Scheduling divisible loads with return messages on heterogeneous master-worker platforms

Olivier Beaumont, Loris Marchal, Yves Robert

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

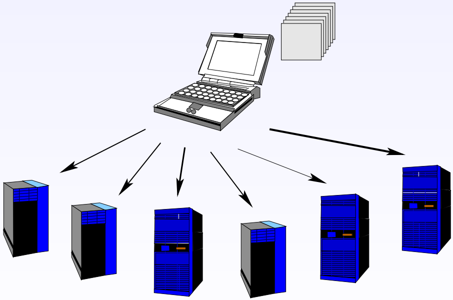
Graal Working Group
June 2005

Outline

Outline

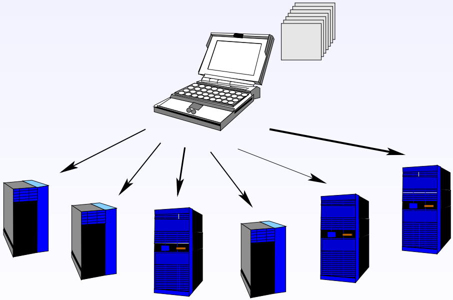
Introduction

- One master, holding a large number of identical tasks
- Some workers



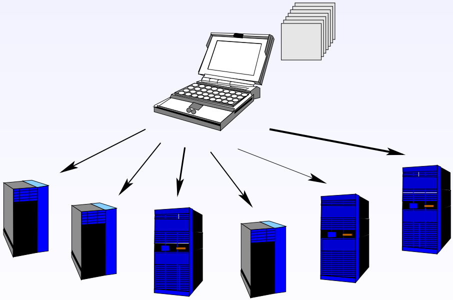
Introduction

- One master, holding a large number of identical tasks
- Some workers
- Heterogeneity in computing speed and bandwidth



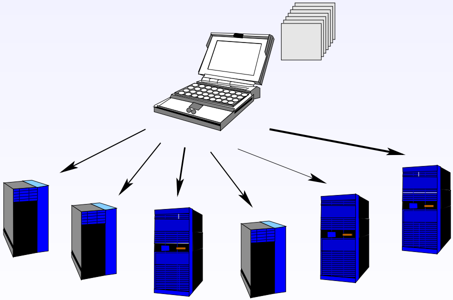
Introduction

- One master, holding a large number of identical tasks
- Some workers
- Heterogeneity in computing speed and bandwidth
- Distribute work to workers



Introduction

- One master, holding a large number of identical tasks
- Some workers
- Heterogeneity in computing speed and bandwidth
- Distribute work to workers
- **Gather the results**



Assumption: divisible load

- Important relaxation of the problem

Assumption: divisible load

- Important relaxation of the problem
- Master holding N tasks

Assumption: divisible load

- Important relaxation of the problem
- Master holding N tasks
- Worker P_i will get a fraction $\alpha_i \times N$ of these tasks

Assumption: divisible load

- Important relaxation of the problem
- Master holding N tasks
- Worker P_i will get a fraction $\alpha_i \times N$ of these tasks
- α_i is **rational**, tasks are divisible

Assumption: divisible load

- Important relaxation of the problem
- Master holding N tasks
- Worker P_i will get a fraction $\alpha_i \times N$ of these tasks
- α_i is **rational**, tasks are divisible
- $\Rightarrow$ possible to derive analytical solutions (tractability)

Assumption: divisible load

- Important relaxation of the problem
- Master holding N tasks
- Worker P_i will get a fraction $\alpha_i \times N$ of these tasks
- α_i is **rational**, tasks are divisible
- $\Rightarrow$ possible to derive analytical solutions (tractability)
- In practice, reasonable assumption with a large number of tasks

Background on Divisible Load Scheduling

Without return messages

- Linear cost model: X units of work:
  - sent to P_i in $X \times c_i$ time units
  - computed by P_i in $X \times w_i$ time units

Background on Divisible Load Scheduling

Without return messages

- Linear cost model: X units of work:
  - sent to P_i in $X \times c_i$ time units
  - computed by P_i in $X \times w_i$ time units

Results:

- Bus network $\Rightarrow$ all processors work and finish at the same time order does not matter closed-formula for the makespan (Bataineh, Hsiung & Robertazzi, 1994)
- Result extended for homogeneous tree: a subtree reduces to one single worker
- Heterogeneous star network:
  all processors work and finish at the same time
  order matters:
  - largest bandwidth first (whatever the computing power)

Background on Divisible Load Scheduling

- With an affine cost model: X units of work:
  - sent to P_i in $C_i + X \times c_i$ time units
  - computed by P_i in $W_i + X \times w_i$ time units

  $\Rightarrow$ not all processors participate to the computation
  selecting the resources is hard:
  - computing the optimal schedule on a star with affine cost model is NP-hard (Casanova, Drodowski, Legrand & Yang, 2005)

Background on Divisible Load Scheduling

- With an affine cost model: X units of work:
  - sent to P_i in $C_i + X \times c_i$ time units
  - computed by P_i in $W_i + X \times w_i$ time units

  $\Rightarrow$ not all processors participate to the computation
  selecting the resources is hard:
  - computing the optimal schedule on a star with affine cost model is NP-hard (Casanova, Drodowski, Legrand & Yang, 2005)
- Affine cost model + multi-round algorithms:
  - UMR, quadratic progression of the chunk size (Casanova & Yang, 2003)
  - asymptotically optimal algorithm based on steady-state (Beaumont, Legrand & Robert, 2003)

Adding return messages

What we need to decide:

- Ordering of the initial data to the workers
- Ordering of the return messages
- Quantity of work for each worker

Adding return messages

What we need to decide:

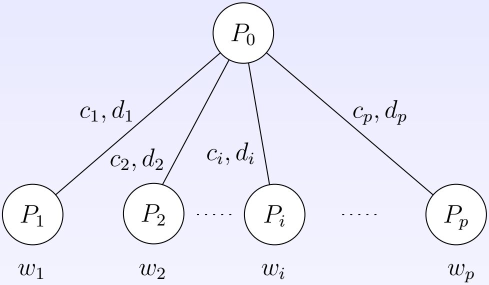
- Ordering of the initial data to the workers
- Ordering of the return messages
- Quantity of work for each worker

Related work:

- Barlas: fixed communication time or bus network + affine computation cost → optimal ordering and closed-from formulas
- Drodowski and Wolniewicz: consider LIFO and FIFO distributions
- Rosenberg et al.: complex (affine) communication model + possibility to slow down computing speed → all FIFO ordering are equivalent and better than any other ordering

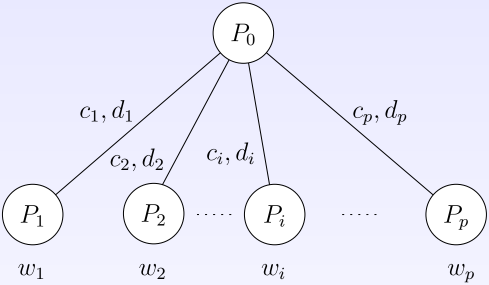
Outline

Model



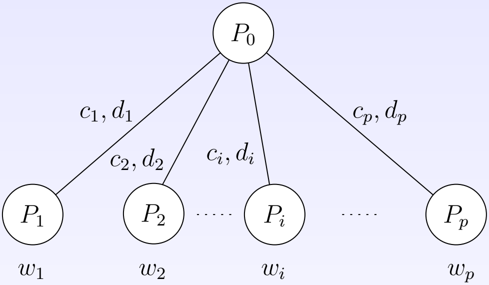
- One master P_0

Model



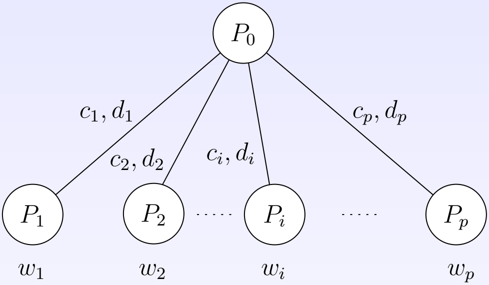
- One master P_0
- p workers $P_1, \ldots, P_p$

Model



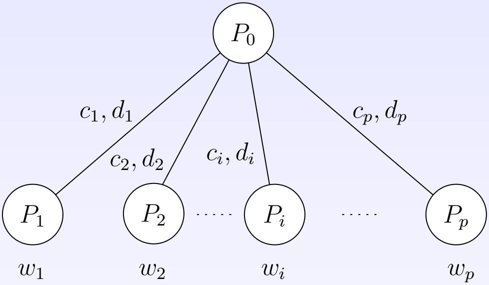
- One master P_0
- p workers $P_1, \ldots, P_p$
- Linear model, X unit of work are:

Model



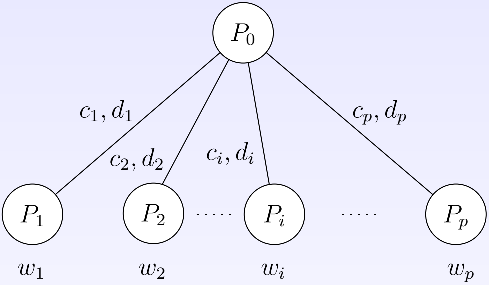
- One master P_0
- p workers $P_1, \ldots, P_p$
- Linear model, X unit of work are:
  - sent to P_i in $X \times c_i$ time units

Model



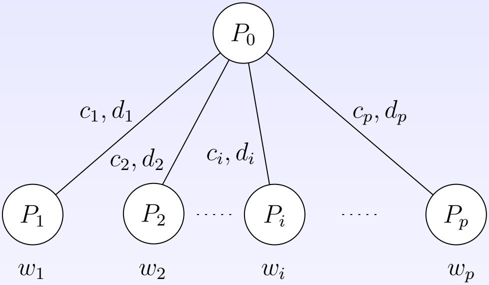
- One master P_0
- p workers $P_1, \ldots, P_p$
- Linear model, X unit of work are:
  - sent to P_i in $X \times c_i$ time units
  - computed by P_i in $X \times w_i$ time units

Model



- One master P_0
- p workers $P_1, \ldots, P_p$
- Linear model, X unit of work are:
  - sent to P_i in $X \times c_i$ time units
  - computed by P_i in $X \times w_i$ time units
  - their result is sent back from P_i to the master in $X \times d_i$ time units

Model



- One master P_0
- p workers $P_1, \ldots, P_p$
- Linear model, X unit of work are:
  - sent to P_i in $X \times c_i$ time units
  - computed by P_i in $X \times w_i$ time units
  - their result is sent back from P_i to the master in $X \times d_i$ time units
- We sometimes make the assumption: $d_i = z \times c_i$
  (e.g. z=0.8 ⇔ result file = 80% of original files)

Model

- Standard model in DLS for communications
  - the master can send data to at most one worker at time t
  - the master can receive data from at most one worker at time t
  - a worker can start computing only once the reception from the master has terminated

Model

- Standard model in DLS for communications
  - the master can send data to at most one worker at time t
  - the master can receive data from at most one worker at time t
  - a worker can start computing only once the reception from the master has terminated
- No idle time in the operation of each worker ?
  - reasonable without return messages
  - send results to the master just after computation ?
    but maybe the communication medium is not free
  - general scheme: allow idle time

Linear Program for a given scenario

A scenario describes:

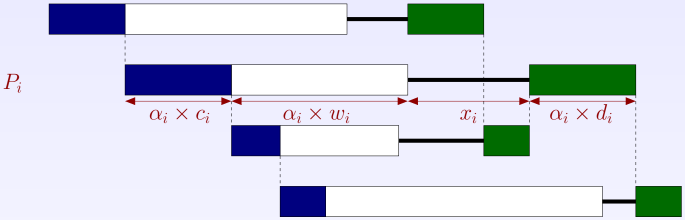
- which workers participate
- in which order are performed the communications (sending data and receiving results)

Then we can suppose that:

- the master sends data as soon as possible
- the workers start computing as soon as possible
- return communications are performed as late as possible

Linear Program for a given scenario

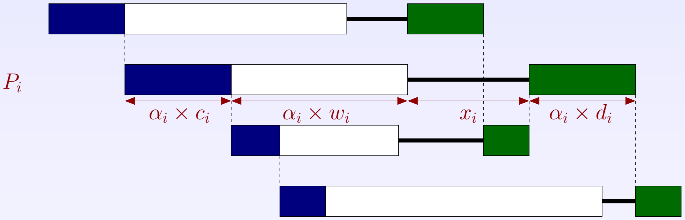
first messages sent to $P_1, P_2, \ldots, P_n$, permutation σ for return messages



Consider worker P_i:

Linear Program for a given scenario

first messages sent to $P_1, P_2, \ldots, P_n$, permutation σ for return messages

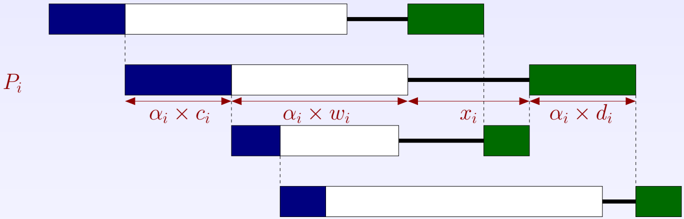


Consider worker P_i:

- starts receiving its data at $t_i^{\text{recv}} = \sum_{j,\ j<i} \alpha_j \times c_j$

Linear Program for a given scenario

first messages sent to $P_1, P_2, \ldots, P_n$, permutation σ for return messages

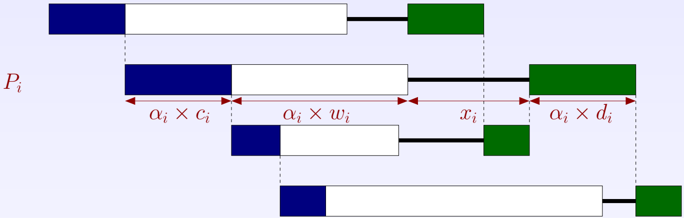


Consider worker P_i:

- starts receiving its data at $t_i^{\text{recv}} = \sum_{j,\ j<i} \alpha_j \times c_j$
- starts execution at $t_i^{\text{recv}} + \alpha_i \times c_i$

Linear Program for a given scenario

first messages sent to $P_1, P_2, \dots, P_n$, permutation σ for return messages

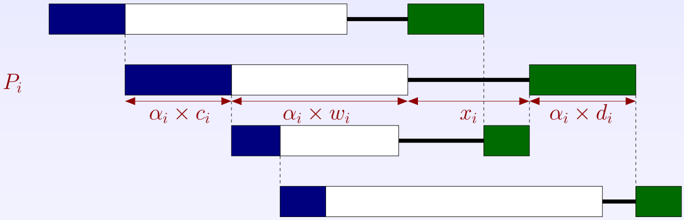


Consider worker P_i:

- starts receiving its data at $t_i^{\text{recv}} = \sum_{j,\ j<i} \alpha_j \times c_j$
- starts execution at $t_i^{\text{recv}} + \alpha_i \times c_i$
- terminates execution at $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$

Linear Program for a given scenario

first messages sent to $P_1, P_2, \ldots, P_n$, permutation σ for return messages

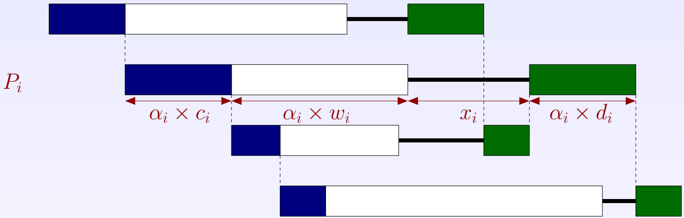


Consider worker P_i:

- starts receiving its data at $t_i^{\text{recv}} = \sum_{j,\ j<i} \alpha_j \times c_j$
- starts execution at $t_i^{\text{recv}} + \alpha_i \times c_i$
- terminates execution at $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$
- starts sending results at $t_i^{\text{back}} = T - \sum_{j,\ \sigma(j) \geqslant \sigma(i)} \alpha_j \times d_j$

Linear Program for a given scenario

first messages sent to $P_1, P_2, \ldots, P_n$, permutation σ for return messages



Consider worker P_i:

- starts receiving its data at $t_i^{\text{recv}} = \sum_{j,\ j<i} \alpha_j \times c_j$
- starts execution at $t_i^{\text{recv}} + \alpha_i \times c_i$
- terminates execution at $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$
- starts sending results at $t_i^{\text{back}} = T - \sum_{j,\ \sigma(j)\geqslant\sigma(i)} \alpha_j \times d_j$
- idle time: $x_i = t_i^{\text{back}} - t_i^{\text{term}} \geqslant 0$

Linear Program for a given scenario

With a fixed T, we get the following LP:

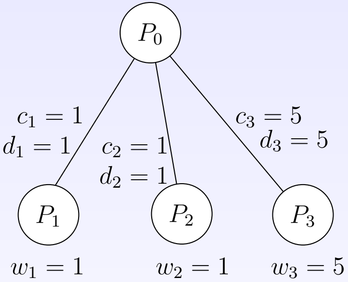
$$\begin{array}{l} \textsc{Maximize } \sum_i \alpha_i, \\ \textsc{under the constraints} \\ \begin{cases} \alpha_i \geqslant 0 \\ t_i^{\text{back}} - t_i^{\text{term}} \geqslant 0 \end{cases} \end{array} \tag{1}$$

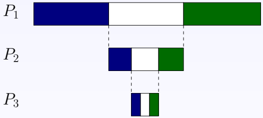
- It gives the optimal throughput
- Given a set of resources and the communications ordering

$\Rightarrow$ not possible to test all configurations

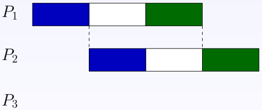
- Even if we force the order of the return messages to be the same as for the initial messages (FIFO), still an exponential number of scenarios

Not all processors always participate to the computation



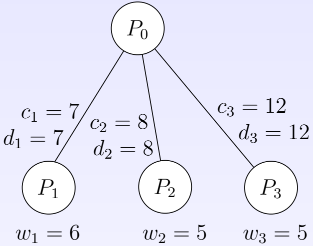


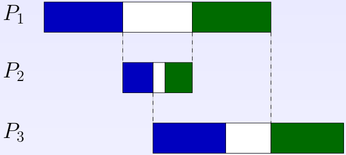
LIFO, throughput $\rho = 61/135$
(best among schedules with 3 processors)



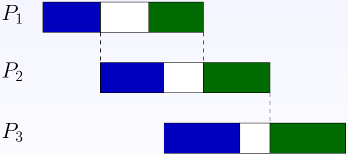
FIFO with 2 processors,
optimal throughput $\rho = 1/2$

Optimal schedule might be non-LIFO and non-FIFO

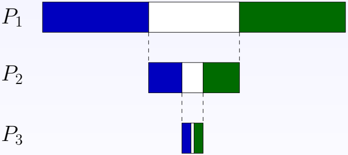




Optimal schedule
($\rho = 38/499 \approx 0.076$)



best FIFO schedule
($\rho = 47/632 \approx 0.074$)

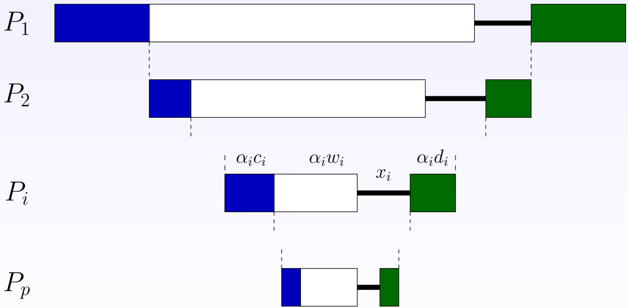


best LIFO schedule
($\rho = 43/580 \approx 0.074$)

Outline

LIFO strategies

- LIFO = Last In First Out
- Processor that receives first its data is the last one sending its results
- The ordering of return messages is the reverse of the ordering of initial messages



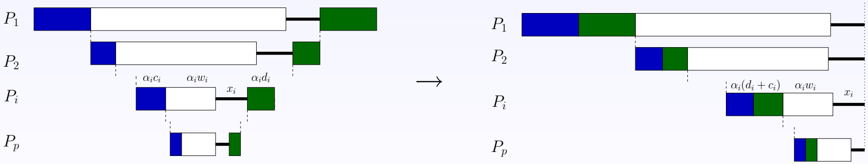
LIFO strategies

Theorem

In the optimal LIFO solution, then

- *All processors participate to the execution*
- *Initial messages must be sent by non-decreasing values of* $c_i + d_i$
- *There is no idle time, i.e.* $x_i = 0$ *for all* i*.*

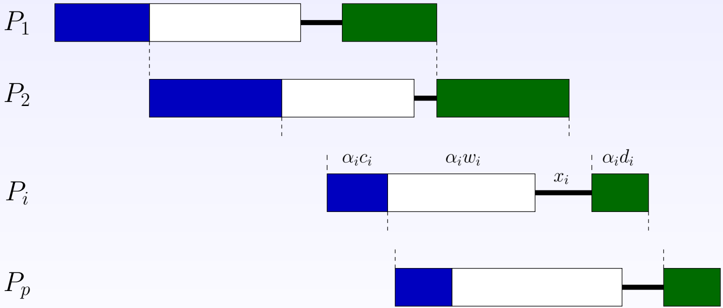
Proof: modify the platform: $c_i \leftarrow c_i + d_i$ and $d_i \leftarrow 0$



$\Rightarrow$ reduces to a classic DLS problem without return messages

FIFO strategies

- FIFO = First In First Out
- The ordering of return messages is the same as the ordering of initial messages



We restrict to the case where $d_i = z \times c_i$ $(z < 1)$

FIFO strategies

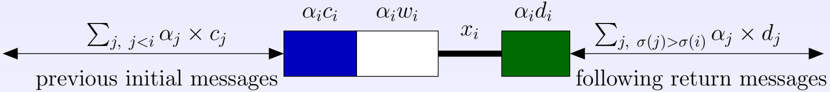
Theorem

In the optimal FIFO solution, then

- *Initial messages must be sent by non-decreasing values of* $c_i + d_i$
- *The set of participating processors is composed of the first* q *processors for the previous ordering, where* q *can be determined in linear time*
- *There is no idle time, i.e.* $x_i = 0$ *for all* i*.*

FIFO strategies

Consider processor P_i in the schedule:



$$\sum_{j,\ j\leqslant i} \alpha_i \times c_i + \alpha_i \times w_i \quad \sum_{j,\ \sigma(j)\geqslant\sigma(i)} \alpha_i \times d_i + x_i = T$$

So we have:

$A\alpha + x = T\mathbb{1}$, where:

$$A = \begin{pmatrix} c_1 + w_1 + d_1 & d_2 & d_3 & \dots & d_k \\ c_1 & c_2 + w_2 + d_2 & d_3 & \dots & d_k \\ \vdots & c_2 & c_3 + w_3 + d_3 & \ddots & \vdots \\ \vdots & \vdots & & \ddots & d_k \\ c_1 & c_2 & c_3 & \dots & c_k + w_k + d_k \end{pmatrix}$$

FIFO strategies

We can write $A = L + \mathbb{1}d^T$, with:

$$L = \begin{pmatrix} c_1 + w_1 & 0 & 0 & \dots & 0 \\ c_1 - d_1 & c_2 + w_2 & 0 & \dots & 0 \\ \vdots & c_2 - d_2 & c_3 + w_3 & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ c_1 - d_1 & c_2 - d_2 & c_3 - d_3 & \dots & c_k + w_k \end{pmatrix} \text{ and } d = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_k \end{pmatrix}$$

Matrix $\mathbb{1}d^t$ is a rank-one matrix, so we can use the Sherman-Morrison formula to compute the inverse of A:

$$A^{-1} = (L + \mathbb{1}d^t)^{-1} = L^{-1} - \frac{L^{-1}\mathbb{1}d^t L^{-1}}{1 + d^t L^{-1}\mathbb{1}}$$

FIFO strategies

Using this formula for A^{-1}, we are able to:

- prove that for all processor P_i, either $\alpha_i = 0$ (processor does not compute at all) or $x_i = 0$ (no idle time)
- derive an analytical formula for the throughput $\rho(T) = \sum_i \alpha_i$
- prove that throughput is better when $c_1 \leqslant c_2 \leqslant c_3 \ldots \leqslant c_n$
- throughput is best when only processors with $d_i \leqslant \dfrac{1}{\rho_{\text{opt}}}$

FIFO strategies - special cases

- Until now, we have suppose that $d_i = z \times c_i$, with $z < 1$
- If $z > 1$, symmetric solution (send initial messages by decreasing value of $d_i + c_i$, select the first q processors in this order)
- $z = 1 \Rightarrow$ order has no importance in this case

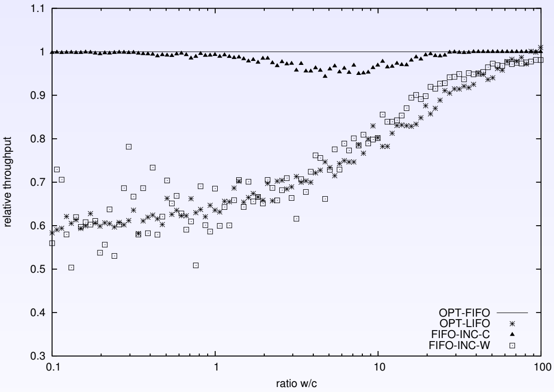
Outline

Simulations

- Not possible to compute the optimal schedule in the general case (for 100 processors $\rightarrow$ $(100!)^2$ linear programs with 100 unknowns to solve...)
- Use optimal FIFO ordering as a comparison basis
- Also compute optimal LIFO solution
- Some FIFO heuristic, with all processors :
  - ordered by increasing value of c_i (fastest communicating worker first)
  - ordered by increasing value of w_i (fastest computing worker first)

Simulations

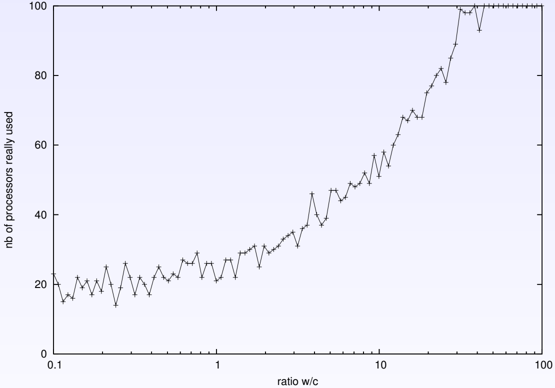
100 processors, $z = 80\%$



- x-axis: ratio between average communication and computation costs (w/c)
- y-axis: throughput versus optimal FIFO throughput

Simulations

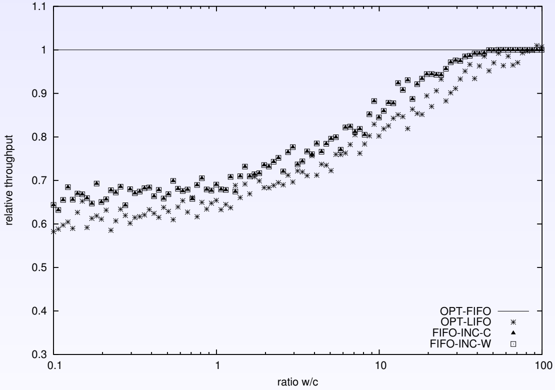
Number of processors used by the optimal FIFO schedule:



- x-axis: ratio between average communication and computation costs (w/c)

Simulations

z=1 (same size for initial messages and return messages)



- x-axis: ratio between average communication and computation costs (w/c)

Outline

Conclusion

- Divisible Load Scheduling with return messages on star platforms
- Natural extension to classical studies
- Leads to considerable difficulties (quite unexpected in the linear model)
- Complexity of the problem is still open
- Characterization of optimal FIFO and LIFO solutions
- Future work:
  - Investigate the general case
  - Extend results to the unidirectional one-port model (master cannot send AND receive at the same time)