Online scheduling on master-slave platforms

Frédéric Vivien

November 27, 2006

Overview

The scheduling problem

The processors

- Parallel
  - Identical
  - Uniforms

The scheduling problem

The processors

- Parallel
  - Identical
  - Uniforms

The tasks

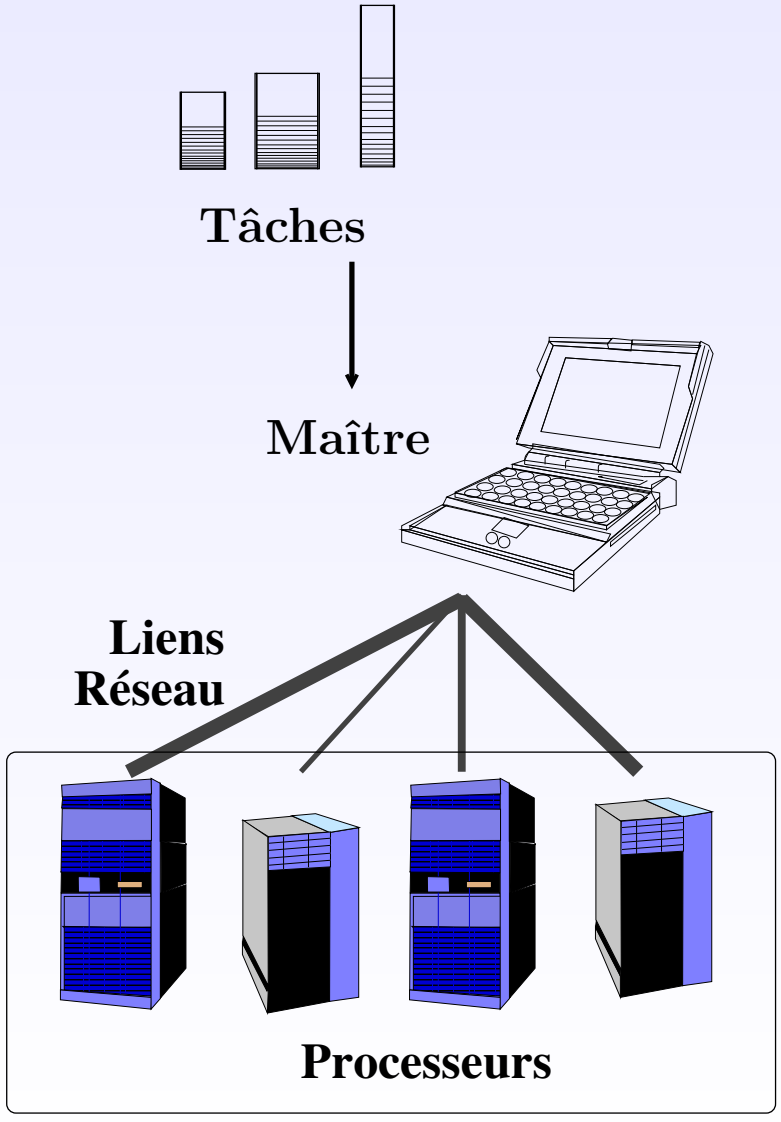
Described by:

- their computation requirements;
- their communication volume;
- their arrival dates.

The scheduling problem

The scheduler

- Gather the tasks
- Send them to the processors



The scheduling problem

The aim

Distribute the tasks to the processors, in order to process these tasks

- While respecting the system constraints
  - on the processors
  - on the tasks
- While optimizing an objective function.

The scheduling problem

Formally

- n tasks, m processors
- $p_{i,j}$: processing time of task i on processor j
- $c_{i,j}$: time to send task i from the master to the slave j
- r_i: arrival date
- C_i: completion time
- The main objective functions:
  - makespan: $\max C_i$
  - maximal flow: $\max C_i - r_i$
  - average flow: $\sum(C_i - r_i)$

The scheduling problem

Simplifying the problem

- identical independent tasks,

The scheduling problem

Simplifying the problem

- identical independent tasks,

Necessary to have efficient algorithms.

The scheduling problem

Simplifying the problem

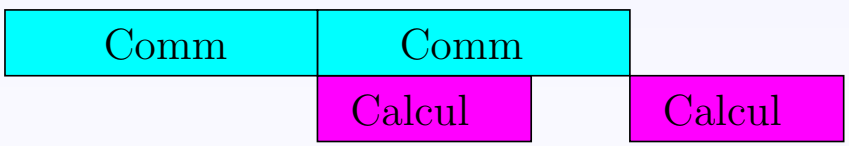
- identical independent tasks,
- Fast communications.

The scheduling problem

Simplifying the problem

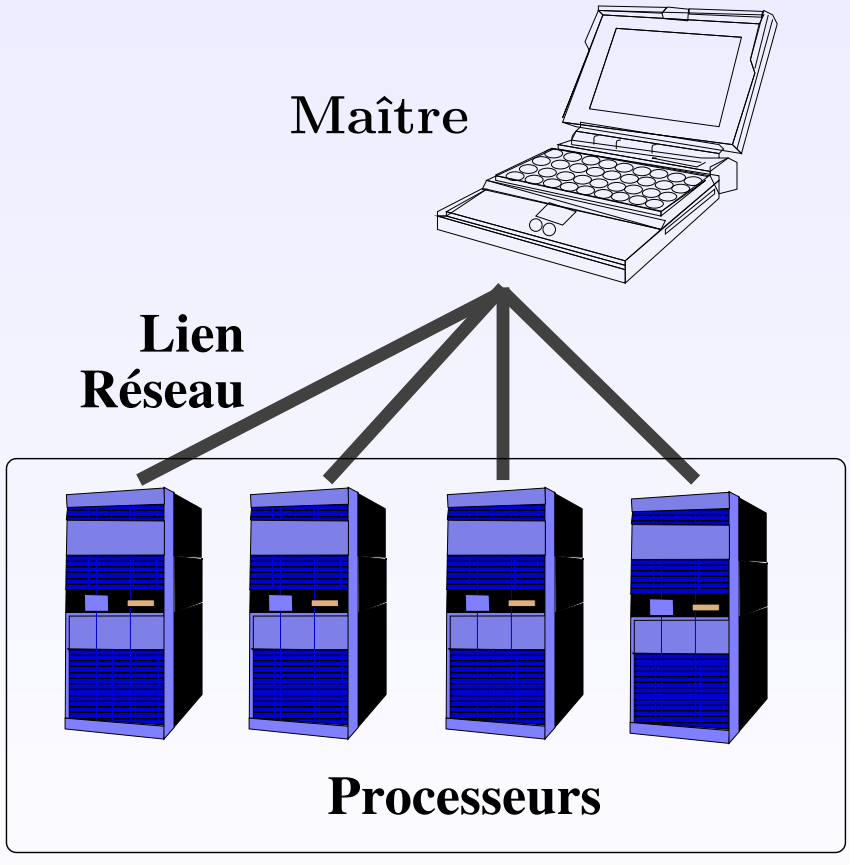
- identical independent tasks,
- Fast communications.

Si $c_{j_0} = min\ c_j$ et $c_{j_0} > p_{j_0}$, then the optimal schedule is trivial.



Overview

Our platform



Our problem

The aim

Minimizing:

- the *makespan*
- the maximal flow
- the sum flow

Our problem

The aim

Minimizing:

- the *makespan*
- the maximal flow
- the sum flow

And the three simultaneously and online, if possible !

Our problem

The aim

Minimizing:

- the *makespan*
- the maximal flow
- the sum flow

And the three simultaneously and online, if possible !

Round-Robin achieves this.

Our problem

The aim

Minimizing:

- the *makespan*
- the maximal flow
- the sum flow

And the three simultaneously and online, if possible !

Round-Robin achieves this.

Round-Robin

Cyclic task allocation:
sends task i to processor $i \ mod \ m$ as soon as possible.

The optimal scheduler: Round-Robin

Principle of the proof

- Explicit an optimal *ASAP* algorithm (ASAP=As Soon As Possible) having a behavior similar to that of *Round-Robin*,
- Show that *ASAP* and *Round-Robin* complete at the same time the execution of each task.

The optimal scheduler: Round-Robin

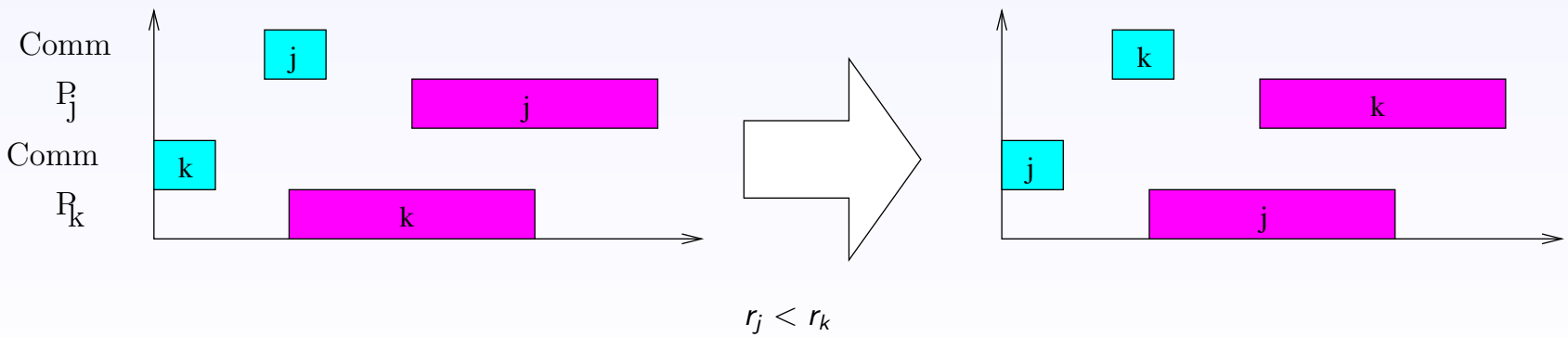
- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$

The optimal scheduler: Round-Robin

- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$
- We build an *ASAP* scheduler from S, also optimal for this metric, and which:
  - sends tasks to slaves by non-decreasing arrival dates

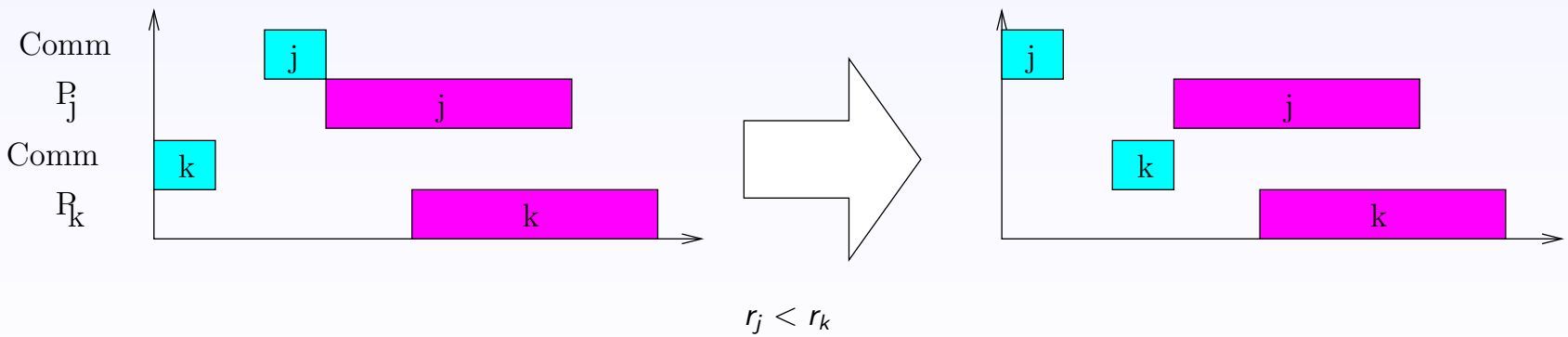
The optimal scheduler: Round-Robin

- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$
- We build an *ASAP* scheduler from S, also optimal for this metric, and which:
  - sends tasks to slaves by non-decreasing arrival dates



The optimal scheduler: Round-Robin

- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$
- We build an *ASAP* scheduler from S, also optimal for this metric, and which:
  - sends tasks to slaves by non-decreasing arrival dates



The optimal scheduler: Round-Robin

- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$
- We build an *ASAP* scheduler from S, also optimal for this metric, and which:
  - sends tasks to slaves by non-decreasing arrival dates
  - makes slaves process tasks by non-decreasing arrival dates

The optimal scheduler: Round-Robin

- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$
- We build an *ASAP* scheduler from S, also optimal for this metric, and which:
  - sends tasks to slaves by non-decreasing arrival dates
  - makes slaves process tasks by non-decreasing arrival dates
  - send tasks to slaves ASAP

The optimal scheduler: Round-Robin

- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$
- We build an *ASAP* scheduler from S, also optimal for this metric, and which:
  - sends tasks to slaves by non-decreasing arrival dates
  - makes slaves process tasks by non-decreasing arrival dates
  - send tasks to slaves ASAP

We show by induction that *ASAP* and *Round-Robin* completes tasks at the very same dates.

The optimal scheduler: Round-Robin

- Let S be an optimal scheduler minimizing $\sum(C_i - r_i)$
- We build an *ASAP* scheduler from S, also optimal for this metric, and which:
  - sends tasks to slaves by non-decreasing arrival dates
  - makes slaves process tasks by non-decreasing arrival dates
  - send tasks to slaves ASAP

We show by induction that *ASAP* and *Round-Robin* completes tasks at the very same dates.

Same reasoning when taking for S an optimal algorithm minimizing the *makespan* or the maximal flow.

The optimal scheduler: Round-Robin

Conclusion

In the end, *Round-Robin* is an optimal algorithm for the minimization of

- *makespan*,
- average flow,
- and maximal flow,

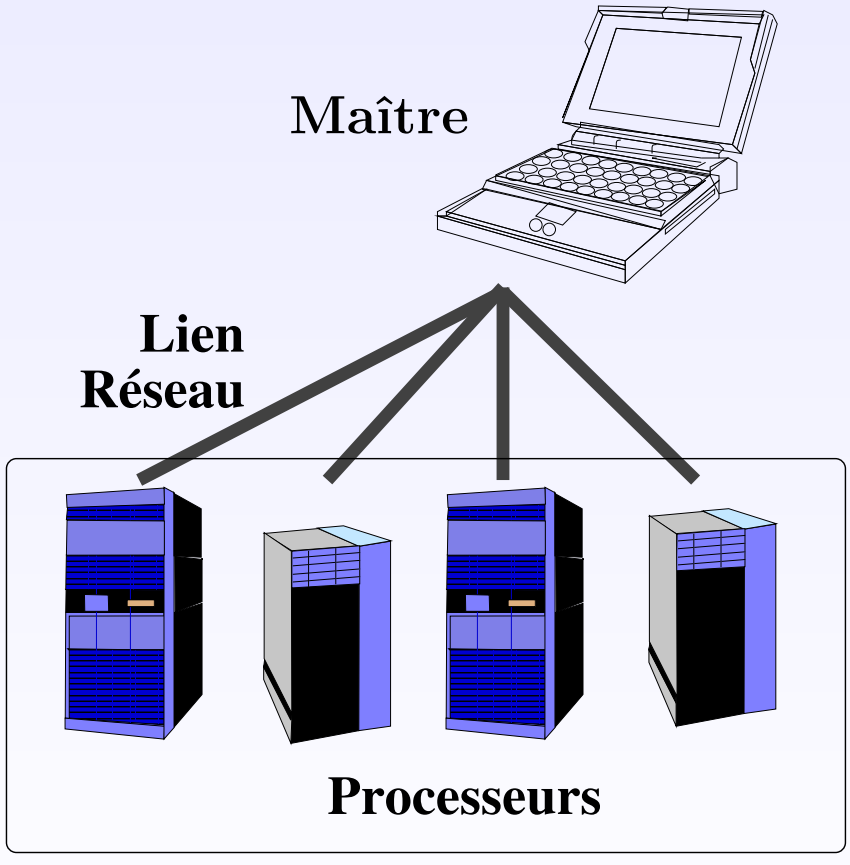
for an online problem (with arrival dates).

Overview

1. The homogeneous case

2. With heterogeneous processors
   - The online problem
   - A solution for a special online problem

The platform



The optimal scheduler...

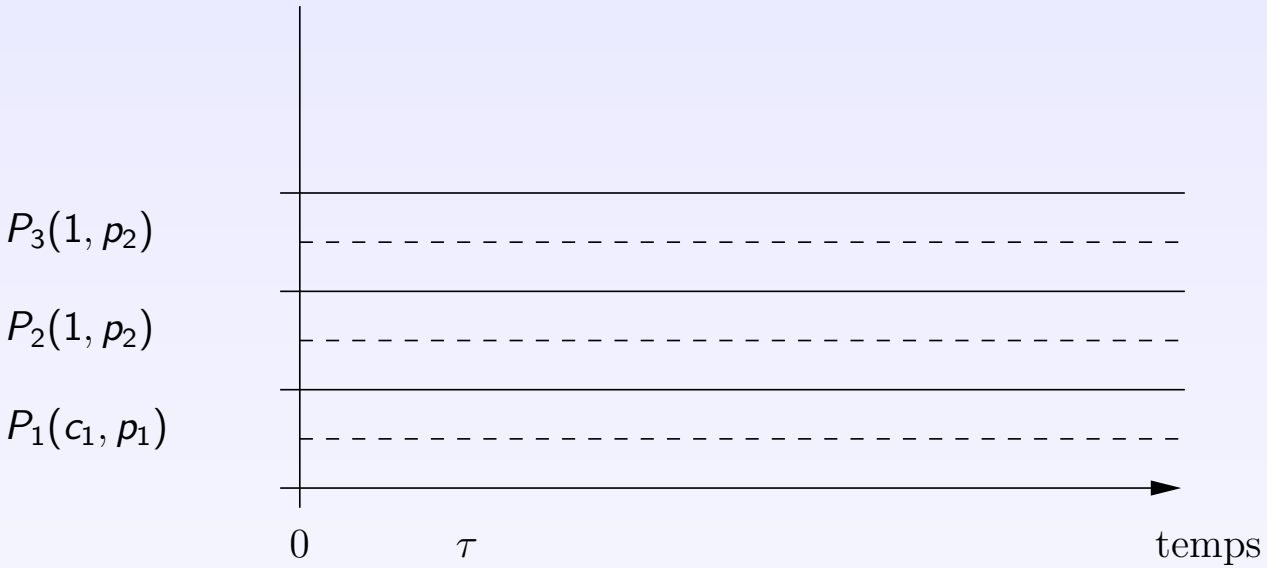
does not exist!!

at least for the online problem and any of the three usual metrics.

Lower bounds on the competitiveness of online algorithms

Platform type	Objective function		
	Makespan	Max-flow	Sum-flow
Communication homogeneous	$\frac{5}{4} = 1.250$	$\frac{5-\sqrt{7}}{2} \approx 1.177$	$\frac{2+4\sqrt{2}}{7} \approx 1.093$
Computation homogeneous	$\frac{6}{5} = 1.200$	$\frac{5}{4} = 1.250$	$\frac{23}{22} \approx 1.045$
Heterogeneous	$\frac{1+\sqrt{3}}{2} \approx 1.366$	$\sqrt{2} \approx 1.414$	$\frac{\sqrt{13}-1}{2} \approx 1.302$

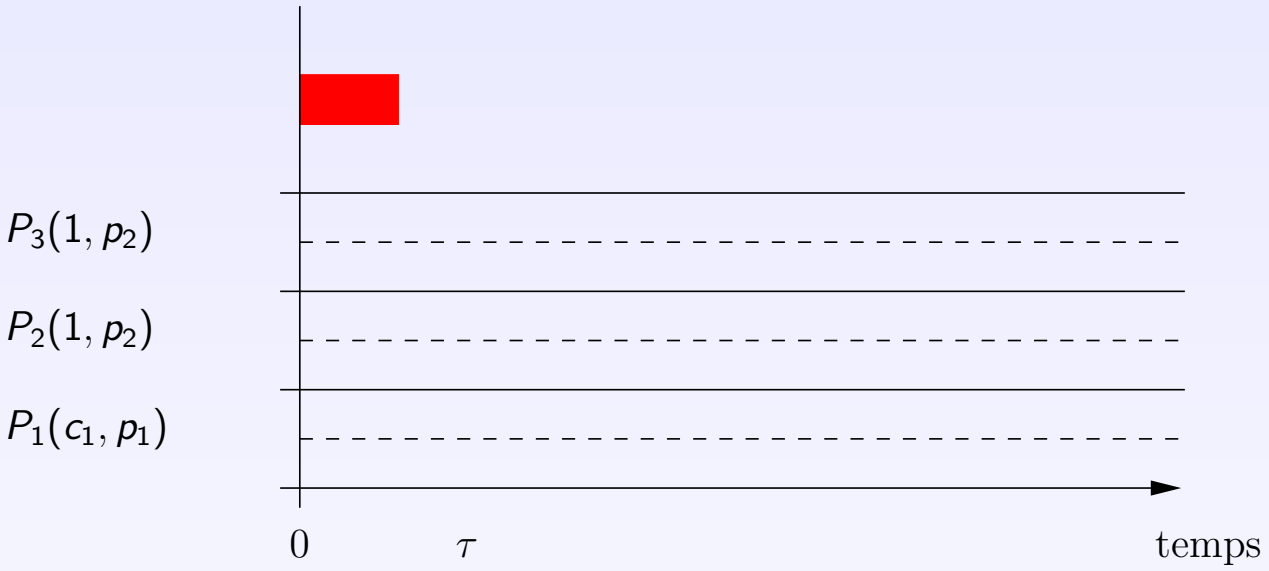
Finding a lower bound on the competitiveness (1)



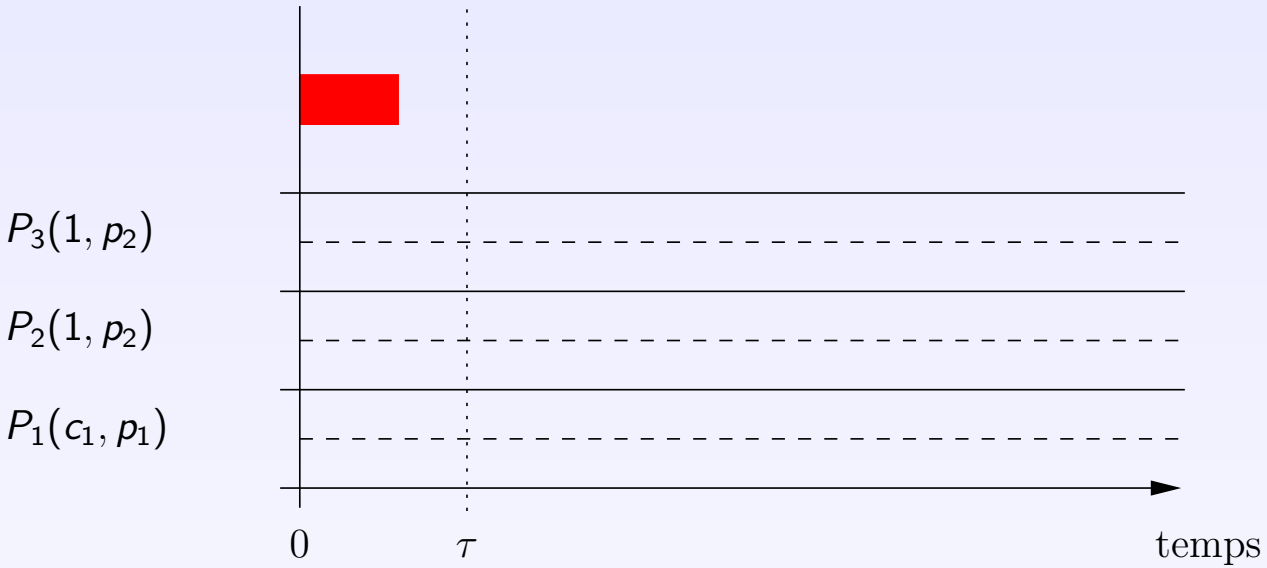
Idea:

- a fast processor with slow communications ($c_1 > 1$);
- two identical and slow processors, but with fast communications;
- if only one task, one must choose the fast processor ($c_1 + p_1 < 1 + p_2$).

Finding a lower bound on the competitiveness (1)

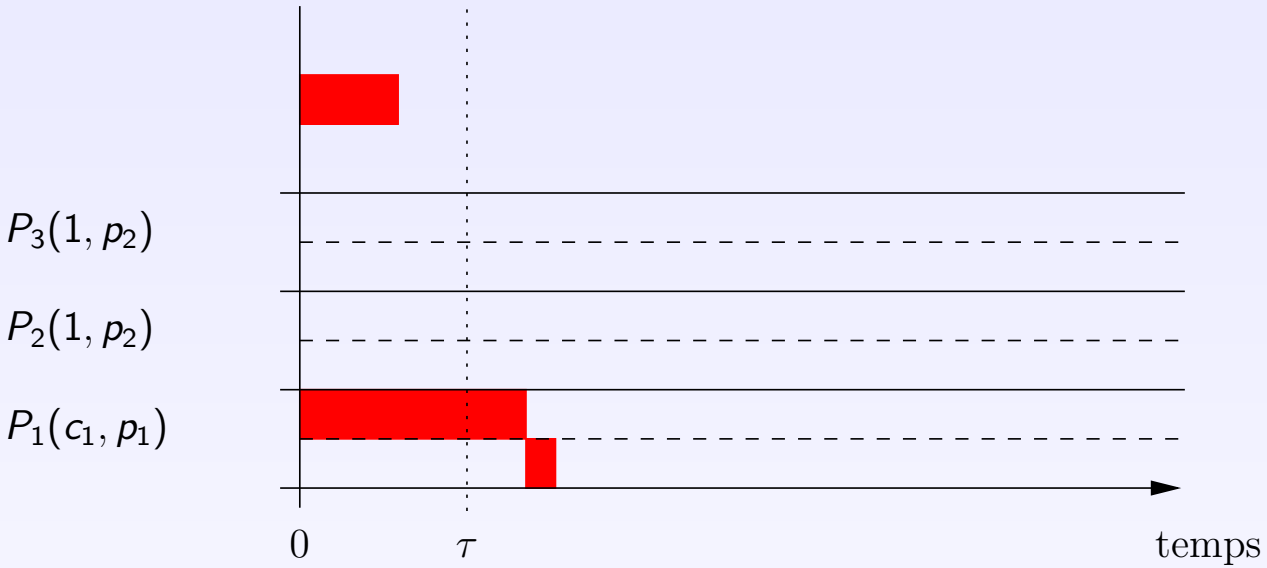


Finding a lower bound on the competitiveness (1)



We look at time $\tau \geq 1$ to see what has happened. Three possibilities:

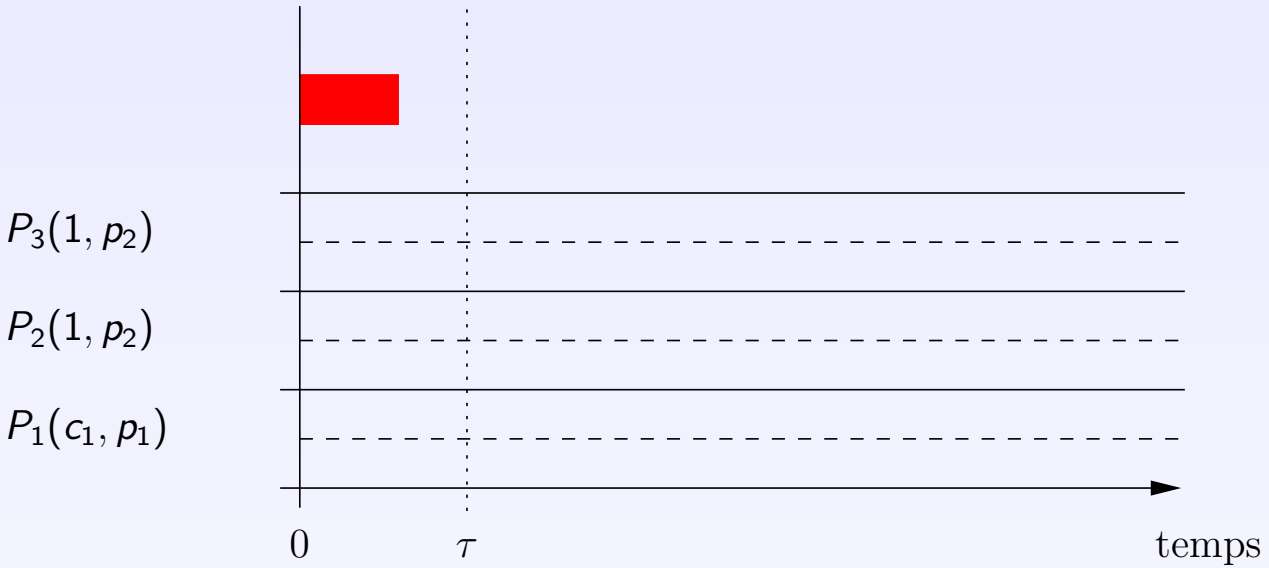
Finding a lower bound on the competitiveness (1)



We look at time $\tau \geq 1$ to see what has happened. Three possibilities:

1. Optimal : task on P_1, max-flow $\geq c_1 + p_1$.

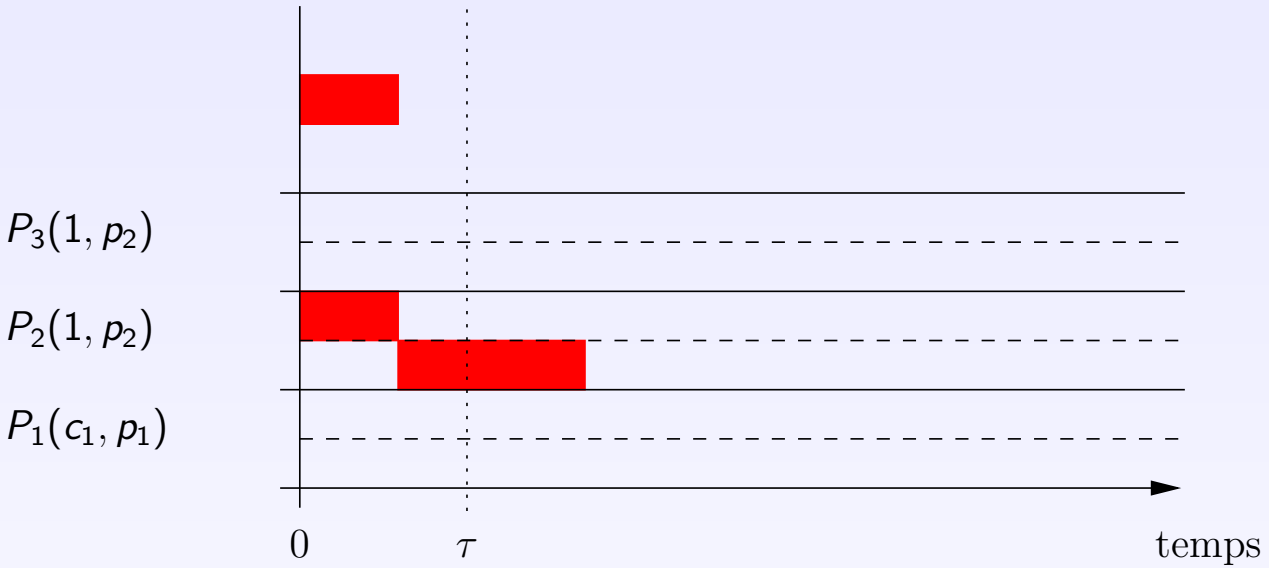
Finding a lower bound on the competitiveness (1)



We look at time $\tau \geq 1$ to see what has happened. Three possibilities:

1. Optimal : task on P_1, max-flow $\geq c_1 + p_1$.
2. Nothing done: max-flow $\geq \tau + c_1 + p_1$, ratio $\geq \frac{\tau + c_1 + p_1}{c_1 + p_1}$.

Finding a lower bound on the competitiveness (1)

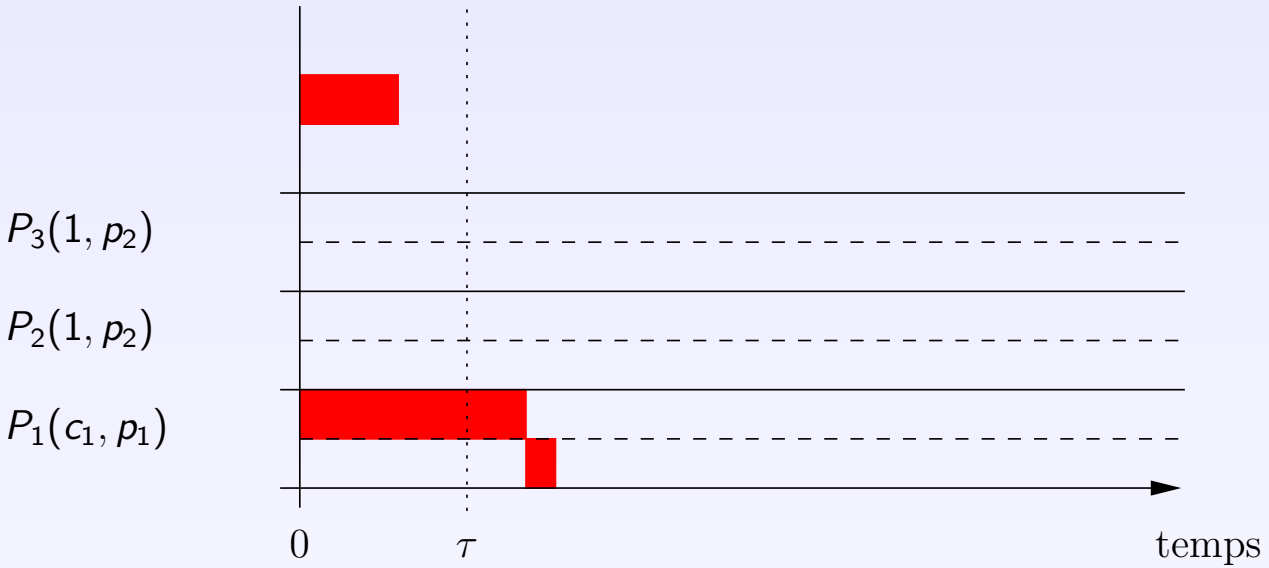


We look at time $\tau \geq 1$ to see what has happened. Three possibilities:

1. Optimal : task on P_1, max-flow $\geq c_1 + p_1$.
2. Nothing done: max-flow $\geq \tau + c_1 + p_1$, ratio $\geq \frac{\tau + c_1 + p_1}{c_1 + p_1}$.
3. Task send to P_2, max-flow $\geq 1 + p_2$. Ratio $\geq \frac{1 + p_2}{c_1 + p_1}$.

We want to force the algorithm to process the first task on P_1.

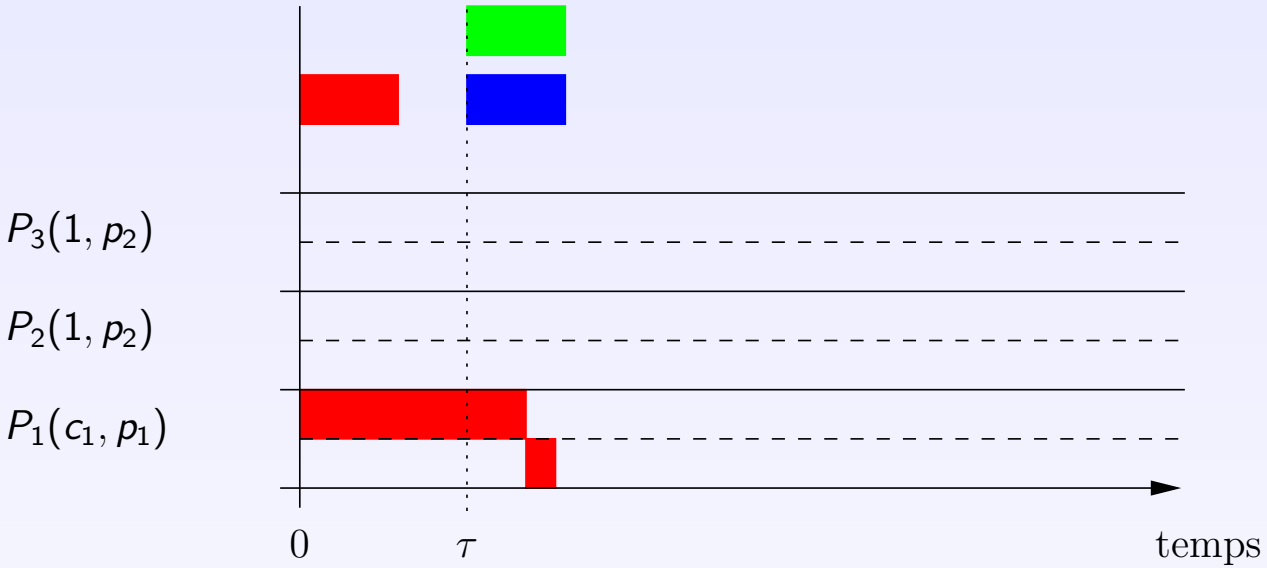
Finding a lower bound on the competitiveness (1)



We look at time $\tau \geq 1$ to see what has happened. We will choose τ, c_1, p_1 and p_2 such that:

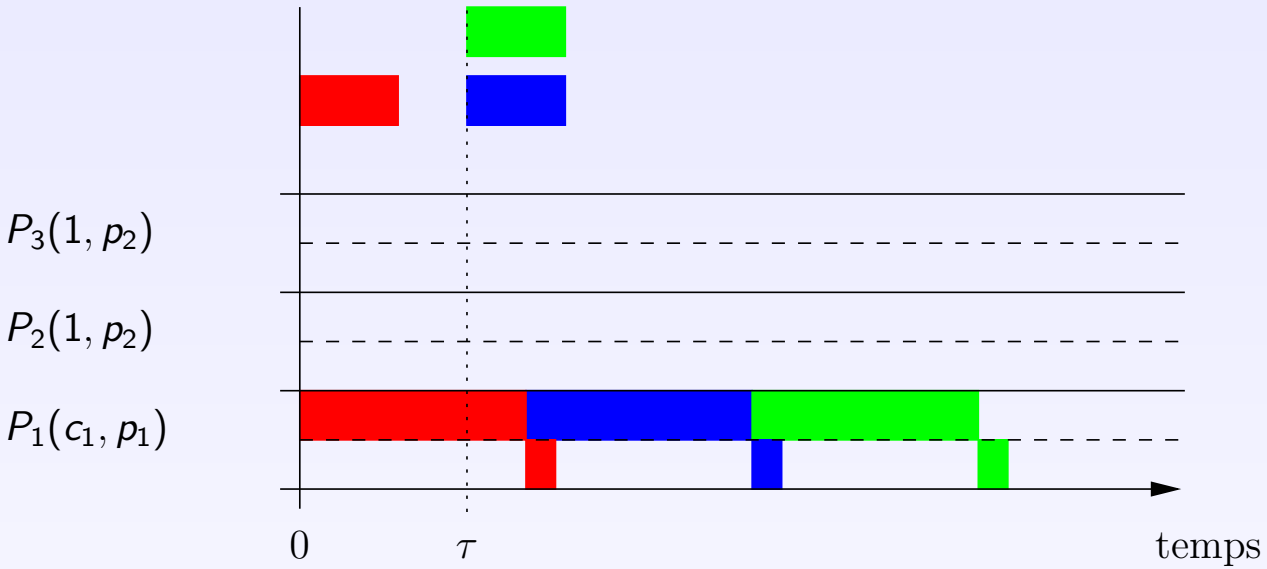
$$\min \left\{ \frac{1 + p_2}{c_1 + p_1}, \frac{\tau + c_1 + p_1}{c_1 + p_1} \right\} \geq \rho$$

Finding a lower bound on the competitiveness (1)



At time τ we send two new tasks.
We consider all the possible cases.

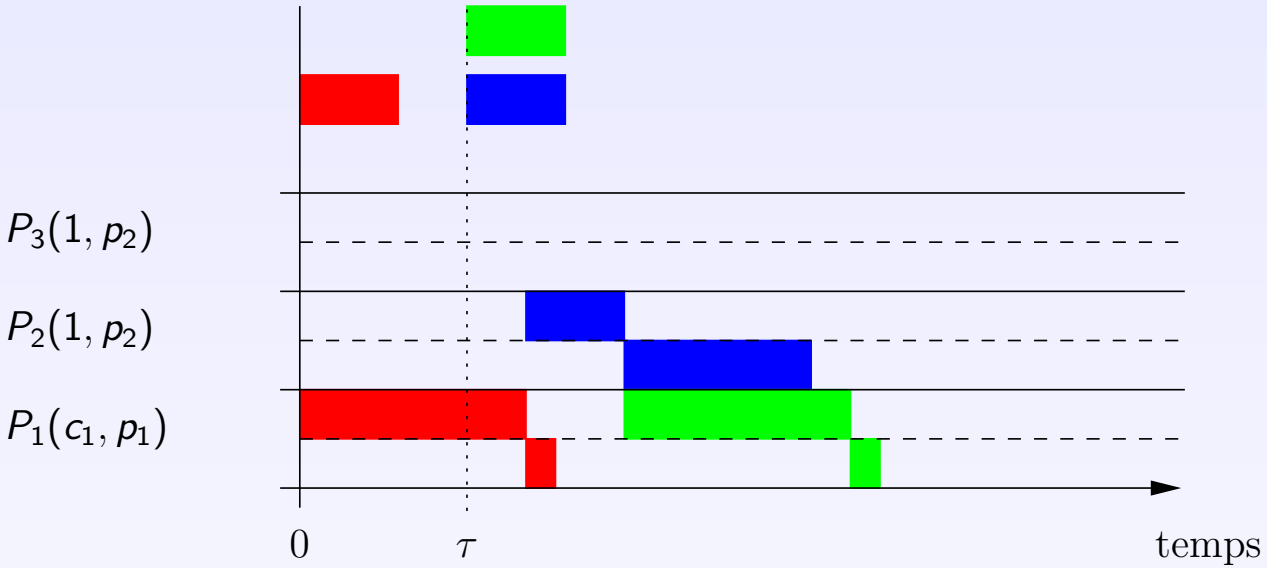
Finding a lower bound on the competitiveness (1)



At time τ we send two new tasks.
The two tasks are executed on P_1:

$$\max\{c_1 + p_1,$$
$$\max\{\max\{c_1, \tau\} + c_1 + p_1, c_1 + 2p_1\} - \tau,$$
$$\max\{\max\{c_1, \tau\} + c_1 + p_1 + \max\{c_1, p_1\}, c_1 + 3p_1\} - \tau\}$$

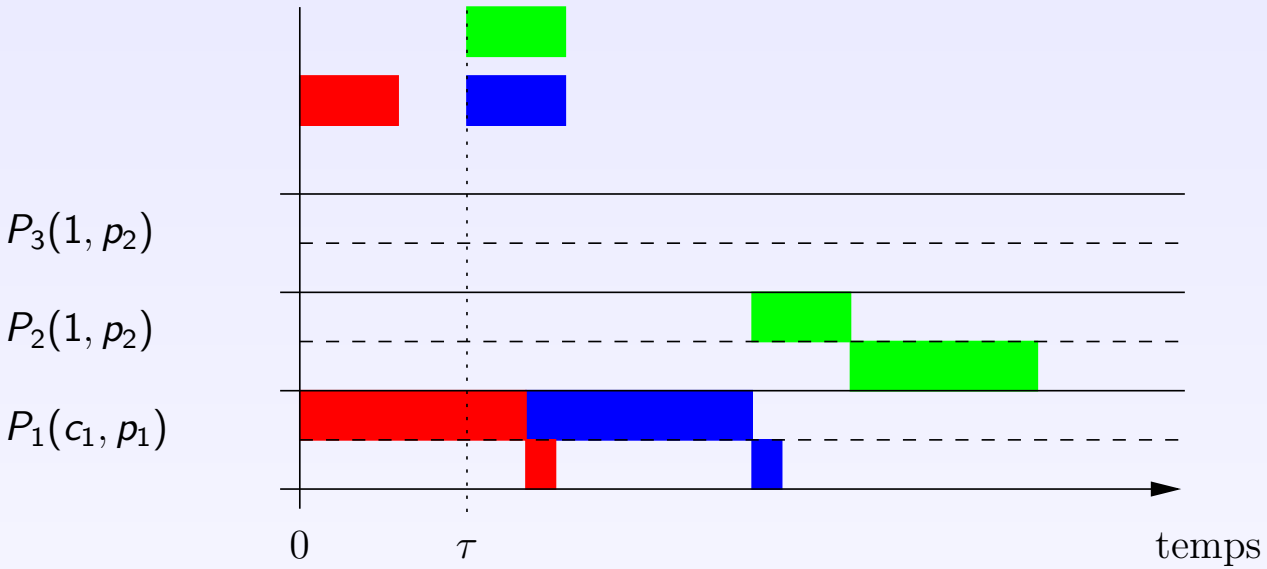
Finding a lower bound on the competitiveness (1)



At time τ we send two new tasks.
The first of the two tasks is executed on P_2 (or P_3), and the other one on P_1.

$$\max\{c_1 + p_1,$$
$$(\max\{c_1, \tau\} + c_2 + p_2) - \tau,$$
$$\max\{\max\{c_1, \tau\} + c_2 + c_1 + p_1, c_1 + 2p_1\} - \tau\}$$

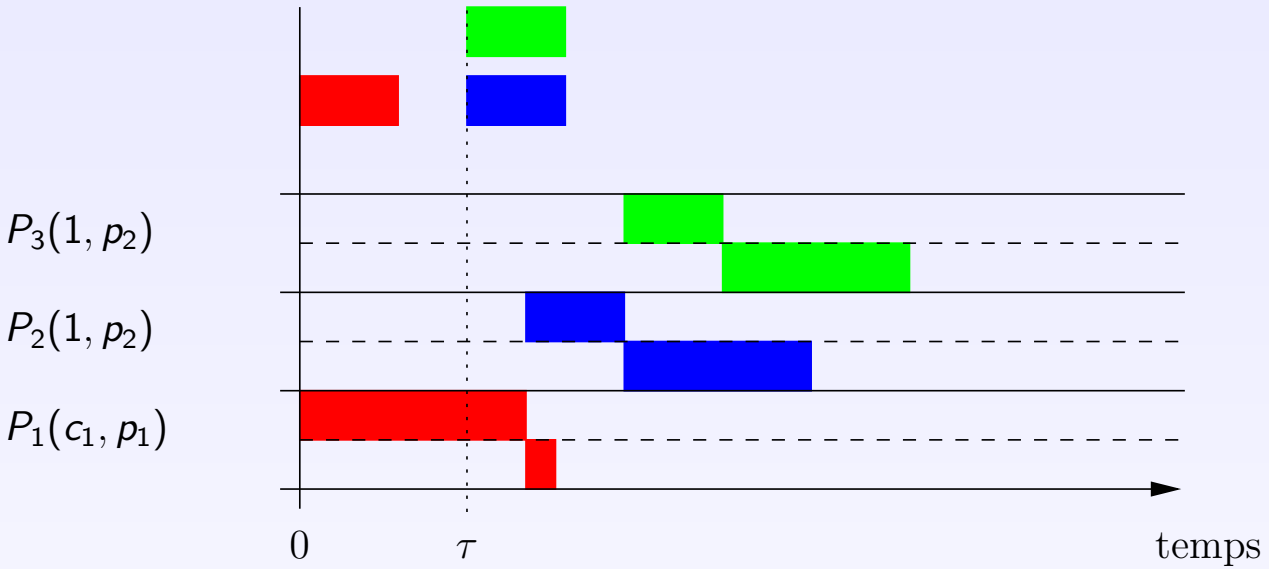
Finding a lower bound on the competitiveness (1)



At time τ we send two new tasks.
The first of the two tasks is executed on P_1, and the other one on P_2 (or P_3).

$$\max\{c_1 + p_1,$$
$$\max\{\max\{c_1, \tau\} + c_1 + p_1, c_1 + 2p_1\} - \tau,$$
$$(\max\{c_1, \tau\} + c_1 + c_2 + p_2) - \tau\}$$

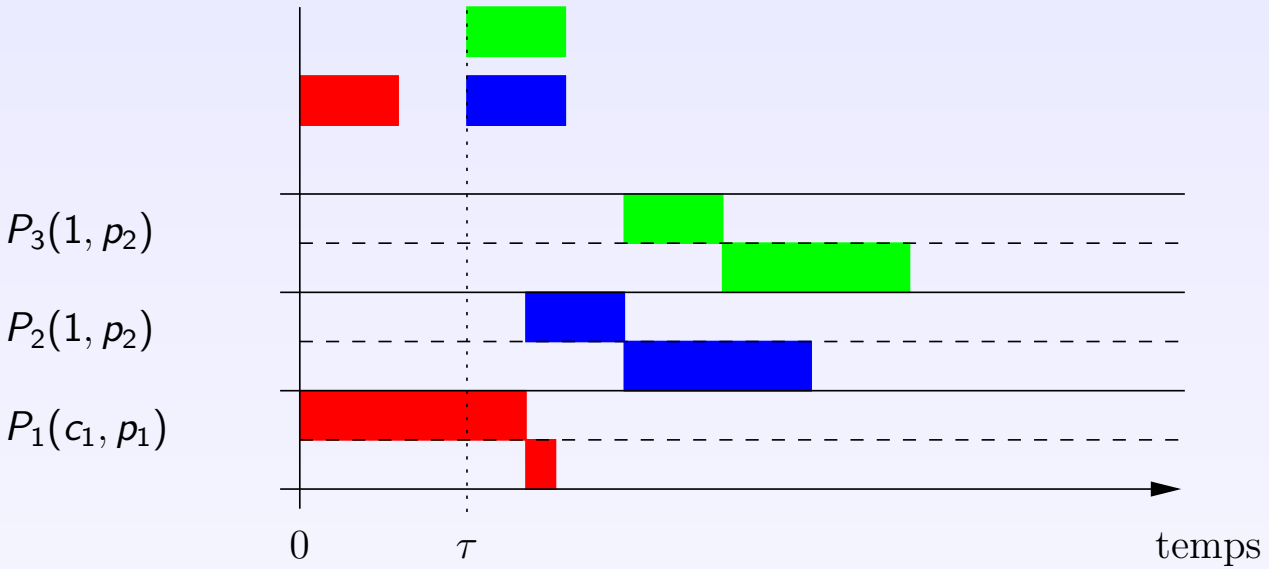
Finding a lower bound on the competitiveness (1)



At time τ we send two new tasks.
One of the two tasks is executed on P_2 and the other one on P_3.

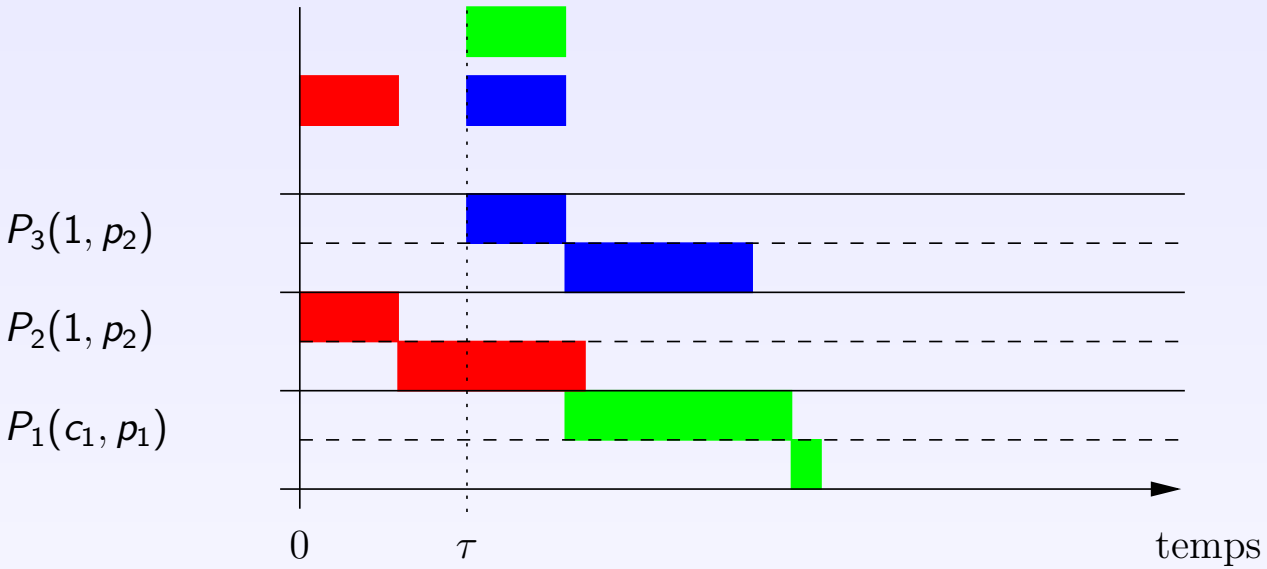
$$\max\{c_1+p_1, (\max\{c_1,\tau\}+c_2+p_2)-\tau, (\max\{c_1,\tau\}+c_2+c_2+p_2)-\tau\}$$

Finding a lower bound on the competitiveness (1)



At time τ we send two new tasks.
The case where both tasks are executed on P_2 (or both on P_3) is worse than the previous one, therefore, we do not need to study it.

Finding a lower bound on the competitiveness (1)



At time τ we send two new tasks.
The (desired) optimal: the first task on P_2, the second on P_3, and the third on P_1.

$$\max\{c_2+p_2, (\max\{c_2,\tau\}+c_2+p_2)-\tau, (\max\{c_2,\tau\}+c_2+c_1+p_1)-\tau\}$$

Finding a lower bound on the competitiveness (2)

Lower bound on the competitiveness of any online algorithm:

$$\min \left\{ \begin{array}{l} \frac{\tau+c_1+p_1}{c_1+p_1}, \\ \\ \frac{1+p_2}{c_1+p_1}, \\ \\ \min \left\{ \begin{array}{l} \max\{c_1+p_1, \max\{\max\{c_1,\tau\}+c_1+p_1, c_1+2p_1\}-\tau, \max\{\max\{c_1,\tau\}+c_1+p_1+\max\{c_1,p_1\}, c_1+3p_1\} \\ \max\{c_1+p_1, (\max\{c_1,\tau\}+c_2+p_2)-\tau, \max\{\max\{c_1,\tau\}+c_2+c_1+p_1, c_1+2p_1\}-\tau\} \\ \max\{c_1+p_1, \max\{\max\{c_1,\tau\}+c_1+p_1, c_1+2p_1\}-\tau, (\max\{c_1,\tau\}+c_1+c_2+p_2)-\tau\} \\ \max\{c_1+p_1, (\max\{c_1,\tau\}+c_2+p_2)-\tau, (\max\{c_1,\tau\}+c_2+c_2+p_2)-\tau\} \end{array} \right. \\ \hline \max\{c_2+p_2, (\max\{c_2,\tau\}+c_2+p_2)-\tau, (\max\{c_2,\tau\}+c_2+c_1+p_1)-\tau\} \end{array} \right.$$

Problem : to find τ, c_1, p_1, and p_2 (as $c_2 = 1$) which maximizes this lower bound.

Constraints : $c_1 + p_1 < p_2$.

Finding a lower bound on the competitiveness (3)

1. Numeric resolution

Finding a lower bound on the competitiveness (3)

1. Numeric resolution
2. Characterization of the shape of the optimal : $\tau < c_1$, $p_1 = 0$, etc.

Finding a lower bound on the competitiveness (3)

1. Numeric resolution
2. Characterization of the shape of the optimal : $\tau < c_1$, $p_1 = 0$, etc.
3. New system:

$$\min \left\{ \begin{array}{l} \frac{\tau+c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \dfrac{\min \left\{ \begin{array}{l} 3c_1 - \tau, \\ c_1 + 1 - \tau + p_2, \\ 2c_1 - \tau + 1 + p_2 \\ c_1 + 2 + p_2 - \tau \end{array} \right.}{1+p_2} \end{array} \right. = \min \left\{ \begin{array}{l} \frac{\tau+c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{c_1+1-\tau+p_2}{1+p_2} \end{array} \right.$$

Finding a lower bound on the competitiveness (3)

1. Numeric resolution
2. Characterization of the shape of the optimal : $\tau < c_1$, $p_1 = 0$, etc.
3. New system:

$$\min \left\{ \begin{array}{l} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \dfrac{\min \left\{ \begin{array}{l} 3c_1 - \tau, \\ c_1 + 1 - \tau + p_2, \\ 2c_1 - \tau + 1 + p_2 \\ c_1 + 2 + p_2 - \tau \end{array} \right.}{1+p_2} \end{array} \right. = \min \left\{ \begin{array}{l} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{c_1+1-\tau+p_2}{1+p_2} \end{array} \right.$$

4. Solution: $c_1 = 2(1+\sqrt{2})$, $p_2 = \sqrt{2}c_1 - 1$, $\tau = 2$, $\rho = \sqrt{2}$.

Online minimization of makespan

We can show that if a scheduler X knows:

- neither the number of tasks it will have to schedules,
- nor their arrival dates,

it cannot find an optimal schedule minimizing in any case one of the three usual metrics.

We even obtain a lower bound on the competitiveness of any algorithm of $\frac{5}{4}$ for the *makespan*.

Online minimization of makespan

We can show that if a scheduler X knows:

- neither the number of tasks it will have to schedules,
- nor their arrival dates,

it cannot find an optimal schedule minimizing in any case one of the three usual metrics.

We even obtain a lower bound on the competitiveness of any algorithm of $\frac{5}{4}$ for the *makespan*.

Let us give it more knowledge.

Scheduling the Last Job First

Optimal algorithm...

for makespan minimization when one knows in advance the total number of tasks to schedule.

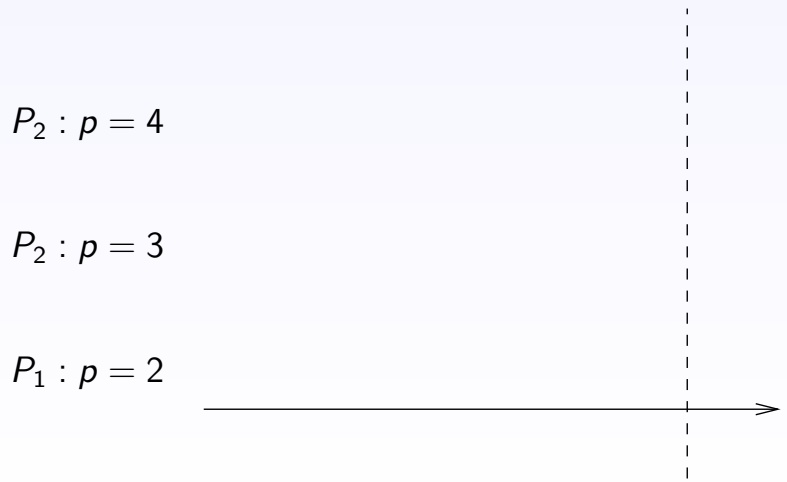
Scheduling the Last Job First

Principle

Scheduling the Last Job First

Principle

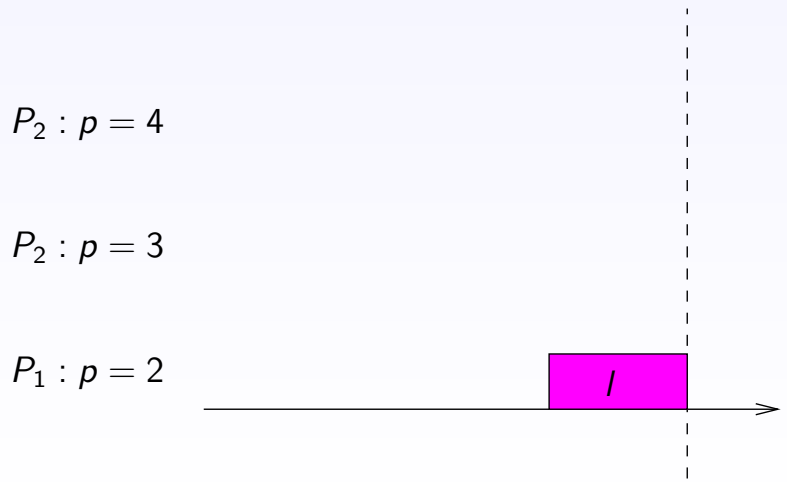
- Define a virtual completion time



Scheduling the Last Job First

Principle

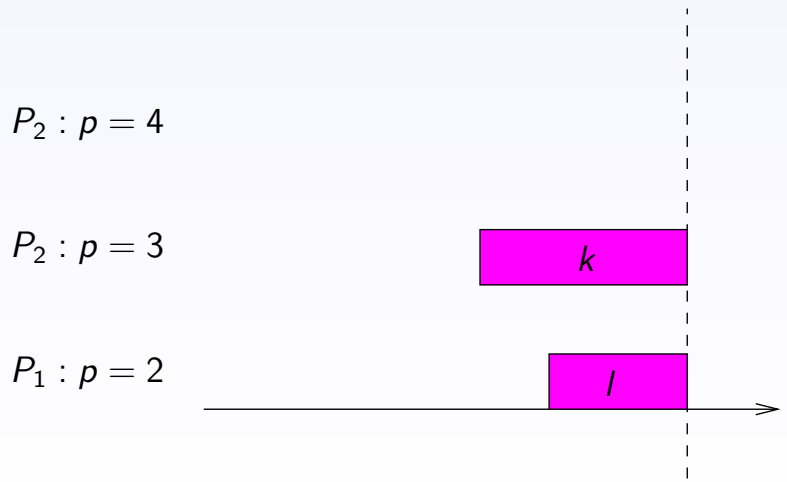
- Define a virtual completion time
- Allocate tasks as late as possible, taking into account the tasks already scheduled



Scheduling the Last Job First

Principle

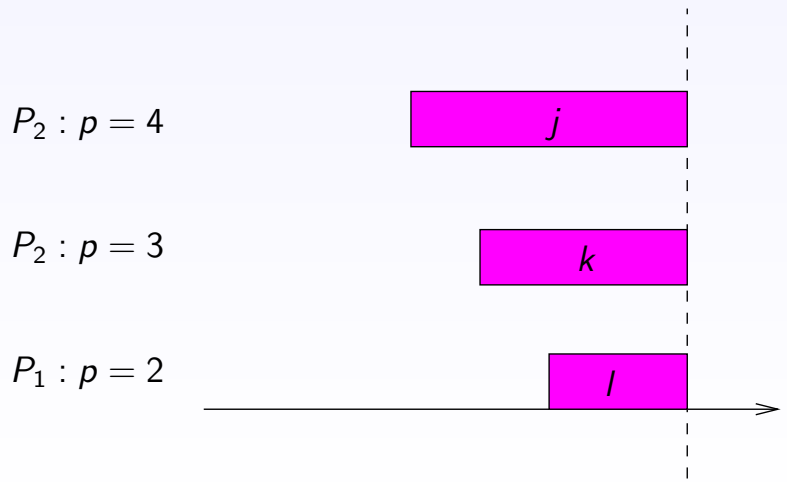
- Define a virtual completion time
- Allocate tasks as late as possible, taking into account the tasks already scheduled



Scheduling the Last Job First

Principle

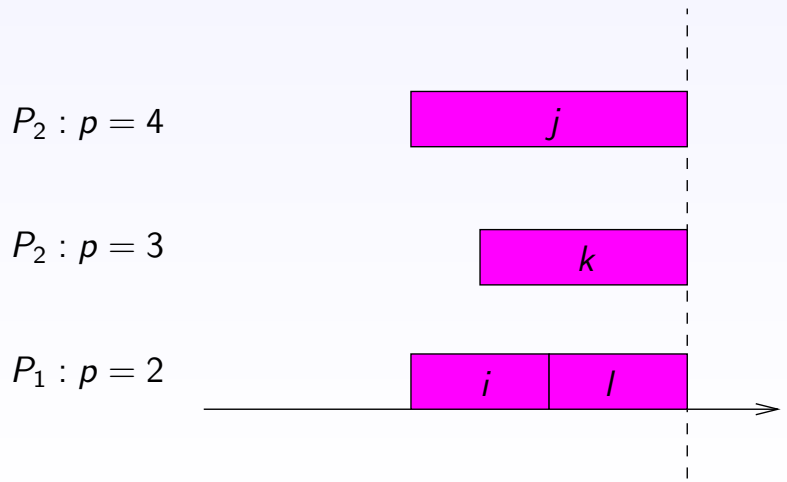
- Define a virtual completion time
- Allocate tasks as late as possible, taking into account the tasks already scheduled



Scheduling the Last Job First

Principle

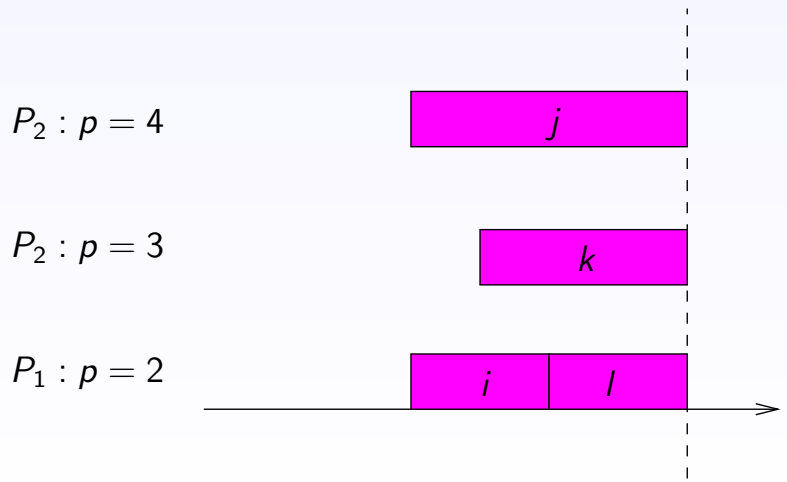
- Define a virtual completion time
- Allocate tasks as late as possible, taking into account the tasks already scheduled



Scheduling the Last Job First

Principle

- Define a virtual completion time
- Allocate tasks as late as possible, taking into account the tasks already scheduled
- Memorize the task allocation to processors



Scheduling the Last Job First

Principle

- Define a virtual completion time
- Allocate tasks as late as possible, taking into account the tasks already scheduled
- Memorize the task allocation to processors
- Distribute tasks as soon as possible