Efficient Algorithms
Summer term 2013

Institut für Theoretische Informatik
Leibniz Universität Hannover

Solution to exercise sheet 10

18.06.2013

Another possibility for parallel computers is to consider the message passing model where a global storage is not used. Formally one sees processors as vertices in a undirected graph and the edges are bidirectional communication channels. Still, the processors work synchronously with the following exceptions: After execution of the operation

- `send(Object, ProcessorId)` the sender-processor can immediately continue,
- `receive(Variable, ProcessorId)` the receiver-processor waits until he gets the desired object.

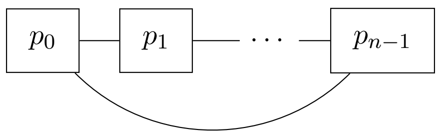
The underlying net-topology has three important attributes:

Diameter Maximum distance of two vertices.

Degree Maximum degree of a vertex.

Bisection width Divide the set of vertices into two almost equal halves. The bisection width is then the number of edges connecting both sets.

Exercise 1: A *processor ring* is an array consisting of n processors which are linearly linked and the last processor in the array is linked to the first:



1. What are the attribute values of this topology?
2. Given an $n \times n$ matrix A and an n-vector $\vec{b}$ one shall compute $A \cdot \vec{b}$. Construct an algorithm on a processor ring with $N \leq n$ processors running in $\Theta(\frac{n^2}{N})$ time with communication time $\Theta(N \cdot n)$.

Solution:

1. **Diameter** $\left\lfloor \frac{N}{2} \right\rfloor$

   Degree 2

   Bisection width 2

2. W.l.o.g. let N be a divisor of n and let $r = \frac{n}{N}$. Now the idea is to divide A into N sub matrices of size $n \times r$,

$$A = \left(\underbrace{A_0}_{r} \;\vdots\; \underbrace{A_1}_{r} \;\vdots\; \underbrace{\cdots}_{r} \;\vdots\; \underbrace{A_{N-1}}_{r} \right)$$

and $\vec{b}$ such that b_i is a tuple of length r. Then it holds that

$$A \cdot \vec{b} = A_0 \cdot b_0 + A_1 \cdot b_1 + \cdots + A_{N-1} \cdot b_{N-1}.$$

Algorithm 1: matMultVec $\langle$Ring$\rangle$

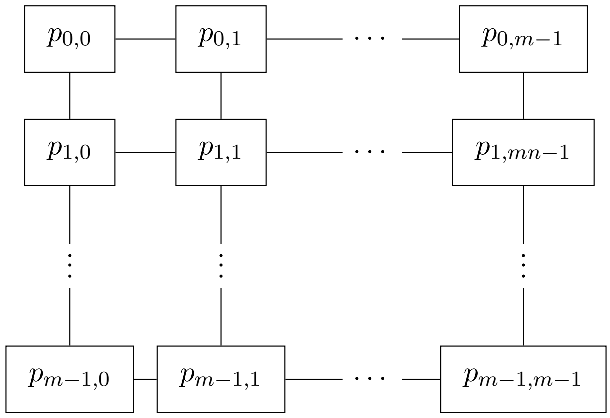
N, p: integer; // initialized $0, \leq p < N$
B: array$[1,\dots,n][1,\dots,r]$ of real; // Submatrix A_p
w: array$[1,\dots,r]$ of real; // Subvector b_p
y, z: array$[1,\dots,n]$ of real;
Compute locally $z = A_p \cdot b_p$;
if $p = 0$ **then** $y := 0$;
else receive$(y, p-1)$;
$y := y + z$;
send$(y, (p+1) \mod N)$;
if $p = 0$ **then** receive$(y, N-1)$;

Observations:

- The processors compute in parallel the products $A_p \cdot b_p$. Then all processors with ids 1 to $N-1$ wait for the data of their left neighbor. P_0 sets y to $A_0 \cdot b_0$ and sends the result to the right neighbor etc. P_0 then waits for the full sum.
- The computation time is $\Theta(r \cdot n) = \Theta(\frac{n^2}{N})$.
- The communication time is $\Theta(N \cdot n)$ because P_0 has to wait until all values are summed up.

$\square$

Exercise 2: A *processor grid* consists of $N = m^2$ processors which are arranged as an $m \times m$ matrix and are connected as follows:



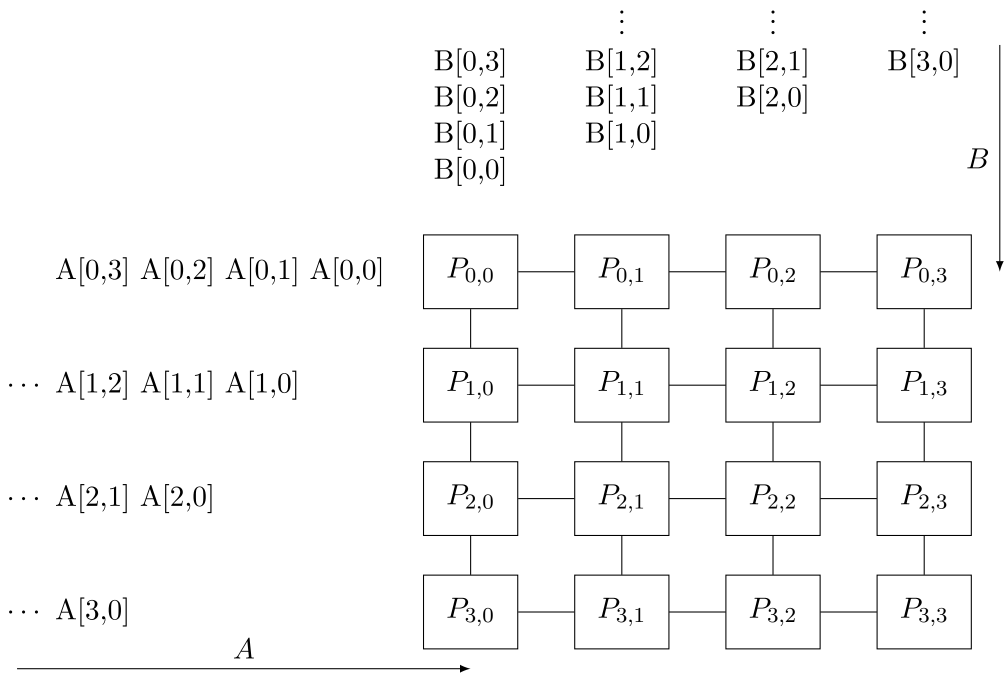
1. What are the attribute values of this topology?
2. Construct an algorithm computing the product of two $n \times n$ matrices in $\Theta(n)$ steps.

Solution:

1. **Diameter** $2(m-1)$

   Degree 4

   Bisection width m
2. **Idea:** The rows of A will be entered in inverted order from left the columns of B from top into the grid. Every processor P_{ij} gets in every step two inputs $A[i,\ell]$ and $B[\ell,j]$, sums the product and sends $A[i,\ell]$ to the right and $B[\ell,j]$ to bottom.



Algorithm 2: matmul $\langle$Grid$\rangle$

n, i, j: integer; // initialized, $(i, j) =$ processor id
A, B, C: real; // A contains $A[i,k]$ and B contains $B[k,j]$
d: integer;
$C := A \cdot B$;
for $d := 1$ *to* $n - 1$ **do**
  send$(A, (i, (j+1) \mod n))$;
  send$(B, ((i+1) \mod n, j))$;
  receive$(A, (i, (j-1) \mod n))$;
  receive$(B, ((i-1) \mod n, j))$;
  $C := C + A \cdot B$;

Observations:

- After $\Theta(n)$ steps processor P_{ij} has computed the correct value of $C[i,j]$.
- Systolic algorithm. The values flow in common mode through the grid.
- After $\Theta(n)$ steps matrix C can be read.
- Speedup of magnitude n^2 — hence optimal speedup.
- The coefficient of A and B can be divided onto the processors of a torus and use the same algorithm.

□