Examen du cours d'Algorithmique et de Programmation Parallèles et Distribuées

Frédéric Vivien

29 janvier, 2016, 9h00–12h00

- No documents are authorized.
- All the requested algorithms *must* be written in pseudo-code.
- Explanations of the algorithms are as important as the pseudo-codes themselves; a perfect algorithm without any explanation will only be awarded half of the points.
- Answers can be provided either in French or in English.

1 Algorithms for PRAMs

1.1 Matrix generation

We consider a PRAM with p processors. We are given two vectors, x and y, both of size n. We want to generate the $n \times n$ matrix A defined by $A_{i,j} = x_i \times y_j$ for $1 \leq i, j \leq n$.

1. Propose an algorithm to generate such a matrix on a CRCW PRAM with $p = n^2$ processors. What is its complexity? Is your algorithm optimal?
2. Propose an algorithm to generate such a matrix on a CREW PRAM with $p = n^2$ processors. What is its complexity? Is your algorithm optimal?
3. Propose an algorithm to generate such a matrix on a EREW PRAM with $p = n^2$ processors. What is its complexity? Is your algorithm optimal?
4. What would be the complexity of your algorithms on PRAMs with $n^2/\log(n)$ processors?
5. Are your algorithms optimal on PRAMs with $n^2/\log(n)$ processors?
6. Propose an algorithm to compute the sum of the elements of each diagonal on an EREW PRAM with n^2 processors. The sum of the k-th diagonal is the sum of the elements $A_{i,i+k}$.
7. What is the complexity of your solution. Is it optimal?

1.2 Changing the playground

Let us assume that we have an algorithm $\mathcal{A}$ that solves a certain problem on a CREW PRAM with n^2 processors in time $O(\log n)$ while executing a total of $n \log n$ operations.

1. What would be the execution time of an algorithm solving the same problem on a CREW PRAM with n processors?
2. What would be the execution time of an algorithm solving the same problem on a EREW PRAM with n^2 processors?
3. What would be the execution time of an algorithm solving the same problem on a EREW PRAM with n processors?

1.3 Tree of largest size

In this exercise we consider CRCW PRAMs. Let $\mathcal{F}$ be a forest, that is, a set of trees. Overall, this forest contains n nodes. Each node i is associated to a processor $P(i)$ and holds a pointer to its father *parent(i)*. If node i is the root of a tree, *parent(i)* is equal to NULL.

1. Propose a CRCW algorithm that computes the size (in number of nodes) of the tree of maximum size in the forest.
2. What is the complexity of your algorithm?
3. Can your algorithm be adapted to run on an EREW PRAM? (If not, could it be adapted assuming some additional hypotheses?.)

2 Message-passing algorithms for grids of processors

We assume here that we have a set of p processors which are arranged along a 3-dimensional grid. Each processor will be identified by its three coordinates in the 3D grid. We will thus consider the set of processors $P_{i,j,k}$ with $1 \leq i \leq p_1$, $1 \leq j \leq p_2$, and $1 \leq k \leq p_3$, with $p_1 \times p_2 \times p_3 = p$. Processor $P_{i,j,k}$ has a direct communication link with the processors $P_{i-1,j,k}$, $P_{i+1,j,k}$, $P_{i,j-1,k}$, $P_{i,j+1,k}$, $P_{i,j,k-1}$, and $P_{i,j,k+1}$ (whenever these processors exist).

We have three matrices of size $n \times n$, A, B, and C and we want to compute $C = C + A \times B$.

2.1 3D Matrix Multiplication

We assume here that the p processors form a "perfect" 3-dimensional grid of size $\sqrt[3]{p}$: $p_1 = p_2 = p_3 = \sqrt[3]{p}$.

We consider the following data layout:

- Matrix A is distributed so that processor $P_{i,j,1}$ owns a $\frac{n}{\sqrt[3]{p}}$-by-$\frac{n}{\sqrt[3]{p}}$ block $\hat{A}_{i,j}$ of A (for $1 \leq i,j \leq \sqrt[3]{p}$).
- Matrix B is distributed so that processor $P_{1,j,k}$ owns a $\frac{n}{\sqrt[3]{p}}$-by-$\frac{n}{\sqrt[3]{p}}$ block $\hat{B}_{j,k}$ of B (for $1 \leq j,k \leq \sqrt[3]{p}$).
- Matrix C is distributed so that processor $P_{i,1,k}$ owns a $\frac{n}{\sqrt[3]{p}}$-by-$\frac{n}{\sqrt[3]{p}}$ block $\hat{C}_{i,k}$ of A (for $1 \leq i,k \leq \sqrt[3]{p}$).

1. Propose a matrix-multiplication algorithm using this data layout.

   Remark: if you need more involved communication primitives than *send* and *receive* (like macro-communications), provide the pseudo-code for the primitives used by your algorithm.
2. What is the complexity of your solution?

2.2 2.5D Matrix Multiplication

We assume here that the p processors form a grid of size $\sqrt{\frac{p}{c}} \times \sqrt{\frac{p}{c}} \times c$. In other words, we assume that: $p_1 = p_2 = \sqrt{\frac{p}{c}}$, and that $p_3 = c$.

Here, A, B, and C have the same initial data layout: processor $P_{i,j,1}$ owns the $\frac{n}{\sqrt{\frac{p}{c}}}$-by-$\frac{n}{\sqrt{\frac{p}{c}}}$ blocks $\hat{A}_{i,j}$, $\hat{B}_{i,j}$, and $\hat{C}_{i,j}$ of matrices A, B and C (for $1 \leq i,j \leq \sqrt{\frac{p}{c}}$).

1. Propose a matrix-multiplication algorithm using this data layout.

   Hint: Initially, processor $P_{i,j,1}$ sends its blocks $\hat{A}_{i,j}$ and $\hat{B}_{i,j}$ to all the $P_{i,j,k}$ processors.
2. What is the complexity of your solution?
3. Could this approach be beneficial with respect to a classic 2D matrix multiplication?

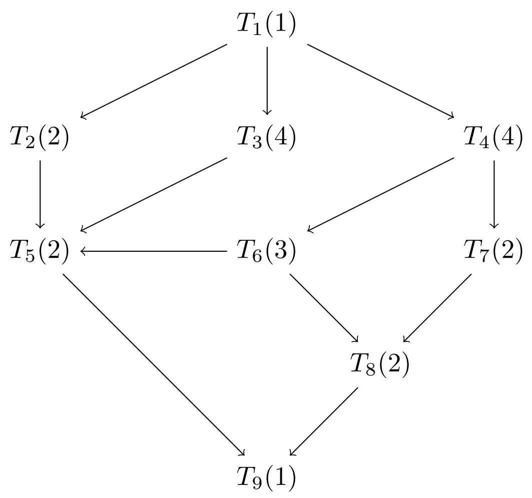

Figure 1: Task graph without communications.

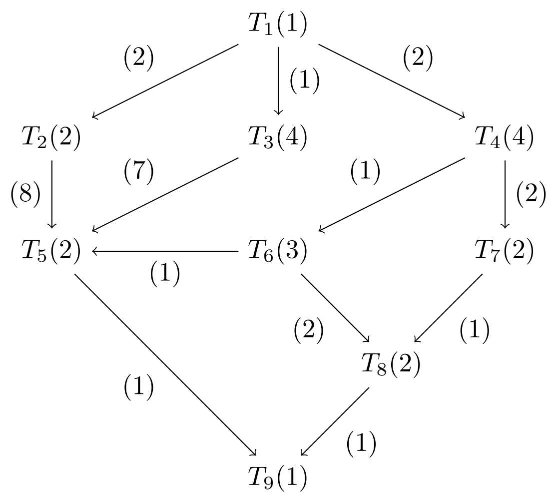

Figure 2: Task graph with communications.

3 Task graph scheduling

3.1 Task graph scheduling without communications

We now consider the directed acyclic task graph on Figure 1 and a computing platform with an infinite number of processors and with negligible communication times. All execution times are indicated between parentheses besides the labels of the vertices.

1. What is the optimal execution time on an unlimited number of processors $\mathrm{MS}_{opt}(\infty)$?
2. What is the minimal number of processors needed to execute the *As Soon As Possible* schedule?
3. What is the minimal number of processors needed to execute the *As Late As Possible* schedule?
4. What is the minimal number p_{opt} of processors needed to schedule this graph in a time $\mathrm{MS}_{opt}(\infty)$?

3.2 Task graph scheduling with communications

We now consider the task graph on Figure 2. All execution times are indicated between parentheses besides the labels of the vertices; the communication costs are written between parentheses besides the edges. We assume that we have 3 processors.

1. Compute the *top-level* and the *bottom-level* of each task.
2. Schedule the task graph using the modified critical path heuristic.
3. Schedule the task graph using at least two of the *clustering* heuristics seen during the lectures (you will mention which heuristics you are using and recall their definition).

4 Dependence analysis and parallelization

We consider the following loop nest:

```
for i = 1 to N do
  for j = i + 2 to N do
    S1: a(i + 1, j − 2) = c(i + 1, j − 1)
    S2: b(i, j + 3) = a(i + 2, j − 6)
    S3: c(i + 1, j) = d(i − 2, j + 1) + c(i + 1, j − 5)
    S4: d(i + 1, j) = a(i + 1, j − 3) + c(i, j + 2)
  endfor
endfor
```

1. Compute the dependences for the above loop nest. The type (flow, anti, output) of dependences must be specified with their distance vectors.
2. What contraints should be satisfied by a Lamport's vector $\pi = (a, b)$ used to schedule this loop nest? (Iteration $I = (i, j)$ being executed at time $\pi \cdot I$.)
3. What is the value of Lamport's vector that minimizes the execution time?

5 Distributed algorithms

We consider a connected distributed system with an arbitrary topology. We assume that each processor stores an integer.

1. Propose a distributed algorithm to compute the maximum of the integers stored on the different processors.

   Remark: you will explicit the assumptions, if any, for your solution to be correct.
2. How many processors hold the answer when the algorithm terminates?
3. What is the complexity of your solution?
4. Could your algorithm be reused to compute the sum of the integers? Why?
5. If the answer to the previous question was negative, provide a distributed algorithm to compute the sum. What is its complexity?