − a_ik a_jk;
```

(b) First we consider the sequential case. For LU decomposition, the number of elements that are updated is $(n-k-1)^2 = m^2$, where we write $m = n-k-1$. These are all the elements in the lower right hand corner. For Cholesky decomposition, we only update the $(m^2-m)/2$ elements below the main diagonal and the m elements on the diagonal. So together, $(m^2+m)/2 \approx m/2$ elements. We save about a factor of 2 in flops. In the parallel case, all processors have about an equal share of the matrix elements. This also holds for the Cholesky decomposition, since the lower triangular part is spread nearly evenly over the processors. Therefore, in the parallel case we also save about a factor of 2 in flops.

(c) To update matrix element a_{ij}, we need two elements from column k, namely a_{ik} and a_{jk}. If we send a_{ik} to both processor row $P(i \bmod M, *)$ and processor column $P(*, i \bmod M)$, every processor obtains the information it needs.

We want to use a two-phase broadcast to make this efficient. First we spread the elements from column k, for instance by sending a_{ik} from $P(i \bmod M, k \bmod M)$ to $P(i \bmod M, (i \operatorname{div} M) \bmod M)$. Then we send the elements to their final destination.

(d) In LU decomposition, we spread column k and row k in the first phase, costing about $2\frac{m}{M}g$. For Cholesky decomposition, we only need to spread column k, costing $\frac{m}{M}g$.

In LU decomposition, each processor then has about $\frac{m}{p}$ values of column k which need to be sent in the second phase to $M-1$ processors, and the same for row k, at a total cost of

$$2\frac{m}{p}\cdot(M-1)g \approx 2\frac{m}{M}g.$$

For Cholesky decomposition, each processor has about $\frac{m}{p}$ values of column k, which need to be sent in the second phase to $2M-1$ processors, at a total cost of

$$\frac{m}{p}\cdot(2M-1)g \approx 2\frac{m}{M}g.$$

The total communication cost for the superstep is then $4\frac{m}{M}g$ for LU decomposition and $3\frac{m}{M}g$ for Cholesky decomposition. Therefore, we save about 25% in communication cost.