Practical Session 2 : Foster Design

Objective

The objective of this practical session is to apply Foster design methodology to some specific problems.

1 – Maximum value

Suppose you are given a two-dimensional array of dimensions M and N containing real numbers. We make the assumption that M and N are much larger than the number of available processing units p. We also assume that the function $\max(a, b)$ is available.

Question 1

Discuss the first step in Foster methodology. Indicate how you make you choices. You may use diagrams to explain the reasoning.

2 – Matrix vector product

We consider the matrix-vector multiplication $y = Ax$, where $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$. This product is to be computed on a system with p processing units. In the first stage, we do no consider the topology of the network.

The matrix A has been partitioned column-block-wise in blocks A_i of size $n \times (n/p)$ and vector x in blocks X_i of size n/p respectively,

$$A = \left(A_1, A_2, \cdots, A_p\right)$$

and

$$x = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix}$$

Question 2

Discuss the first step in Foster methodology. Indicate how you make you choices. You may use diagrams to explain the reasoning.

3 – Matrix product

We consider the following algorithm for the product of two matrices

Algorithme 1 : Matrices product

Data : Matrices A and B
Result : Matrix C

for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** p **do**
    sum $= 0$
    for $k \leftarrow 1$ **to** m **do**
      sum $= A_{ik}B_{kj}$
    end
    C_{ij} $=$ sum
  end
end

Question 3

Design an algorithm that enable the computation of matrices product in parallel.

4 – Maximum subarray

In order to find the maximum subarray, that is find the subarray of a matrix with the maximum sum, we can use Kadane's algorithm algorithm

Algorithme 2 : 1D Kadane's algorithm

```
Data : array
Result : maxSum, left, right
(maxSum, left, right) = (−∞, 0, 0)
currentMaxSum = 0
currentStartIndex = 0
for i ← 1 to n do
    currentMaxSum = currentMaxSum + array(i)
    if currentMaxSum > maxSum then
        (maxSum, left, right) = (currentMaxSum, currentStartIndex, i)
    end
    if currentMaxSum < 0 then
        currentMaxSum = 0
        currentStartIndex = i + 1
    end
end
```

The 2D algorithm relies on the use of Kadane's algorithm

Algorithme 3 : 2D maximum subarray

```
ă Data : array
Result : maxSum, left, right, top, bottom
(maxSum, left, right, top, bottom) = (−∞, 0, 0, 0, 0)
for i ← 0 to n do
    temp(0 :n-1) = 0
    for j ← i to n do
        for k ← 0 to m do
            temp(k) += array(j,k)
        end
        sum = kadane(temp, start, finish)
        if sum > maxSum then
            (maxSum, left, right, top, bottom) = (sum, i, j, start, finish)
        end
    end
end
```

Question 4

Design an algorithm that enable the resolution of the maximum subarray problem in parallel in 2D.