The impact of heterogeneity on master-slave on-line scheduling

Jean-François PINEAU, Yves ROBERT and Frédéric VIVIEN

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, France

Jean-Francois.Pineau@ens-lyon.fr

http://graal.ens-lyon.fr/~jfpineau

March 16, 2006

Outline

Outline

Background on Scheduling

The processors

- Parallel
  - Identical
  - Uniform

Background on Scheduling

The processors

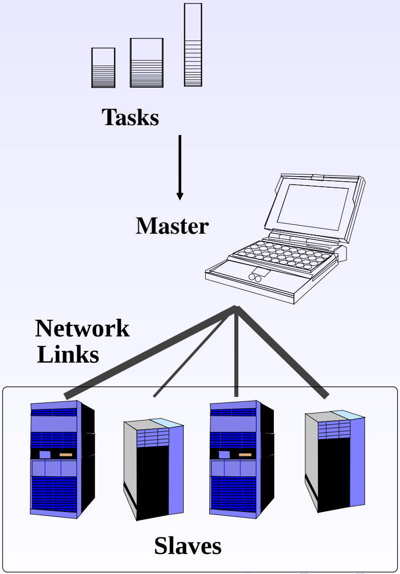
- Parallel
  - Identical
  - Uniform

The tasks

described by:

- their amount of computation
- their amount of communication
- their release date

Background on Scheduling



The master

- Receive the tasks
- Send them to the processors

Background on Scheduling

Goal

Scheduling tasks onto processors

- according to the constraints,
  - of the processors
  - of the tasks
- and optimizing some objective function

Background on Scheduling

Notations

- n tasks, m processors
- $p_{i,j}$: processing time of task i on processor j
- $c_{i,j}$: sending time of task i from master to processor j
- r_i: release date
- C_i: date of end of execution
- Main objective functions:
  - makespan: $\max C_i$
  - maximum flow time: $\max (C_i - r_i)$
  - average flow time: $\sum (C_i - r_i)$

Background on Scheduling

Definition

An algorithm $\mathcal{X}$ has a lower bound on its competitive ratio of ρ for the minimization of one objective function (for example *makespan*) if for one set of tasks:

$$\left(\max C_i\right)_{\mathcal{X}} \geq \rho\left(\max C_i\right)_{Opt}$$

Background on Scheduling

Let's specify the problem

- Identical independent tasks,

Otherwise, problem NP-hard even for 2 processors.

Background on Scheduling

Let's specify the problem

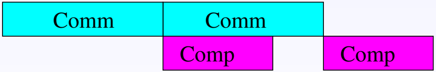
- Identical independent tasks,
- Fast communications.

Background on Scheduling

Let's specify the problem

- Identical independent tasks,
- Fast communications.

If $c_{j_0} = min\ c_j$ and $c_{j_0} > p_{j_0}$, then the optimal algorithm is trivial.



Outline

Outline

On homogeneous platforms

Round-Robin

is an optimal algorithm to minimize **all three**

- *makespan*,
- max flow time,
- sum flow time,

for an on-line problem with release dates.

Outline

On heterogeneous platforms

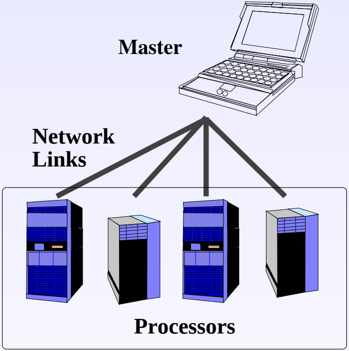
Optimal algorithm

does not exist, to minimize **one** objective function among

- *makespan*,
- max flow time,
- sum flow time,

This can be proved by an adversary method.

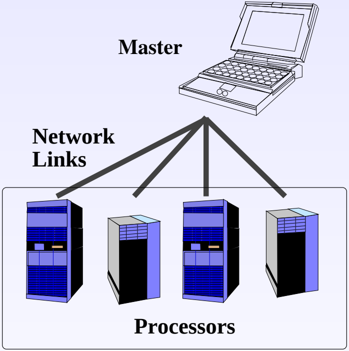
Example



Theorem

There is no scheduling algorithm for the problem $Q, MS \mid online, r_i, p_j, c_j = c \mid \max C_i$ with a competitive ratio less than $\frac{5}{4}$.

Example



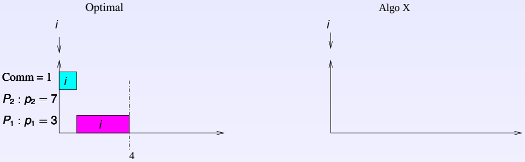
Theorem

There is no scheduling algorithm for the problem $Q, MS \mid online, r_i, p_j, c_j = c \mid \max \; C_i$ with a competitive ratio less than $\frac{5}{4}$.

Proof

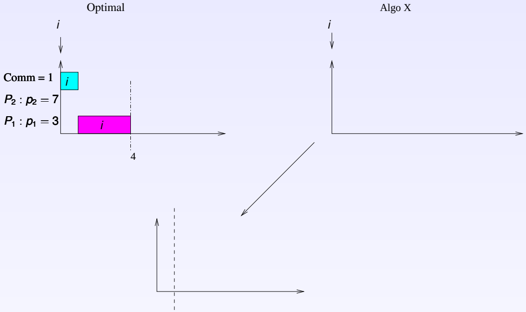
1. Suppose the existence of an on-line algorithm $\mathcal{X}$ with a competitive ratio $\rho = \frac{5}{4} - \epsilon$, with $\epsilon > 0$.
2. Let's study the behavior of $\mathcal{X}$ opposed to our adversary on a platform composed of two processors, where $p_1 = 3$, $p_2 = 7$, and $c = 1$.

Proof



Adversary sends a single task *i* at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

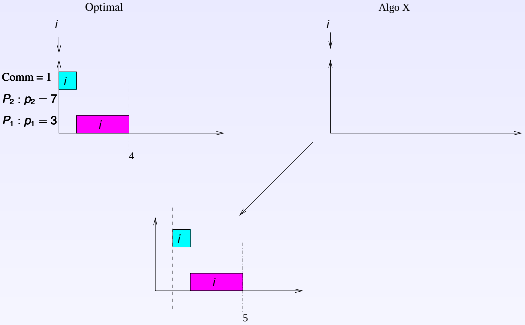
Proof



Adversary sends a single task *i* at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

- adversary does not send other tasks.

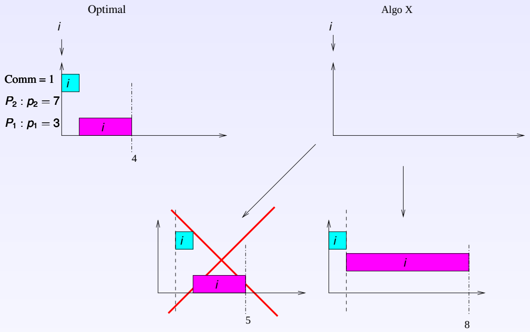
Proof



Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

- adversary does not send other tasks.
  competitive ratio : $\frac{t_1+c+p_1}{4} = \frac{5}{4} > \rho$

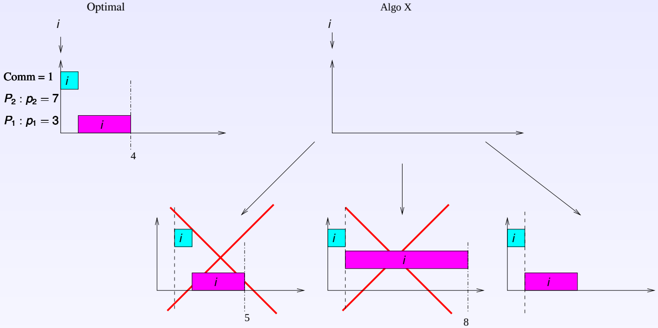
Proof



Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

- adversary does not send other tasks.
  competitive ratio : $\frac{c+p_2}{4} = 2 > \rho$

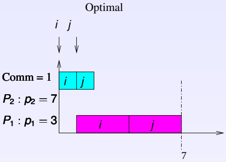
Proof

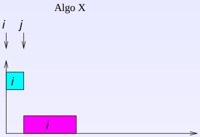


Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

- $\mathcal{X}$ has no choice but to schedule task i on P_1 to enforce its competitive ratio.

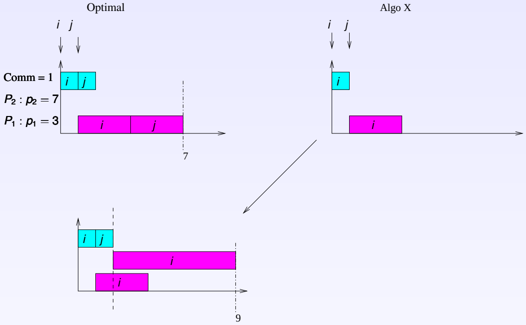
Proof





At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

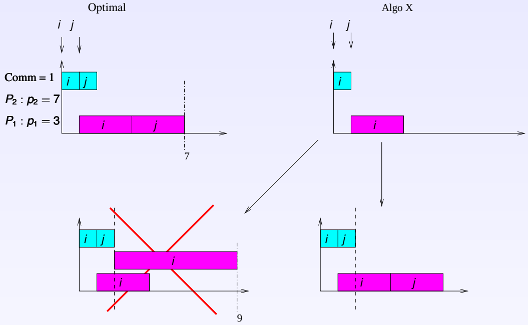
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- adversary sends no more task.
  competitive ratio : $\frac{2c+p_2}{7} = \frac{9}{7} > \frac{5}{4} > \rho$.

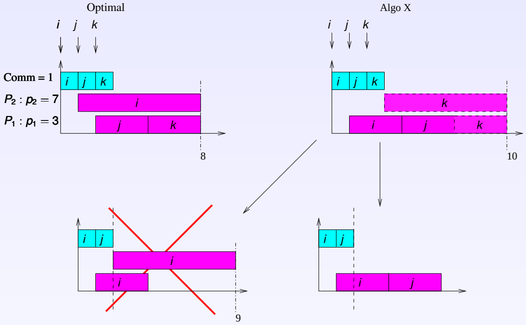
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- adversary sends a last task at time $t_2 = 2c$.

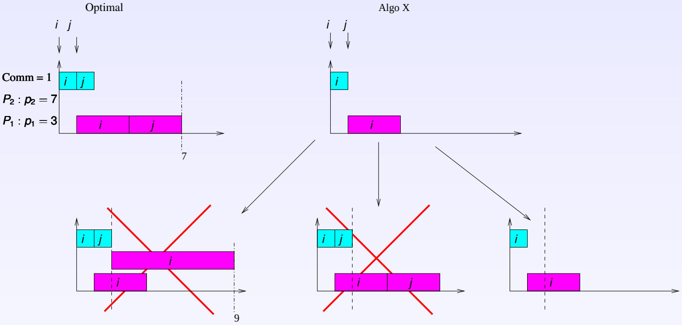
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- adversary sends a last task at time $t_2 = 2c$. competitive ratio: $\frac{10}{8} = \frac{5}{4} > \rho$.

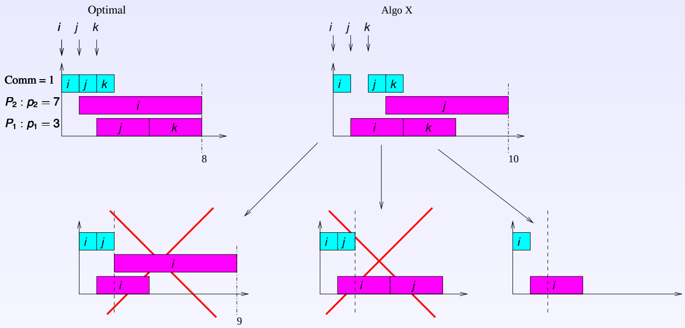
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- adversary sends a last task at time $t_2 = 2c$.

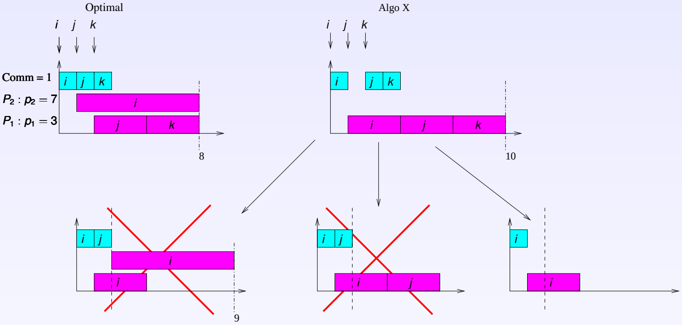
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- adversary sends a last task at time $t_2 = 2c$.

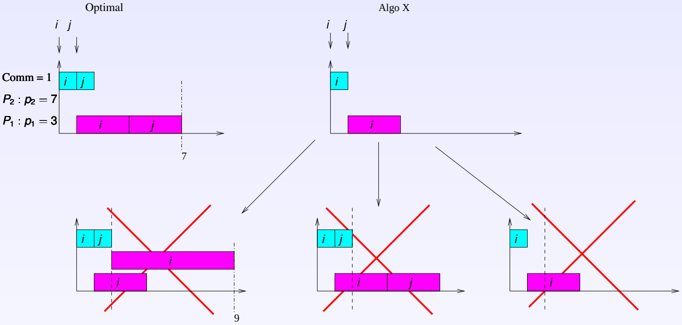
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- adversary sends a last task at time $t_2 = 2c$.

Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- adversary sends a last task at time $t_2 = 2c$.
  competitive ratio : $\frac{10}{8} = \frac{5}{4} > \rho$.

Outline

General approach

How does it work?

Let's see how we find the worst platform for an on-line algorithm.

Example

- Fully heterogeneous platform
- Minimization of max flow

General approach

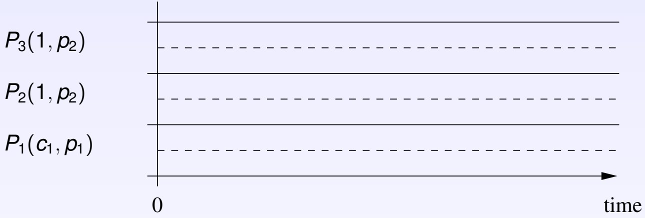
How does it work?

Let's see how we find the worst platform for an on-line algorithm.

Example

- Fully heterogeneous platform
- Minimization of max flow

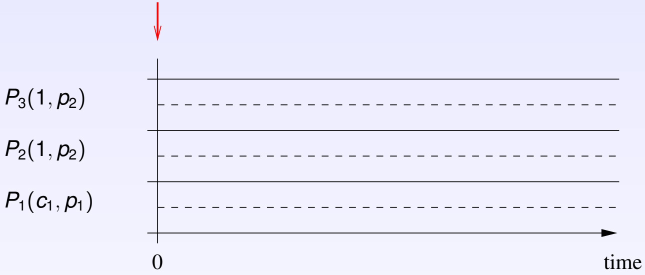
Generalisation



Idea:

- one fast processor with slow communication ($c_1 > 1$);
- two slow identical processors with fast communication;
- if only one task, send it on fast processor ($c_1 + p_1 < 1 + p_2$).
- if more than one task, do not send the first task on the fast processor

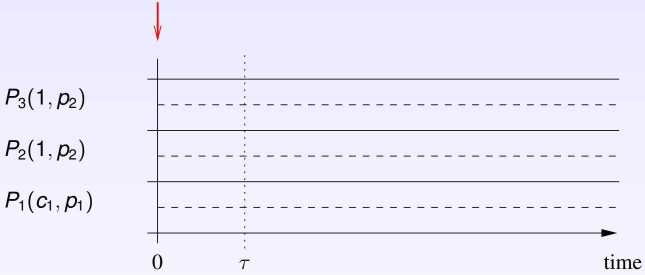
Generalisation



Idea:

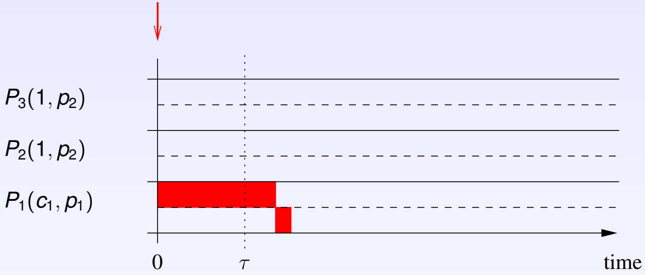
- one fast processor with slow communication ($c_1 > 1$);
- two slow identical processors with fast communication;
- if only one task, send it on fast processor ($c_1 + p_1 < 1 + p_2$).
- if more than one task, do not send the first task on the fast processor

Generalisation



At time $\tau \geq 1$ we look at what happened:

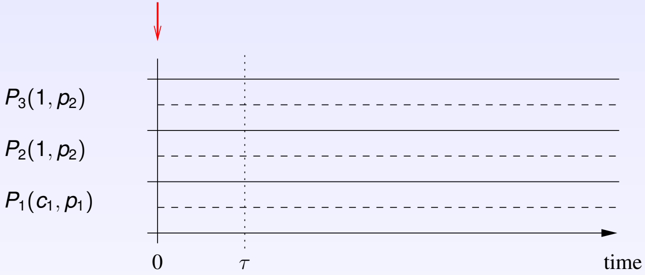
Generalisation



At time $\tau \geq 1$ we look at what happened:

1. Optimal : max flow $= c_1 + p_1$.

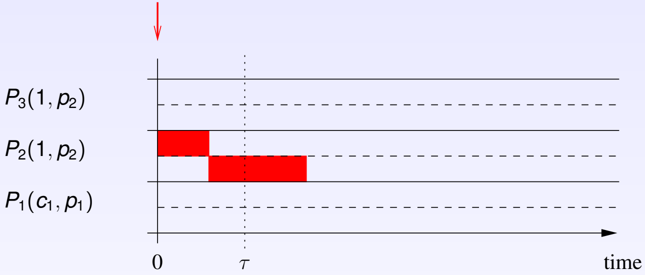
Generalisation



At time $\tau \geq 1$ we look at what happened:

1. Optimal : max flow $= c_1 + p_1$.
2. max flow $\geq \tau + c_1 + p_1$, ratio $\geq \frac{\tau + c_1 + p_1}{c_1 + p_1}$.

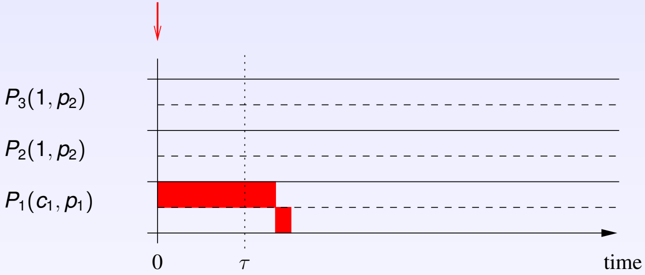
Generalisation



At time $\tau \geq 1$ we look at what happened:

1. Optimal : max flow $= c_1 + p_1$.
2. max flow $\geq \tau + c_1 + p_1$, ratio $\geq \frac{\tau + c_1 + p_1}{c_1 + p_1}$.
3. max flow $\geq 1 + p_2$, ratio $\geq \frac{1 + p_2}{c_1 + p_1}$.

Generalisation

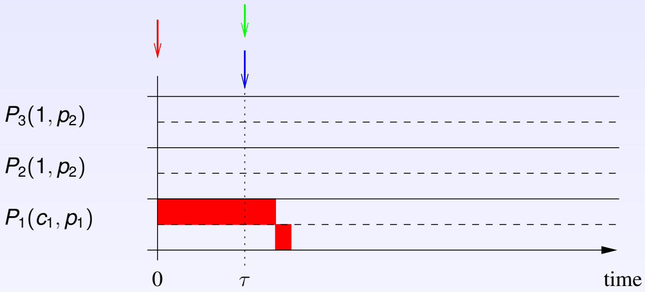


We choose τ, c_1, p_1 and p_2 to have:

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

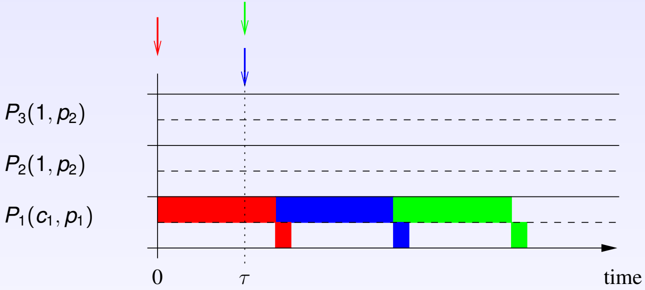
So algorithm has to execute the first task on P_1.

Generalisation



At time τ we send two new tasks. Let's see all possible schedulings.

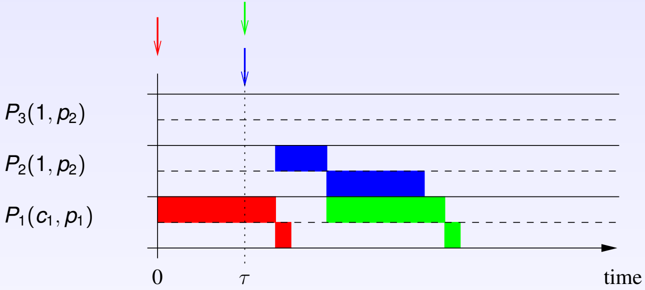
Generalisation



three tasks on P_1:

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

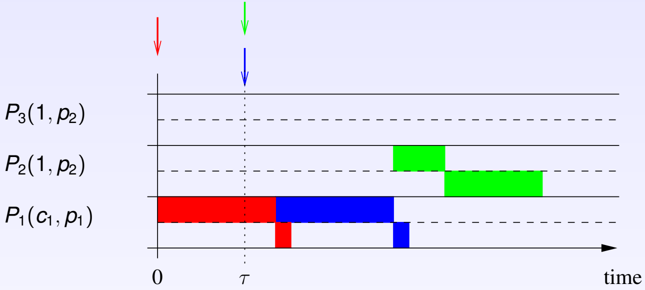
Generalisation



Last task on P_1.

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

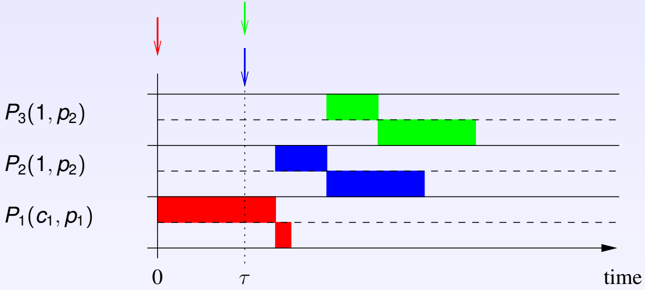
Generalisation



First task on P_1.

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

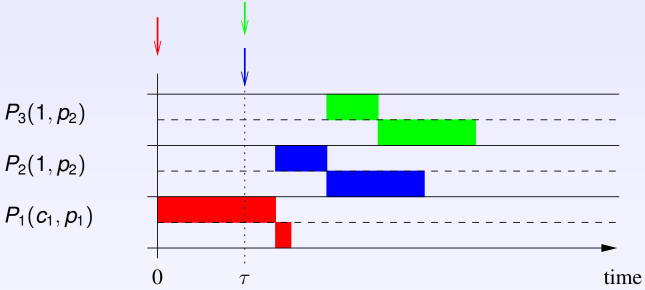
Generalisation



No more tasks on P_1.

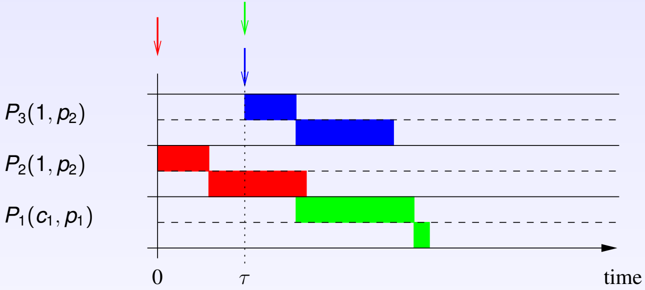
$$\max\{c_1+p_1, (\max\{c_1,\tau\}+c_2+p_2)-\tau, (\max\{c_1,\tau\}+c_2+c_2+p_2)-\tau\}$$

Generalisation



The case where two tasks are allocated on P_2 is even worse than the previous case.

Generalisation



Better solution : 1st task on P_2, 2nd on P_3 and 3rd on P_1.

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

How we found lower bound of competitiveness (1)

Lower bound of competitiveness:

$$\min \begin{cases} \frac{\tau+c_1+p_1}{c_1+p_1}, \\ \frac{1+p_2}{c_1+p_1}, \\ \dfrac{\min\begin{cases} \max\{c_1+p_1, \max\{\max\{c_1,\tau\}+c_1+p_1, c_1+2p_1\}-\tau, \max\{\max\{c_1,\tau\}+c_1+p_1+\max\{c_1,p_1\}, c_1+3p_1\}-\tau\} \\ \max\{c_1+p_1, (\max\{c_1,\tau\}+c_2+p_2)-\tau, \max\{\max\{c_1,\tau\}+c_2+c_1+p_1, c_1+2p_1\}-\tau\} \\ \max\{c_1+p_1, \max\{\max\{c_1,\tau\}+c_1+p_1, c_1+2p_1\}-\tau, (\max\{c_1,\tau\}+c_1+c_2+p_2)-\tau\} \\ \max\{c_1+p_1, (\max\{c_1,\tau\}+c_2+p_2)-\tau, (\max\{c_1,\tau\}+c_2+c_2+p_2)-\tau\} \end{cases}}{\max\{c_2+p_2, (\max\{c_2,\tau\}+c_2+c_1+p_1)-\tau\}} \end{cases}$$

Problem

Find τ, c_1, p_1 and p_2 ($c_2 = 1$) which maximize this lower bound, such as : $c_1 + p_1 < p_2$.

How we found lower bound of competitiveness (1)

Lower bound of competitiveness:

$$
\min \begin{cases}
\frac{\tau + c_1 + p_1}{c_1 + p_1}, \\[2ex]
\frac{1 + p_2}{c_1 + p_1}, \\[2ex]
\dfrac{\min \begin{cases}
\max\{c_1 + p_1, \max\{\max\{c_1, \tau\} + c_1 + p_1, c_1 + 2p_1\} - \tau, \max\{\max\{c_1, \tau\} + c_1 + p_1 + \max\{c_1, p_1\}, c_1 + 3p_1\} - \tau\} \\
\max\{c_1 + p_1, (\max\{c_1, \tau\} + c_2 + p_2) - \tau, \max\{\max\{c_1, \tau\} + c_2 + c_1 + p_1, c_1 + 2p_1\} - \tau\} \\
\max\{c_1 + p_1, \max\{\max\{c_1, \tau\} + c_1 + p_1, c_1 + 2p_1\} - \tau, (\max\{c_1, \tau\} + c_1 + c_2 + p_2) - \tau\} \\
\max\{c_1 + p_1, (\max\{c_1, \tau\} + c_2 + p_2) - \tau, (\max\{c_1, \tau\} + c_2 + c_2 + p_2) - \tau\}
\end{cases}}{\max\{c_2 + p_2, (\max\{c_2, \tau\} + c_2 + c_1 + p_1) - \tau\}}
\end{cases}
$$

Problem

Find τ, c_1, p_1 and p_2 ($c_2 = 1$) which maximize this lower bound, such as : $c_1 + p_1 < p_2$.

How we found lower bound of competitiveness (2)

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

$$\min \begin{cases} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{\min \begin{cases} 3c_1 - \tau, \\ c_1 + 1 - \tau + p_2, \\ 2c_1 - \tau + 1 + p_2 \\ c_1 + 2 + p_2 - \tau \end{cases}}{1+p_2} \end{cases} = \min \begin{cases} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{c_1 + 1 - \tau + p_2}{1+p_2}, \end{cases}$$

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

How we found lower bound of competitiveness (2)

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

$$\min \begin{cases} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{\min \begin{cases} 3c_1 - \tau, \\ c_1 + 1 - \tau + p_2, \\ 2c_1 - \tau + 1 + p_2 \\ c_1 + 2 + p_2 - \tau \end{cases}}{1+p_2} \end{cases} = \min \begin{cases} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{c_1 + 1 - \tau + p_2}{1+p_2}, \end{cases}$$

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

How we found lower bound of competitiveness (2)

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

$$\min \begin{cases} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \dfrac{\min \begin{cases} 3c_1 - \tau, \\ c_1 + 1 - \tau + p_2, \\ 2c_1 - \tau + 1 + p_2 \\ c_1 + 2 + p_2 - \tau \end{cases}}{1+p_2} \end{cases} = \min \begin{cases} \frac{\tau + c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{c_1 + 1 - \tau + p_2}{1+p_2}, \end{cases}$$

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

How we found lower bound of competitiveness (2)

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

$$\min \begin{cases} \frac{\tau+c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{\min\begin{cases} 3c_1 - \tau, \\ c_1 + 1 - \tau + p_2, \\ 2c_1 - \tau + 1 + p_2 \\ c_1 + 2 + p_2 - \tau \end{cases}}{1+p_2} \end{cases} = \min \begin{cases} \frac{\tau+c_1}{c_1}, \\ \frac{1+p_2}{c_1}, \\ \frac{c_1+1-\tau+p_2,}{1+p_2} \end{cases}$$

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

Outline

All results

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

Table: Lower bounds on the competitive ratio of on-line algorithms, depending on the platform type and on the objective function.

Outline

The platform

Hardware

- 5 computers (1 master, 4 slaves)
- 1 Fast-Ethernet switch

Software

- MPI communications
- Modification of slave parameters

Algorithms

- Algorithm **1** is a dynamic one
- Algorithm **4** and **7** take into account communication heterogeneity
- Algorithms **5** and **6** take into account computation heterogeneity
- Algorithms **2** and **3** take into account both communication and computation heterogeneity

Algorithm **6** is optimal to minimize *makespan* if it knows the total number of tasks.
Algorithm **7** is meant to be used on computation homogeneeous platform

Algorithms

- Algorithm **1** is a dynamic one
- Algorithm **4** and **7** take into account communication heterogeneity
- Algorithms **5** and **6** take into account computation heterogeneity
- Algorithms **2** and **3** take into account both communication and computation heterogeneity

Algorithm **6** is optimal to minimize *makespan* if it knows the total number of tasks.

Algorithm **7** is meant to be used on computation homogeneeous platform

Algorithms

- Algorithm **1** is a dynamic one
- Algorithm **4** and **7** take into account communication heterogeneity
- Algorithms **5** and **6** take into account computation heterogeneity
- Algorithms **2** and **3** take into account both communication and computation heterogeneity

Algorithm **6** is optimal to minimize *makespan* if it knows the total number of tasks.
Algorithm **7** is meant to be used on computation homogeneeous platform

Results

General case:

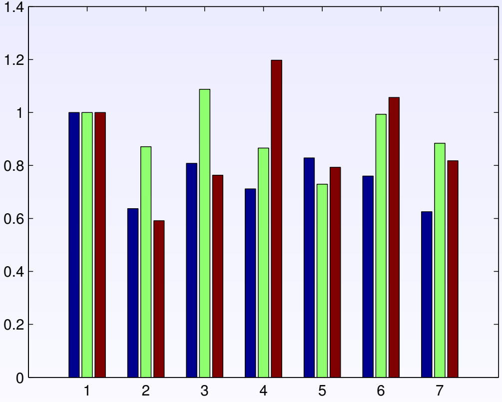


Figure: Normalized objective functions

Results

Homogeneous processors:

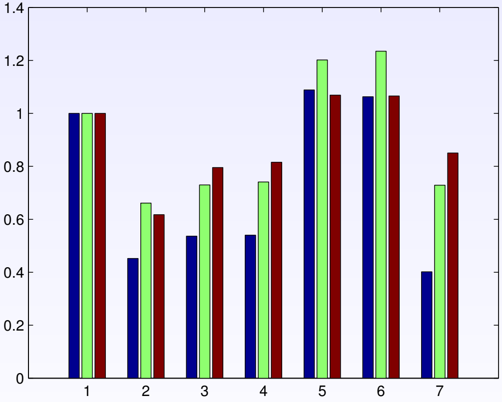


Figure: Normalized objective functions

Results

Summary

- The heuristic meant to be used on a communication heterogeneous platform is better than the other most part of the time (95%), and close to the best found algorithm (2%) elsewhere
- *SLJF* is outperformed by some classical algorithms

Results

Summary

- The heuristic meant to be used on a communication heterogeneous platform is better than the other most part of the time (95%), and close to the best found algorithm (2%) elsewhere
- *SLJF* is outperformed by some classical algorithms

Point out the importance to take into account the relative speed of communication links when searching a close-to-optimal solution to our scheduling problem.

Outline

Contributions and perspectives

Contributions

- Comprehensive set of lower bounds for the competitive ratio of any deterministic scheduling algorithm, for each source of heterogeneity and for each target objective function,
- Experiments on real small-size master-slave platform.

Perspectives

- See which bounds can be met, if any, and design the corresponding approximation algorithms,
- Theoretical study of off-line scheduling problems,
- Detailed comparison of all previous heuristics on significantly larger platforms,
- Widen the scope of the MPI experiments.

Contributions and perspectives

Contributions

- Comprehensive set of lower bounds for the competitive ratio of any deterministic scheduling algorithm, for each source of heterogeneity and for each target objective function,
- Experiments on real small-size master-slave platform.

Perspectives

- See which bounds can be met, if any, and design the corresponding approximation algorithms,
- Theoretical study of off-line scheduling problems,
- Detailed comparison of all previous heuristics on significantly larger platforms,
- Widen the scope of the MPI experiments.