# Analysis of Work Stealing with latency

## Work Stealing with latency

Nicolas Gast[1] Mohammed Khatiri[12] Denis Trystram[1]
Frédéric Wagner[1]

[1]Univ. Grenoble Alpes, France

[2]University Mohammed First, Morocco

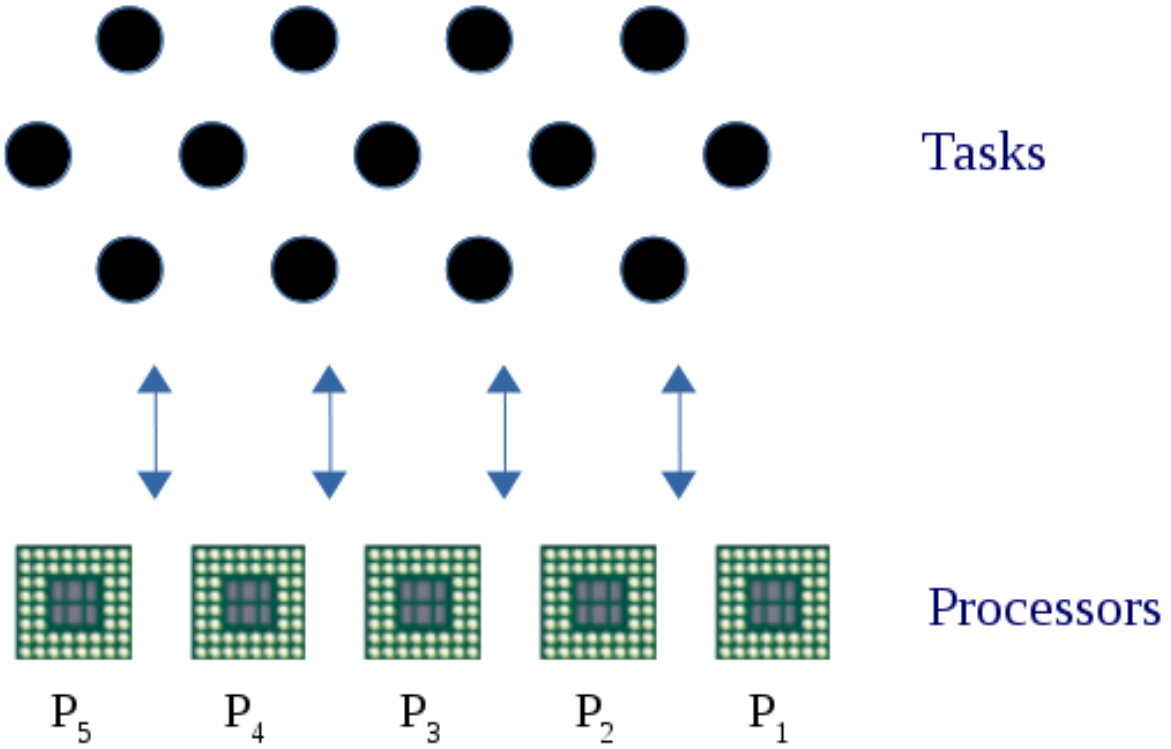
16 juin 2018

**ECCO XXXI - CO 2018**

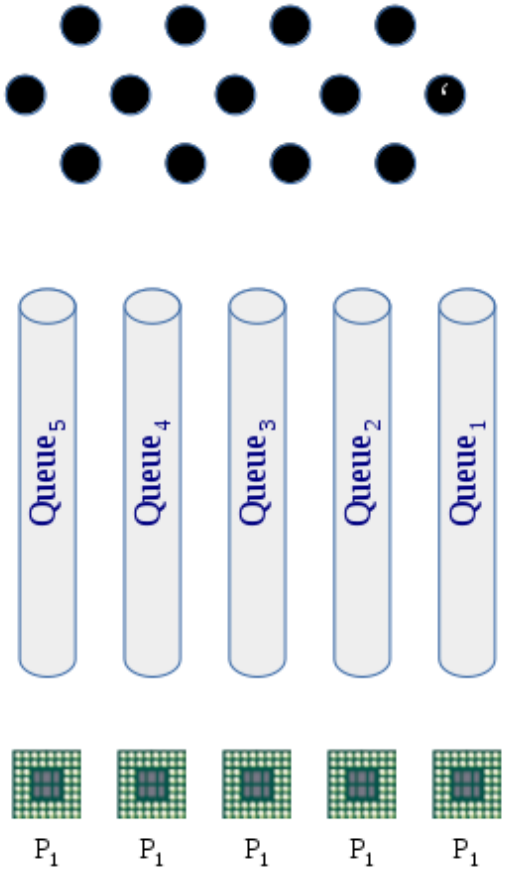
# Outline

# Outline

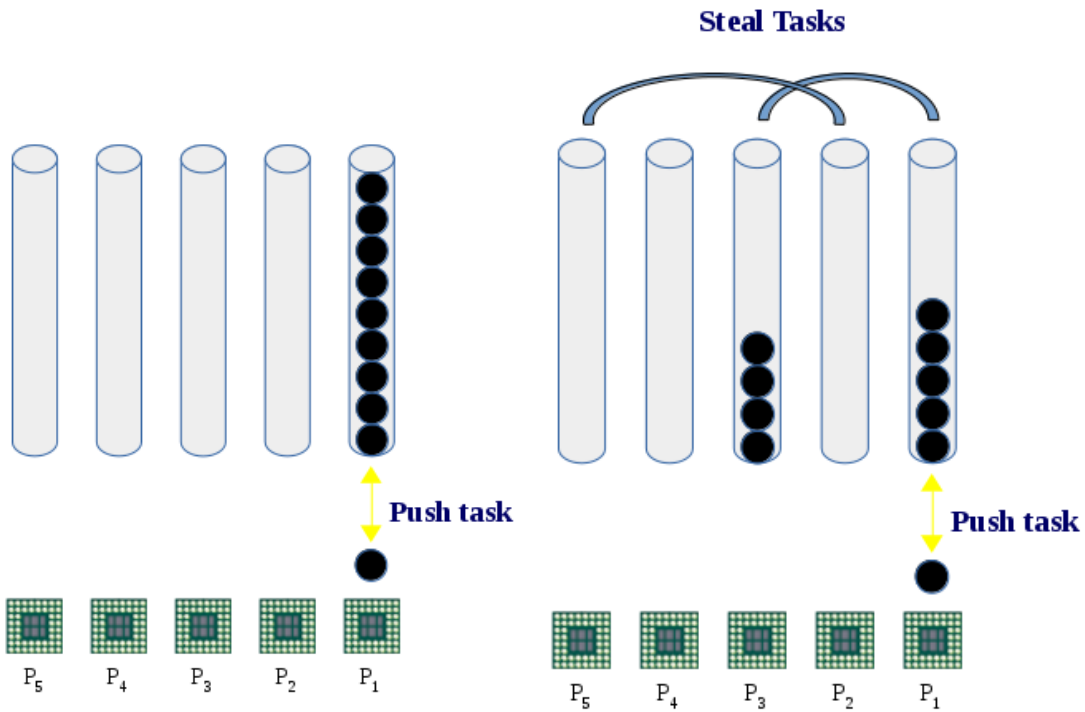
# Preliminary



**Decentralized List**



## Decentralized List
## Work Stealing Algorithm



# Objectives and Content

- Study the Work Stealing algorithm in distributed-memory platforms where communications matter.
- Create a new realistic model for distributed-memory cluster of $p$ identical processors including communication latency.
- Provide an upper bound of the expected Makespan.
- Provide simulation results to assess this bound.

# Context

## Scheduling Problem

The scheduling problem is to answer the two following questions for each of the n independent tasks which represents an application.

- Where? determine the computing resources for executing each task.
- When? determined when each task will start.

- **WHERE is crucial** since the load between resources must be balanced.
- There are several possibilities: centralized decision, static or dynamic algorithm to determine such an assignment.

# Setting of the problem

- Workload $\mathcal{W}$ composed of $n$ unit independent tasks.
- $p$ identical processors in a distributed-memory cluster.
- Each processor has its private memory
- The processors are linked by an interconnection network

**Makespan**

The optimized objective is the completion time of the $n$ tasks which called the makespan and denoted by $C_{max}$.

- Communication cost between processors is most often ignored.

# Classical Work Stealing algorithm

- **Decentralized list scheduling algorithm.**
- Each processor has its own queue of tasks
- An active processor (defined as a non-empty tasks queue) executes tasks from its queue.
- An idle processor with empty queue (Stealer) randomly chooses another processor (Victim) to steal some work:
  - If the victim's queue is not empty, the stealer steals half of the tasks and resumes execution at **the next time slot**
  - Otherwise, the stealer tries again to steal at **the next time slot**
- If several stealers target the same victim, a random succeeds and the others fail.

# Classical Work Stealing algorithm

- **Decentralized list scheduling algorithm.**
- Each processor has its own queue of tasks
- An active processor (defined as a non-empty tasks queue) executes tasks from its queue.
- An idle processor with empty queue (Stealer) randomly chooses another processor (Victim) to steal some work:
  - If the victim's queue is not empty, the stealer steals half of the tasks and resumes execution at **the next time slot**
  - Otherwise, the stealer tries again to steal at **the next time slot**
- If several stealers target the same victim, a random succeeds and the others fail.

# Related Work (in brief)

[Blumofe and Leiserson 1999, Arora et al. 2001 ]

- Bound the expected Makespan of a series-parallel precedence graph with $\mathcal{W}$ unit tasks on $p$ processors.
- $E(C_{\max}) \leq \frac{\mathcal{W}}{p} + \mathcal{O}(D)$ Where $D$ is the length of the critical path of the graph.

[Tchiboukdjian *et al.* 2013]

- $\mathcal{W}$ unit independent tasks on $p$ processors (communication cost is ignored).
- $E(C_{\max}) \leq \frac{\mathcal{W}}{p} + c.(\log_2 \mathcal{W}) + \Theta(1)$ where $c \simeq 3.24$

Need a precise analysis of work stealing when communication is not ignored

# Outline

# Problem

## Problem

- When the communication cost is not ignored, the steal work takes more than one unit of time.
- The stealer takes more time to receive the answer from the victim.
- The stealer must wait several time steps before receiving the answer.

**Question :**

What happens when the communication cost is explicit?

# Model with communications cost (latency)

## Objective

Analyze Work Stealing algorithm to execute a $\mathcal{W}$ workload of $n$ unit tasks on a distributed-memory cluster with $p$ identical processors and communications between them take $\lambda$ unit of times (latency).

# Work stealing algorithm with communications (latency)

- **Principe of Work stealing (Recall)**
  - Each processor has its own queue of tasks
  - An active processor with **not-empty** queue executes tasks from its queue
  - When a processor becomes **idle**, it chooses randomly another processor (Victim) to steal work
- **Latency**
  - A work request takes a time $\lambda$ **(latency)**
  - The answer takes again $\lambda$ units of time (fail answer or work answer)
- **Steal threshold:**
  - If the victim has less than $\lambda$ units of work, the steal request fails.
- **Single work transfer**
  - While a victim sends some work to a stealer, it replies by a fail request to all other steal requests.

# Analysis

We consider a discrete model with:

- $p$ processors indexed by $i$ where $i \in [1...p]$.
- All communications between the processors take $\lambda$ units of time.
- $\mathcal{W}_i(t) \in [1, 2, ...]$ the amount of work on processor $P_i$ at time $t$.
- $\mathcal{W}(t) = \sum_{i=1}^{p} \mathcal{W}_i(t)$ the total amount of Work on all the processors.
- $\mathcal{S}_i(t)$ the amount of Work in transit from $P_i$ at time $t$ to another stealer.
- At $t = 0$ the whole work is on $P_1$ and $\mathcal{W} = \mathcal{W}_1(0)$

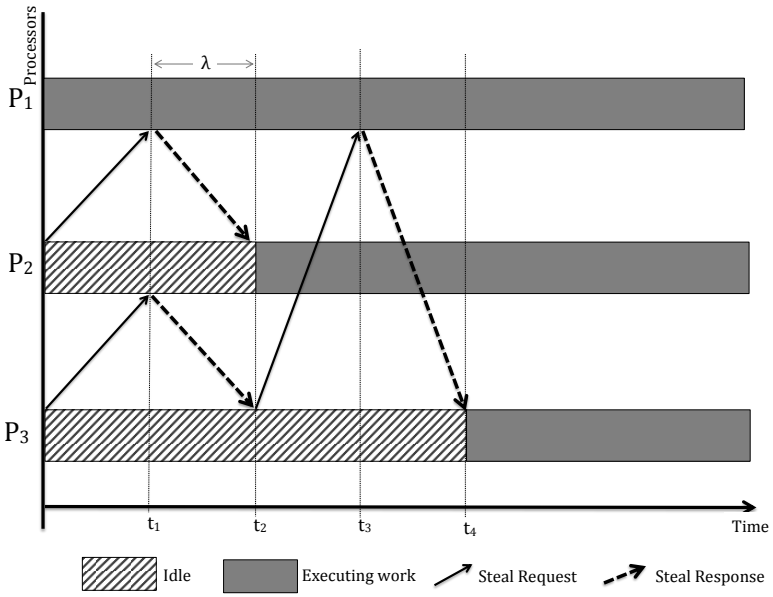
## Theorem

$$\mathbb{E}(C_{max}) \leq \frac{\mathcal{W}}{p} + c\lambda \log_2(\frac{\mathcal{W}}{2\lambda}) + 3\lambda$$

where $c \leq 16.12$

# Outline

# Principle of the analysis (1)




$$pC_{\max} \leq \mathcal{W} + 2\lambda \#StealRequests$$

$$C_{max} \leq \frac{\mathcal{W}}{p} + 2\lambda \frac{\#StealRequests}{p} \quad (1)$$

To bound $C_{max}$, we need to bound $\#StealRequests$

# Principle of the analysis (2)

The analysis is based on a **potential function** that represents *how well the jobs are balanced in the system*.

## Potential function

Our potential function **at time-step** $k\lambda$ is denoted by $\phi(k)$ and :

$$\phi(k) = \sum_{i=1}^{p} \phi_i(k), \text{ where } \phi_i(k) = \mathcal{W}_i(k\lambda)^2 + 2\mathcal{S}_i(k\lambda)^2 \quad \text{for } i \in \{1 \ldots, p\}.$$

**Properties**

- Always decreases.
- Its maximal at time 0, $\phi(0) = \mathcal{W}^2$
- Becomes 0 at the end of schedule.
- Decreases according to each processor event (executing work, sending or answering steal request)

**Analysis method**

We analyse how much each case contributes to the decrease of the potential between $\lambda k$ and $\lambda(k+1)$

# Principle of the analysis (3)

## Lemme 1

We denote by $\mathcal{F}_k$ all events up to the interval $((k-1)\lambda,\ \lambda k]$
For all steps $k$, the expected ratio between $\phi(k+1)$ and $\phi(k)$ knowing $\mathcal{F}_k$ is bounded by:

$$\mathbb{E}[\phi(k+1) \mid \mathcal{F}_k] \leq \left(1 - \frac{q(r_k)}{4}\right) \phi(k)$$

Where $q(r_k)$ is the probability for a processor to receive one or more steal requests in the interval $((k-1)\lambda, k\lambda]$ knowing that there are $r_k$ incoming steal requests.

$$q(r_k) = 1 - \left(\frac{p-2}{p-1}\right)^{r_k}$$

# Principle of the analysis (4)

- Using :
  - The potential function at time 0 $(\mathcal{W}^2)$ and at the end of schedule $(0)$
  - The result of lemme 1.
- We bound the expected number of steal request between 0 and the end of schedule.
- Then, we derive the bound of the expected makespan $C_{max}$ using the equation:

$$C_{max} \leq \frac{\mathcal{W}}{p} + 2\lambda \frac{\#StealRequests}{p}$$

# Outline

# Experimental Results

$$\mathbb{E}(C_{max}) \leq \frac{\mathcal{W}}{p} + c\lambda \log_2(\frac{\mathcal{W}}{2\lambda}) + 3\lambda$$

$\frac{\mathcal{W}}{p}$ does not depend on the configuration and the algorithm

$$\text{Overhead ratio} = \frac{c\lambda \log_2(\mathcal{W}/2\lambda) + 3\lambda}{SimulationTime - \mathcal{W}/p}$$

**Simulator :**

- A python discrete event simulator.
- The simulator follows the model described to schedule an amount $\mathcal{W}$ of work on a distributed platform with $p$ identical processors
- The communication cost is equal to the latency $\lambda$

**Configuration**

- $10^5 \leq \mathcal{W} \leq 10^8$ , $32 \leq p \leq 256$ and $2 \leq \lambda \leq 500$
- Each experiment has been reproduced 1000 times

# Experimental Results

$$\mathbb{E}(C_{max}) \leq \frac{\mathcal{W}}{p} + c\lambda \log_2(\frac{\mathcal{W}}{2\lambda}) + 3\lambda$$

$\frac{\mathcal{W}}{p}$ does not depend on the configuration and the algorithm

$$\text{Overhead ratio} = \frac{c\lambda \log_2(\mathcal{W}/2\lambda) + 3\lambda}{SimulationTime - \mathcal{W}/p}$$

**Simulator :**

- A python discrete event simulator.
- The simulator follows the model described to schedule an amount $\mathcal{W}$ of work on a distributed platform with $p$ identical processors
- The communication cost is equal to the latency $\lambda$

**Configuration**

- $10^5 \leq \mathcal{W} \leq 10^8$ , $32 \leq p \leq 256$ and $2 \leq \lambda \leq 500$
- Each experiment has been reproduced 1000 times

# Experimental Results

$$\mathbb{E}(C_{max}) \leq \frac{\mathcal{W}}{p} + c\lambda \log_2(\frac{\mathcal{W}}{2\lambda}) + 3\lambda$$

$\frac{\mathcal{W}}{p}$ does not depend on the configuration and the algorithm

$$\text{Overhead ratio} = \frac{c\lambda \log_2(\mathcal{W}/2\lambda) + 3\lambda}{SimulationTime - \mathcal{W}/p}$$

**Simulator :**

- A python discrete event simulator.
- The simulator follows the model described to schedule an amount $\mathcal{W}$ of work on a distributed platform with $p$ identical processors
- The communication cost is equal to the latency $\lambda$

**Configuration**

- $10^5 \leq \mathcal{W} \leq 10^8$ , $32 \leq p \leq 256$ and $2 \leq \lambda \leq 500$
- Each experiment has been reproduced 1000 times

# Experimental Results

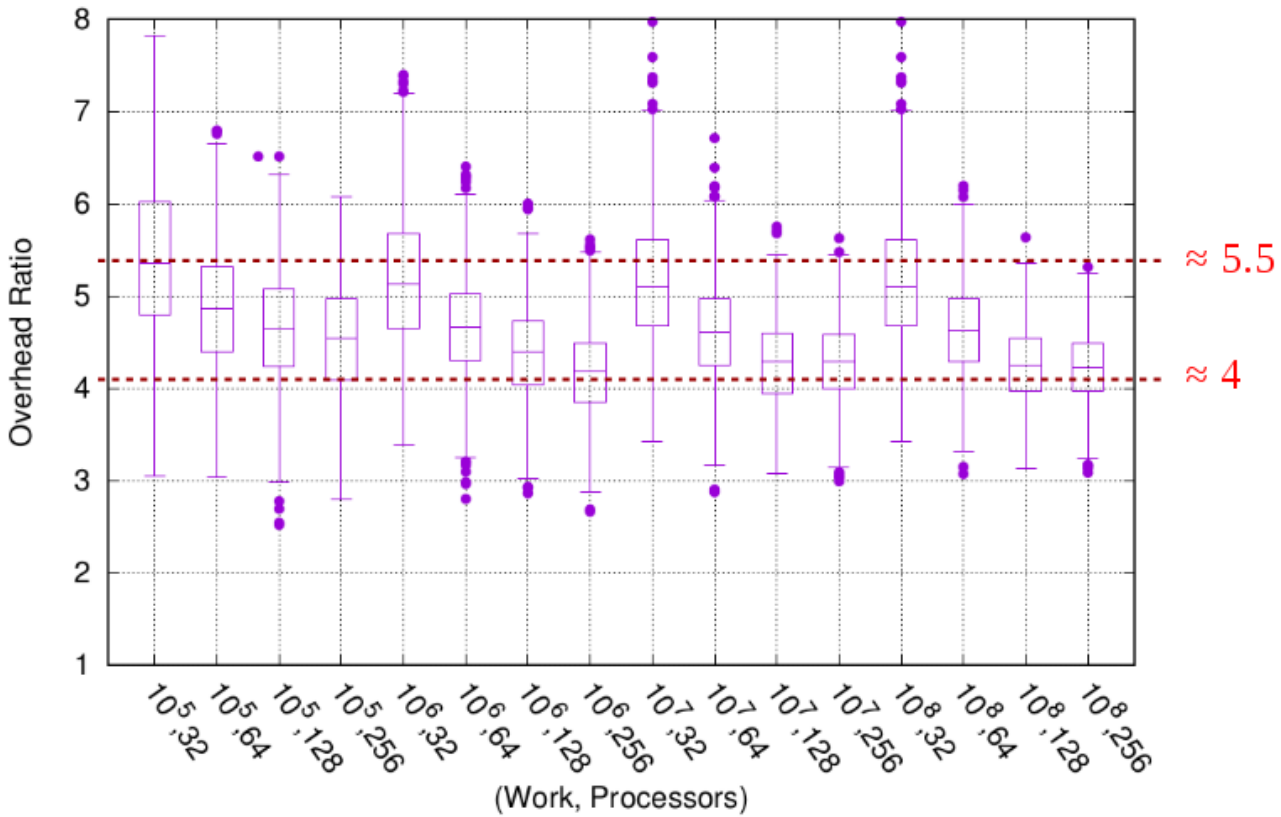
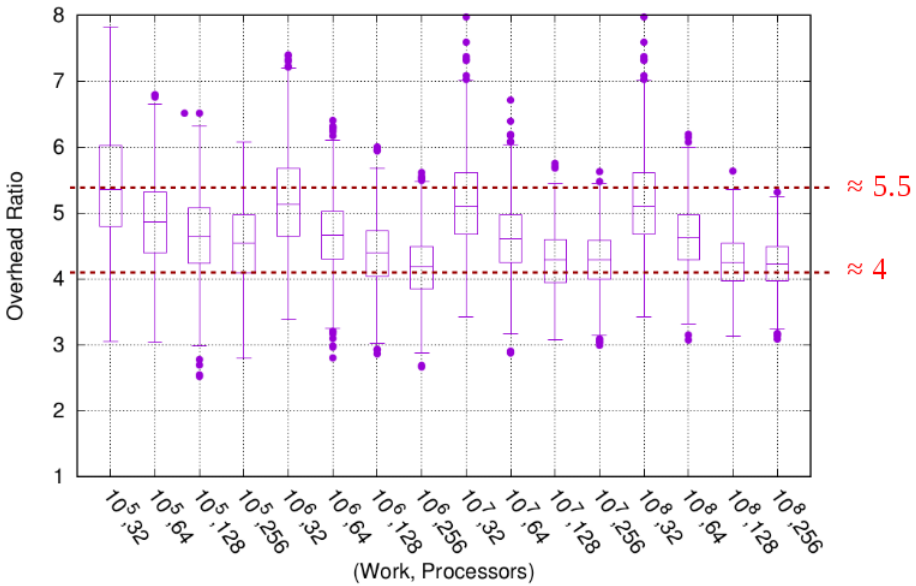



Figure: Overhead ratio as a function of $(\mathcal{W}, p)$ $(\lambda = 262)$

**Similar observations have been observed with all values of latency used**

# Experimental Results

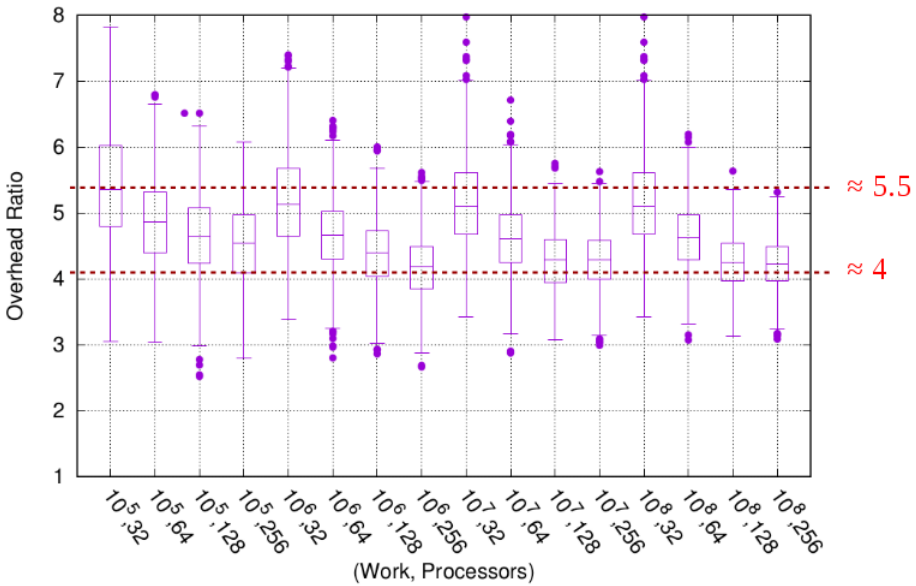



## Observations

- Our bound is about 4 to 5.5 times greater to the one computed by simulation
- Because we take the worst case in different cases of the analysis
- The ratio decrease with the number of processors **but seems fairly independent to** $\mathcal{W}$

- Because in the theoretical bound of the second term, we major the number of processors, which makes the born independent to the number of processors, but it still reacted on the simulation results.

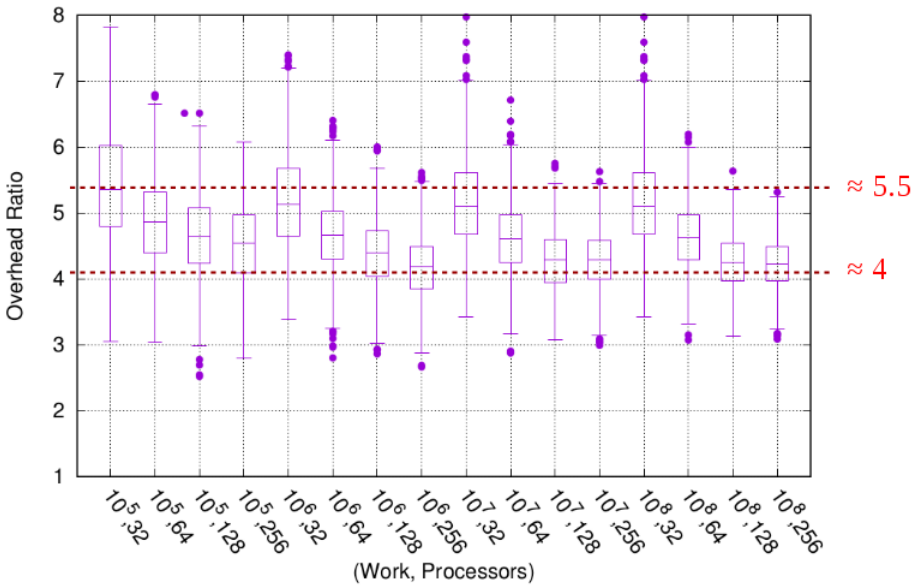
# Experimental Results




## Observations

- Our bound is about 4 to 5.5 times greater to the one computed by simulation
- Because we take the worst case in different cases of the analysis
- The ratio decrease with the number of processors **but seems fairly independent to** $\mathcal{W}$
- Because in the theoretical bound of the second term, we major the number of processors, which makes the born independent to the number of processors, but it still reacted on the simulation results.

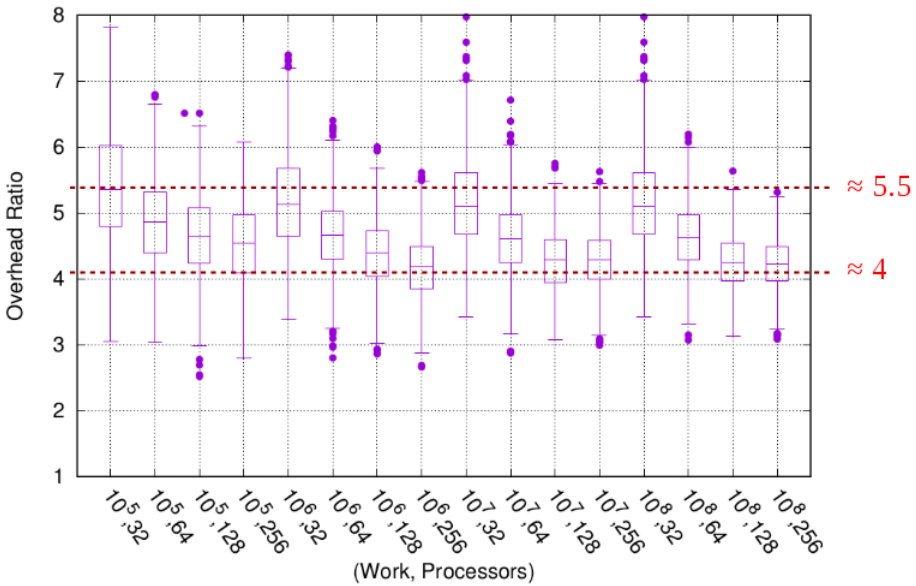
# Experimental Results




## Observations

- Our bound is about 4 to 5.5 times greater to the one computed by simulation
- Because we take the worst case in different cases of the analysis
- The ratio decrease with the number of processors **but seems fairly independent to** $\mathcal{W}$
- Because in the theoretical bound of the second term, we major the number of processors, which makes the born independent to the number of processors, but it still reacted on the simulation results.

# Experimental Results




**Observations**

- Our bound is about 4 to 5.5 times greater to the one computed by simulation
- Because we take the worst case in different cases of the analysis
- The ratio decrease with the number of processors **but seems fairly independent to** $\mathcal{W}$

- Because in the theoretical bound of the second term, we major the number of processors, which makes the born independent to the number of processors, but it still reacted on the simulation results.

# Outline

# Conclusion and Perspective

**Conclusion**

- Created a new realistic scheduling model for distributed-memory cluster of $p$ identical processors including latency $\lambda$
- Analyzed this model and provide an upper bound of the Makespan
- Provided simulation results to assess this bound.

**Perspective**

This work forms the basis of incoming studies on more complex hierarchical topologies :

- Communication cost inside clusters is small
- Communication cost outside clusters is important

Our paper is available on Arxiv : https://arxiv.org/abs/1805.00857

E-mail : khatiri.med@gmail.com

Tank you,