Divisible load theory

Loris Marchal, material from Frédéric Vivien

30 janvier 2009
Overview

The context

Bus-like network: classical resolution

Bus-like network: resolution under the divisible load model

Star-like network

With return messages

Conclusion
Overview

The context

Bus-like network: classical resolution

Bus-like network: resolution under the divisible load model

Star-like network

With return messages

Conclusion
Context of the study

- Scientific computing: large needs in computation or storage resources.
- Need to use systems with "several processors":
  - Parallel computers with shared memory
  - Parallel computers with distributed memory
  - Clusters
  - Heterogeneous clusters
  - Clusters of clusters
  - Network of workstations
  - The Grid
- Problematic: to take into account the heterogeneity at the algorithmic level.
New platforms, new problems

Execution platforms: Distributed heterogeneous platforms (network of workstations, clusters, clusters of clusters, grids, etc.)

New sources of problems

- Heterogeneity of processors (computational power, memory, etc.)
- Heterogeneity of communications links.
- Irregularity of interconnection network.
- Non dedicated platforms.

We need to adapt our algorithmic approaches and our scheduling strategies: new objectives, new models, etc.
An example of application: seismic tomography of the Earth

- Model of the inner structure of the Earth

- The model is validated by comparing the propagation time of a seismic wave in the model to the actual propagation time.

- Set of all seismic events of the year 1999: 817101

- Original program written for a parallel computer:

```c
if (rank == ROOT)
    raydata ← read n lines from data file;
MPI_Scatter(raydata, n/P, ..., rbuff, ..., ROOT, MPI_COMM_WORLD);
compute_work(rbuff);
```
Applications covered by the divisible loads model

Applications made of a very (very) large number of fine grain computations.

Computation time proportional to the size of the data to be processed.

Independent computations: neither synchronizations nor communications.
Overview

The context

Bus-like network: classical resolution

Bus-like network: resolution under the divisible load model

Star-like network

With return messages

Conclusion
Bus-like network

- The links between the master and the slaves all have the same characteristics.
- The slave have different computation power.
Notations

- A set $P_1, \ldots, P_p$ of processors
- $P_1$ is the master processor: initially, it holds all the data.
- The overall amount of work: $W_{\text{total}}$.
- Processor $P_i$ receives an amount of work: $n_i \in \mathbb{N}$ with $\sum_i n_i = W_{\text{total}}$.
- Length of a unit-size work on processor $P_i$: $w_i$.
- Computation time on $P_i$: $n_i w_i$.
- Time needed to send a unit-message from $P_1$ to $P_i$: $c$.

One-port bus: $P_1$ sends a single message at a time over the bus, all processors communicate at the same speed with the master.
Behavior of the master and of the slaves (illustration)
Behavior of the master and of the slaves (hypotheses)

- The master sends its chunk of $n_i$ data to processor $P_i$ in a single sending.
- The master sends their data to the processors, serving one processor at a time, in the order $P_2, \ldots, P_p$.
- During this time the master processes its $n_1$ data.
- A slave does not start the processing of its data before it has received all of them.
Equations

- \( P_1 : T_1 = n_1 \cdot w_1 \)
- \( P_2 : T_2 = n_2 \cdot c + n_2 \cdot w_2 \)
- \( P_3 : T_3 = (n_2 \cdot c + n_3 \cdot c) + n_3 \cdot w_3 \)
- \( P_i : T_i = \sum_{j=2}^{i} n_j \cdot c + n_i \cdot w_i \) for \( i \geq 2 \)
- \( P_i : T_i = \sum_{j=1}^{i} n_j \cdot c_j + n_i \cdot w_i \) for \( i \geq 1 \) with \( c_1 = 0 \) and \( c_j = c \) otherwise.
Execution time

\[ T = \max_{1 \leq i \leq p} \left( \sum_{j=1}^{i} n_j \cdot c_j + n_i \cdot w_i \right) \]

We look for a data distribution \( n_1, \ldots, n_p \) which minimizes \( T \).
Execution time: rewriting

\[ T = \max \left( n_1.c_1 + n_1.w_1, \, \max_{2 \leq i \leq p} \left( \sum_{j=1}^{i} n_j.c_j + n_i.w_i \right) \right) \]

\[ T = n_1.c_1 + \max \left( n_1.w_1, \, \max_{2 \leq i \leq p} \left( \sum_{j=2}^{i} n_j.c_j + n_i.w_i \right) \right) \]

An optimal solution for the distribution of \( W_{\text{total}} \) data over \( p \) processors is obtained by distributing \( n_1 \) data to processor \( P_1 \) and then optimally distributing \( W_{\text{total}} - n_1 \) data over processors \( P_2 \) to \( P_p \).
Algorithm

1: \textit{solution}[0, p] \leftarrow \text{cons}(0, \text{NIL}) ; \textit{cost}[0, p] \leftarrow 0
2: \textbf{for} \ d \leftarrow 1 \ \textbf{to} \ W_{\text{total}} \ \textbf{do}
3: \quad \textit{solution}[d, p] \leftarrow \text{cons}(d, \text{NIL})
4: \quad \textit{cost}[d, p] \leftarrow d \cdot c_p + d \cdot w_p
5: \quad \textbf{for} \ i \leftarrow p - 1 \ \textbf{downto} \ 1 \ \textbf{do}
6: \quad \textit{solution}[0, i] \leftarrow \text{cons}(0, \textit{solution}[0, i + 1])
7: \quad \textit{cost}[0, i] \leftarrow 0
8: \quad \textbf{for} \ d \leftarrow 1 \ \textbf{to} \ W_{\text{total}} \ \textbf{do}
9: \quad \quad (\textit{sol}, \textit{min}) \leftarrow (0, \textit{cost}[d, i + 1])
10: \quad \quad \textbf{for} \ e \leftarrow 1 \ \textbf{to} \ d \ \textbf{do}
11: \quad \quad \quad m \leftarrow e \cdot c_i + \max(e \cdot w_i, \textit{cost}[d - e, i + 1])
12: \quad \quad \quad \textbf{if} \ m < \textit{min} \ \textbf{then}
13: \quad \quad \quad \quad (\textit{sol}, \textit{min}) \leftarrow (e, m)
14: \quad \quad \quad \textit{solution}[d, i] \leftarrow \text{cons}(\textit{sol}, \textit{solution}[d - \textit{sol}, i + 1])
15: \quad \quad \quad \textit{cost}[d, i] \leftarrow \textit{min}
16: \quad \textbf{return} \ (\textit{solution}[W_{\text{total}}, 1], \textit{cost}[W_{\text{total}}, 1])
Complexity

▶ Theoretical complexity

\[ O(W_{\text{total}}^2 \cdot p) \]

▶ Complexity in practice

If \( W_{\text{total}} = 817101 \) and \( p = 16 \), on a Pentium III running at 933 MHz: more than two days...

(Optimized version ran in 6 minutes.)
Disadvantages

- Cost
- Solution is not reusable
- Solution is only partial (processor order is fixed)

We do not need the solution to be so precise
Overview

The context

Bus-like network: classical resolution

Bus-like network: resolution under the divisible load model

Star-like network

With return messages

Conclusion
Notations

- A set \( P_1, \ldots, P_p \) of processors
- \( P_1 \) is the master processor: initially, it holds all the data.
- The overall amount of work: \( W_{\text{total}} \).
- Processor \( P_i \) receives an amount of work \( \alpha_i W_{\text{total}} \) with \( \alpha_i W_{\text{total}} \in \mathbb{Q} \) and \( \sum_i \alpha_i = 1 \).
  - Length of a unit-size work on processor \( P_i \): \( w_i \).
  - Computation time on \( P_i \): \( \alpha_i w_i \).
- Time needed to send a unit-message from \( P_1 \) to \( P_i \): \( c \).
  - One-port model: \( P_1 \) sends a \textit{single} message at a time, all processors communicate at the same speed with the master.
Equations

For processor $P_i$ (with $c_1 = 0$ and $c_j = c$ otherwise):

$$T_i = \sum_{j=1}^{i} \alpha_j W_{total} \cdot c_j + \alpha_i W_{total} \cdot w_i$$

$$T = \max_{1 \leq i \leq p} \left( \sum_{j=1}^{i} \alpha_j W_{total} \cdot c_j + \alpha_i W_{total} \cdot w_i \right)$$

We look for a data distribution $\alpha_1, ..., \alpha_p$ which minimizes $T$. 
Properties of load-balancing

Lemma

*In an optimal solution, all processors end their processing at the same time.*
Demonstration of lemma 1

Two slaves $i$ and $i + 1$ with $T_i < T_{i+1}$.
(the same results holds with $i - 1$ and $i$)

We decrease $\alpha_{i+1}$ by $\epsilon$. 
Demonstration of lemma 1

Two slaves $i$ and $i + 1$ with $T_i < T_{i+1}$.
(the same results holds with $i - 1$ and $i$)

We decrease $\alpha_{i+1}$ by $\epsilon$. 
Demonstration of lemma 1

Two slaves $i$ and $i + 1$ with $T_i < T_{i+1}$.
(the same results holds with $i - 1$ and $i$)

We decrease $\alpha_{i+1}$ by $\epsilon$. 
Demonstration of lemma 1

Two slaves $i$ and $i + 1$ with $T_i < T_{i+1}$.
(the same results holds with $i - 1$ and $i$)

We increase $\alpha_i$ by $\epsilon$. 
Demonstration of lemma 1

Two slaves $i$ and $i + 1$ with $T_i < T_{i+1}$.
(the same results holds with $i - 1$ and $i$)

We increase $\alpha_i$ by $\epsilon$. 
Demonstration of lemma 1

Two slaves $i$ and $i + 1$ with $T_i < T_{i+1}$.
(the same results holds with $i - 1$ and $i$)

The communication time for the following processors is unchanged.
Demonstration of lemma 1

Two slaves $i$ and $i + 1$ with $T_i < T_{i+1}$.
(the same results holds with $i - 1$ and $i$)

We end up with a better solution!
Demonstration of lemma 1 (continuation and conclusion)

- Ideal: $T'_i = T'_{i+1}$.

  We choose $\epsilon$ such that:

  $$(\alpha_i + \epsilon)W_{\text{total}}(c + w_i) = (\alpha_i + \epsilon)W_{\text{total}}c + (\alpha_{i+1} - \epsilon)W_{\text{total}}(c + w_{i+1})$$

- The master stops before the slaves: absurde.
- The master stops after the slaves: we decrease $P_1$ by $\epsilon$. 
Lemma

In an optimal solution all processors work.

Demonstration: this is just a corollary of lemma 1...
Resolution

\[ T = \alpha_1 W_{\text{total}} w_1. \]

\[ T = \alpha_2 (c + w_2) W_{\text{total}}. \text{ Therefore } \alpha_2 = \frac{w_1}{c + w_2} \alpha_1. \]

\[ T = (\alpha_2 c + \alpha_3 (c + w_3)) W_{\text{total}}. \text{ Therefore } \alpha_3 = \frac{w_2}{c + w_3} \alpha_2. \]

\[ \alpha_i = \frac{w_{i-1}}{c + w_i} \alpha_{i-1} \text{ for } i \geq 2. \]

\[ \sum_{i=1}^{n} \alpha_i = 1. \]

\[ \alpha_1 \left( 1 + \frac{w_1}{c + w_2} + \ldots + \prod_{k=2}^{j} \frac{w_{k-1}}{c + w_k} + \ldots \right) = 1 \]
Impact of the order of communications

How important is the influence of the ordering of the processor on the solution?
No impact of the order of the communications

Volume processed by processors $P_i$ and $P_{i+1}$ during a time $T$.

**Processor** $P_i$: $\alpha_i(c + w_i)W_{\text{total}} = T$. Therefore $\alpha_i = \frac{1}{c+w_i} \frac{T}{W_{\text{total}}}$.

**Processor** $P_{i+1}$: $\alpha_i c W_{\text{total}} + \alpha_{i+1}(c + w_{i+1})W_{\text{total}} = T$.
Thus $\alpha_{i+1} = \frac{1}{c+w_{i+1}} \left( \frac{T}{W_{\text{total}}} - \alpha_i c \right) = \frac{w_i}{(c+w_i)(c+w_{i+1})} \frac{T}{W_{\text{total}}}$.

**Processors** $P_i$ and $P_{i+1}$:

$$\alpha_i + \alpha_{i+1} = \frac{c + w_i + w_{i+1}}{(c + w_i)(c + w_{i+1})}$$
Choice of the master processor

We compare processors $P_1$ and $P_2$.

**Processor** $P_1$ : $\alpha_1 w_1 W_{\text{total}} = T$. Then, $\alpha_1 = \frac{1}{w_1} \frac{T}{W_{\text{total}}}$.

**Processor** $P_2$ : $\alpha_2 (c + w_2) W_{\text{total}} = T$. Thus, $\alpha_2 = \frac{1}{c + w_2} \frac{T}{W_{\text{total}}}$.

Total volume processed :

$$\alpha_1 + \alpha_2 = \frac{c + w_1 + w_2}{w_1 (c + w_2)} = \frac{c + w_1 + w_2}{cw_1 + w_1 w_2}$$

Minimal when $w_1 < w_2$.

Master = the most powerfull processor (for computations).
Conclusion

- Closed-form expressions for the execution time and the distribution of data.

- Choice of the master.

- The ordering of the processors has no impact.

- All processors take part in the work.
Overview

The context

Bus-like network: classical resolution

Bus-like network: resolution under the divisible load model

Star-like network

With return messages

Conclusion
The links between the master and the slaves have *different* characteristics.

The slaves have different computational power.
Notations

- A set $P_1, ..., P_p$ of processors
- $P_1$ is the master processor: initially, it holds all the data.
- The overall amount of work: $W_{\text{total}}$.
- Processor $P_i$ receives an amount of work $\alpha_i W_{\text{total}}$ with $\sum_i n_i = W_{\text{total}}$ with $\alpha_i W_{\text{total}} \in \mathbb{Q}$ and $\sum_i \alpha_i = 1$.
- Length of a unit-size work on processor $P_i$: $w_i$.
- Computation time on $P_i$: $n_i w_i$.
- **Time needed to send a unit-message from $P_1$ to $P_i$: $c_i$.**
- One-port model: $P_1$ sends a *single* message at a time.
Ressource selection

Lemma

*In an optimal solution, all processors work.*

We take an optimal solution. Let $P_k$ be a processor which does not receive any work: we put it last in the processor ordering and we give it a fraction $\alpha_k$ such that $\alpha_k(c_k + w_k)W_{\text{total}}$ is equal to the processing time of the last processor which received some work.

Why should we put this processor last?
Lemma

In an optimal solution, all processors end at the same time.
**Demonstration of lemma 4**

- Most existing proofs are false.

\[
\text{Minimize } T, \\
\text{subject to } \begin{cases} 
\sum_{i=1}^{n} \alpha_i = 1 \\
\forall i, \quad \alpha_i \geq 0 \\
\forall i, \quad \sum_{k=1}^{i} \alpha_k c_k + \alpha_i w_i \leq T 
\end{cases}
\]

- The constraints define a polyhedron
- One of the optimal solution is a vertex of the polyhedron, that is at least \( n \) among the \( 2n \) inequalities are equalities,
- It can not be a lower bound, because all processors participate, thus this point is an optimal solution
- Assume there is another optimal solution; it lies within the polyhedron
- We can build a linear combination of both optimal solution (with optimal objective), such that one variable is zero, which
Demonstration of lemma 4

- Most existing proofs are false.

\[
\begin{align*}
\text{Minimize } & T, \\
\text{subject to } & \sum_{i=1}^{n} \alpha_i = 1 \\
& \forall i, \quad \alpha_i \geq 0 \\
& \forall i, \quad \sum_{k=1}^{i} \alpha_k c_k + \alpha_i w_i \leq T
\end{align*}
\]

- The constraints define a polyhedron
  - One of the optimal solution is a vertex of the polyhedron, that is at least \( n \) among the \( 2n \) inequalities are equalities,
  - It can not be a lower bound, because all processors participate, thus this point is an optimal solution
  - Assume there is another optimal solution; it lies within the polyhedron
  - We can build a linear combination of both optimal solution (with optimal objective), such that one variable is zero, which
Demonstration of lemma 4

- Most existing proofs are false.

\[ \text{Minimize } T, \text{ subject to } \]
\[ \sum_{i=1}^{n} \alpha_i = 1 \]
\[ \forall i, \quad \alpha_i \geq 0 \]
\[ \forall i, \quad \sum_{k=1}^{i} \alpha_k c_k + \alpha_i w_i \leq T \]

- The constraints define a polyhedron
- One of the optimal solution is a vertex of the polyhedron, that is at least \( n \) among the \( 2n \) inequalities are equalities,
- It can not be a lower bound, because all processors participate, thus this point is an optimal solution
- Assume there is another optimal solution; it lies within the polyhedron
- We can build a linear combination of both optimal solution (with optimal objective), such that one variable is zero, which
Demonstration of lemma 4

- Most existing proofs are false.

Minimize $T$,
subject to
\[
\begin{align*}
\forall i, & \quad \sum_{i=1}^{n} \alpha_i = 1 \\
\forall i, & \quad \alpha_i \geq 0 \\
\forall i, & \quad \sum_{k=1}^{i} \alpha_k c_k + \alpha_i w_i \leq T
\end{align*}
\]

- The constraints define a polyhedron
- One of the optimal solution is a vertex of the polyhedron, that is at least $n$ among the $2n$ inequalities are equalities,
- It can not be a lower bound, because all processors participate, thus this point is an optimal solution
- Assume there is another optimal solution; it lies within the polyhedron
- We can build a linear combination of both optimal solution (with optimal objective), such that one variable is zero, which
Demonstration of lemma 4

Most existing proofs are false.

Minimize $T$, subject to

\[
\begin{align*}
\sum_{i=1}^{n} \alpha_i &= 1 \\
\forall i, \quad \alpha_i &\geq 0 \\
\forall i, \quad \sum_{k=1}^{i} \alpha_k c_k + \alpha_i w_i &\leq T
\end{align*}
\]

The constraints define a polyhedron

One of the optimal solution is a vertex of the polyhedron, that is at least $n$ among the $2n$ inequalities are equalities,

It can not be a lower bound, because all processors participate, thus this point is an optimal solution

Assume there is another optimal solution; it lies within the polyhedron

We can build a linear combination of both optimal solution (with optimal objective), such that one variable is zero, which
Demonstration of lemma 4

▶ Most existing proofs are false.

\[
\text{MINIMIZE } T, \\
\text{SUBJECT TO} \\
\begin{array}{ll}
\sum_{i=1}^{n} \alpha_i &= 1 \\
\forall i, & \alpha_i \geq 0 \\
\forall i, & \sum_{k=1}^{i} \alpha_k c_k + \alpha_i w_i \leq T
\end{array}
\]

▶ The constraints define a polyhedron
▶ One of the optimal solution is a vertex of the polyhedron, that is at least \( n \) among the \( 2n \) inequalities are equalities,
▶ It can not be a lower bound, because all processors participate, thus this point is an optimal solution
▶ Assume there is another optimal solution; it lies within the polyhedron
▶ We can build a linear combination of both optimal solution (with optimal objective), such that one variable is zero, which
Impact of the order of communications

Volume processed by processors $P_i$ and $P_{i+1}$ during a time $T$.

**Processor** $P_i$: $\alpha_i (c_i + w_i) W_{\text{total}} = T$. Thus, $\alpha_i = \frac{1}{c_i + w_i} \frac{T}{W_{\text{total}}}$.

**Processor** $P_{i+1}$: $\alpha_i c_i W_{\text{total}} + \alpha_{i+1} (c_{i+1} + w_{i+1}) W_{\text{total}} = T$. Thus, $\alpha_{i+1} = \frac{1}{c_{i+1} + w_{i+1}} (1 - \frac{c_i}{c_i + w_i}) \frac{T}{W_{\text{total}}} = \frac{w_i}{(c_i + w_i)(c_{i+1} + w_{i+1})} \frac{T}{W_{\text{total}}}$.

Volume processed: $\alpha_i + \alpha_{i+1} = \frac{c_{i+1} + w_i + w_{i+1}}{(c_i + w_i)(c_{i+1} + w_{i+1})}$

Communication time: $\alpha_i c_i + \alpha_{i+1} c_{i+1} = \frac{c_i c_{i+1} + c_{i+1} w_i + c_i w_{i+1}}{(c_i + w_i)(c_{i+1} + w_{i+1})}$

Processors must be served by decreasing bandwidths.
Conclusion

- The processors must be ordered by decreasing bandwidths
- All processors are working
- All processors end their work at the same time
- Formulas for the execution time and the distribution of data
Overview

The context

Bus-like network: classical resolution

Bus-like network: resolution under the divisible load model

Star-like network

With return messages

Conclusion
With return messages

Once it has finished processing its share of the total load, a slave sends back a result to the master.

Problems to be solved:
- Resource selection.
- Defining an order for sending the data to the slaves.
- Defining an order for receiving the data from the slaves.
- Defining the amount of work each processor has to process.
Notations

- A set $P_1, \ldots, P_p$ of processors
- $P_1$ is the master processor: initially, it holds all the data.
- The overall amount of work: $W_{\text{total}}$.
- Processor $P_i$ receives an amount of work $\alpha_i W_{\text{total}}$ with $\sum_i n_i = W_{\text{total}}$ with $\alpha_i W_{\text{total}} \in \mathbb{Q}$ and $\sum_i \alpha_i = 1$.
- Length of a unit-size work on processor $P_i$: $w_i$.
- Computation time on $P_i$: $n_i w_i$.
- **Time needed to send a unit-message from $P_1$ to $P_i$:** $c_i$.
- **Time needed to send a unit-message from $P_i$ to $P_1$:** $d_i$.
- One-port model: $P_1$ sends and receives a single message at a time.
Solutions with idle time?

- How about waiting between the end of the reception of the data and the start of the computation?

  Not interesting!

- How about waiting between the end of the computation and the time the results start to be sent back to the master?

  Mandatory if the communication link is not available.

We need to anticipate, when building a solution, the possibility of idle times.
Review of known results in January 2005

(the first paper on divisible loads dates back to 1988)

- Barlas
  - Fixed communication times or bus-like network $c_i = c$.
  - Optimal ordering and closed-form formulas (trivial).
- Drozdowski and Wolniewicz: experimental study of LIFO and FIFO distributions.
- Rosenberg et al.:
  - Complex communication model (affine).
  - Possibility to slow down a processor (to avoid idle times).
  - In practice: communication capabilities are not heterogeneous.
  - All FIFO distributions are equivalent and are better than any other solution (proof made by exchange).
A scenario is described by:

- which processor is given work to;
- in which order the communications take place (sending of the data and gathering of the results).

With a given scenario, one can suppose that:

- the master sends the data as soon as possible;
- the slaves start working as soon as possible;
- the slaves send their as late as possible.
Linear program for a given scenario (2)

Consider slave $P_i$:

- it starts receiving data at time $t_i^{\text{recv}} = \sum_{j=1}^{i-1} \alpha_j \times c_j$
- it starts working at time $t_i^{\text{recv}} + \alpha_i \times c_i$
- it ends processing its load at time $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$
- it starts sending back its results at time $t_i^{\text{back}} = T - \sum_{j \text{ successor of } i} \alpha_j \times d_j$
- its idle time is: $x_i = t_i^{\text{back}} - t_i^{\text{term}} \geq 0$
Linear program for a given scenario (3)

For a given value of $T$, we obtain the linear program:

\[
\text{Maximize } \sum_i \alpha_i, \text{ under the constraints }
\begin{align*}
\alpha_i & \geq 0 \\
t^\text{back}_i - t^\text{term}_i & \geq 0
\end{align*}
\]

▶ Optimal throughput, an ordering and the resource selection being given.

For a given amount of work $\sum_i \alpha_i = W$:

\[
\text{Minimize } T, \text{ under the constraints }
\begin{align*}
\alpha_i & \geq 0 \\
\sum_i \alpha_i & = W \\
t^\text{back}_i - t^\text{term}_i & \geq 0
\end{align*}
\]

▶ Minimal time, an ordering and the resource selection being given.
One cannot test all possible configurations

- Even if we decide that the order of return messages should be the same than the order of data distribution messages (FIFO), there still is an exponential number of scenarios to be tested.
All processors do not always participate

\[
\begin{align*}
\text{LIFO, throughput } &\rho = 61/135 \\
&\text{(best schedule with 3 processors)}
\end{align*}
\]

\[
\begin{align*}
\text{FIFO with 2 processors, } &\rho = 1/2
\end{align*}
\]
The optimal schedule may be neither LIFO nor FIFO

Optimal schedule
\( \left( \rho = \frac{38}{499} \approx 0.076 \right) \)

Best FIFO schedule
\( \left( \rho = \frac{47}{632} \approx 0.074 \right) \)

Best LIFO schedule
\( \left( \rho = \frac{43}{580} \approx 0.074 \right) \)
LIFO strategies (1)

- LIFO = Last In First Out
- The processor which receives its data first is the last to send its results back.
- The order of the return messages is the inverse of the order in which data are sent.
**LIFO strategies (2)**

**Theorem**

*In the best LIFO solution:*

- All processors work
- The processors are sent by increasing values of $c_i + d_i$
- There is no idle time, i.e. $x_i = 0$ for each $i$.

**Demonstration**: We change the platform: $c_i \leftarrow c_i + d_i$ and $d_i \leftarrow 0$

$\Rightarrow$ reduction to a classical problem without return messages.
FIFO strategies (1)

- FIFO = First In First Out
- The order the data are sent is the same than the order the return messages are sent.

We only consider the case \( d_i = z \times c_i \) \((z < 1)\)
FIFO strategies (2)

Theorem

In the best FIFO solution:

- The data are sent by increasing values of: $c_i + d_i$
- The set of all working processors are made of the first $q$ processors under this order; $q$ can be computed in linear time.
- There is no idle time, i.e. $x_i = 0$ for each $i$. 
FIFO strategies (3)

We consider $i$ in the schedule:

\[
\sum_{j=1}^{i-1} \alpha_j \times c_j + \alpha_i \times c_i + \alpha_i \times w_i + \alpha_i \times d_i + x_i = T
\]

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

\[
A = \begin{pmatrix}
c_1 + w_1 + d_1 & d_2 & d_3 & \ldots & d_k \\
c_1 & c_2 + w_2 + d_2 & d_3 & \ldots & d_k \\
\vdots & \vdots & c_3 + w_3 + d_3 & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & d_k \\
c_1 & c_2 & c_3 & \ldots & c_k + w_k + d_k
\end{pmatrix}
\]
FIFO strategies (4)

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

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

The matrix $\mathbb{1}d^t$ is a matrix of rank one, we can thus use Sherman-Morrison’s 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 (5)

With the formula which gives $A^{-1}$, one can:

- show that for each processor $P_i$, either $\alpha_i = 0$ (the processor does not work) or $x_i = 0$ (no idle time);
- define analytically the throughput $\rho(T) = \sum_i \alpha_i$;
- show that the throughput is best when $c_1 \leq c_2 \leq c_3 \ldots \leq c_n$;
- show that the throughput is best when the only working processors are the one satisfying $d_i \leq \frac{1}{\rho_{opt}}$. 
FIFO strategies — special cases

- So far, we have supposed that $d_i = z \times c_i$, with $z < 1$.
- If $z > 1$, symmetrical solution (the data are sent by decreasing values of $d_i + c_i$, the first $q$ processors are selected under this order).
- $z = 1 \Rightarrow$ the order has no impact (but all processors do not always work).
Overview

The context

Bus-like network: classical resolution

Bus-like network: resolution under the divisible load model

Star-like network

With return messages

Conclusion
Que retenir de tout ça ?

- Idée de base simple : une solution approchée est amplement suffisante.

- Les temps de communication jouent un plus grand rôle que les vitesses de calcul.