Divisible Load Scheduling

Master 2 Research Tutorial: High-Performance Architectures

Arnaud Legrand et Jean-François Méhaut

ID laboratory, arnaud.legrand@imag.fr

November 22, 2006

Outline

1. The context
2. Bus-like network: classical resolution
3. Bus-like network: resolution under the divisible load model
4. Star-like network
5. With return messages
6. Multi-round algorithms
7. Conclusion

Outline

1 The context

2 Bus-like network: classical resolution

3 Bus-like network: resolution under the divisible load model

4 Star-like network

5 With return messages

6 Multi-round algorithms

7 Conclusion

Context of the study

- Scientific computing : large needs in computation or storage resources.

Context of the study

- Scientific computing : large needs in computation or storage resources.
- Need to use systems with "several processors":

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.

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.

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.

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.

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.

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.

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.

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.

New platforms, new problems

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

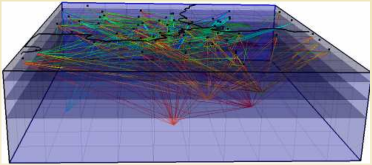
New sources of problems

- 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:

```
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.

Outline

1 The context

2 Bus-like network: classical resolution

3 Bus-like network: resolution under the divisible load model

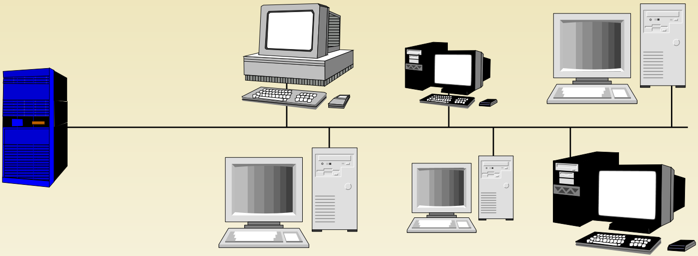
4 Star-like network

5 With return messages

6 Multi-round algorithms

7 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, ..., P_p of processors

Notations

- A set P_1, ..., P_p of processors
- P_1 is the master processor: initially, it holds all the data.

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_{total}.

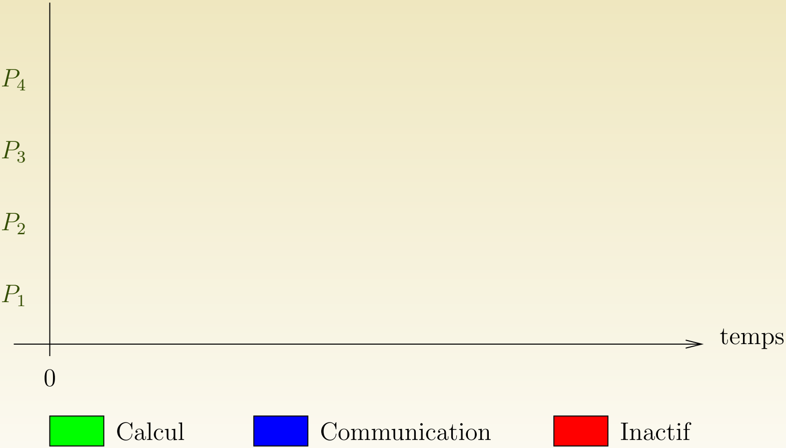
Notations

- 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_{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: $n_i w_i$.

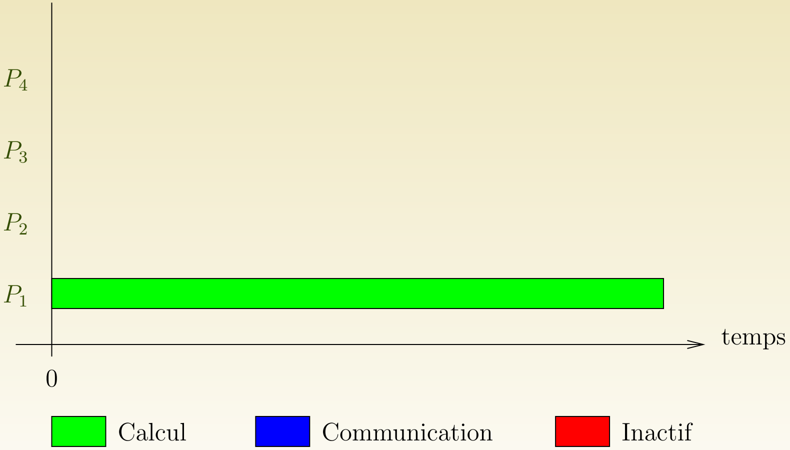
Notations

- 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_{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.

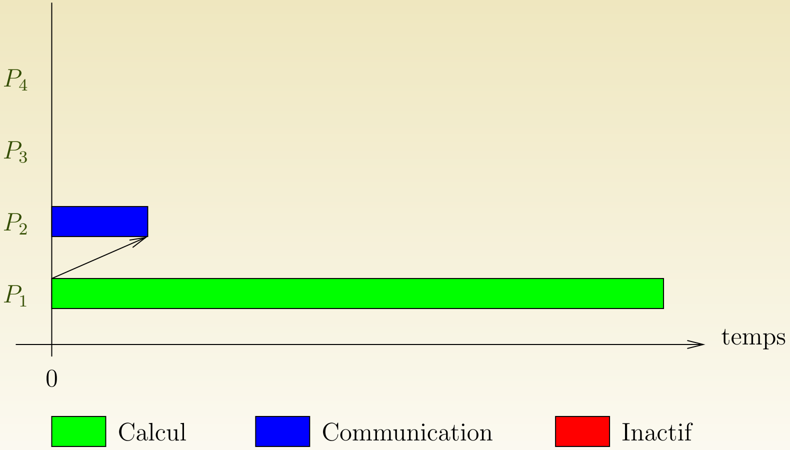
Behavior of the master and of the slaves (illustration)



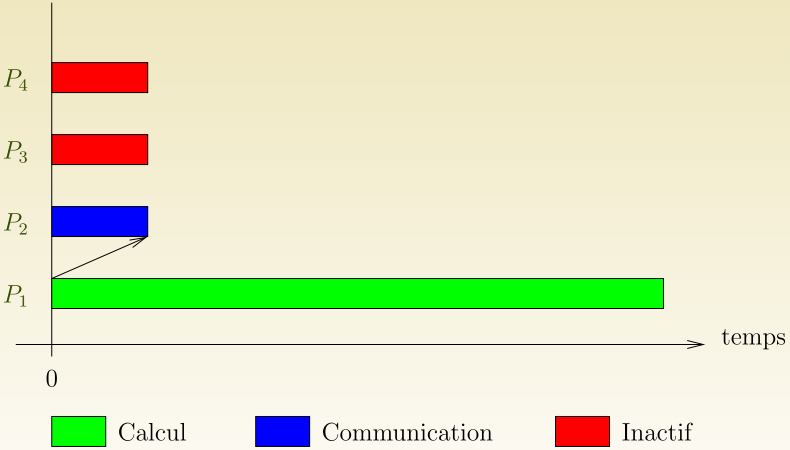
Behavior of the master and of the slaves (illustration)



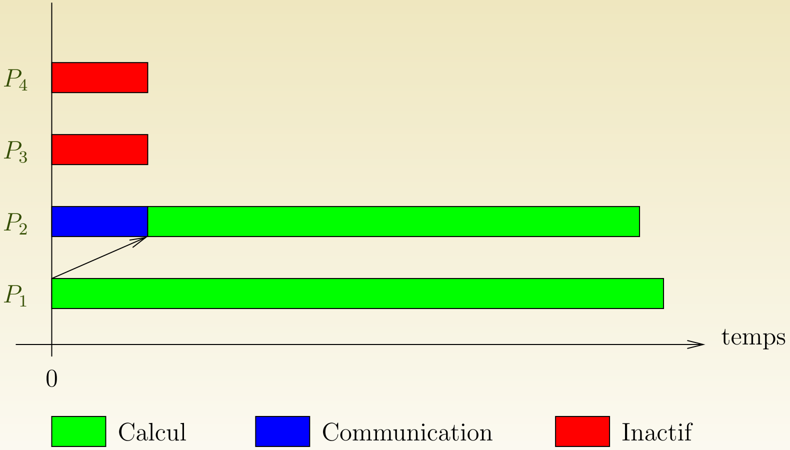
Behavior of the master and of the slaves (illustration)



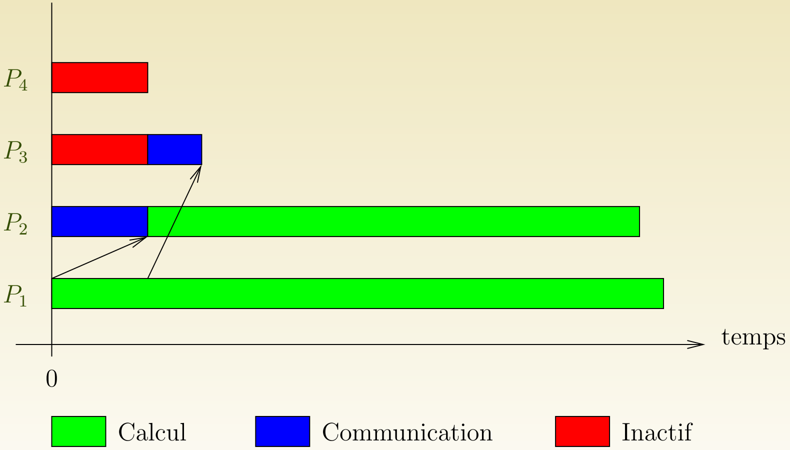
Behavior of the master and of the slaves (illustration)



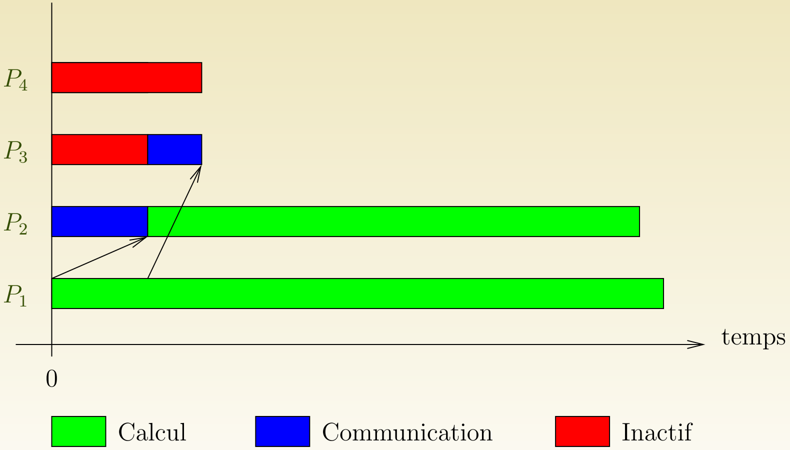
Behavior of the master and of the slaves (illustration)



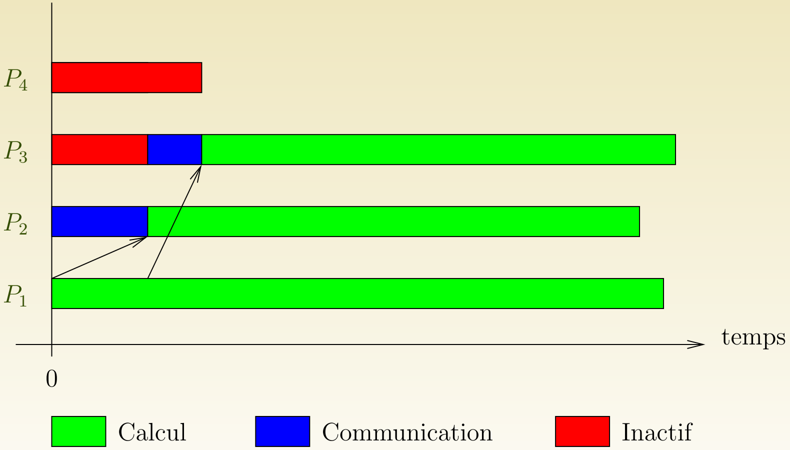
Behavior of the master and of the slaves (illustration)



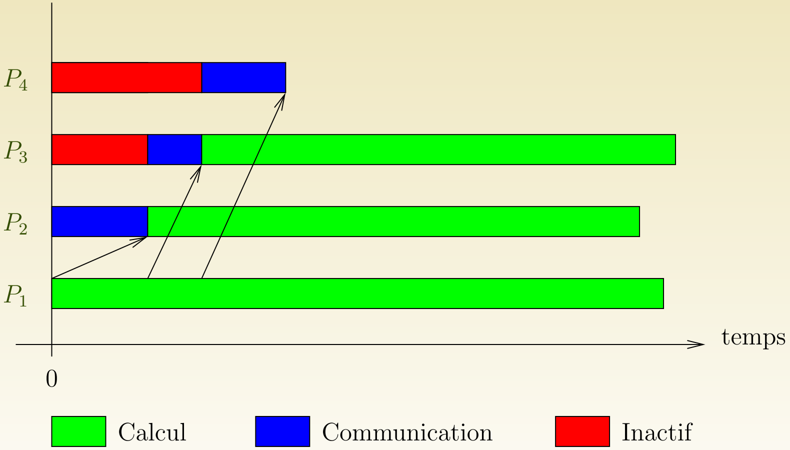
Behavior of the master and of the slaves (illustration)



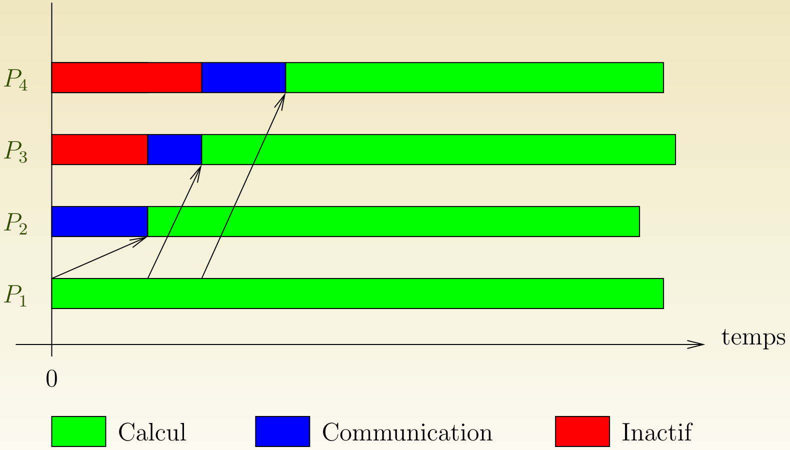
Behavior of the master and of the slaves (illustration)



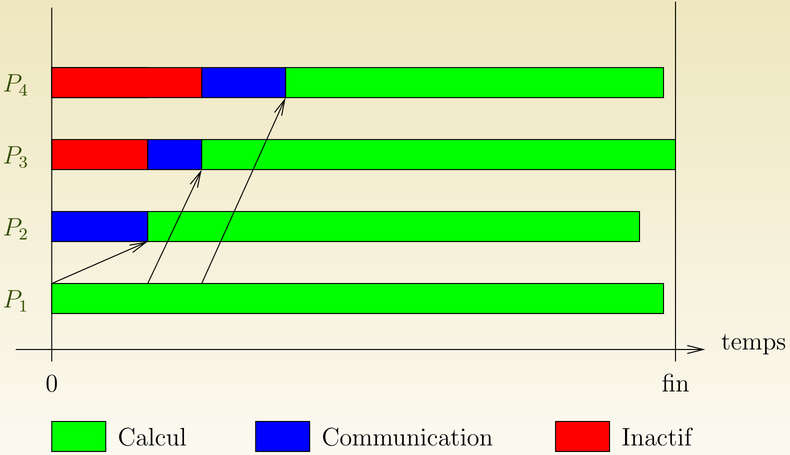
Behavior of the master and of the slaves (illustration)



Behavior of the master and of the slaves (illustration)



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.

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, ..., P_p.

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, ..., P_p.
- During this time the master processes its n_1 data.

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, ..., 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: $T_1 = n_1.w_1$

Equations

- P_1: $T_1 = n_1.w_1$
- P_2: $T_2 = n_2.c + n_2.w_2$

Equations

- P_1: $T_1 = n_1.w_1$
- P_2: $T_2 = n_2.c + n_2.w_2$
- P_3: $T_3 = (n_2.c + n_3.c) + n_3.w_3$

Equations

- P_1: $T_1 = n_1.w_1$
- P_2: $T_2 = n_2.c + n_2.w_2$
- P_3: $T_3 = (n_2.c + n_3.c) + n_3.w_3$
- P_i: $T_i = \sum_{j=2}^{i} n_j.c + n_i.w_i$ for $i \geqslant 2$

Equations

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

Execution time

$$T = \max_{1 \leqslant i \leqslant p} \left(\sum_{j=1}^{i} n_j.c_j + n_i.w_i \right)$$

We look for a data distribution n_1, ..., n_p which minimizes T.

Execution time: rewriting

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

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

An optimal solution for the distribution of W_{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

```
solution[0, p] ← cons(0, NIL); cost[0, p] ← 0
for d ← 1 to W_total do
    solution[d, p] ← cons(d, NIL)
    cost[d, p] ← d · c_p + d · w_p
end for
for i ← p − 1 downto 1 do
    solution[0, i] ← cons(0, solution[0, i + 1])
    cost[0, i] ← 0
    for d ← 1 to W_total do
        (sol, min) ← (0, cost[d, i + 1])
        for e ← 1 to d do
            m ← e · c_i + max(e · w_i, cost[d − e, i + 1])
            if m < min then
                (sol, min) ← (e, m)
            end if
        end for
        solution[d, i] ← cons(sol, solution[d − sol, i + 1])
        cost[d, i] ← min
    end for
end for
return (solution[W_total, 1], cost[W_total, 1])
```

Complexity

- **Theorical 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

Disadvantages

- Cost
- Solution is not reusable
- Solution is only partial

We do not need the solution to be so precise

Outline

1 The context
2 Bus-like network: classical resolution
3 Bus-like network: resolution under the divisible load model
4 Star-like network
5 With return messages
6 Multi-round algorithms
7 Conclusion

Notations

- A set P_1, ..., P_p of processors

Notations

- A set P_1, ..., P_p of processors
- P_1 is the master processor: initially, it holds all the data.

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_{total}.

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_{total}.
- **Processor P_i receives an amount of work $\alpha_i W_{\textbf{total}}$ with $\alpha_i W_{\textbf{total}} \in \mathbb{Q}$ and $\sum_i \alpha_i = 1$.**
  Length of a unit-size work on processor P_i: w_i.
  Computation time on P_i: $n_i w_i$.

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_{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: $n_i w_i$.
- Time needed to send a unit-message from P_1 to P_i: c.
  One-port model: P_1 sends a *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):

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

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

We look for a data distribution α_1, ..., α_p which minimizes T.

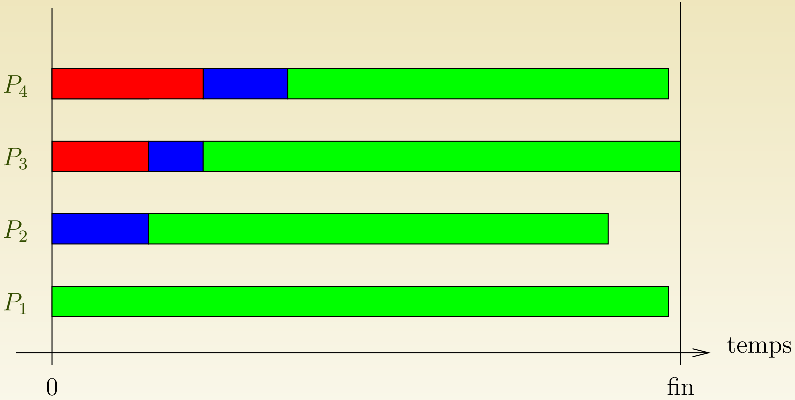
Properties of load-balancing

Lemma 1.

In an optimal solution, all processors end their processing at the same time.

Demonstration of lemma 1

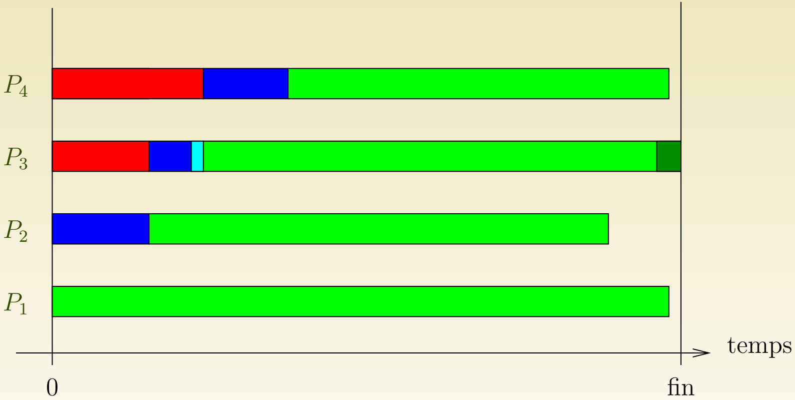
Two slaves i and $i+1$ with $T_i < T_{i+1}$.



We decrease α_{i+1} by ϵ.

Demonstration of lemma 1

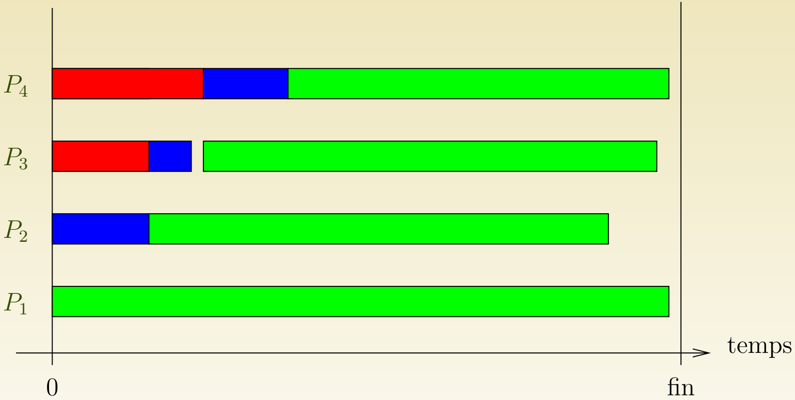
Two slaves i and $i+1$ with $T_i < T_{i+1}$.



We decrease α_{i+1} by ϵ.

Demonstration of lemma 1

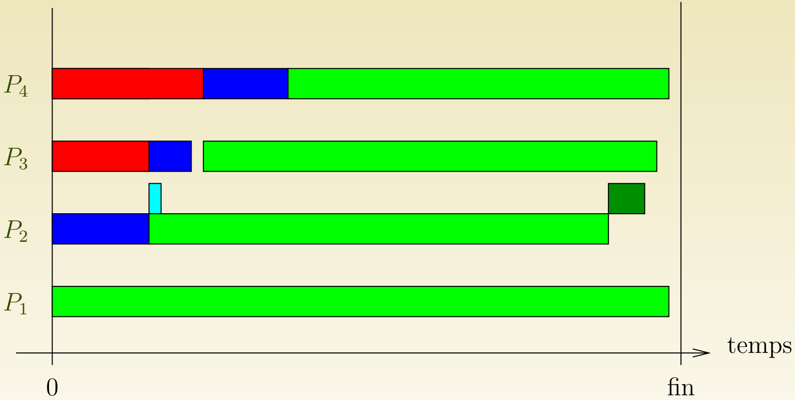
Two slaves i and $i+1$ with $T_i < T_{i+1}$.



We decrease α_{i+1} by ϵ.

Demonstration of lemma 1

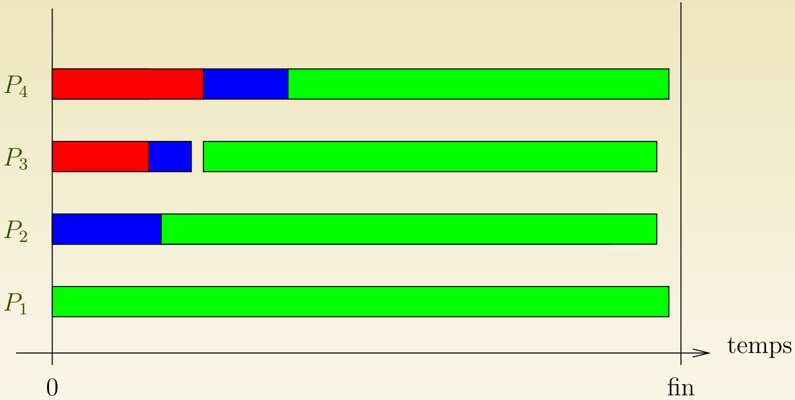
Two slaves i and $i+1$ with $T_i < T_{i+1}$.



We increase α_i by ϵ.

Demonstration of lemma 1

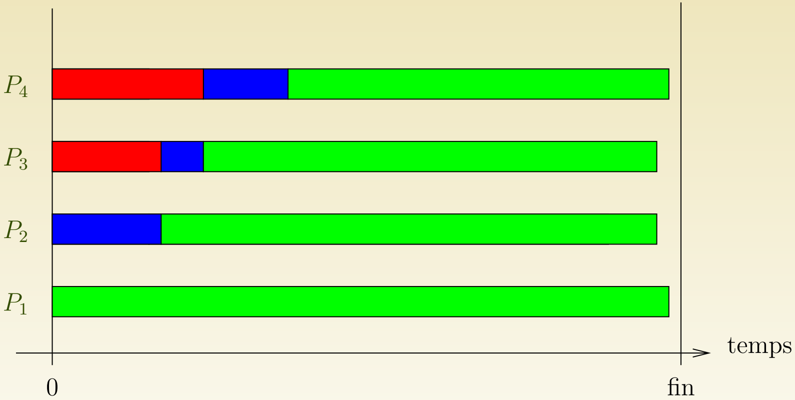
Two slaves i and $i+1$ with $T_i < T_{i+1}$.



We increase α_i by ϵ.

Demonstration of lemma 1

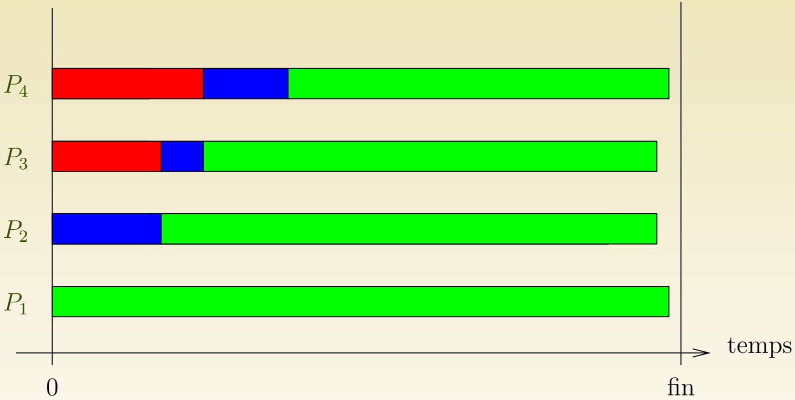
Two slaves i and $i+1$ with $T_i < T_{i+1}$.



The communication time for the following processors is unchanged.

Demonstration of lemma 1

Two slaves i and $i+1$ with $T_i < T_{i+1}$.



We end up with a better solution !

Demonstration of lemma 1 (continuation and conclusion)

- Ideal: $T'_i = T'_{i+1}$.
  We choose ϵ such that:

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

- The master stops before the slaves: absurde.
- The master stops after the slaves: we decrease P_1 by ϵ.

Lemma 2.

In an optimal solution all processors work.

Property for the selection of ressources

Lemma 2.

In an optimal solution all processors work.

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

Resolution

$$T = \alpha_1 W_{\mathsf{total}} w_1 .$$

Resolution

$T = \alpha_1 W_{\text{total}} w_1$.

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

Resolution

$T = \alpha_1 W_{\mathsf{total}} w_1$.

$T = \alpha_2(c + w_2)W_{\mathsf{total}}$. Therefore $\alpha_2 = \frac{w_1}{c+w_2}\alpha_1$.

$T = (\alpha_2 c + \alpha_3(c + w_3))W_{\mathsf{total}}$. Therefore $\alpha_3 = \frac{w_2}{c+w_3}\alpha_2$.

Resolution

$T = \alpha_1 W_{\text{total}} w_1$.

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

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

$\alpha_i = \frac{w_{i-1}}{c+w_i}\alpha_{i-1}$ for $i \geqslant 2$.

Resolution

$T = \alpha_1 W_{\text{total}} w_1$.

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

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

$\alpha_i = \frac{w_{i-1}}{c+w_i}\alpha_{i-1}$ for $i \geqslant 2$.

$\sum_{i=1}^{n} \alpha_i = 1$.

Resolution

$T = \alpha_1 W_{\text{total}} w_1$.

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

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

$\alpha_i = \frac{w_{i-1}}{c+w_i}\alpha_{i-1}$ for $i \geqslant 2$.

$\sum_{i=1}^{n} \alpha_i = 1$.

$$\alpha_1 \left(1 + \frac{w_1}{c + w_2} + \ldots + \prod_{k=2}^{j} \frac{w_{k-1}}{c + w_k} + \ldots \right) = 1$$

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.

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 $\alpha_i = \frac{1}{c+w_i}\frac{T}{W_{\text{total}}}$.

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}**:** $\alpha_i c W_{\text{total}} + \alpha_{i+1}(c + w_{i+1})W_{\text{total}} = T$.

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

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

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.

Choice of the master processor

We compare processors P_1 and P_2.

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

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, $\alpha_2 = \frac{1}{c+w_2} \frac{T}{W_{\text{total}}}$.

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:

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

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:

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

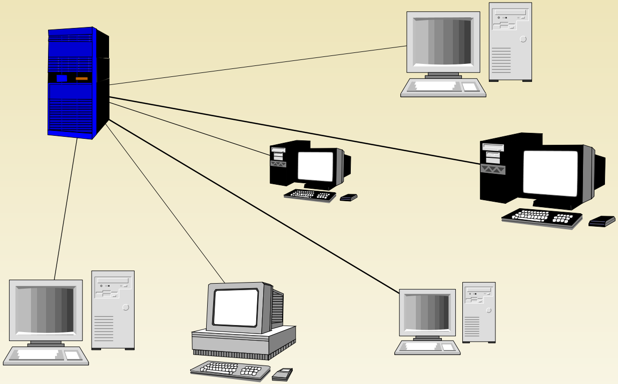
Minimal when $w_1 < w_2$.
Master = the most powerfull processor (for computations).

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.

Outline

1. The context
2. Bus-like network: classical resolution
3. Bus-like network: resolution under the divisible load model
4. Star-like network
5. With return messages
6. Multi-round algorithms
7. 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

Notations

- A set P_1, ..., P_p of processors
- P_1 is the master processor: initially, it holds all the data.

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_{total}.

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_{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: $n_i w_i$.

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_{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.

Star network and linear cost model

Goal : maximize the number of processed tasks within a time-bound T_f : $\sum \alpha_i$.

Star network and linear cost model

Goal : maximize the number of processed tasks within a time-bound T_f : $\sum \alpha_i$.

Lemma 3.

In any optimal solution of the $\textsc{StarLinear}$ problem, all workers participate in the computation, and they all finish computing simultaneously.

Star network and linear cost model

Goal : maximize the number of processed tasks within a time-bound T_f : $\sum \alpha_i$.

Lemma 3.

In any optimal solution of the StarLinear problem, all workers participate in the computation, and they all finish computing simultaneously.

Lemma 4.

An optimal ordering for the StarLinear problem is obtained by serving the workers in the ordering of non decreasing link capacities c_i.

Sketch of the proof of Lemma 3

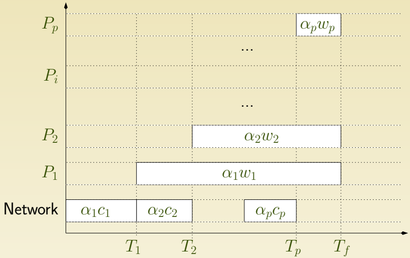
Two steps :

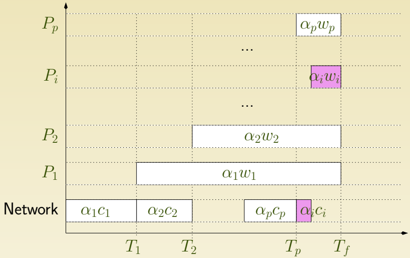
- All workers participate in the computation...

Sketch of the proof of Lemma 3

Two steps :

- All workers participate in the computation. . . otherwise it would not be optimal.





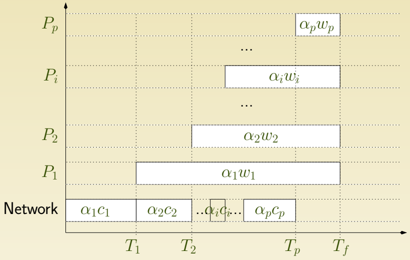
Two steps :

- All workers participate in the computation... otherwise it would not be optimal.
- All processors finish their work at the same time.

Sketch of the proof of Lemma 3

Two steps :

- All workers participate in the computation... otherwise it would not be optimal.
- All processors finish their work at the same time.

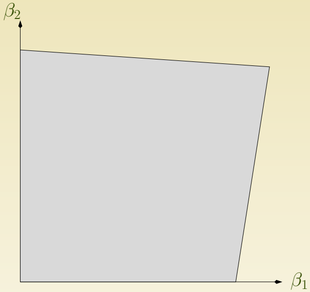


$$\text{MAXIMIZE } \sum \beta_i,$$
$$\text{SUBJECT TO}$$
$$\begin{cases} \text{LB}(i) & \forall i, \quad \beta_i \geqslant 0 \\ \text{UB}(i) & \forall i, \quad \sum_{k=1}^{i} \beta_k c_k + \beta_i w_i \leqslant T_f \end{cases}$$

Sketch of the proof of Lemma 3

Two steps :

- All workers participate in the computation... otherwise it would not be optimal.
- All processors finish their work at the same time.

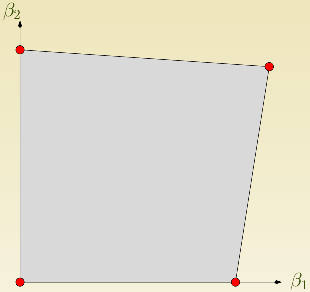


$$\text{MAXIMIZE} \sum \beta_i,$$
$$\text{SUBJECT TO}$$
$$\begin{cases} \mathsf{LB}(i) & \forall i, \quad \beta_i \geqslant 0 \\ \mathsf{UB}(i) & \forall i, \quad \sum_{k=1}^{i} \beta_k c_k + \beta_i w_i \leqslant T_f \end{cases}$$

Sketch of the proof of Lemma 3

Two steps :

- All workers participate in the computation. . . otherwise it would not be optimal.
- All processors finish their work at the same time.

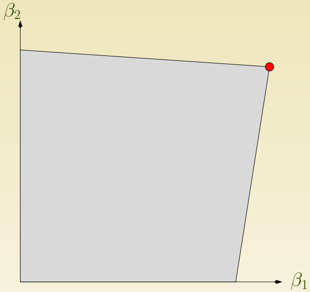


$$
\begin{array}{l}
\textsc{Maximize} \sum \beta_i, \\
\textsc{subject to} \\
\left\{ \begin{array}{lll}
\mathsf{LB}(i) & \forall i, & \beta_i \geqslant 0 \\
\mathsf{UB}(i) & \forall i, & \sum_{k=1}^{i} \beta_k c_k + \beta_i w_i \leqslant T_f
\end{array} \right.
\end{array}
$$

Sketch of the proof of Lemma 3

Two steps :

- All workers participate in the computation... otherwise it would not be optimal.
- All processors finish their work at the same time.

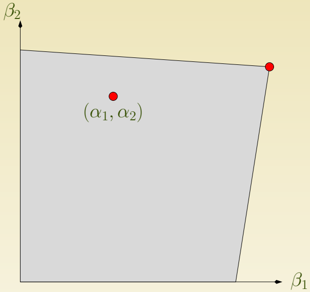


$$
\begin{array}{l}
\textsc{Maximize } \sum \beta_i, \\
\textsc{subject to} \\
\left\{ \begin{array}{lll} \mathsf{LB}(i) & \forall i, & \beta_i \geqslant 0 \\ \mathsf{UB}(i) & \forall i, & \sum_{k=1}^{i} \beta_k c_k + \beta_i w_i \leqslant T_f \end{array} \right.
\end{array}
$$

Sketch of the proof of Lemma 3

Two steps :

- All workers participate in the computation. . . otherwise it would not be optimal.
- All processors finish their work at the same time.

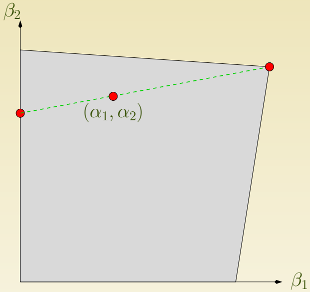


$$
\begin{array}{l}
\textsc{Maximize } \sum \beta_i, \\
\textsc{subject to} \\
\left\{ \begin{array}{lll} \mathsf{LB}(i) & \forall i, & \beta_i \geqslant 0 \\ \mathsf{UB}(i) & \forall i, & \sum_{k=1}^{i} \beta_k c_k + \beta_i w_i \leqslant T_f \end{array} \right.
\end{array}
$$

Sketch of the proof of Lemma 3

Two steps :

- All workers participate in the computation... otherwise it would not be optimal.
- All processors finish their work at the same time.



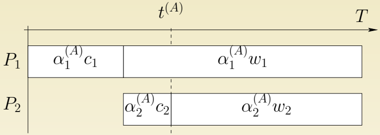
$$\textsc{Maximize } \sum \beta_i,$$
$$\textsc{subject to}$$
$$\begin{cases} \mathsf{LB}(i) & \forall i, \quad \beta_i \geqslant 0 \\ \mathsf{UB}(i) & \forall i, \quad \sum_{k=1}^{i} \beta_k c_k + \beta_i w_i \leqslant T_f \end{cases}$$

Sketch of the proof of Lemma 4

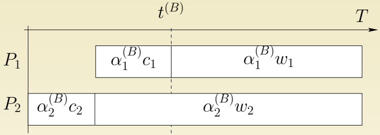
The proof is based on the comparison of the amount of work that is performed by the first two workers, and then proceeds by induction.

Sketch of the proof of Lemma 4

The proof is based on the comparison of the amount of work that is performed by the first two workers, and then proceeds by induction.



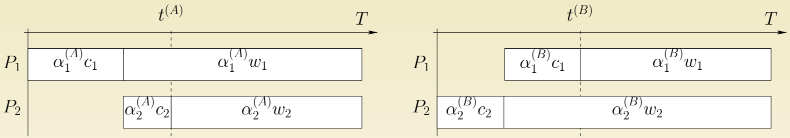
(A) P_1 starts before P_2



(B) P_2 starts before P_1

Sketch of the proof of Lemma 4

The proof is based on the comparison of the amount of work that is performed by the first two workers, and then proceeds by induction.

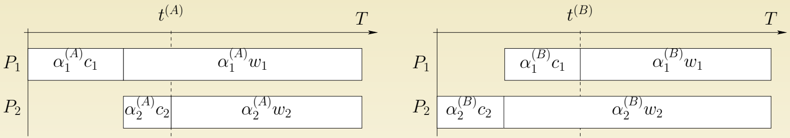


(A) P_1 starts before P_2 (B) P_2 starts before P_1

$$t^{(A)} = t^{(B)}, \tag{1}$$

Sketch of the proof of Lemma 4

The proof is based on the comparison of the amount of work that is performed by the first two workers, and then proceeds by induction.



(A) P_1 starts before P_2 (B) P_2 starts before P_1

$$t^{(A)} = t^{(B)}, \tag{1}$$

and

$$\left(\alpha_1^{(A)} + \alpha_2^{(A)}\right) - \left(\alpha_1^{(B)} + \alpha_2^{(B)}\right) = \frac{T(c_2 - c_1)}{(c_1 + w_1)(c_2 + w_2)}. \tag{2}$$

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

Outline

1 The context

2 Bus-like network: classical resolution

3 Bus-like network: resolution under the divisible load model

4 Star-like network

5 With return messages

6 Multi-round algorithms

7 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, ..., P_p of processors

Notations

- A set P_1, ..., P_p of processors
- P_1 is the master processor: initially, it holds all the data.

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_{total}.

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_{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: $n_i w_i$.

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_{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.**

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_{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 ?

Solutions with idle time ?

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

  Not interesting !

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 bask to the master ?

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 bask to the master ?

  Mandatory if the communication link is not available.

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 bask 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

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 $c_i = c$.

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).

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.

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.:

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).

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).

(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.

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).

Linear program for a given scenario (1)

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;

Linear program for a given scenario (1)

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;

Linear program for a given scenario (1)

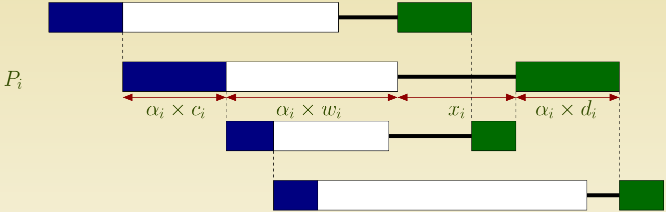
A scenario is described by:

- 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:

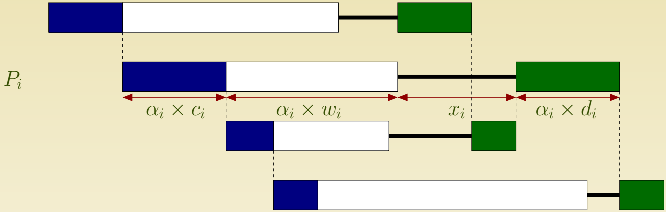
- the master sends the data as soon as possible;
- the slaves start working as soon as possible;
- the slaves send their as late as possible.

Linear program for a given scenario (2)



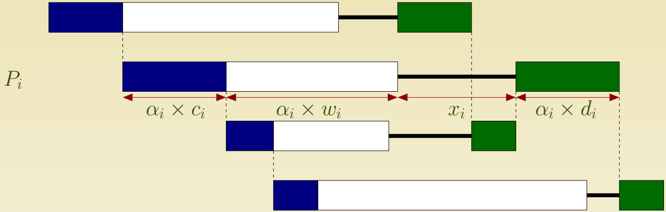
Consider slave P_i:

Linear program for a given scenario (2)



Consider slave P_i:

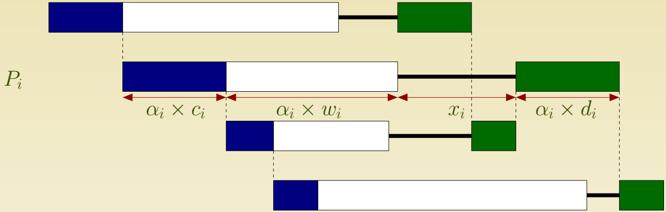
- it starts receiving data at time $t_i^{\text{recv}} = \sum_{j=1}^{i-1} \alpha_j \times c_j$



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 $t_i^{\text{recv}} + \alpha_i \times c_i$

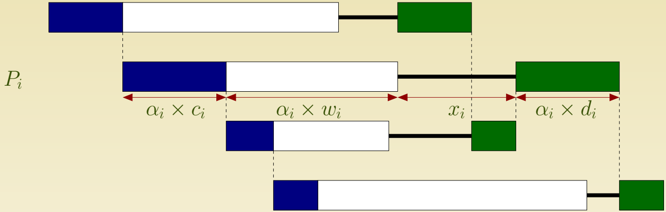
Linear program for a given scenario (2)



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 $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$

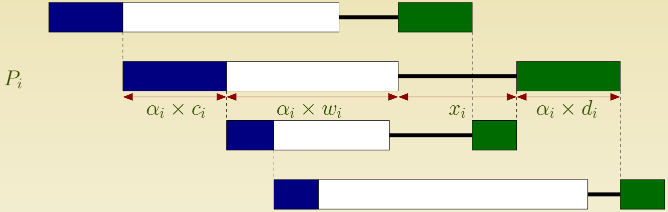
Linear program for a given scenario (2)



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 $t_i^{\text{back}} = T - \sum_{j \text{ successor of } i} \alpha_j \times d_j$

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: $x_i = t_i^{\text{back}} - t_i^{\text{term}} \geqslant 0$

Linear program for a given scenario (3)

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

$$\text{MAXIMIZE } \sum_i \alpha_i, \text{ UNDER THE CONSTRAINTS}$$
$$\begin{cases} \alpha_i \geqslant 0 \\ t_i^{\mathsf{back}} - t_i^{\mathsf{term}} \geqslant 0 \end{cases} \tag{3}$$

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

Linear program for a given scenario (3)

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

$$\textsc{Maximize } \sum_i \alpha_i, \textsc{ under the constraints}$$
$$\begin{cases} \alpha_i \geqslant 0 \\ t_i^{\text{back}} - t_i^{\text{term}} \geqslant 0 \end{cases} \tag{3}$$

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

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

$$\textsc{Minimize } T, \textsc{ under the constraints}$$
$$\begin{cases} \alpha_i \geqslant 0 \\ \sum_i \alpha_i = W \\ t_i^{\text{back}} - t_i^{\text{term}} \geqslant 0 \end{cases} \tag{4}$$

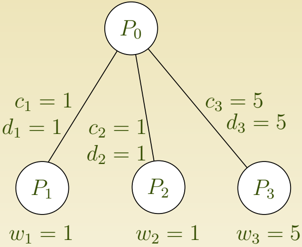
- Minimal time, an ordering and the resource selection being given.

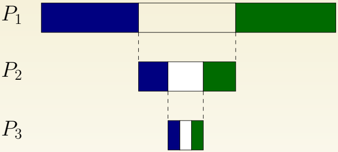
Linear program for a given scenario (4)

One cannot test all possible configurations

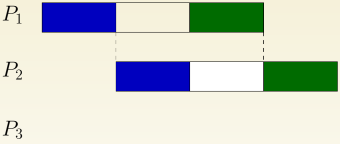
- Even if we decide that the order of return messages should be the same than the order of data distribution messages (FIFO), there still is an exponential number of scenarios to be tested.

All processors do not always participate



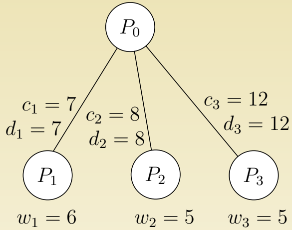


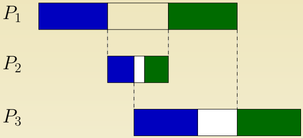
LIFO, throughput $\rho = 61/135$
(best schedule
with 3 processors)



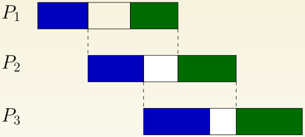
FIFO with 2 processors,
optimal throughput$\rho = 1/2$

The optimal schedule may be neither LIFO nor FIFO

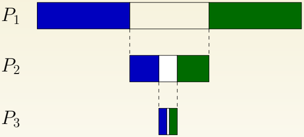




Optimal schedule
$(\rho = 38/499 \approx 0.076)$



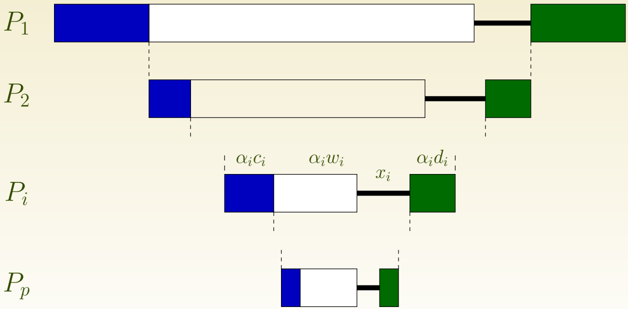
Best FIFO schedule
$(\rho = 47/632 \approx 0.074)$



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

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.

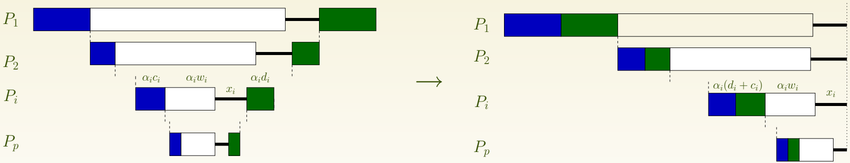


Theorem 1.

In the best LIFO solution:

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

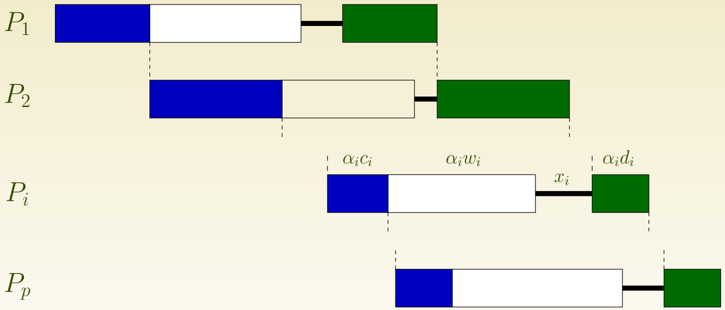
Demonstration: We change the platform: $c_i \leftarrow c_i + d_i$ and $d_i \leftarrow 0$



$\Rightarrow$ 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 $d_i = z \times c_i$ $(z < 1)$

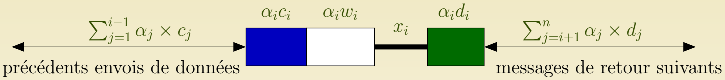
FIFO strategies (2)

Theorem 2.

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.

We consider i in the schedule:



$$\sum_{j=1}^{i} \alpha_i \times c_i + \alpha_i \times w_i + \sum_{j=i}^{n} \alpha_i \times d_i + x_i = T$$

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

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

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

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

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 $c_1 \leqslant c_2 \leqslant c_3 \ldots \leqslant c_n$;
- show that the throughput is best when the only working processors are the one satisfying $d_i \leqslant \dfrac{1}{\rho_{\text{opt}}}$

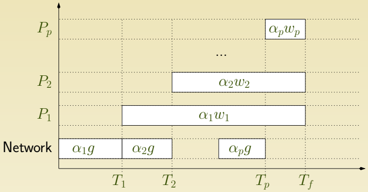
FIFO strategies — special cases

- 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).

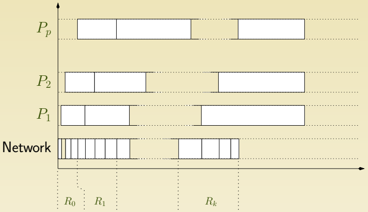
Outline

1. The context
2. Bus-like network: classical resolution
3. Bus-like network: resolution under the divisible load model
4. Star-like network
5. With return messages
6. Multi-round algorithms
7. Conclusion

One round vs. multi-round

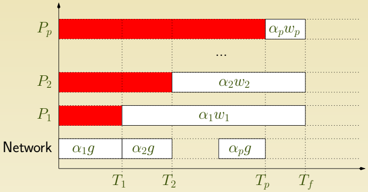


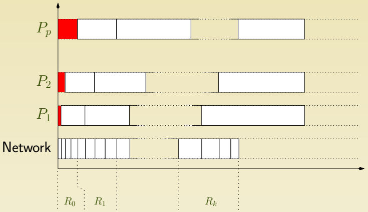
One round



Multi-round

One round vs. multi-round





One round

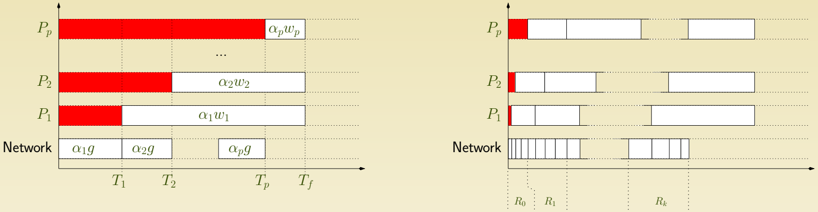
$\leadsto$ long idle-times

Multi-round

Efficient when W_{total} large

Intuition: start with small rounds, then increase chunks.

Problems :



One round
$\rightsquigarrow$ long idle-times

Multi-round
Efficient when W_{total} large

Intuition: start with small rounds, then increase chunks.
Problems :

- linear communication model leads to absurd solution
- resource selection
- number of rounds
- size of each round

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_{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 message of size α_i P_1 to P_i: $L_i + c_i \times \alpha_i$.**
  One-port model: P_1 sends and receives a *single* message at a time.

Complexity

Definition: One round, $\forall i, c_i = 0$.

Given W_{total}, p workers, $(P_i)_{1 \leqslant i \leqslant p}$, $(L_i)_{1 \leqslant i \leqslant p}$, and a rational number $T \geqslant 0$, and assuming that bandwidths are infinite, is it possible to compute all W_{total} load units within T time units?

Theorem 3.

The problem with one-round and infinite bandwidths is NP-complete.

Definition: One round, $\forall i, c_i = 0$.

Given W_{total}, p workers, $(P_i)_{1 \leqslant i \leqslant p}$, $(L_i)_{1 \leqslant i \leqslant p}$, and a rational number $T \geqslant 0$, and assuming that bandwidths are infinite, is it possible to compute all W_{total} load units within T time units?

Theorem 3.

The problem with one-round and infinite bandwidths is NP-complete.

What is the complexity of the general problem with finite bandwidths and several rounds ?

Complexity

Definition: One round, $\forall i, c_i = 0$.

Given W_{total}, p workers, $(P_i)_{1 \leqslant i \leqslant p}$, $(L_i)_{1 \leqslant i \leqslant p}$, and a rational number $T \geqslant 0$, and assuming that bandwidths are infinite, is it possible to compute all W_{total} load units within T time units?

Theorem 3.

The problem with one-round and infinite bandwidths is NP-complete.

What is the complexity of the general problem with finite bandwidths and several rounds ?

The general problem is NP-hard, but does not appear to be in NP (no polynomial bound on the number of activations).

Fixed activation sequence

Hypotheses

1. Number of activations : N_{act};
2. Whether P_i is **the** processor used during activation j : $\chi_i^{(j)}$

MINIMIZE T, UNDER THE CONSTRAINTS

$$\begin{cases} \displaystyle\sum_{j=1}^{N_{\text{act}}} \sum_{i=1}^{p} \chi_i^{(j)} \alpha_i^{(j)} = W_{\text{total}} \\ \forall k \leqslant N_{\text{act}}, \forall l : \left(\displaystyle\sum_{j=1}^{k} \sum_{i=1}^{p} \chi_i^{(j)} (L_i + \alpha_i^{(j)} c_i) \right) + \displaystyle\sum_{j=k}^{N_{\text{act}}} \chi_l^{(j)} \alpha_l^{(j)} w_l \leqslant T \\ \forall i, j : \alpha_i^{(j)} \geqslant 0 \end{cases} \tag{5}$$

Can be solved in polynomial time.

Fixed number of activations

MINIMIZE T, UNDER THE CONSTRAINTS

$$
\begin{cases}
\displaystyle\sum_{j=1}^{N_{\text{act}}} \sum_{i=1}^{p} \chi_i^{(j)} \alpha_i^{(j)} = W_{\text{total}} \\
\forall k \leqslant N_{\text{act}}, \forall l : \left(\displaystyle\sum_{j=1}^{k} \sum_{i=1}^{p} \chi_i^{(j)} (L_i + \alpha_i^{(j)} c_i) \right) + \displaystyle\sum_{j=k}^{N_{\text{act}}} \chi_l^{(j)} \alpha_l^{(j)} w_l \leqslant T \\
\forall k \leqslant N_{\text{act}} : \displaystyle\sum_{i=1}^{p} \chi_i^{(k)} \leqslant 1 \\
\forall i, j : \chi_i^{(j)} \in \{0, 1\} \\
\forall i, j : \alpha_i^{(j)} \geqslant 0
\end{cases} \tag{6}
$$

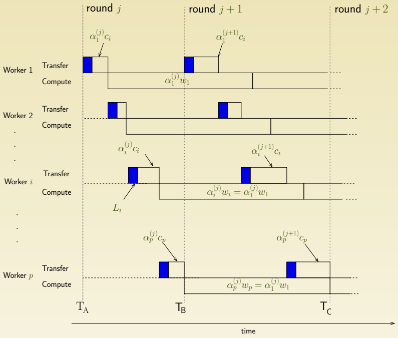
Exact but exponential

Can lead to branch-and-bound algorithms

In a round: all workers have same computation time

Geometrical increase of rounds size

No idle time in communications:



$$\alpha_i^{(j)} w_i = \sum_{k=1}^{p} (L_k + \alpha_k^{(j+1)} c_k).$$

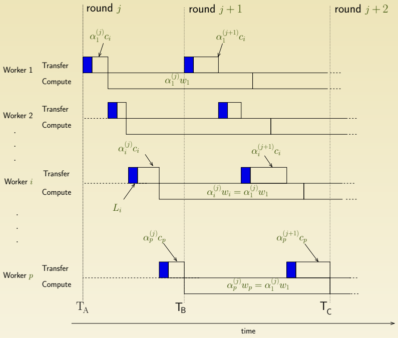
Heuristic processor selection: by decreasing bandwidths

Uniform multi-round

In a round: all workers have same computation time

Geometrical increase of rounds size

No idle time in communications:

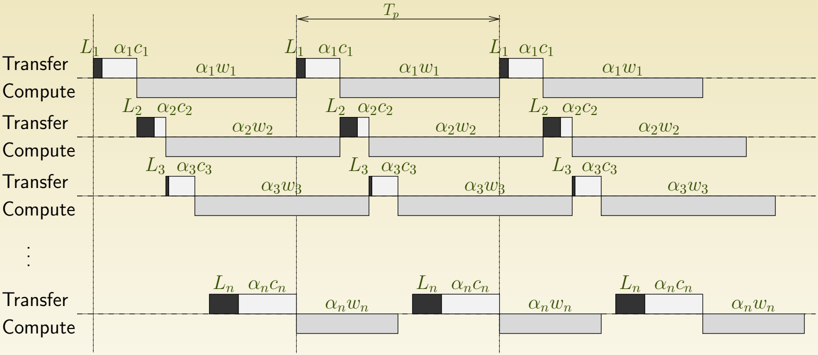


$$\alpha_i^{(j)} w_i = \sum_{k=1}^{p} (L_k + \alpha_k^{(j+1)} c_k).$$

Heuristic processor selection: by decreasing bandwidths

No guarantee...

Periodic schedule



How to choose T_p? Which resources to select?

With no overlap (1/4)

Equations

- Divide total execution time T into k periods of duration T_p.

With no overlap (1/4)

Equations

- Divide total execution time T into k periods of duration T_p.
- $\mathcal{I} \subset \{1, \ldots, p\}$ participating processors.

With no overlap (1/4)

Equations

- Divide total execution time T into k periods of duration T_p.
- $\mathcal{I} \subset \{1, \ldots, p\}$ participating processors.
- Bandwidth limitation:

$$\sum_{i \in \mathcal{I}} (L_i + \alpha_i c_i) \leqslant T_p.$$

With no overlap (1/4)

Equations

- Divide total execution time T into k periods of duration T_p.
- $\mathcal{I} \subset \{1, \ldots, p\}$ participating processors.
- Bandwidth limitation:

$$\sum_{i \in \mathcal{I}} (L_i + \alpha_i c_i) \leqslant T_p.$$

- No overlap:

$$\forall i \in \mathcal{I}, \qquad L_i + \alpha_i (c_i + w_i) \leqslant T_p.$$

With no overlap (2/4)

Normalization

- β_i average number of tasks processed by P_i during one time unit.

With no overlap (2/4)

Normalization

- β_i average number of tasks processed by P_i during one time unit.
- Linear program: $$\text{MAXIMIZE} \sum_{i=1}^{p} \beta_i \quad \begin{cases} \forall i \in \mathcal{I}, & \beta_i(c_i + w_i) \leqslant 1 - \frac{L_i}{T_p} \\ \sum_{i\in\mathcal{I}} \beta_i c_i \leqslant 1 - \frac{\sum_{i\in\mathcal{I}} L_i}{T_p} \end{cases} .$$

Normalization

- β_i average number of tasks processed by P_i during one time unit.
- Linear program: $$\begin{array}{l}\textsc{Maximize} \sum_{i=1}^{p} \beta_i \\ \begin{cases} \forall i \in \mathcal{I}, \quad \beta_i(c_i + w_i) \leqslant 1 - \frac{L_i}{T_p} \\ \sum_{i \in \mathcal{I}} \beta_i c_i \leqslant 1 - \frac{\sum_{i \in \mathcal{I}} L_i}{T_p} \end{cases}\end{array} .$$

Relaxed version

$$\begin{array}{l}\textsc{Maximize} \sum_{i=1}^{p} x_i \\ \begin{cases} \forall 1 \leqslant i \leqslant p, \quad x_i(c_i + w_i) \leqslant 1 - \frac{L_i}{T_p} \\ \sum_{i=1}^{p} x_i c_i \leqslant 1 - \frac{\sum_{i=1}^{p} L_i}{T_p} \end{cases}\end{array}$$

With no overlap (2/4)

Normalization

- β_i average number of tasks processed by P_i during one time unit.
- Linear program: $$\text{MAXIMIZE} \sum_{i=1}^{p} \beta_i \\ \begin{cases} \forall i \in \mathcal{I}, \quad \beta_i(c_i + w_i) \leqslant 1 - \frac{L_i}{T_p} \\ \sum_{i \in \mathcal{I}} \beta_i c_i \leqslant 1 - \frac{\sum_{i \in \mathcal{I}} L_i}{T_p} \end{cases} .$$

Relaxed version

$$\text{MAXIMIZE} \sum_{i=1}^{p} x_i \\ \begin{cases} \forall 1 \leqslant i \leqslant p, \quad x_i(c_i + w_i) \leqslant 1 - \frac{\sum_{i=1}^{p} L_i}{T_p} \\ \sum_{i=1}^{p} x_i c_i \leqslant 1 - \frac{\sum_{i=1}^{p} L_i}{T_p} \end{cases}$$

With no overlap (3/4)

Bandwidth-centric solution

- Sort: $c_1 \leqslant c_2 \leqslant \ldots \leqslant c_p$.
- Let q be the largest index so that $\sum_{i=1}^{q} \frac{c_i}{c_i+w_i} \leqslant 1$.
- If $q < p$, $\epsilon = 1 - \sum_{i=1}^{q} \frac{c_i}{c_i+w_i}$.
- Optimal solution to relaxed program:

$$\forall 1 \leqslant i \leqslant q, \quad x_i = \frac{1 - \frac{\sum_{i=1}^{p} L_i}{T_p}}{c_i + w_i}$$

and (if $q < p$):

$$x_{q+1} = \left(1 - \frac{\sum_{i=1}^{p} L_i}{T_p}\right) \left(\frac{\epsilon}{c_{q+1}}\right),$$

and $x_{q+2} = x_{q+3} = \ldots = x_p = 0$.

With no overlap (4/4)

Asymptotic optimality

- Let $T_p = \sqrt{T^*_{\max}}$ and $\alpha_i = x_i T_p$ for all i.

Asymptotic optimality

- Let $T_p = \sqrt{T^*_{\max}}$ and $\alpha_i = x_i T_p$ for all i.
- Then $T \leqslant T^*_{\max} + O(\sqrt{T^*_{\max}})$.

With no overlap (4/4)

Asymptotic optimality

- Let $T_p = \sqrt{T^*_{\max}}$ and $\alpha_i = x_i T_p$ for all i.
- Then $T \leqslant T^*_{\max} + O(\sqrt{T^*_{\max}})$.
- Closed-form expressions for resource selection and task assignment provided by the algorithm.

With overlap

Key points

- Still sort resources according to the c_i.
- Greedily select resources until the sum of the ratios $\frac{c_i}{w_i}$ $\left(\text{instead of } \frac{c_i}{c_i+w_i}\right)$ exceeds 1.

Outline

1. The context
2. Bus-like network: classical resolution
3. Bus-like network: resolution under the divisible load model
4. Star-like network
5. With return messages
6. Multi-round algorithms
7. 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.