Le modèle des tâches divisibles (*divisible load theory*)

Frédéric Vivien

e-mail: Frederic.Vivien@ens-lyon.fr
Bureau 333

30 septembre et 7 octobre 2005

Plan du cours

Plan du cours

Tour d'horizon du parallélisme

- cf. le cours de Jean-Yves L'Excellent : « Algorithmique numérique parallèle ».

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.
  - Ordinateurs parallèles à mémoire distribuée.

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.
  - Ordinateurs parallèles à mémoire distribuée.
  - Grappes (*clusters*).

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.
  - Ordinateurs parallèles à mémoire distribuée.
  - Grappes (*clusters*).
  - Grappes hétérogènes.

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.
  - Ordinateurs parallèles à mémoire distribuée.
  - Grappes (*clusters*).
  - Grappes hétérogènes.
  - Grappes de grappes.

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.
  - Ordinateurs parallèles à mémoire distribuée.
  - Grappes (*clusters*).
  - Grappes hétérogènes.
  - Grappes de grappes.
  - Réseau de stations de travail.

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.
  - Ordinateurs parallèles à mémoire distribuée.
  - Grappes (*clusters*).
  - Grappes hétérogènes.
  - Grappes de grappes.
  - Réseau de stations de travail.
  - La grille.

Contexte de travail

- Calcul scientifique : gros besoins en calcul ou de capacité de stockage.
- Nécessité d'utiliser des « systèmes multiprocesseurs » :
  - Ordinateurs parallèles à mémoire partagée.
  - Ordinateurs parallèles à mémoire distribuée.
  - Grappes (*clusters*).
  - Grappes hétérogènes.
  - Grappes de grappes.
  - Réseau de stations de travail.
  - La grille.
- Problématique : tenir compte de l'hétérogénéité au niveau algorithmique.

Nouvelles plates-formes, nouveaux problèmes

Plates-formes d'exécution : Plates-formes distribuées hétérogènes (réseau de stations de travail, clusters, clusters de clusters, grilles, etc.)

De nouvelles sources de problèmes

- Hétérogénéité des processeurs (puissance de calcul, mémoire, etc.)
- Hétérogénéité des liens de communications.
- Irrégularité du réseau d'interconnexion.
- Plates-formes non-dédiées.

Nouvelles plates-formes, nouveaux problèmes

Plates-formes d'exécution : Plates-formes distribuées hétérogènes (réseau de stations de travail, clusters, clusters de clusters, grilles, etc.)

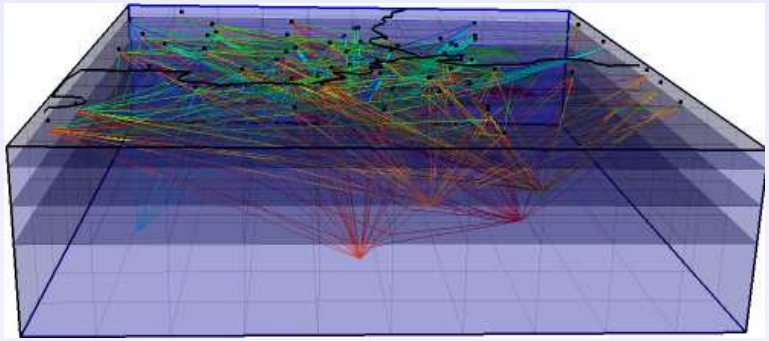
De nouvelles sources de problèmes

- Hétérogénéité des processeurs (puissance de calcul, mémoire, etc.)
- Hétérogénéité des liens de communications.
- Irrégularité du réseau d'interconnexion.
- Plates-formes non-dédiées.

Il faut adapter les approches algorithmiques et les stratégies d'ordonnancement : nouveaux objectifs, nouveaux modèles, etc.

Exemple d'application : tomographie sismique de la terre

- Modélisation de la structure interne de la terre



- Validation du modèle par comparaison du temps de propagation dans le modèle d'une onde sismique par rapport à la réalité
- Ensemble des événements sismiques de l'année 1999 : 817101
- Code original écrit pour un ordinateur parallèle :

```
if (rank = ROOT)
   raydata ← read n lines from data file;
MPI_Scatter(raydata, n/P, ..., rbuff, ...,
            ROOT, MPI_COMM_WORLD);
compute_work(rbuff);
```

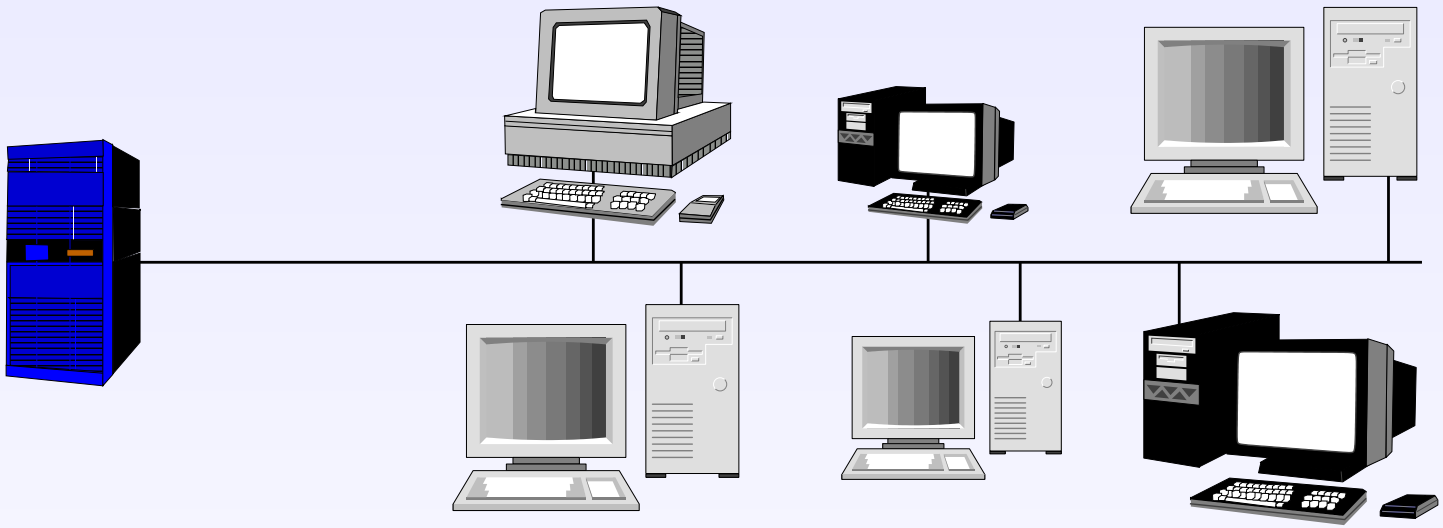
Applications concernées par le modèle des tâches divisibles

Application composée d'un très (très) grand nombre de calculs de faibles granularité.
Temps de calcul proportionnel au volume de données à traiter.

Calculs indépendants entre eux : pas de synchronisations ni de communications.

Plan du cours

Réseau en bus



- Les liaisons entre le maître et les esclaves ont toutes les mêmes caractéristiques.
- Les esclaves ont des puissances de calcul différentes.

Notations

- Un ensemble P_1, ..., P_p de processeurs

Notations

- Un ensemble P_1, ..., P_p de processeurs
- P_1 est le processeur maître : il détient initialement toutes les données.

Notations

- Un ensemble P_1, ..., P_p de processeurs
- P_1 est le processeur maître : il détient initialement toutes les données.
- Quantité totale de travail : W_{total}.

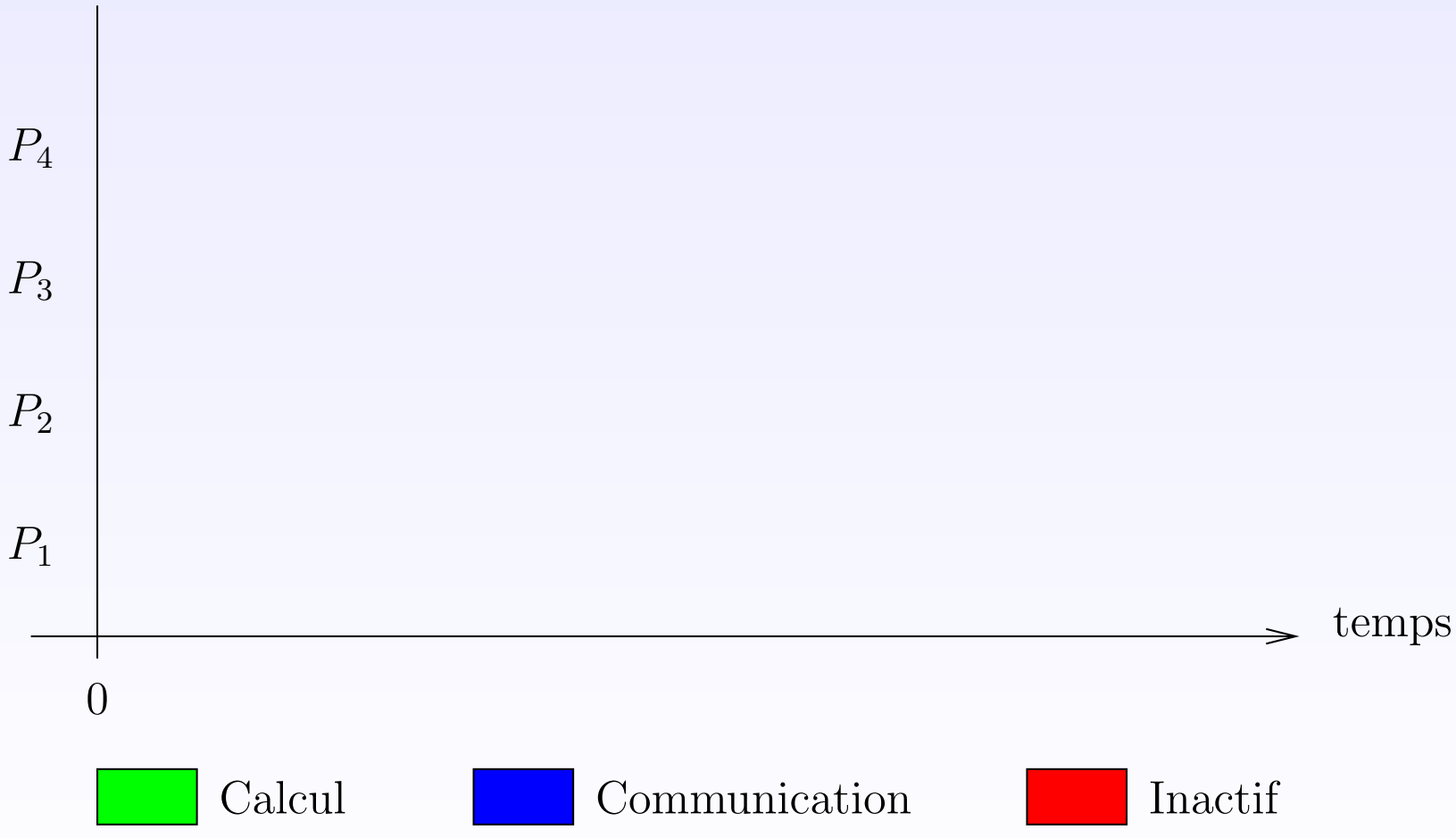
Notations

- Un ensemble P_1, ..., P_p de processeurs
- P_1 est le processeur maître : il détient initialement toutes les données.
- Quantité totale de travail : W_{total}.
- Le processeur P_i reçoit une quantité de travail : $n_i \in \mathbb{N}$ avec $\sum_i n_i = W_{\text{total}}$.
  Durée d'un calcul unitaire sur P_i : w_i.
  Temps de calcul sur P_i : $n_i w_i$.

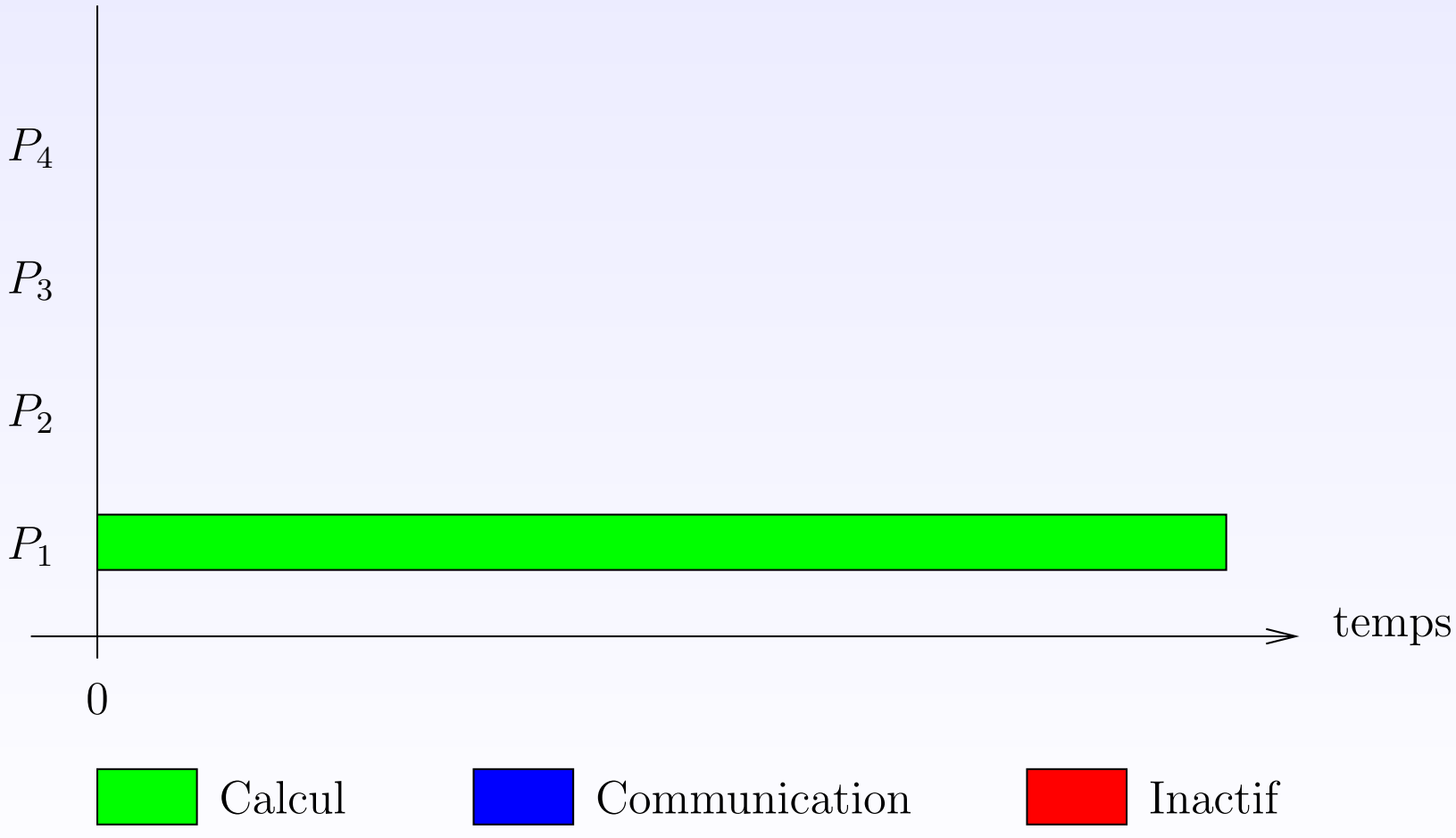
Notations

- Un ensemble P_1, ..., P_p de processeurs
- P_1 est le processeur maître : il détient initialement toutes les données.
- Quantité totale de travail : W_{total}.
- Le processeur P_i reçoit une quantité de travail : $n_i \in \mathbb{N}$ avec $\sum_i n_i = W_{\text{total}}$.
  Durée d'un calcul unitaire sur P_i : w_i.
  Temps de calcul sur P_i : $n_i w_i$.
- Durée de communication d'un message unitaire de P_1 à P_i : c.
  Bus un-port : P_1 envoie un *unique* message à la fois sur le bus, tous les processeurs communiquent à la même vitesse avec le maître.

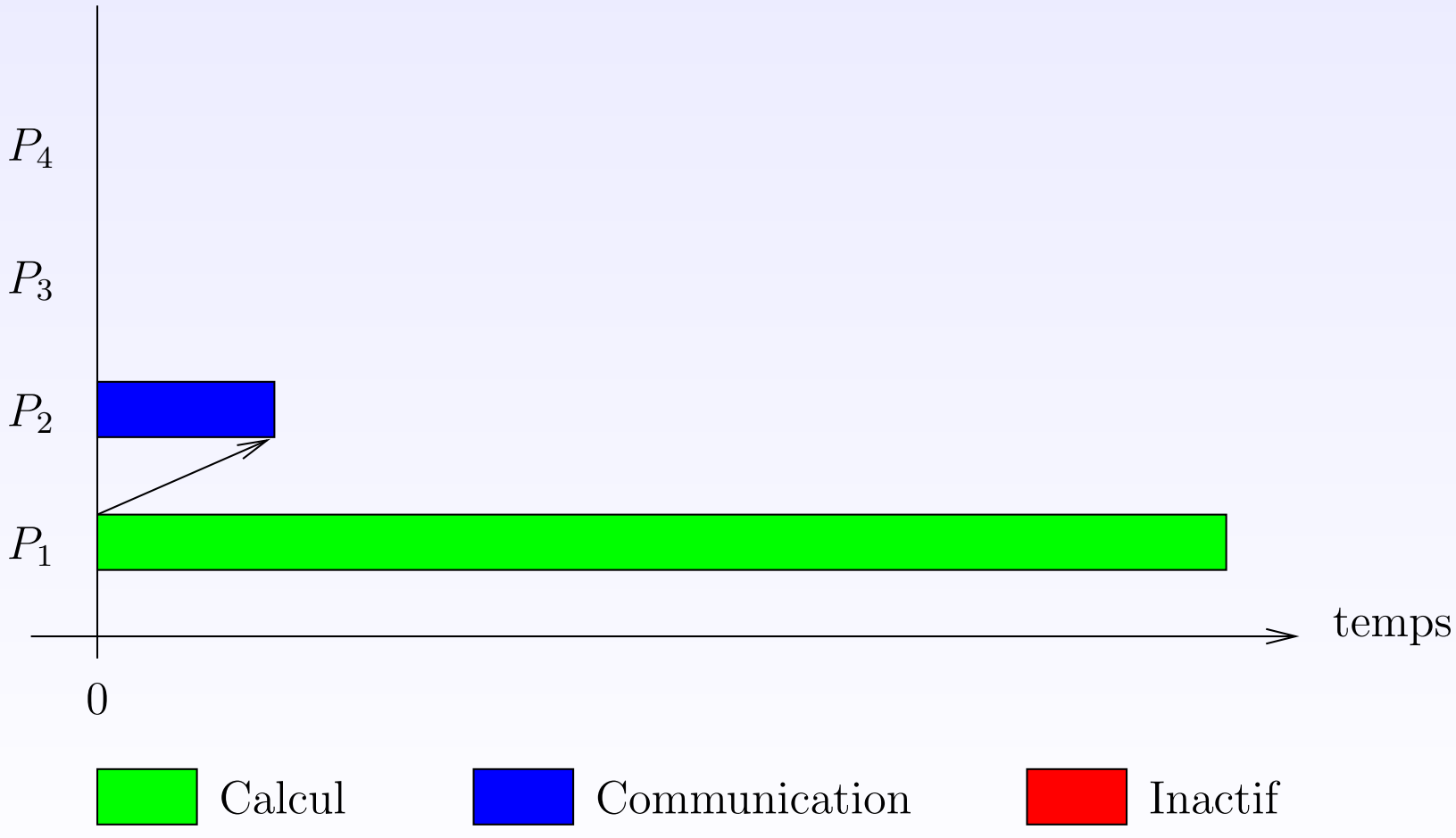
Comportement du maître et des esclaves (illustration)



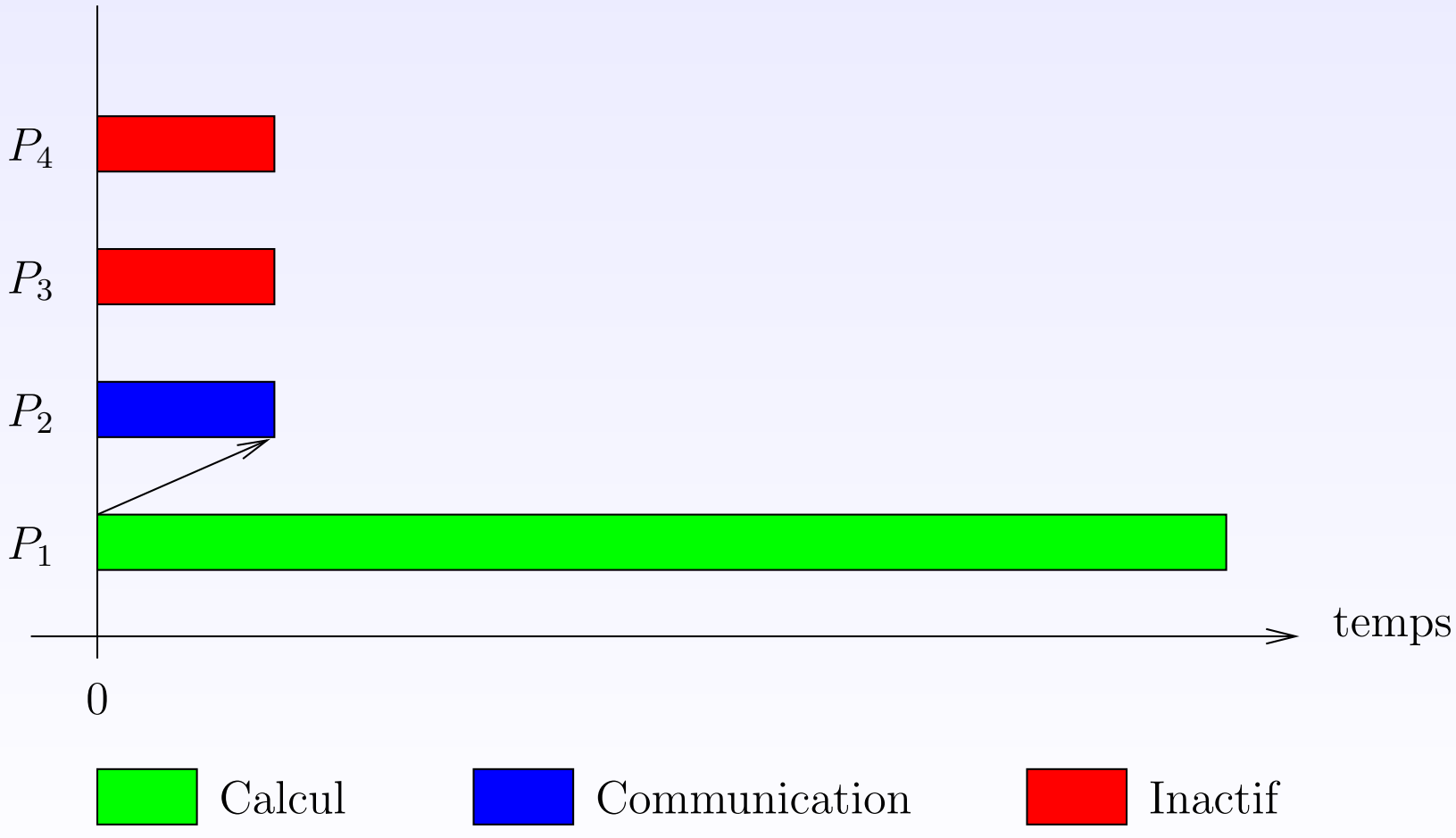
Comportement du maître et des esclaves (illustration)



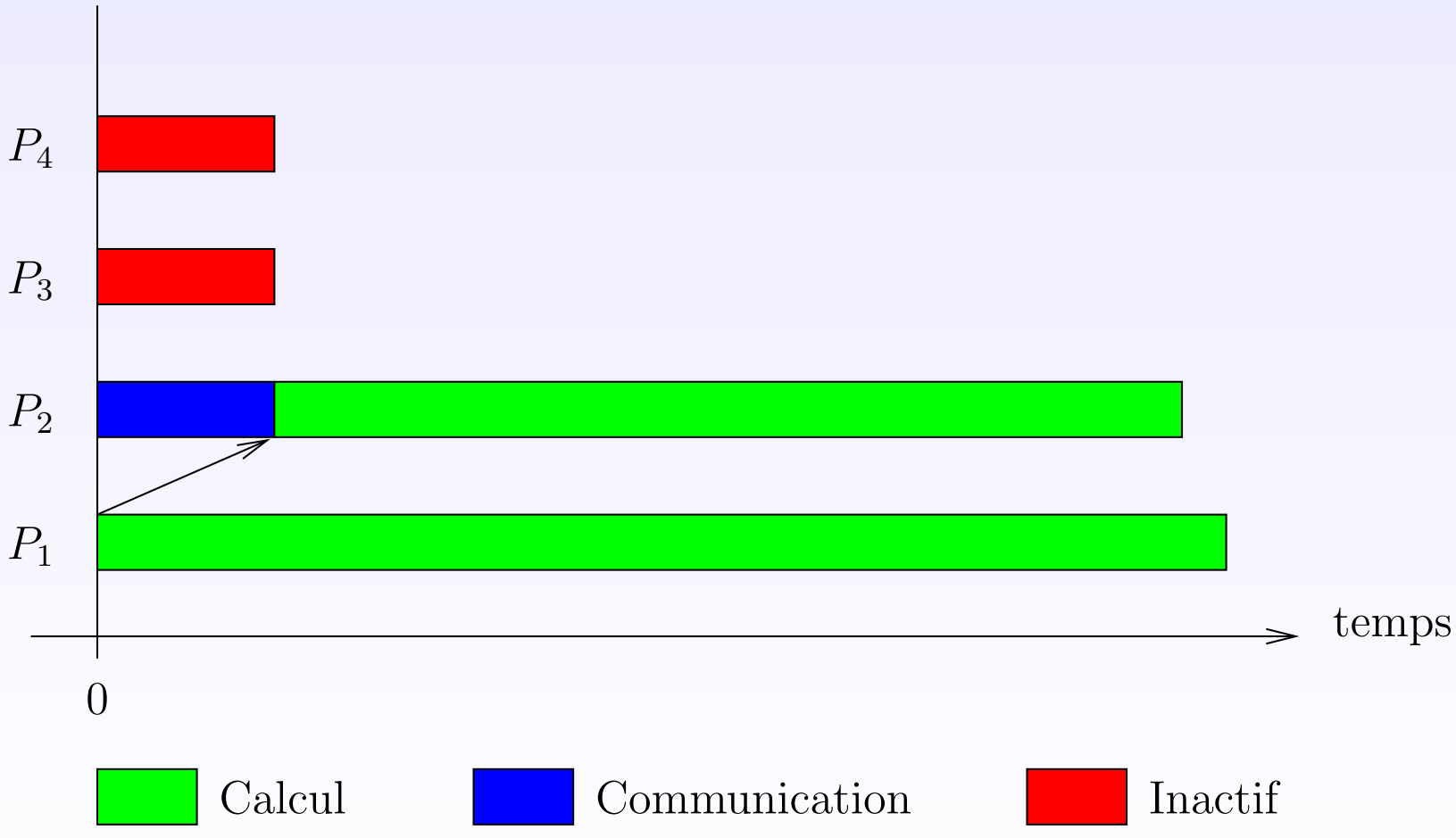
Comportement du maître et des esclaves (illustration)



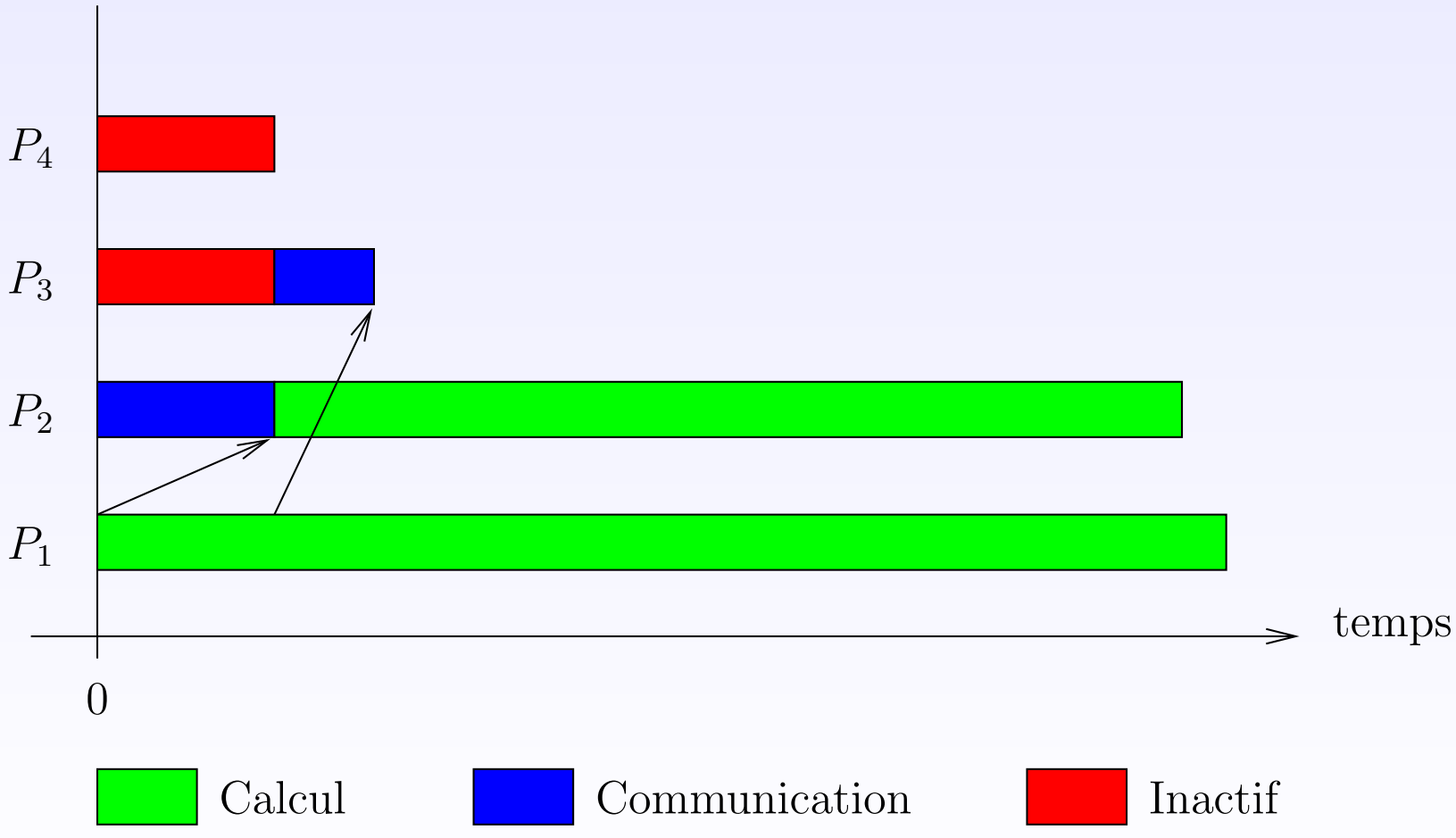
Comportement du maître et des esclaves (illustration)



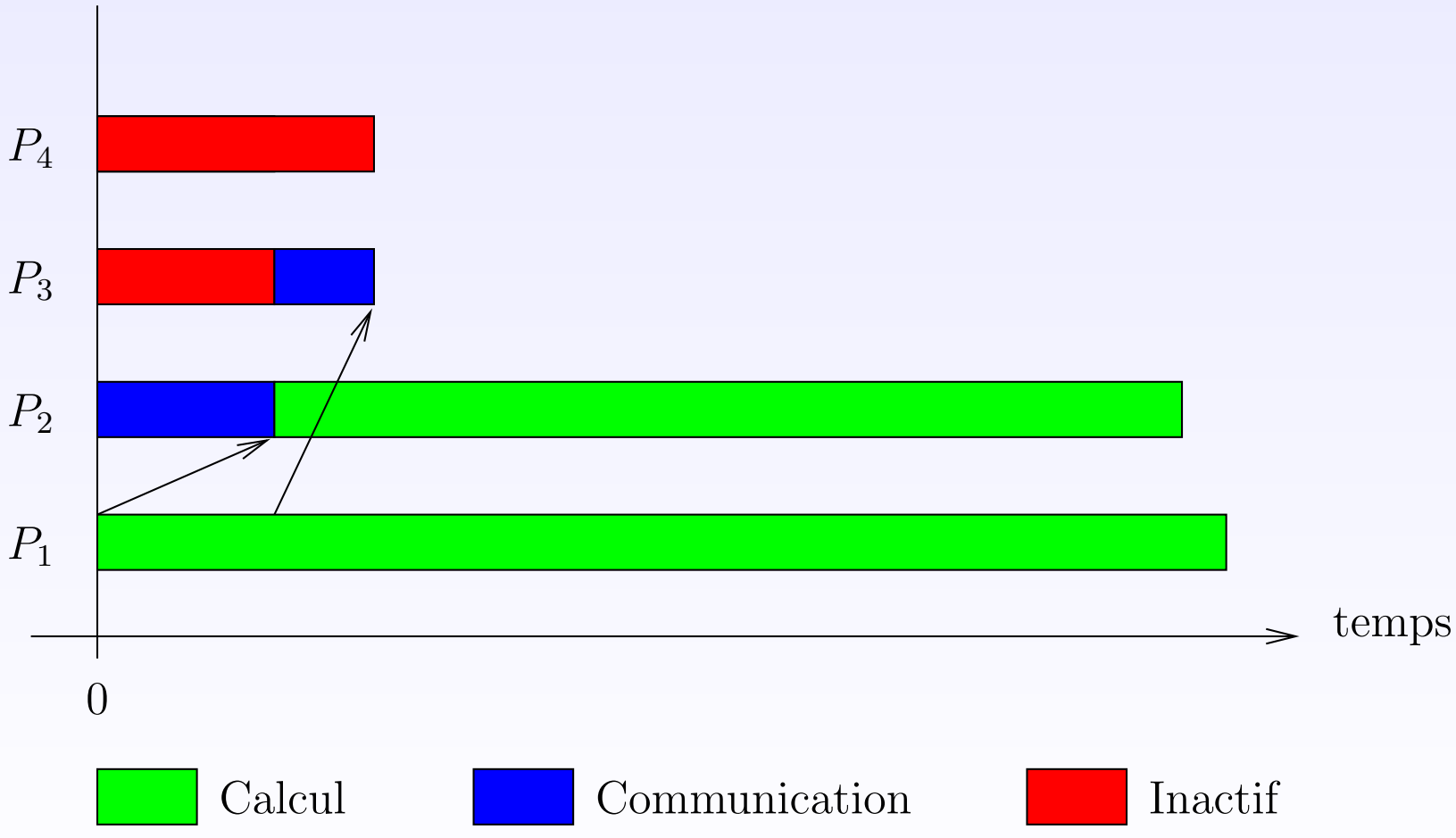
Comportement du maître et des esclaves (illustration)



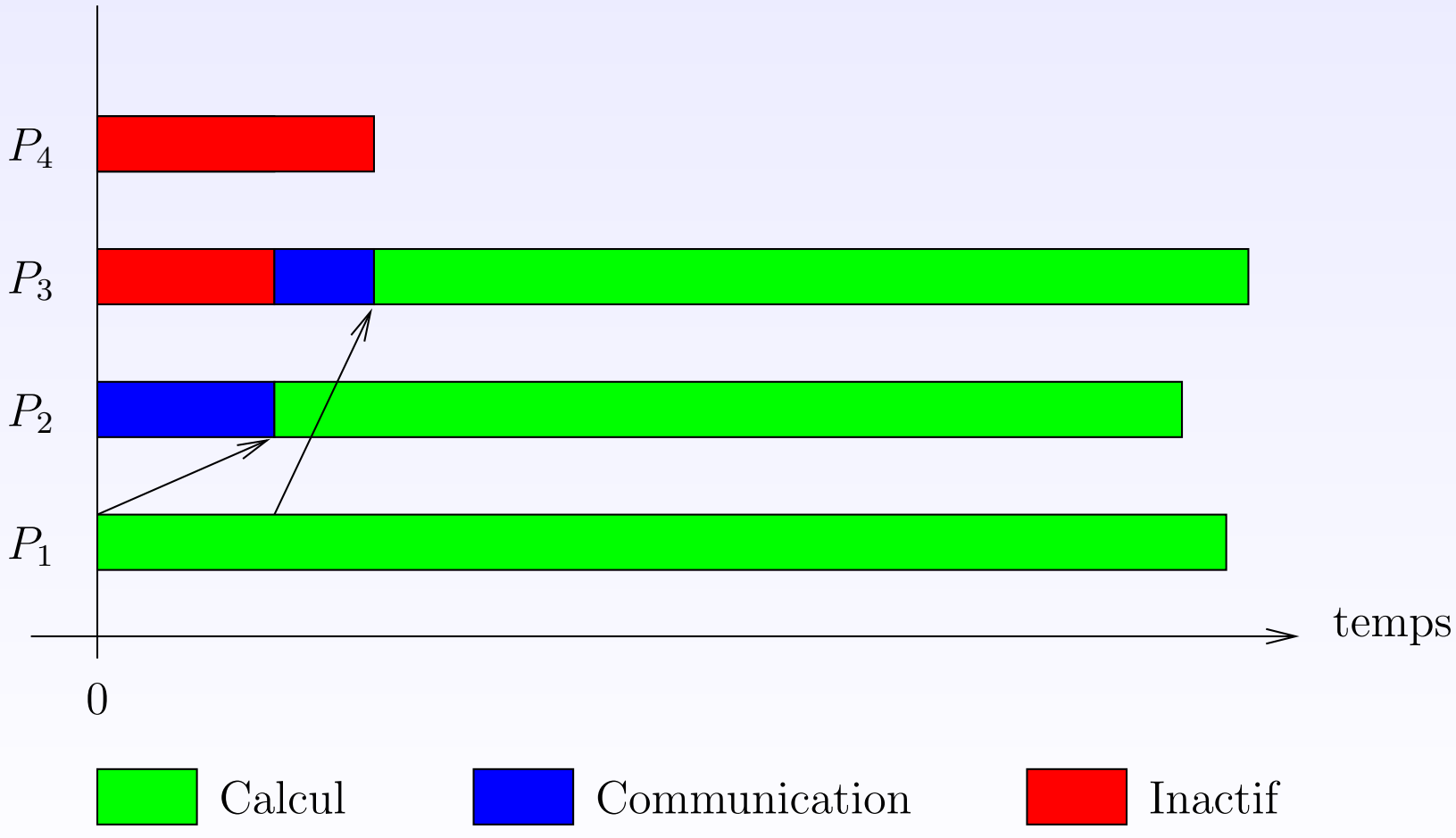
Comportement du maître et des esclaves (illustration)



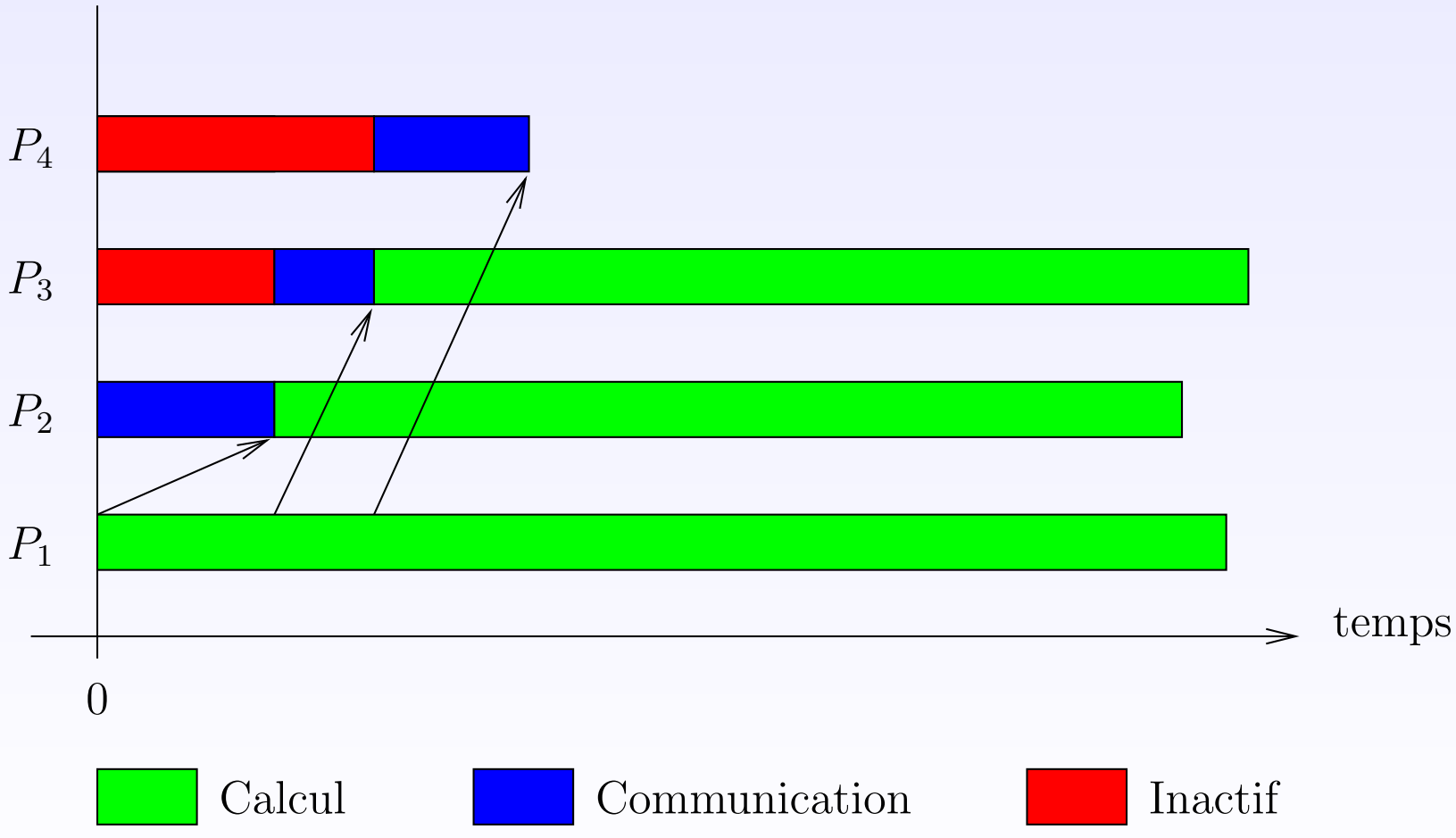
Comportement du maître et des esclaves (illustration)



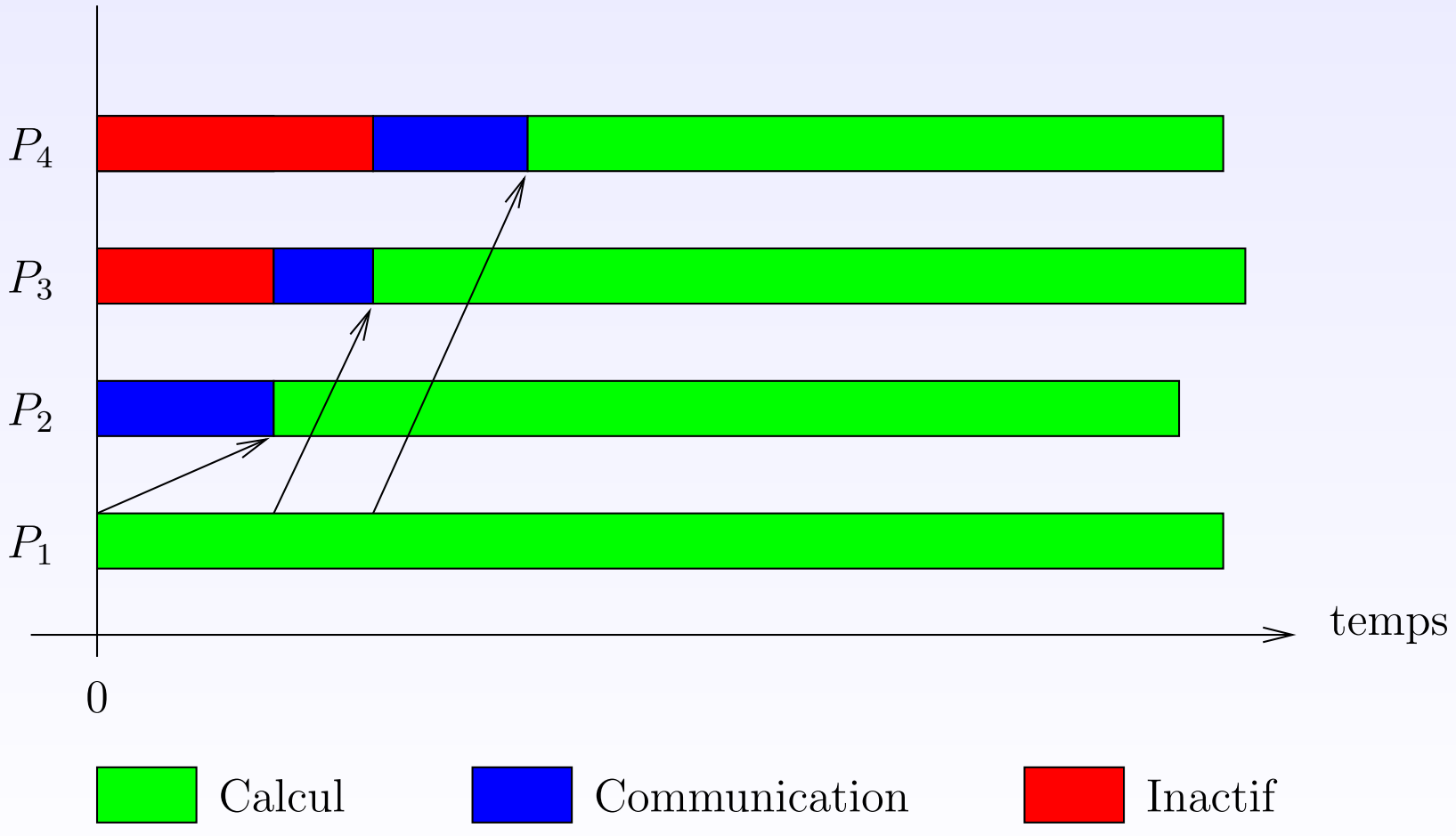
Comportement du maître et des esclaves (illustration)



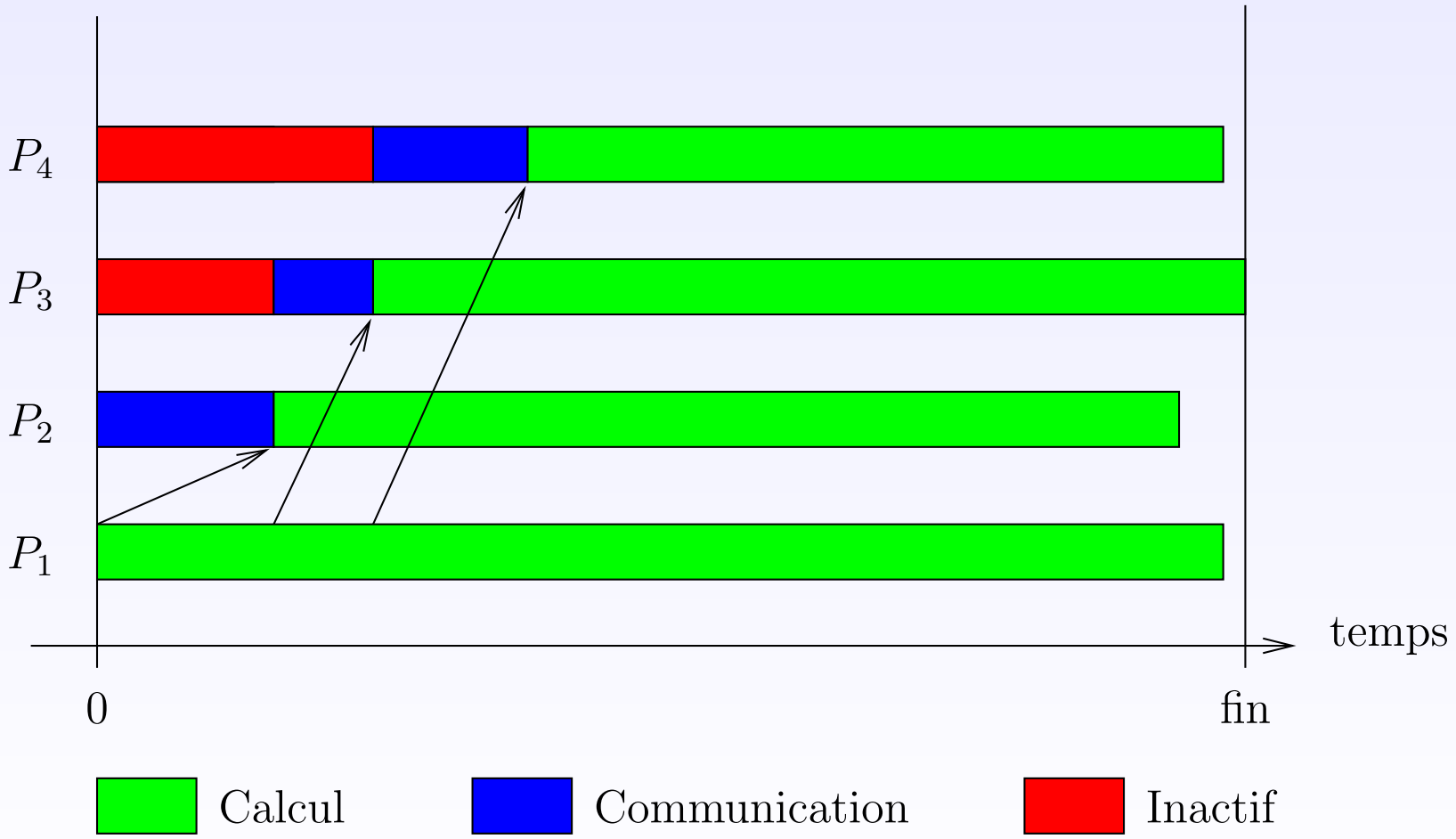
Comportement du maître et des esclaves (illustration)



Comportement du maître et des esclaves (illustration)



Comportement du maître et des esclaves (illustration)



Comportement du maître et des esclaves (hypothèses)

- Le maître envoie ses n_i données au processeur P_i en un seul envoi.

Comportement du maître et des esclaves (hypothèses)

- Le maître envoie ses n_i données au processeur P_i en un seul envoi.
- Le maître envoie leurs données aux processeurs tour à tour dans l'ordre P_2, ..., P_p.

Comportement du maître et des esclaves (hypothèses)

- Le maître envoie ses n_i données au processeur P_i en un seul envoi.
- Le maître envoie leurs données aux processeurs tour à tour dans l'ordre P_2, ..., P_p.
- Pendant ce temps, le maître traite ses n_1 données.

Comportement du maître et des esclaves (hypothèses)

- Le maître envoie ses n_i données au processeur P_i en un seul envoi.
- Le maître envoie leurs données aux processeurs tour à tour dans l'ordre P_2, ..., P_p.
- Pendant ce temps, le maître traite ses n_1 données.
- Un esclave ne commence à traiter ses données que lorsqu'il les a toutes reçues.

Mise en équations

- $P_1 : T_1 = n_1.w_1$

Mise en équations

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

Mise en équations

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

Mise en équations

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

Mise en équations

- $P_1 : T_1 = n_1.w_1$
- $P_2 : T_2 = n_2.c + n_2.w_2$
- $P_3 : T_3 = (n_2.c + n_3.c) + n_3.w_3$
- $P_i : T_i = \sum_{j=2}^{i} n_j.c + n_i.w_i$ pour $i \geq 2$
- $P_i : T_i = \sum_{j=1}^{i} n_j.c_j + n_i.w_i$ pour $i \geq 1$ avec $c_1 = 0$ et $c_j = c$ sinon.

Temps d'exécution

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

On cherche une distribution n_1, ..., n_p des données minimisant T.

Temps d'exécution : réécriture

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

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

Une solution optimale pour répartir W_{total} données entre p processeurs est obtenue en distribuant n_1 données au processeur P_1 et en répartissant optimalement $W_{\text{total}} - n_1$ données entre les processeurs P_2 à P_p.

Algorithme

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

Complexité

- **Théorique**

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

- **En pratique**
  Si $W_{\text{total}} = 817101$ et $p = 16$, sur un Pentium III à 933MHz : plus de deux jours...
  (Version optimisée 6 minutes.)

Inconvénients

- Coût
- Solution non réutilisable
- Solution partielle

Inconvénients

- Coût
- Solution non réutilisable
- Solution partielle

On n'a pas besoin d'une solution aussi précise

Plan du cours

Notations

- Un ensemble P_1, ..., P_p de processeurs
- P_1 est le processeur maître : il détient initialement toutes les données.
- Quantité totale de travail : W_{total}.
- **Le processeur P_i reçoit une quantité de travail : $\alpha_i W_{\text{total}}$ avec $\alpha_i W_{\text{total}} \in \mathbb{Q}$ et $\sum_i \alpha_i = 1$.**
  Durée d'un calcul unitaire sur P_i : w_i.
  Temps de calcul sur P_i : $\alpha_i W_{\text{total}} w_i$.
- Durée de communication d'un message unitaire de P_1 à P_i : c.
  Bus un-port : P_1 envoie un message à la fois sur le bus, tous les processeurs communiquent à la même vitesse avec le maître.

Mise en équations

Pour le processeur P_i (avec $c_1 = 0$ et $c_j = c$ sinon) :

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

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

On cherche une distribution α_1, ..., α_p des données minimisant T.

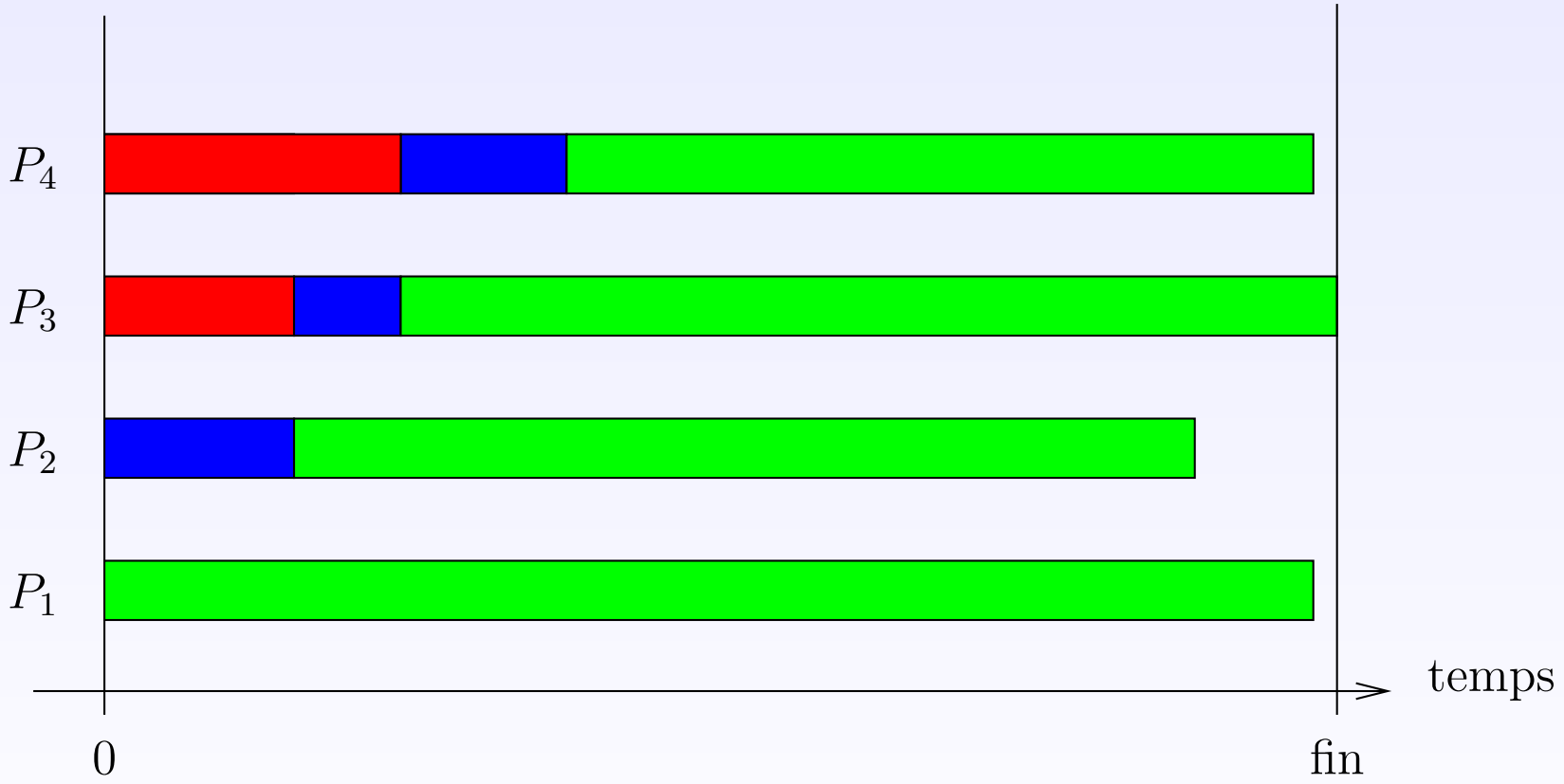
Propriété d'équilibrage

Lemme

Dans une solution optimale, tous les processeurs finissent en même temps.

Démonstration du lemme 1

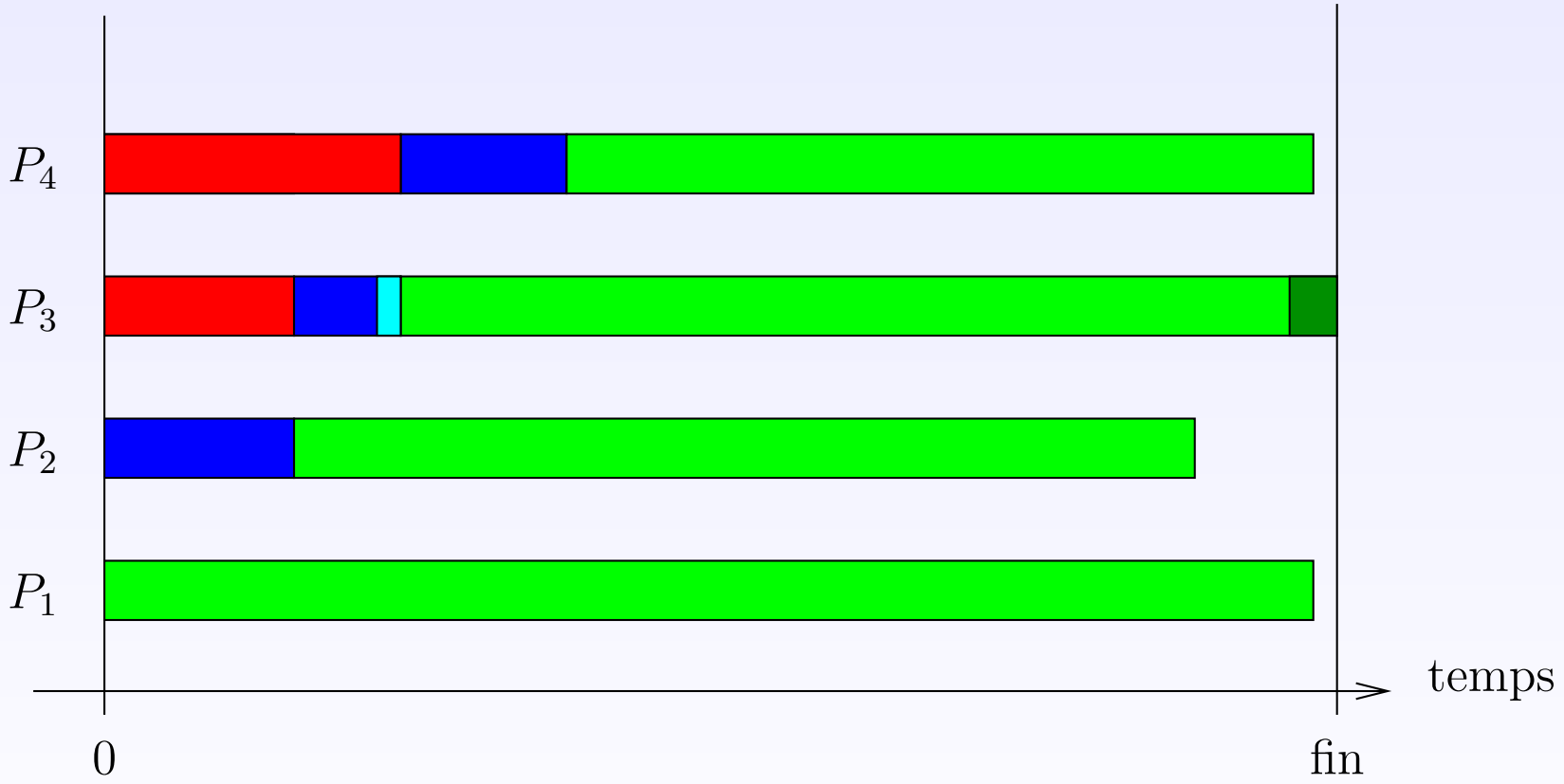
Deux esclaves i et $i+1$ avec $T_i < T_{i+1}$.



On diminue α_{i+1} de ϵ.

Démonstration du lemme 1

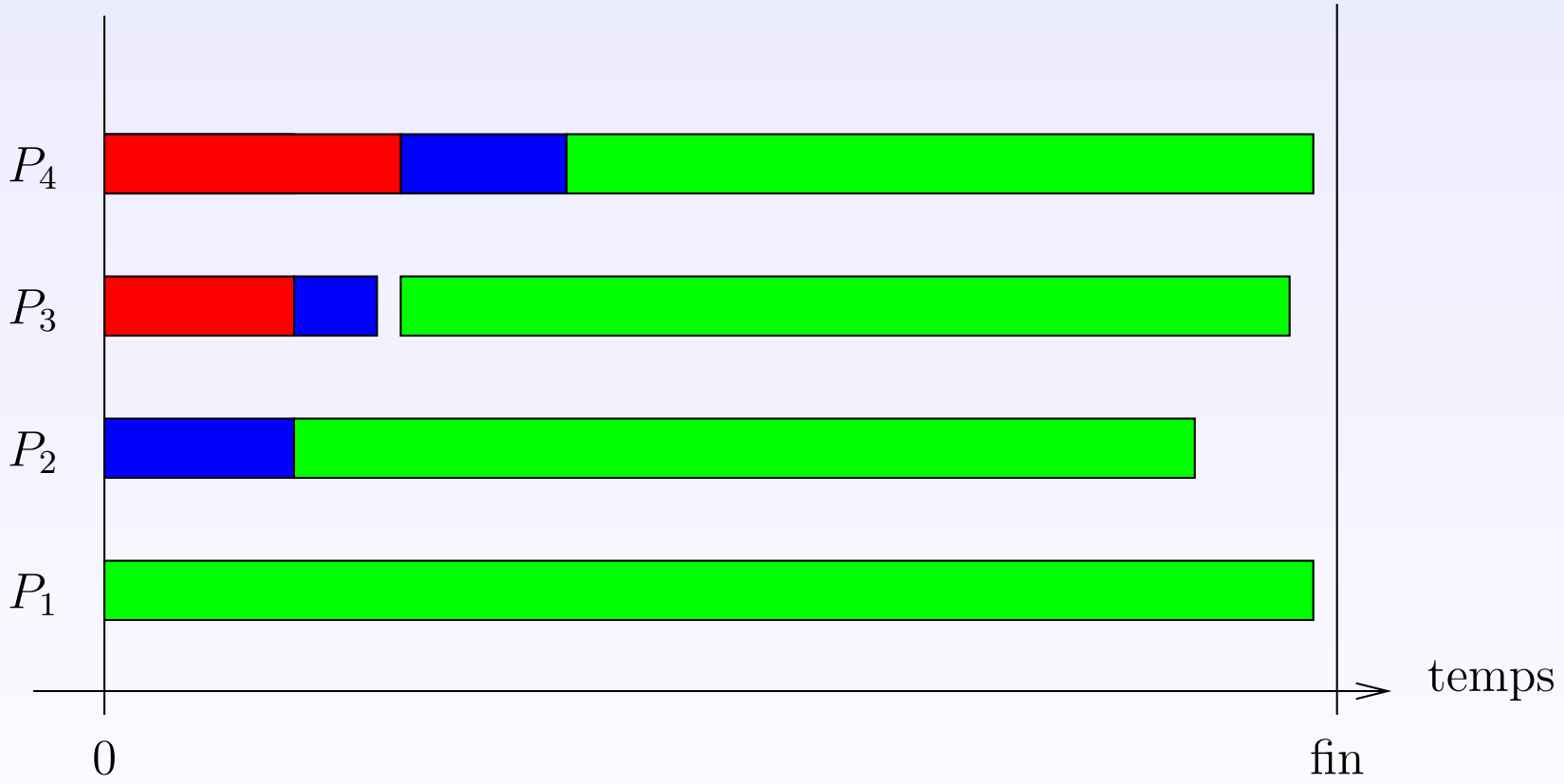
Deux esclaves i et $i+1$ avec $T_i < T_{i+1}$.



On diminue α_{i+1} de ϵ.

Démonstration du lemme 1

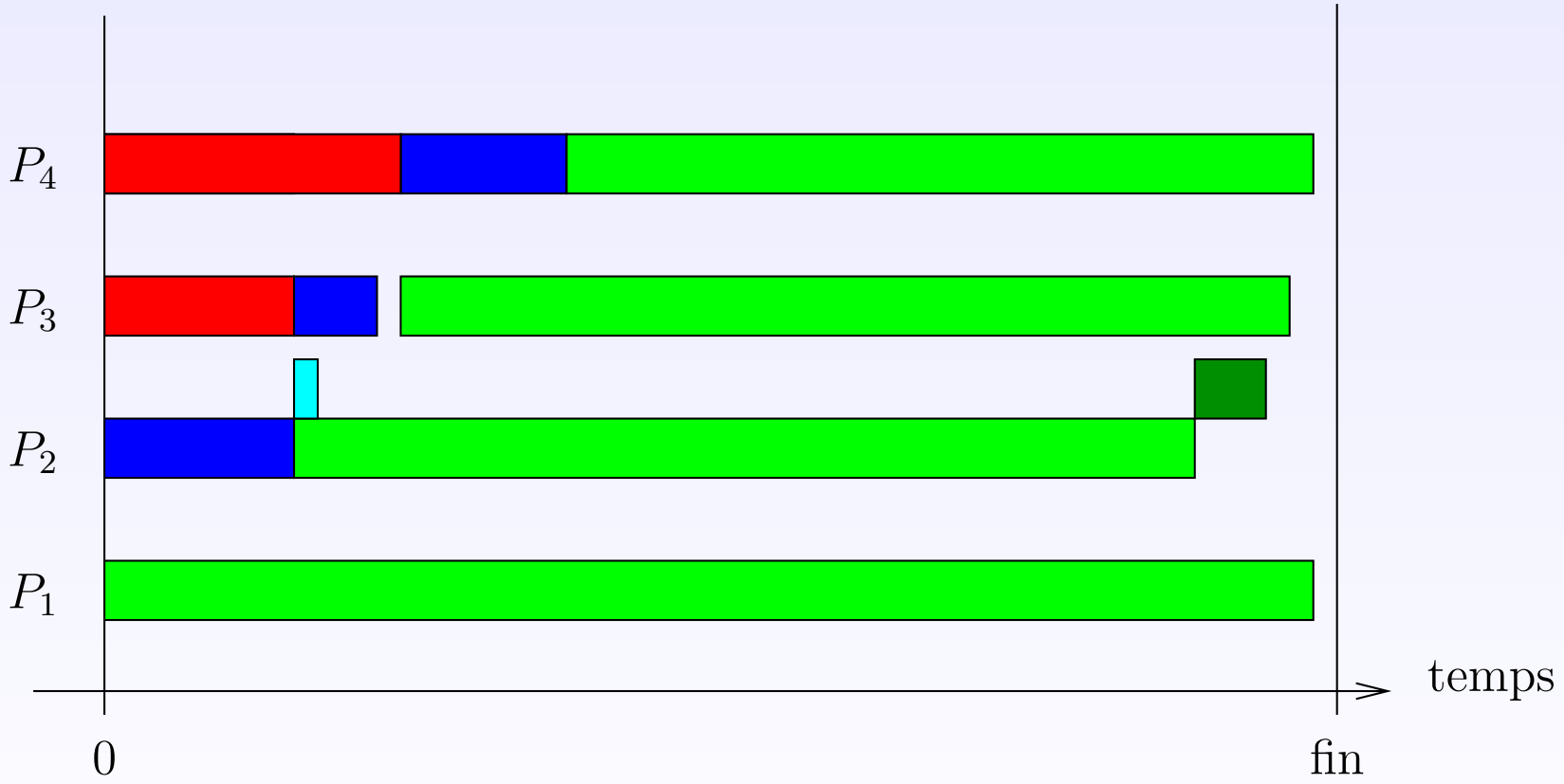
Deux esclaves i et $i+1$ avec $T_i < T_{i+1}$.



On diminue α_{i+1} de ϵ.

Démonstration du lemme 1

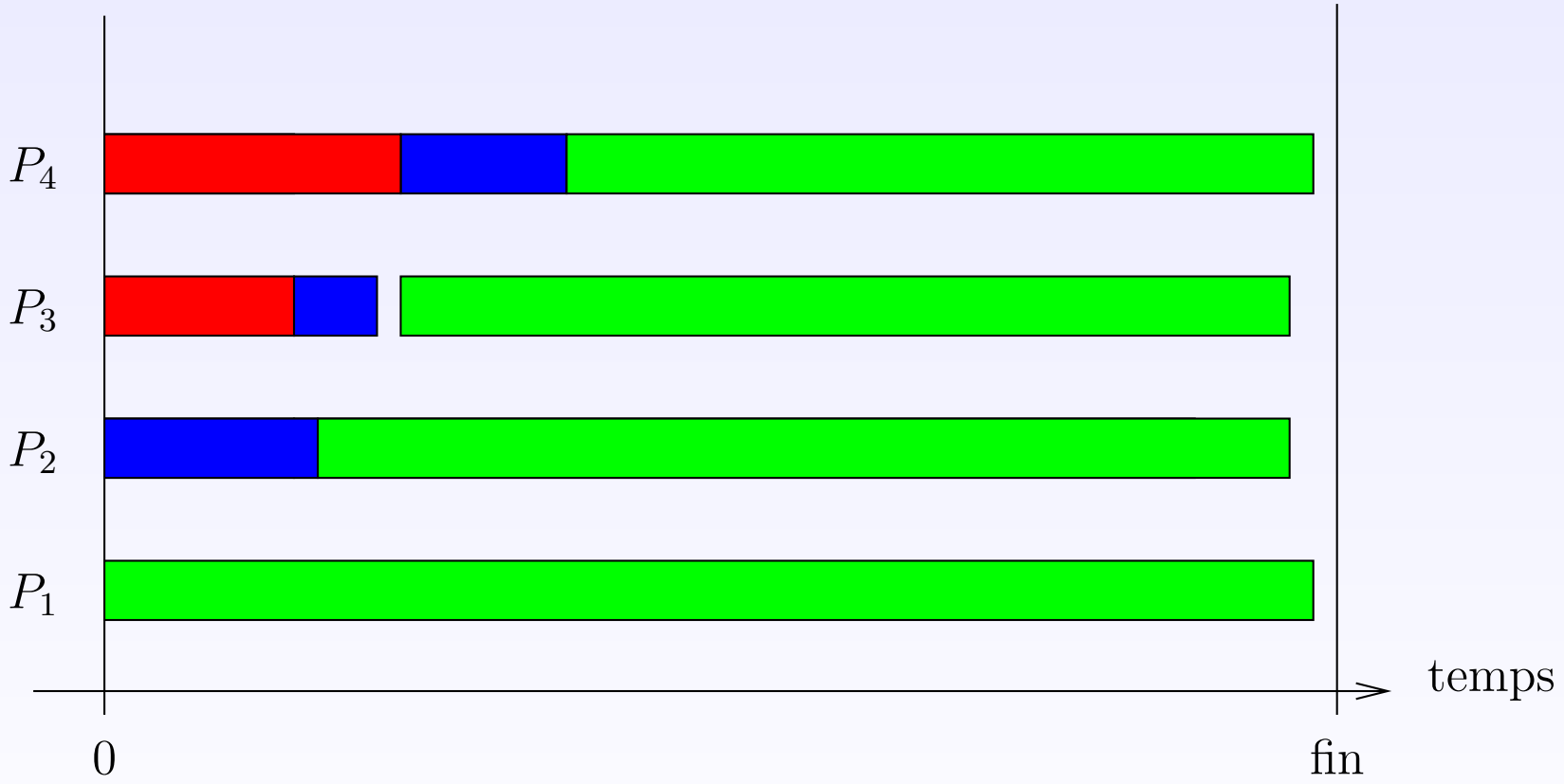
Deux esclaves i et $i+1$ avec $T_i < T_{i+1}$.



On augmente α_i de ϵ.

Démonstration du lemme 1

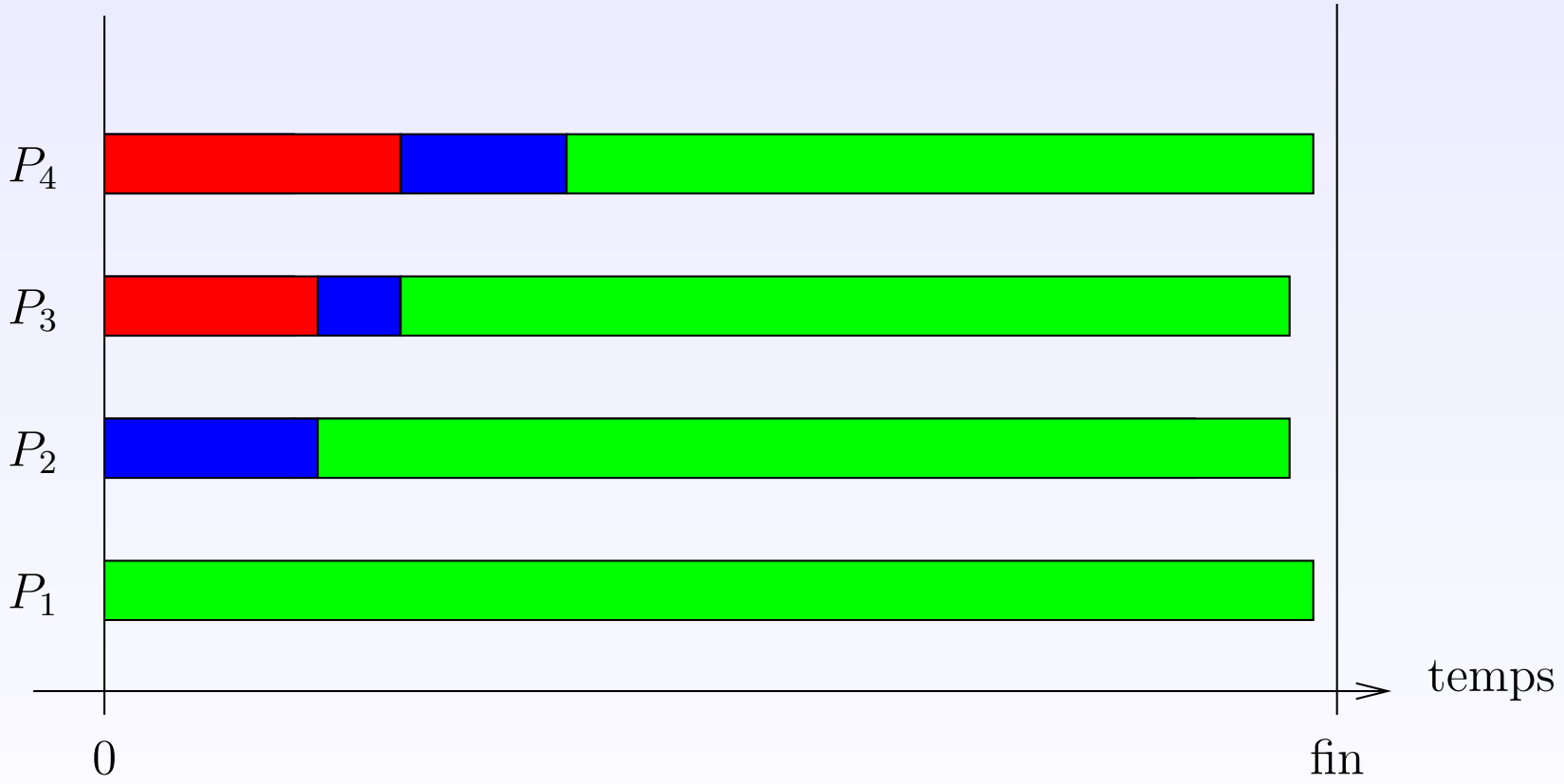
Deux esclaves i et $i+1$ avec $T_i < T_{i+1}$.



On augmente α_i de ϵ.

Démonstration du lemme 1

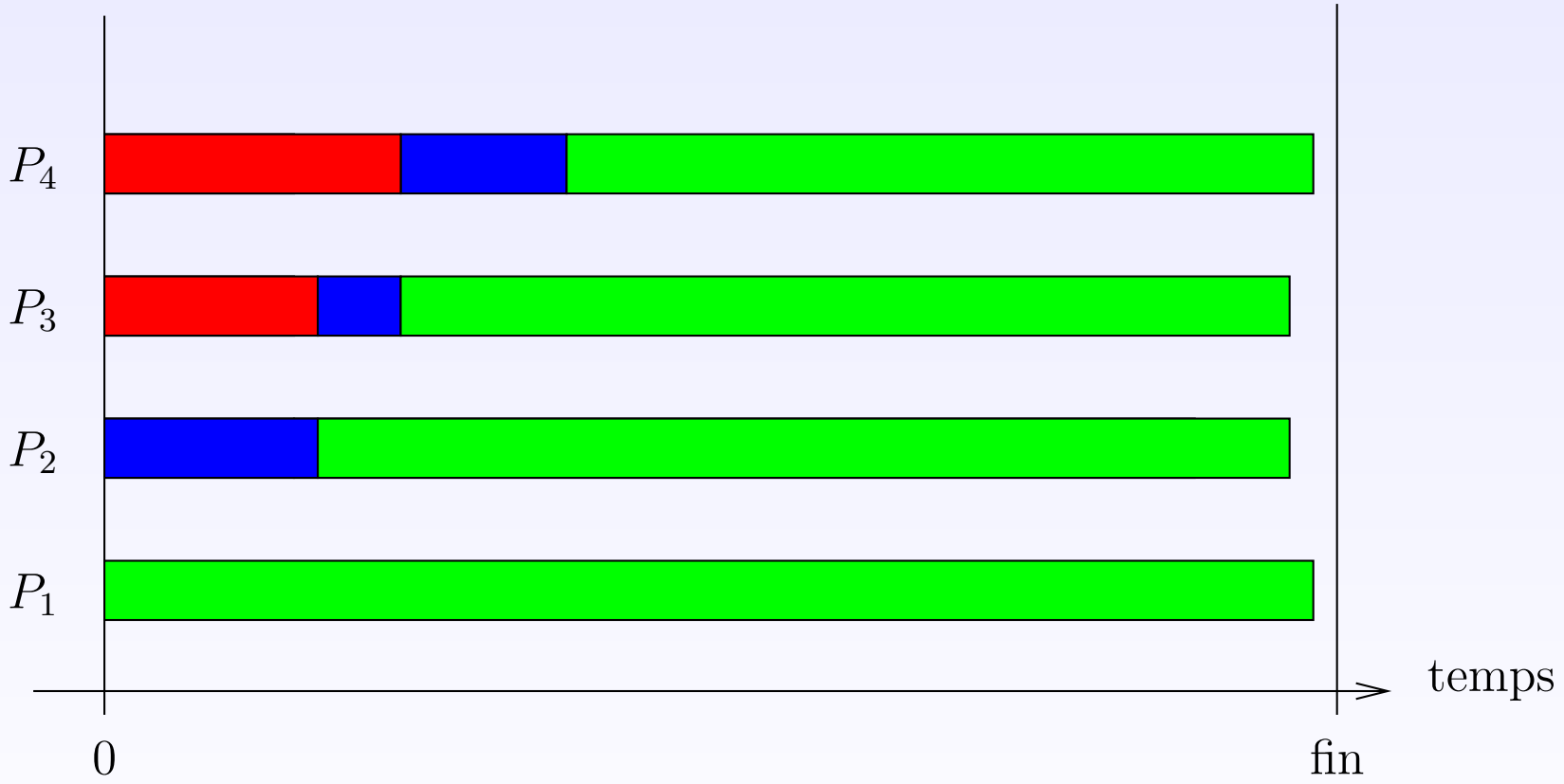
Deux esclaves i et $i+1$ avec $T_i < T_{i+1}$.



Le temps de communication pour les processeurs suivants ne change pas.

Démonstration du lemme 1

Deux esclaves i et $i+1$ avec $T_i < T_{i+1}$.



On obtient une meilleure solution !

Démonstration du lemme 1 (suite et fin)

- Idéal : $T'_i = T'_{i+1}$.
  On choisit ϵ tel que :
  $(\alpha_i + \epsilon)W_{\text{total}}(c + w_i) = (\alpha_i + \epsilon)W_{\text{total}}c + (\alpha_{i+1} - \epsilon)W_{\text{total}}(c + w_{i+1})$.
- Le maître finit avant : absurde.
- Le maître finit après : on diminue de même P_1 de ϵ.

Propriété sur la sélection des ressources

Lemme

Dans une solution optimale, tous les processeurs travaillent.

Propriété sur la sélection des ressources

Lemme

Dans une solution optimale, tous les processeurs travaillent.

Démonstration : Simple corollaire du lemme 1...

Résolution

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

Résolution

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

$T = \alpha_2 (c + w_2) W_{\text{total}}$. D'où $\alpha_2 = \frac{w_1}{c+w_2} \alpha_1$.

Résolution

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

$T = \alpha_2 (c + w_2) W_{\text{total}}$. D'où $\alpha_2 = \frac{w_1}{c+w_2} \alpha_1$.

$T = (\alpha_2 c + \alpha_3 (c + w_3)) W_{\text{total}}$. D'où $\alpha_3 = \frac{w_2}{c+w_3} \alpha_2$.

Résolution

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

$T = \alpha_2 (c + w_2) W_{\text{total}}$. D'où $\alpha_2 = \frac{w_1}{c+w_2} \alpha_1$.

$T = (\alpha_2 c + \alpha_3 (c + w_3)) W_{\text{total}}$. D'où $\alpha_3 = \frac{w_2}{c+w_3} \alpha_2$.

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

Résolution

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

$T = \alpha_2 (c + w_2) W_{\text{total}}$. D'où $\alpha_2 = \frac{w_1}{c+w_2} \alpha_1$.

$T = (\alpha_2 c + \alpha_3 (c + w_3)) W_{\text{total}}$. D'où $\alpha_3 = \frac{w_2}{c+w_3} \alpha_2$.

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

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

Résolution

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

$T = \alpha_2 (c + w_2) W_{\text{total}}$. D'où $\alpha_2 = \frac{w_1}{c+w_2} \alpha_1$.

$T = (\alpha_2 c + \alpha_3 (c + w_3)) W_{\text{total}}$. D'où $\alpha_3 = \frac{w_2}{c+w_3} \alpha_2$.

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

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

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

Influence de l'ordre des communications

?

Pas d'influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Pas d'influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i : $\alpha_i(c + w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c+w_i}\frac{T}{W_{\text{total}}}$.

Pas d'influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i **:** $\alpha_i(c+w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c+w_i}\frac{T}{W_{\text{total}}}$.

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

Pas d'influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i **:** $\alpha_i(c+w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c+w_i}\frac{T}{W_{\text{total}}}$.

Processeur P_{i+1} **:** $\alpha_i c W_{\text{total}} + \alpha_{i+1}(c+w_{i+1})W_{\text{total}} = T$.
D'où $\alpha_{i+1} = \frac{1}{c+w_{i+1}}(\frac{T}{W_{\text{total}}} - \alpha_i c) = \frac{w_i}{(c+w_i)(c+w_{i+1})}\frac{T}{W_{\text{total}}}$.

Pas d'influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i : $\alpha_i(c+w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c+w_i}\frac{T}{W_{\text{total}}}$.

Processeur P_{i+1} : $\alpha_i c W_{\text{total}} + \alpha_{i+1}(c+w_{i+1})W_{\text{total}} = T$.
D'où $\alpha_{i+1} = \frac{1}{c+w_{i+1}}(\frac{T}{W_{\text{total}}} - \alpha_i c) = \frac{w_i}{(c+w_i)(c+w_{i+1})}\frac{T}{W_{\text{total}}}$.

Processeurs P_i et P_{i+1} :

$$\alpha_i + \alpha_{i+1} = \frac{c+w_i+w_{i+1}}{(c+w_i)(c+w_{i+1})}$$

Choix du processeur maître

On compare les processeurs P_1 et P_2.

Choix du processeur maître

On compare les processeurs P_1 et P_2.

Processeur P_1 : $\alpha_1 w_1 W_{\text{total}} = T$. D'où $\alpha_1 = \frac{1}{w_1} \frac{T}{W_{\text{total}}}$.

Choix du processeur maître

On compare les processeurs P_1 et P_2.

Processeur P_1 : $\alpha_1 w_1 W_{\text{total}} = T$. D'où $\alpha_1 = \frac{1}{w_1} \frac{T}{W_{\text{total}}}$.

Processeur P_2 : $\alpha_2 (c + w_2) W_{\text{total}} = T$. D'où $\alpha_2 = \frac{1}{c+w_2} \frac{T}{W_{\text{total}}}$.

Choix du processeur maître

On compare les processeurs P_1 et P_2.

Processeur P_1 : $\alpha_1 w_1 W_{\text{total}} = T$. D'où $\alpha_1 = \frac{1}{w_1} \frac{T}{W_{\text{total}}}$.

Processeur P_2 : $\alpha_2 (c + w_2) W_{\text{total}} = T$. D'où $\alpha_2 = \frac{1}{c+w_2} \frac{T}{W_{\text{total}}}$.

Volume total traité :

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

Choix du processeur maître

On compare les processeurs P_1 et P_2.

Processeur P_1 : $\alpha_1 w_1 W_{\text{total}} = T$. D'où $\alpha_1 = \frac{1}{w_1} \frac{T}{W_{\text{total}}}$.

Processeur P_2 : $\alpha_2 (c + w_2) W_{\text{total}} = T$. D'où $\alpha_2 = \frac{1}{c+w_2} \frac{T}{W_{\text{total}}}$.

Volume total traité :

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

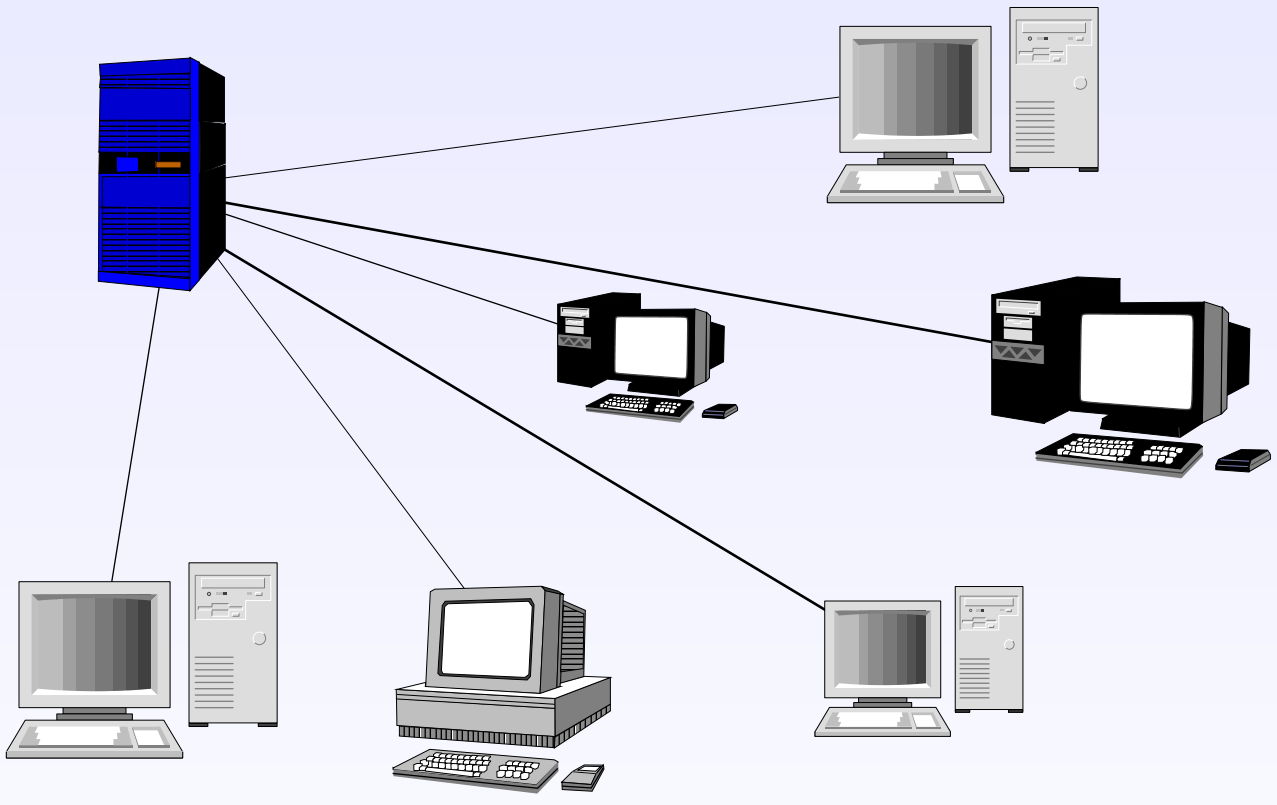
Minimal quand $w_1 < w_2$.
Maître = processeur le plus rapide (en calcul).

Conclusion

- Formule pour le temps d'exécution et la répartition des données.
- Choix du maître.
- Ordre des processeurs indifférents.
- Tous les processeurs participent.

Plan du cours

Réseau en étoile



- Les liaisons entre le maître et les esclaves ont des caractéristiques *différentes*.
- Les esclaves ont des puissances de calcul différentes.

Notations

- Un ensemble P_1, ..., P_p de processeurs
- P_1 est le processeur maître : il détient initialement toutes les données.
- Quantité totale de travail : W_{total}.
- Le processeur P_i reçoit une quantité de travail : $\alpha_i W_{\text{total}}$
  avec $\alpha_i W_{\text{total}} \in \mathbb{Q}$ et $\sum_i \alpha_i = 1$.
  Durée d'un calcul unitaire sur P_i : w_i.
  Temps de calcul sur P_i : $\alpha_i W_{\text{total}} w_i$.
- **Durée de communication d'un message unitaire de P_1 à P_i : c_i.**
  Modèle un-port : P_1 envoie un seul message à la fois.

Influence de l'ordre des communications

?

Influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i : $\alpha_i(c_i + w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c_i+w_i}\frac{T}{W_{\text{total}}}$.

Influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i **:** $\alpha_i(c_i + w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c_i+w_i}\frac{T}{W_{\text{total}}}$.

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

Influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i : $\alpha_i(c_i + w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c_i+w_i}\frac{T}{W_{\text{total}}}$.

Processeur P_{i+1} : $\alpha_i c_i W_{\text{total}} + \alpha_{i+1}(c_{i+1} + w_{i+1})W_{\text{total}} = T$.
D'où $\alpha_{i+1} = \frac{1}{c_{i+1}+w_{i+1}}(1 - \frac{c_i}{c_i+w_i})\frac{T}{W_{\text{total}}} = \frac{w_i}{(c_i+w_i)(c_{i+1}+w_{i+1})}\frac{T}{W_{\text{total}}}$.

Influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i **:** $\alpha_i(c_i + w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c_i+w_i}\frac{T}{W_{\text{total}}}$.

Processeur P_{i+1} **:** $\alpha_i c_i W_{\text{total}} + \alpha_{i+1}(c_{i+1} + w_{i+1})W_{\text{total}} = T$.
D'où $\alpha_{i+1} = \frac{1}{c_{i+1}+w_{i+1}}(1 - \frac{c_i}{c_i+w_i})\frac{T}{W_{\text{total}}} = \frac{w_i}{(c_i+w_i)(c_{i+1}+w_{i+1})}\frac{T}{W_{\text{total}}}$.

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

Influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i **:** $\alpha_i(c_i + w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c_i+w_i}\frac{T}{W_{\text{total}}}$.

Processeur P_{i+1} **:** $\alpha_i c_i W_{\text{total}} + \alpha_{i+1}(c_{i+1} + w_{i+1})W_{\text{total}} = T$.
D'où $\alpha_{i+1} = \frac{1}{c_{i+1}+w_{i+1}}(1 - \frac{c_i}{c_i+w_i})\frac{T}{W_{\text{total}}} = \frac{w_i}{(c_i+w_i)(c_{i+1}+w_{i+1})}\frac{T}{W_{\text{total}}}$.

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

Temps de communication : $\alpha_i c_i + \alpha_{i+1}c_{i+1} = \frac{c_i c_{i+1}+c_{i+1}w_i+c_i w_{i+1}}{(c_i+w_i)(c_{i+1}+w_{i+1})}$

Influence de l'ordre des communications

Volume traité par les processeurs P_i et P_{i+1} en un temps T.

Processeur P_i **:** $\alpha_i(c_i + w_i)W_{\text{total}} = T$. D'où $\alpha_i = \frac{1}{c_i+w_i}\frac{T}{W_{\text{total}}}$.

Processeur P_{i+1} **:** $\alpha_i c_i W_{\text{total}} + \alpha_{i+1}(c_{i+1} + w_{i+1})W_{\text{total}} = T$.
D'où $\alpha_{i+1} = \frac{1}{c_{i+1}+w_{i+1}}(1 - \frac{c_i}{c_i+w_i})\frac{T}{W_{\text{total}}} = \frac{w_i}{(c_i+w_i)(c_{i+1}+w_{i+1})}\frac{T}{W_{\text{total}}}$.

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

Temps de communication : $\alpha_i c_i + \alpha_{i+1}c_{i+1} = \frac{c_i c_{i+1}+c_{i+1}w_i+c_i w_{i+1}}{(c_i+w_i)(c_{i+1}+w_{i+1})}$

Processeurs servis par bande passante décroissante.

Sélection de ressources

Lemme

Dans une solution optimale, tous les processeurs travaillent.

Démonstration du lemme 3

On prend une solution optimale. Soit P_k un processeur qui ne reçoit rien : on le met en dernier et on peut lui donner une fraction α_k telle que $\alpha_k(c_k + w_k)W_{\text{total}}$ soit égal au temps de traitement du dernier processeur ayant reçu du travail.

Démonstration du lemme 3

On prend une solution optimale. Soit P_k un processeur qui ne reçoit rien : on le met en dernier et on peut lui donner une fraction α_k telle que $\alpha_k(c_k + w_k)W_{\text{total}}$ soit égal au temps de traitement du dernier processeur ayant reçu du travail.

Pourquoi doit-on mettre le processeur en dernier ?

Propriété d'équilibrage

Lemme

Dans une solution optimale, tous les processeurs finissent en même temps.

Démonstration du lemme 4

- La majorité des preuves existantes sont fausses.
- Arguments très techniques (propriété des solutions des systèmes linéaires) ou ennuyeux (étude cas par cas).

Conclusion

- Les processeurs doivent être ordonnés par bande passante décroissante
- Ils travaillent tous
- Ils finissent en même temps
- Formules pour le temps d'exécution et la répartition des données

Plan du cours

Messages de retour

- Une fois qu'il a fini sa part de travail, chaque fils renvoie un message de résultat au maître.

- Problèmes à résoudre
  - Sélectionner les ressources.
  - Définir l'ordre d'envoi des données aux fils.
  - Définir l'ordre de réception des résultats.
  - Définir la quantité de travail de chacun des fils.

Notations

- Un ensemble P_1, ..., P_p de processeurs
- P_1 est le processeur maître : il détient initialement toutes les données.
- Quantité totale de travail : W_{total}.
- Le processeur P_i reçoit une quantité de travail : $\alpha_i W_{\text{total}}$
  avec $\alpha_i W_{\text{total}} \in \mathbb{Q}$ et $\sum_i \alpha_i = 1$.
  Durée d'un calcul unitaire sur P_i : w_i.
  Temps de calcul sur P_i : $\alpha_i W_{\text{total}} w_i$.
- **Durée de communication d'un message unitaire de P_1 à P_i : c_i.**
- **Durée de communication d'un message unitaire de P_i à P_1 : d_i.** (Parfois on posera $d_i = z c_i$.)
- Modèle un-port : P_1 envoie et reçoit un seul message à la fois.

Solutions avec temps d'attente ?

- Attendre entre la fin de la réception des données et le début des calculs ?

Solutions avec temps d'attente ?

- Attendre entre la fin de la réception des données et le début des calculs ?
  Aucun intérêt !

Solutions avec temps d'attente ?

- Attendre entre la fin de la réception des données et le début des calculs ?
  Aucun intérêt !
- Attendre entre la fin des calculs et le début de l'envoi du message retour ?

Solutions avec temps d'attente ?

- Attendre entre la fin de la réception des données et le début des calculs ?
  Aucun intérêt !
- Attendre entre la fin des calculs et le début de l'envoi du message retour ?

  Nécessaire si le lien de communication n'est pas disponible.

Solutions avec temps d'attente ?

- Attendre entre la fin de la réception des données et le début des calculs ?
  Aucun intérêt !
- Attendre entre la fin des calculs et le début de l'envoi du message retour ?

  Nécessaire si le lien de communication n'est pas disponible.

Il faut prévoir la possibilité de temps d'attente dans la construction de la solution.

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.
  - Ordre optimal et formules clauses (trivial).

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.
  - Ordre optimal et formules clauses (trivial).
- Drozdowski et Wolniewicz : étude expérimentale de distributions LIFO et FIFO.

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.
  - Ordre optimal et formules clauses (trivial).
- Drozdowski et Wolniewicz : étude expérimentale de distributions LIFO et FIFO.
- Rosenberg et al. :

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.
  - Ordre optimal et formules clauses (trivial).
- Drozdowski et Wolniewicz : étude expérimentale de distributions LIFO et FIFO.
- Rosenberg et al. :
  - Modèle de communication complexe (affine).

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.
  - Ordre optimal et formules clauses (trivial).
- Drozdowski et Wolniewicz : étude expérimentale de distributions LIFO et FIFO.
- Rosenberg et al. :
  - Modèle de communication complexe (affine).
  - Possibilité de ralentir un processeur (pour éviter les attentes).

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.
  - Ordre optimal et formules clauses (trivial).
- Drozdowski et Wolniewicz : étude expérimentale de distributions LIFO et FIFO.
- Rosenberg et al. :
  - Modèle de communication complexe (affine).
  - Possibilité de ralentir un processeur (pour éviter les attentes).
  - En pratique : pas d'hétérogénéité des temps de communications.

État des connaissances en janvier 2005

(le premier papier sur les tâches divisibles date de 1988)

- Barlas
  - Temps de communication fixe ou réseau en bus $c_i = c$.
  - Ordre optimal et formules clauses (trivial).
- Drozdowski et Wolniewicz : étude expérimentale de distributions LIFO et FIFO.
- Rosenberg et al. :
  - Modèle de communication complexe (affine).
  - Possibilité de ralentir un processeur (pour éviter les attentes).
  - En pratique : pas d'hétérogénéité des temps de communications.
  - Toutes les distributions FIFO sont équivalentes et sont meilleures que les autres distributions (preuve par échange).

Programme linéaire pour un scénario donné (1)

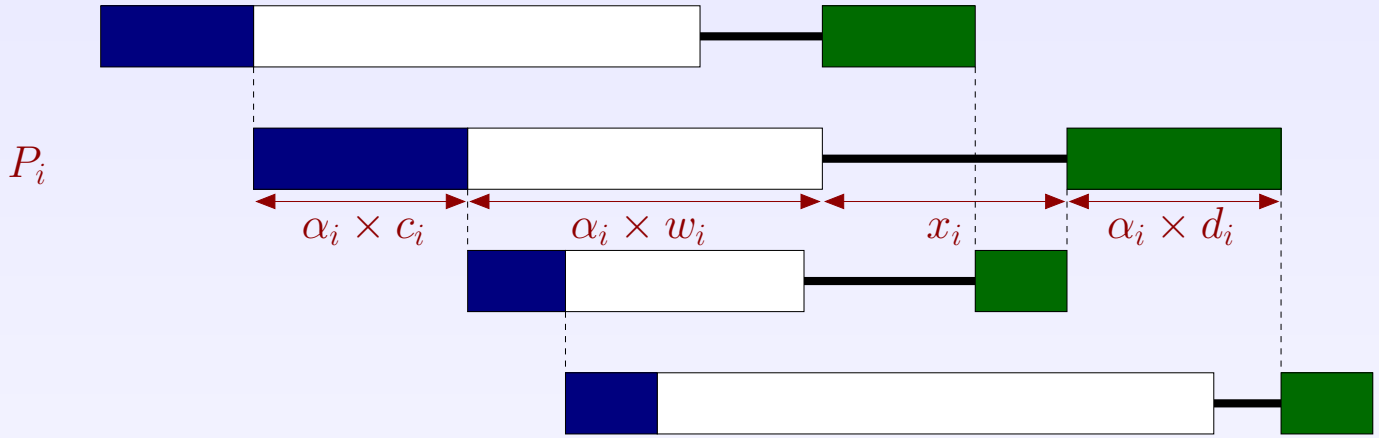
Un scénario décrit :

- à quel processeur on donne du travail ;
- dans quel ordre ont lieu les communications (envoi des données et récupération des résultats)

Étant donné un scénario, on peut supposer

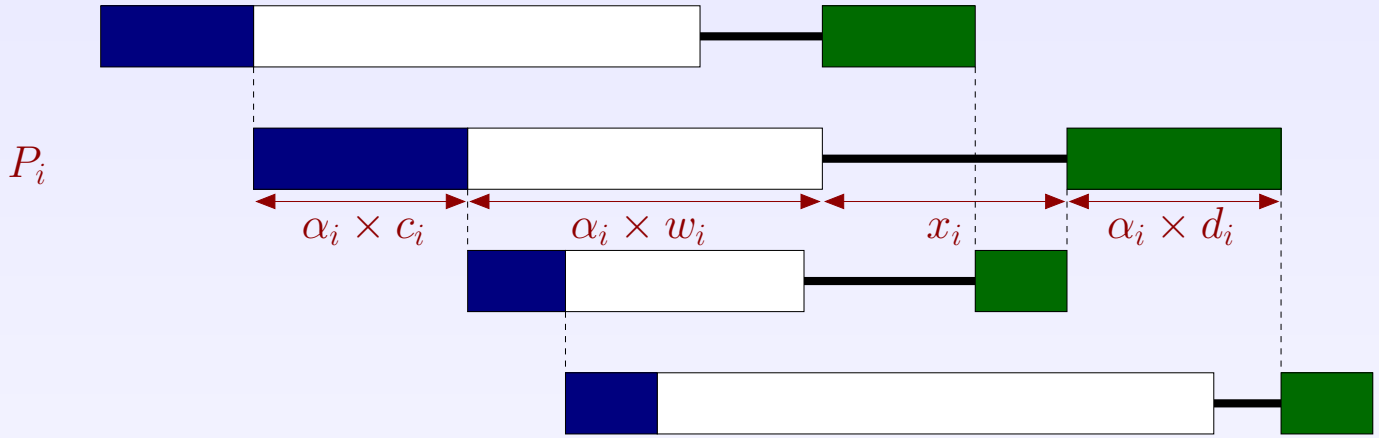
- que le maître envoi les données au plus tôt ;
- que les esclaves commencent à calculer au plus tôt ;
- que les esclaves renvoient leurs résultats le plus tard possible.

Programme linéaire pour un scénario donné (2)



Considérons l'esclave P_i :

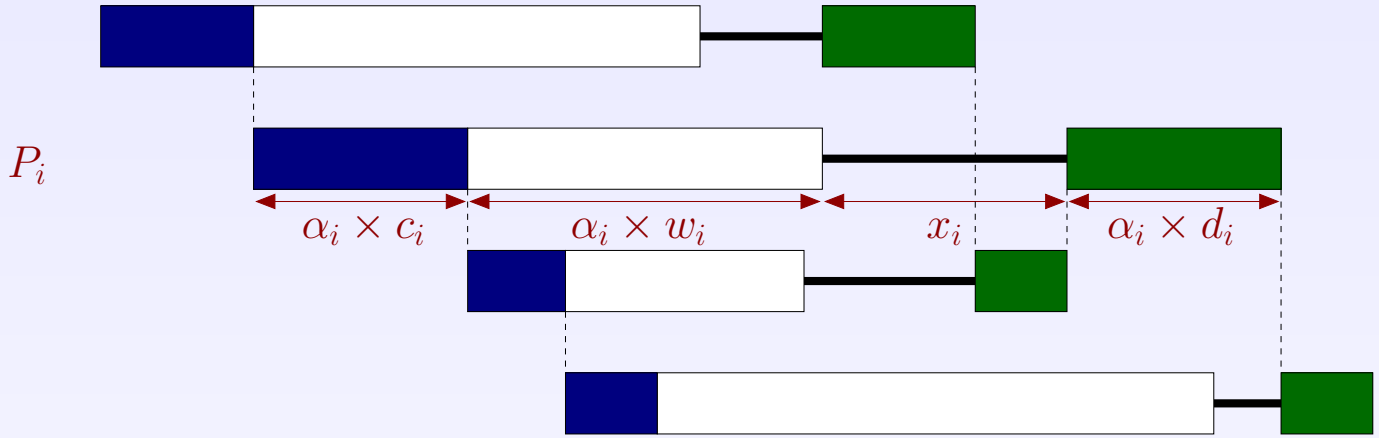
Programme linéaire pour un scénario donné (2)



Considérons l'esclave P_i :

- reçoit ses données à partir de $t_i^{\text{recv}} = \sum_{j=1}^{i-1} \alpha_j \times c_j$

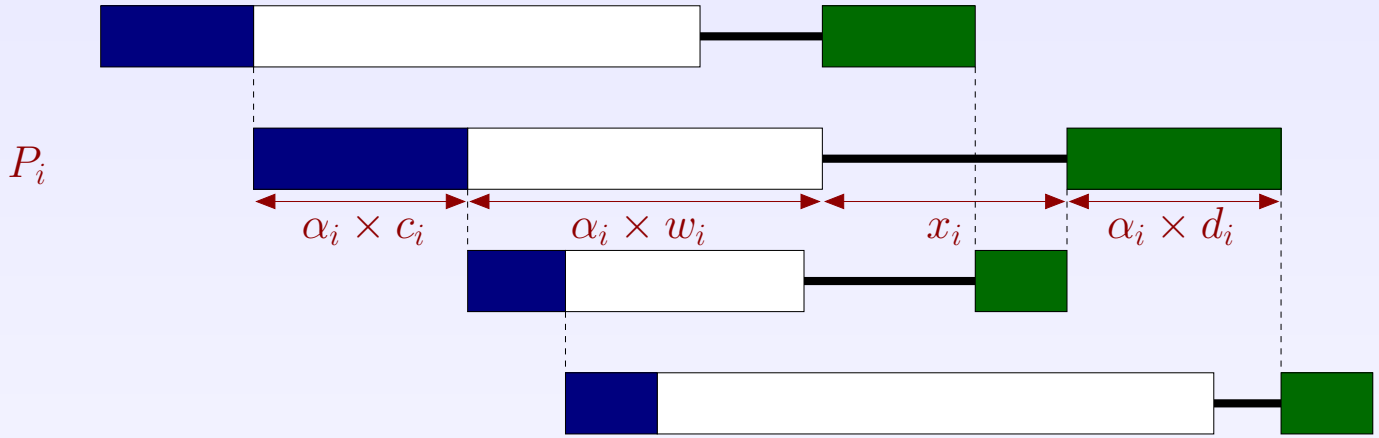
Programme linéaire pour un scénario donné (2)



Considérons l'esclave P_i :

- reçoit ses données à partir de $t_i^{\text{recv}} = \sum_{j=1}^{i-1} \alpha_j \times c_j$
- travaille à partir de $t_i^{\text{recv}} + \alpha_i \times c_i$

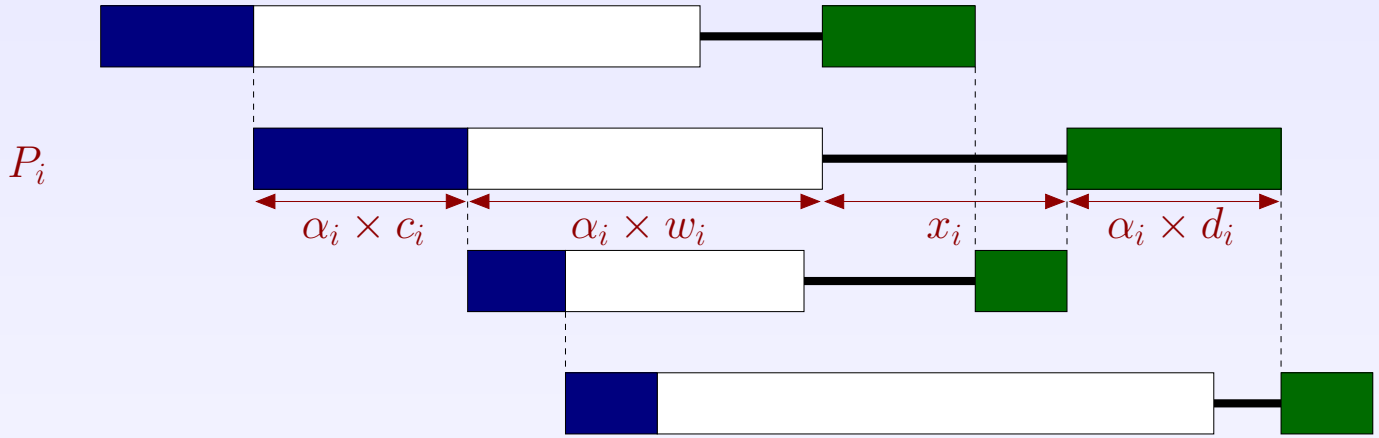
Programme linéaire pour un scénario donné (2)



Considérons l'esclave P_i :

- reçoit ses données à partir de $t_i^{\text{recv}} = \sum_{j=1}^{i-1} \alpha_j \times c_j$
- travaille à partir de $t_i^{\text{recv}} + \alpha_i \times c_i$
- termine son travail à la date $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$

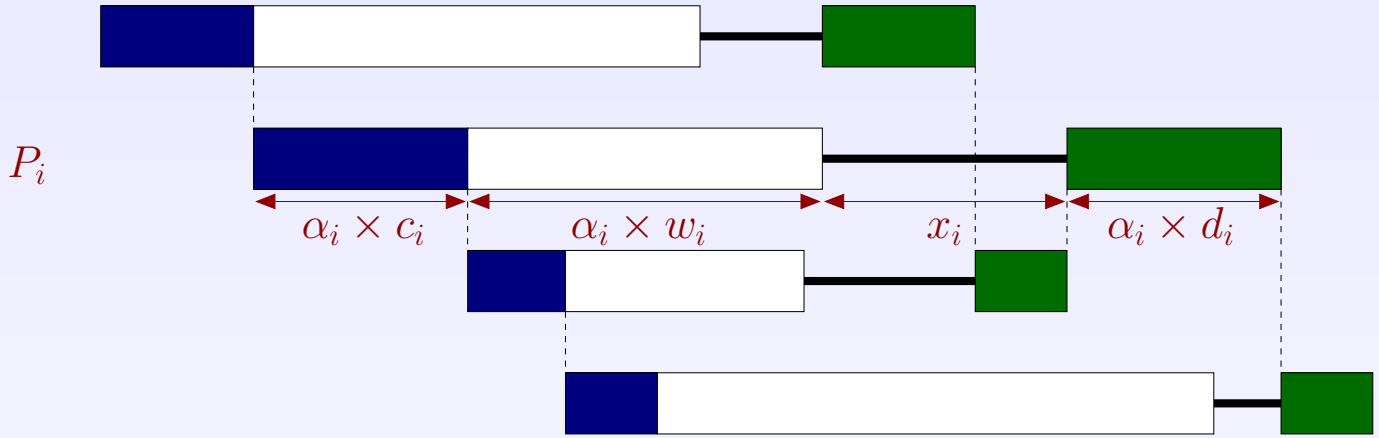
Programme linéaire pour un scénario donné (2)



Considérons l'esclave P_i :

- reçoit ses données à partir de $t_i^{\text{recv}} = \sum_{j=1}^{i-1} \alpha_j \times c_j$
- travaille à partir de $t_i^{\text{recv}} + \alpha_i \times c_i$
- termine son travail à la date $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$
- envoie ses résultats à la date $t_i^{\text{back}} = T - \sum_{j \text{ predecessor of } i} \alpha_j \times d_j$

Programme linéaire pour un scénario donné (2)



Considérons l'esclave P_i :

- reçoit ses données à partir de $t_i^{\text{recv}} = \sum_{j=1}^{i-1} \alpha_j \times c_j$
- travaille à partir de $t_i^{\text{recv}} + \alpha_i \times c_i$
- termine son travail à la date $t_i^{\text{term}} = t_i^{\text{recv}} + \alpha_i \times c_i + \alpha_i \times w_i$
- envoie ses résultats à la date $t_i^{\text{back}} = T - \sum_{j \text{ predecessor of } i} \alpha_j \times d_j$
- temps d'attente : $x_i = t_i^{\text{back}} - t_i^{\text{term}} \geq 0$

Programme linéaire pour un scénario donné (3)

Pour T fixé, nous obtenons le programme linéaire :

$$\text{Maximiser } \sum_i \alpha_i, \text{ sous les constraintes } \begin{cases} \alpha_i \geq 0 \\ t_i^{\mathsf{back}} - t_i^{\mathsf{term}} \geq 0 \end{cases} \tag{1}$$

- Débit optimal *étant donnés* les ressources et un ordre.

Programme linéaire pour un scénario donné (3)

Pour T fixé, nous obtenons le programme linéaire :

$$\text{Maximiser } \sum_i \alpha_i, \text{ sous les constraintes } \begin{cases} \alpha_i \geq 0 \\ t_i^{\text{back}} - t_i^{\text{term}} \geq 0 \end{cases} \tag{1}$$

- Débit optimal *étant donnés* les ressources et un ordre.

Pour une quantité de travail $\sum_i \alpha_i = W$ fixée :

$$\text{Minimiser } T, \text{ sous les constraints } \begin{cases} \alpha_i \geq 0 \\ \sum_i \alpha_i = W \\ t_i^{\text{back}} - t_i^{\text{term}} \geq 0 \end{cases} \tag{2}$$

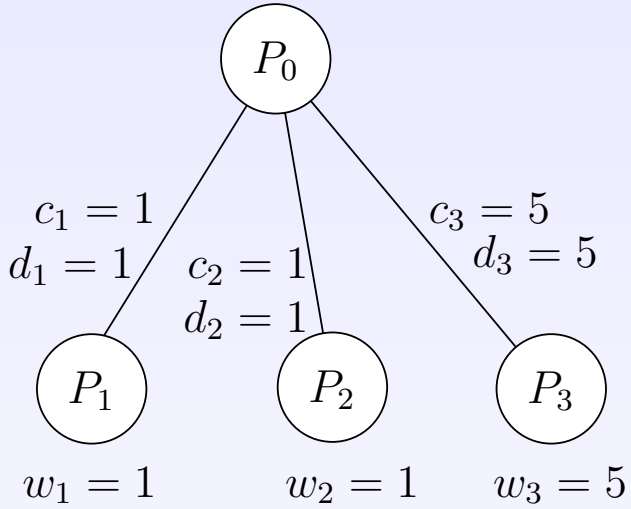
- Temps minimal *étant donnés* les ressources et un ordre.

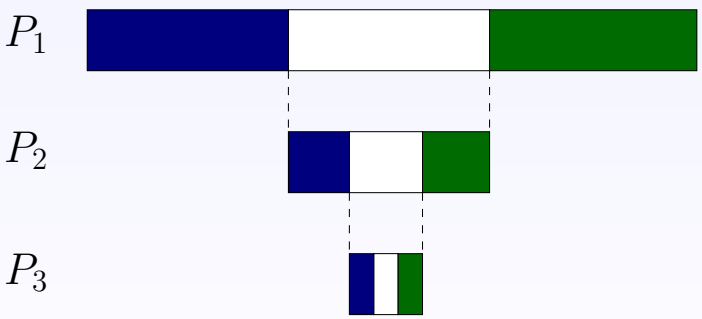
Programme linéaire pour un scénario donné (4)

Impossible de tester toutes les configurations

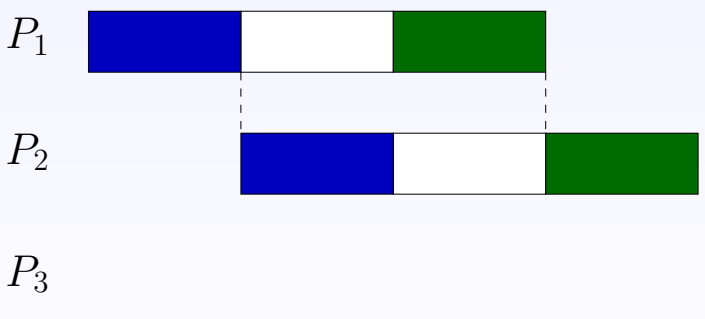
- Même si l'on impose que l'ordre des messages de retour soit le même que l'ordre d'envoi des messages (FIFO), il reste un nombre exponentiel de scénarios à tester.

Tous les processeurs ne participent pas toujours



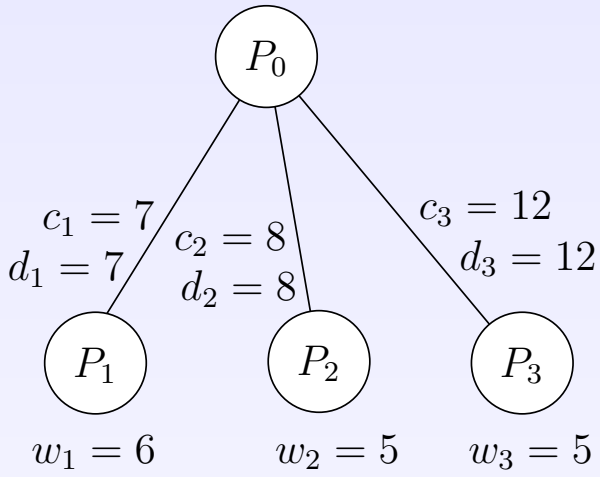


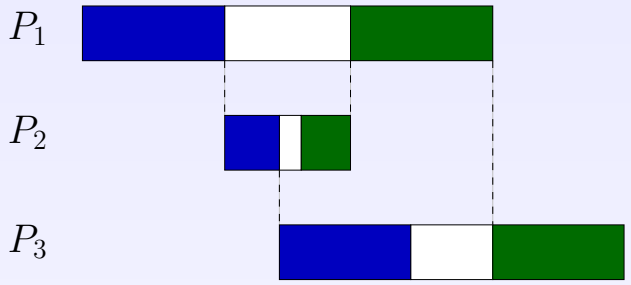
LIFO, débit $\rho = 61/135$
(meilleur ordonnancement avec 3 processeurs)



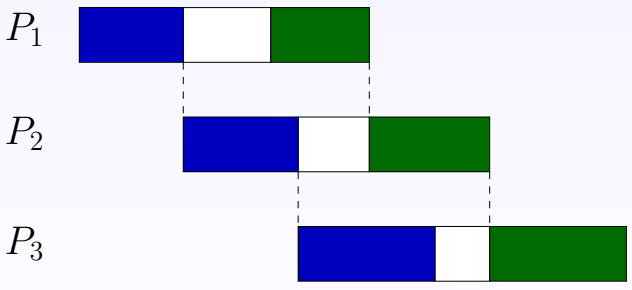
FIFO avec 2 processeurs, débit optimal $\rho = 1/2$

L'ordonnancement optimal peut n'être ni LIFO ni FIFO

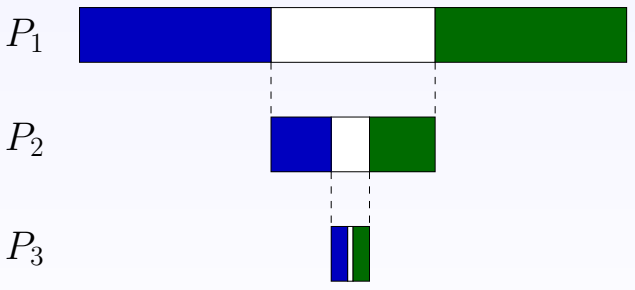




Ordonnancement optimal
$(\rho = 38/499 \approx 0.076)$



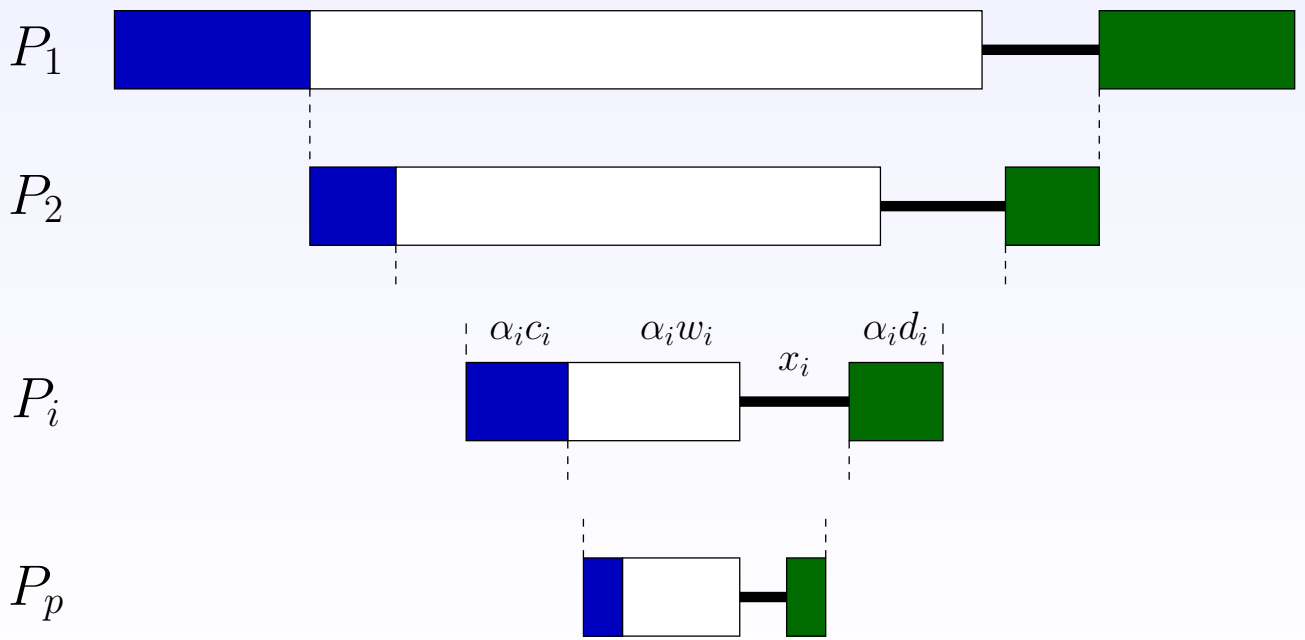
meilleur ordonnancement FIFO
$(\rho = 47/632 \approx 0.074)$



meilleur ordonnancement LIFO
$(\rho = 43/580 \approx 0.074)$

Stratégies LIFO

- LIFO = Last In First Out
- Le processeur qui reçoit en premier ses données est le dernier à renvoyer ses résultats.
- L'ordre de réception des messages-retour est l'inverse de l'ordre d'envoi des données.



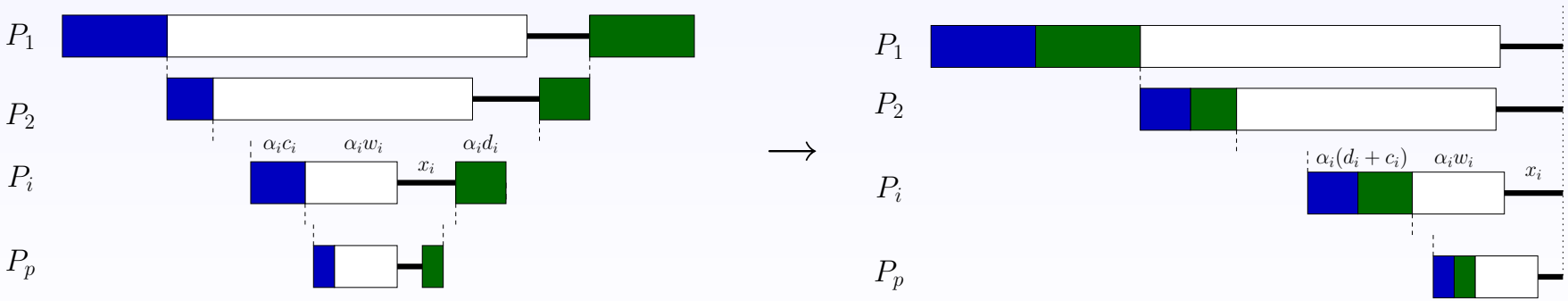
Stratégies LIFO

Théorème

Dans la meilleure solution LIFO :

- *Tous les processeurs travaillent*
- *Les données sont envoyées par valeurs croissantes de* $c_i + d_i$
- *Il n'y a pas de temps d'attente, i.e.* $x_i = 0$ *pour tout* i.

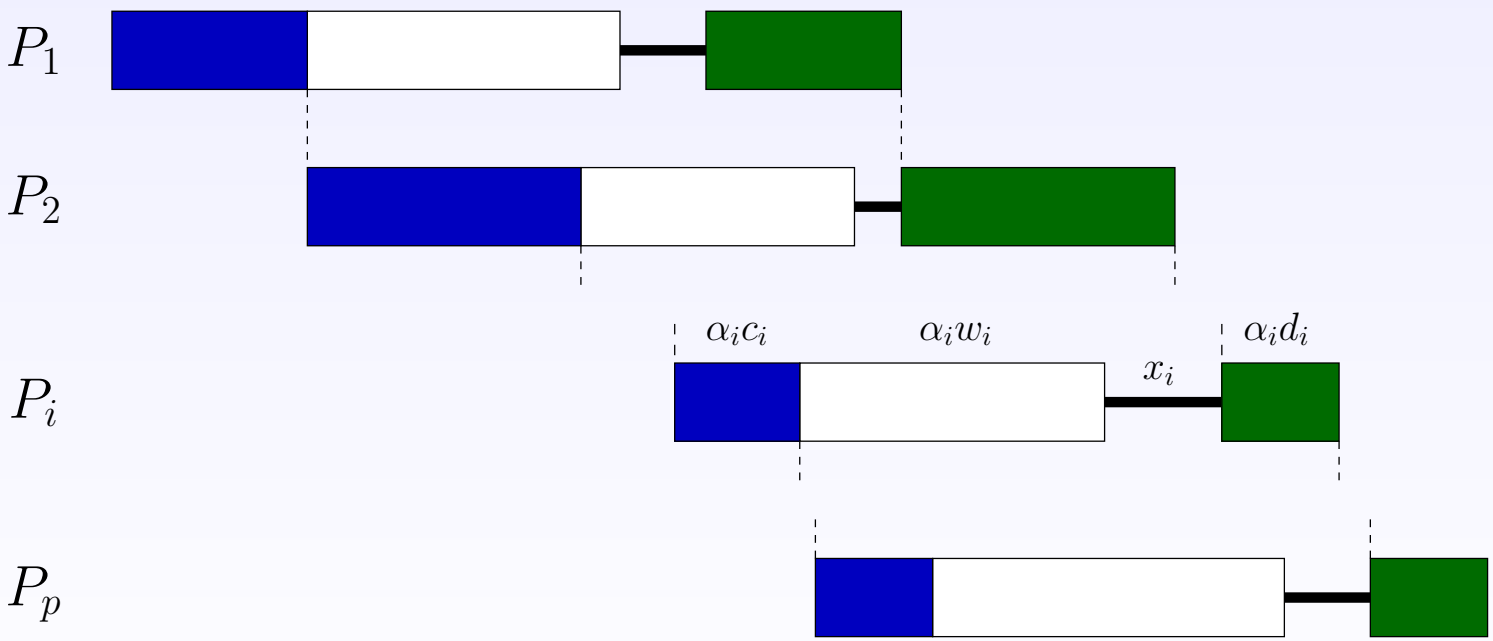
Démonstration : on modifie la plate-forme : $c_i \leftarrow c_i + d_i$ et $d_i \leftarrow 0$



$\Rightarrow$ réduction à un problème classique sans messages de retour.

Stratégies FIFO (1)

- FIFO = First In First Out
- L'ordre d'envoi des résultats est le même que l'ordre d'envoi des données.



Nous nous restreignons au cas $d_i = z \times c_i$ $(z < 1)$

Stratégies FIFO (2)

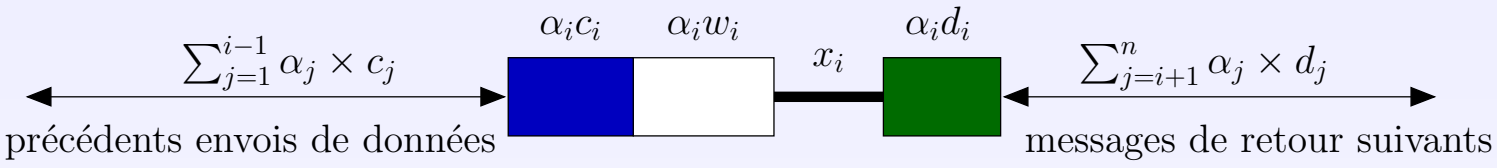
Théorème

Dans la meilleure solution FIFO :

- *Les données sont envoyées par valeurs croissantes de* $c_i + d_i$
- *L'ensemble des processeurs qui travaillent est composé des* q *premiers processeurs dans cet ordre ;* q *peut être déterminé en temps linéaire.*
- *Il n'y a pas de temps d'attente, i.e.* $x_i = 0$ *pour tout* i*.*

Stratégies FIFO (3)

On considère le processeur i dans l'ordonnancement :



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

Nous avons donc : $A\alpha + x = T\mathbb{1}$, où :

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

Stratégies FIFO (4)

On peut écrire $A = L + \mathbb{1}d^T$, avec :

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

La matrice $\mathbb{1}d^t$ est une matrice de rang un, on peut donc utiliser la formule de Sherman-Morrison pour calculer l'inverse de A :

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

Stratégies FIFO (5)

Avec la formule donnant A^{-1}, on peut :

- prouver que pour tout processeur P_i, soit $\alpha_i = 0$ (le processeur ne travaille pas) ou $x_i = 0$ (pas de temps d'attente).
- définir analytiquement le débit $\rho(T) = \sum_i \alpha_i$
- prouver que le débit est meilleur quand $c_1 \leq c_2 \leq c_3 \ldots \leq c_n$
- prouver que le débit est meilleur quand ne travaille que des processeurs vérifiant $d_i \leq \frac{1}{\rho_{\text{opt}}}$

Stratégies FIFO - cas particuliers

- Jusqu'à présent, nous avons supposé que $d_i = z \times c_i$, avec $z < 1$
- Si $z > 1$, solution symétrique (les données sont envoyées par valeurs décroissantes de $d_i + c_i$, on sélectionne les q premiers processeurs dans cet ordre)
- $z = 1 \Rightarrow$ l'ordre n'a pas d'importance (mais tous les processeurs ne travaillent pas forcément)

Plan du cours

Du linéaire à l'affine

- Temps de communications et de calculs : fonctions affines et non plus linéaires.

Du linéaire à l'affine

- Temps de communications et de calculs : fonctions affines et non plus linéaires.
- Dans une solution optimale, tous les processeurs qui *participent* terminent en même temps.

Du linéaire à l'affine

- Temps de communications et de calculs : fonctions affines et non plus linéaires.
- Dans une solution optimale, tous les processeurs qui *participent* terminent en même temps.
- Si la charge est suffisamment grande : 1) tous les processeurs participent ; 2) ils reçoivent les données dans l'ordre des bandes passantes décroissantes.

Du linéaire à l'affine

- Temps de communications et de calculs : fonctions affines et non plus linéaires.
- Dans une solution optimale, tous les processeurs qui *participent* terminent en même temps.
- Si la charge est suffisamment grande : 1) tous les processeurs participent ; 2) ils reçoivent les données dans l'ordre des bandes passantes décroissantes.
- Cas général : programme linéaire mixte (entier et rationnel) au temps de résolution potentiellement exponentiel.

Distribution *multi-round*

- Les données sont distribuées en plusieurs tournées.

Distribution *multi-round*

- Les données sont distribuées en plusieurs tournées.
- Algorithme asymptotiquement optimal de sélection des ressources et de distribution du travail.

Plan du cours

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.