Algorithmes itératifs (de l'influence de la modélisation du réseau)

Frédéric Vivien

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

21 octobre 2005

Plan de l'exposé

Plan de l'exposé

Le contexte : plates-formes distribuées hétérogènes

Nouvelles sources de problèmes

- Hétérogénéité des processeurs
- Hétérogénéité des liens de communications
- Topologie réseau (plus ou moins) inconnue
- Machines et réseau non dédiés

Applications considérées : algorithmes itératifs

- Un ensemble de données (typiquement une matrice)

Applications considérées : algorithmes itératifs

- Un ensemble de données (typiquement une matrice)
- Schéma de l'algorithme :

Applications considérées : algorithmes itératifs

- Un ensemble de données (typiquement une matrice)
- Schéma de l'algorithme :
  1. Chaque processeur effectue un calcul sur sa part des données

Applications considérées : algorithmes itératifs

- Un ensemble de données (typiquement une matrice)
- Schéma de l'algorithme :
  1. Chaque processeur effectue un calcul sur sa part des données
  2. Chaque processeur échange la « frontière » de son ensemble de données avec ses processeurs voisins

Applications considérées : algorithmes itératifs

- Un ensemble de données (typiquement une matrice)
- Schéma de l'algorithme :
  1. Chaque processeur effectue un calcul sur sa part des données
  2. Chaque processeur échange la « frontière » de son ensemble de données avec ses processeurs voisins
  3. On recommence à l'étape 1

Applications considérées : algorithmes itératifs

- Un ensemble de données (typiquement une matrice)
- Schéma de l'algorithme :
  1. Chaque processeur effectue un calcul sur sa part des données
  2. Chaque processeur échange la « frontière » de son ensemble de données avec ses processeurs voisins
  3. On recommence à l'étape 1

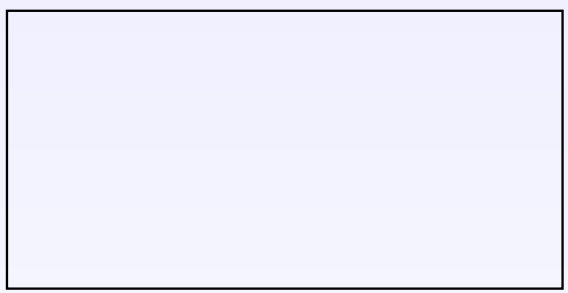
Question : comment exécuter efficacement un tel algorithme sur une telle plate-forme ?

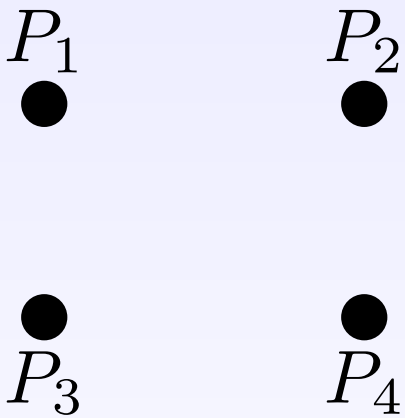
Les questions

- Quels processeurs doivent participer au calcul ?
- Quel fraction des données doit-on leur allouer ?
- Comment doit-on découper l'ensemble des données ?

Simplification préalable : découpage par bande

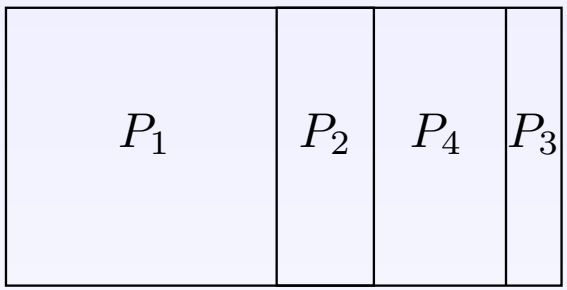
- Données : un tableau 2D

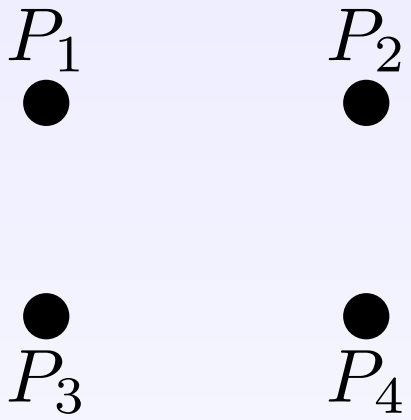




Simplification préalable : découpage par bande

- Données : un tableau 2D

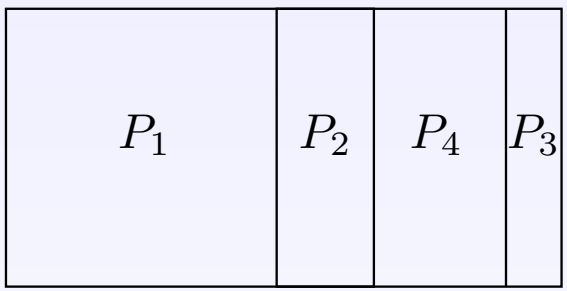


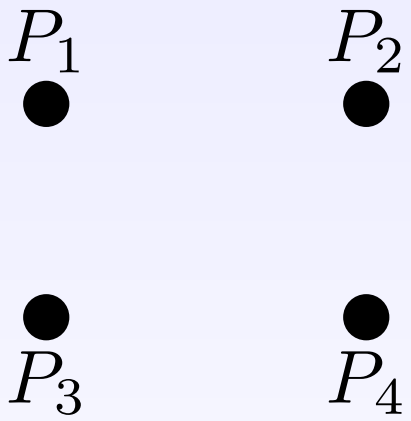


- Découpage unidimensionnel par bandes verticales

Simplification préalable : découpage par bande

- Données : un tableau 2D

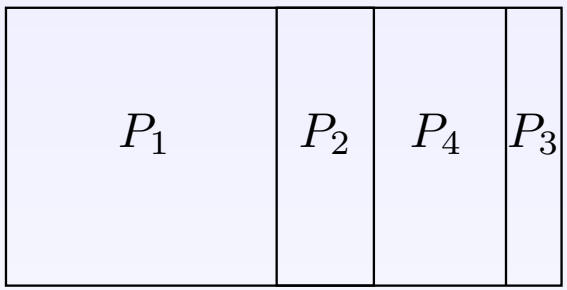


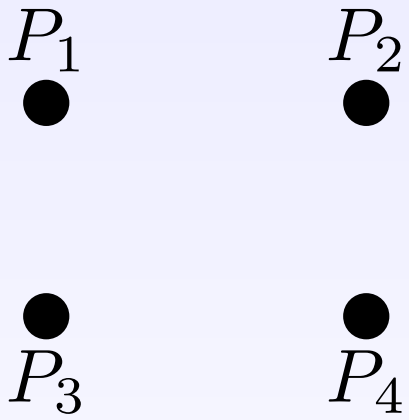


- Découpage unidimensionnel par bandes verticales
- Conséquences :

Simplification préalable : découpage par bande

- Données : un tableau 2D

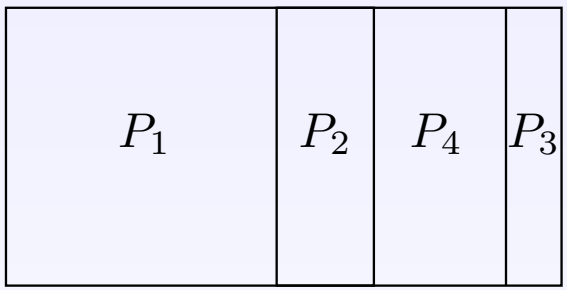


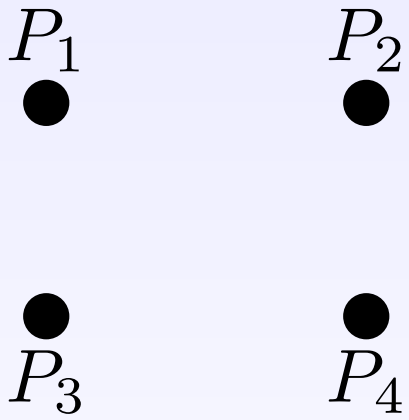


- Découpage unidimensionnel par bandes verticales
- Conséquences :
  1. Détermination simple des frontières et des voisins

Simplification préalable : découpage par bande

- Données : un tableau 2D

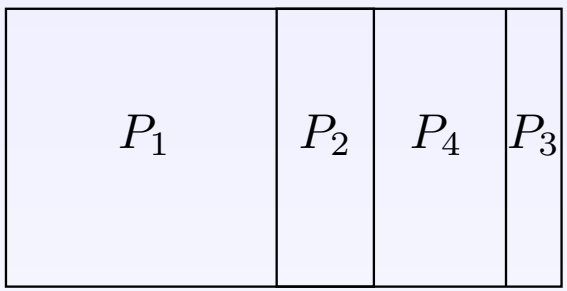


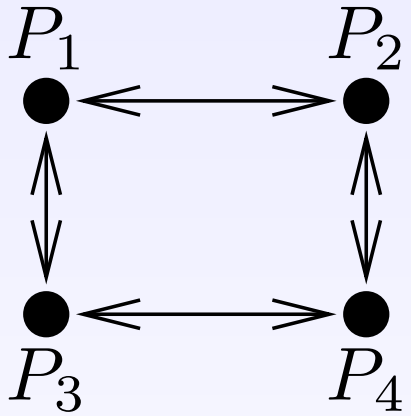


- Découpage unidimensionnel par bandes verticales
- Conséquences :
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants : D_c

Simplification préalable : découpage par bande

- Données : un tableau 2D





- Découpage unidimensionnel par bandes verticales
- Conséquences :
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants : D_c
  3. Les processeurs sont virtuellement organisés en anneau

Notations

- Processeurs : P_1, ..., P_p

Notations

- Processeurs : P_1, ..., P_p
- Le processeur P_i exécute une tâche unitaire en un temps w_i

Notations

- Processeurs : P_1, ..., P_p
- Le processeur P_i exécute une tâche unitaire en un temps w_i
- Quantité total de travail D_w ;
  Part de P_i : $\alpha_i.D_w$ traitée en un temps $\alpha_i.D_w.w_i$
  $(\alpha_i \geq 0, \sum_j \alpha_j = 1)$

Notations

- Processeurs : P_1, ..., P_p
- Le processeur P_i exécute une tâche unitaire en un temps w_i
- Quantité total de travail D_w ;
  Part de P_i : $\alpha_i.D_w$ traitée en un temps $\alpha_i.D_w.w_i$
  $(\alpha_i \geq 0, \sum_j \alpha_j = 1)$

- Coût d'un envoi unitaire de P_i à P_j : $c_{i,j}$

Notations

- Processeurs : P_1, ..., P_p
- Le processeur P_i exécute une tâche unitaire en un temps w_i
- Quantité total de travail D_w ;
  Part de P_i : $\alpha_i.D_w$ traitée en un temps $\alpha_i.D_w.w_i$
  $(\alpha_i \geq 0, \sum_j \alpha_j = 1)$

- Coût d'un envoi unitaire de P_i à P_j : $c_{i,j}$
- Coût d'un envoi de P_i à son successeur dans l'anneau : $D_c.c_{i,\text{succ}(i)}$

Communications : modèle 1-port

Un processeur peut :

- envoyer au plus un message à la fois ;
- recevoir au plus un message à la fois ;
- envoyer un message et en recevoir un simultanément.

Objectif

1. Sélectionner q processeurs parmi p

Objectif

1. Sélectionner q processeurs parmi p
2. Les ordonner en un anneau

Objectif

1. Sélectionner q processeurs parmi p
2. Les ordonner en un anneau
3. Leur répartir les données

Objectif

1. Sélectionner q processeurs parmi p
2. Les ordonner en un anneau
3. Leur répartir les données

Afin de minimiser :

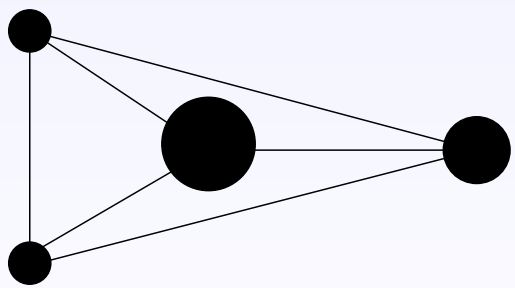
$$\max_{1 \leq i \leq p} \mathbb{I}\{i\}[\alpha_i . D_w . w_i + D_c . (c_{i,\text{pred}(i)} + c_{i,\text{succ}(i)})]$$

Où $\mathbb{I}\{i\}[x] = x$ si P_i participe au calcul et 0 sinon

Plan de l'exposé

Hypothèses particulières

1. Il existe un lien de communication entre chaque paire de processeurs
2. Tous les liens ont même capacité de communication $(\exists c, \forall i, j \ c_{i,j} = c)$



Conséquences

- Soit le processeur le plus rapide fait tout le travail, soit tous les processeurs participent

Conséquences

- Soit le processeur le plus rapide fait tout le travail, soit tous les processeurs participent
- Si tous les processeurs participent, ils finissent tous leur part de calcul en même temps

Conséquences

- Soit le processeur le plus rapide fait tout le travail, soit tous les processeurs participent
- Si tous les processeurs participent, ils finissent tous leur part de calcul en même temps $\alpha_i.D_w$ *rationnels ? ? ?*

Conséquences

- Soit le processeur le plus rapide fait tout le travail, soit tous les processeurs participent
- Si tous les processeurs participent, ils finissent tous leur part de calcul en même temps $\alpha_i.D_w$ *rationnels ? ? ?*
  ($\exists \tau, \quad \alpha_i.D_w.w_i = \tau$, d'où $1 = \sum_i \frac{\tau}{D_w.w_i}$)

Conséquences

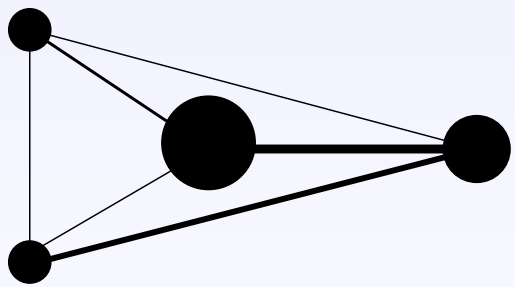
- Soit le processeur le plus rapide fait tout le travail, soit tous les processeurs participent
- Si tous les processeurs participent, ils finissent tous leur part de calcul en même temps $\alpha_i.D_w$ *rationnels ? ? ?*
  ($\exists \tau, \quad \alpha_i.D_w.w_i = \tau$, d'où $1 = \sum_i \frac{\tau}{D_w.w_i}$)
- Temps pour la solution optimale :

$$T_{\text{step}} = \min \left\{ D_w.w_{\min}, D_w.\frac{1}{\sum_i \frac{1}{w_i}} + 2.D_c.c \right\}$$

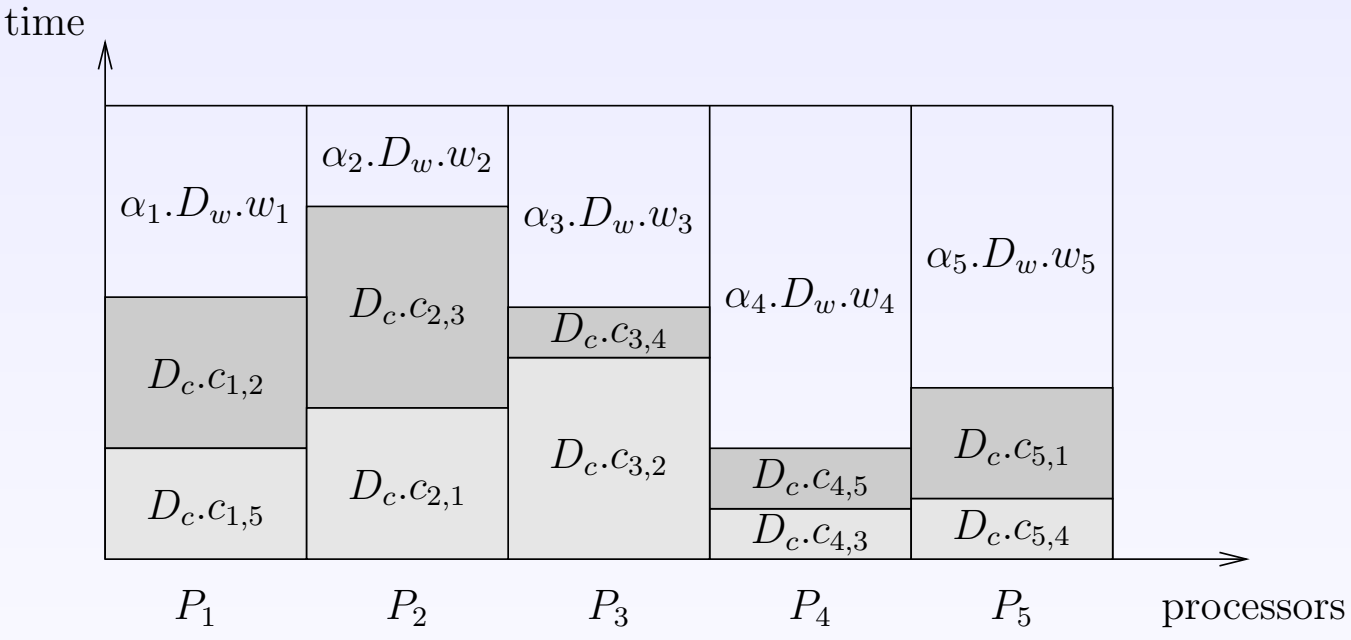
Plan de l'exposé

Hypothèse particulière

1. Il existe un lien de communication entre chaque paire de processeurs



Tous les processeurs participent : étude (1)



Tous les processeurs finissent en même temps

Tous les processeurs participent : étude (2)

- Ils finissent tous en même temps :

$$T_{\text{step}} = \alpha_i.D_w.w_i + D_c.(c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)})$$

Tous les processeurs participent : étude (2)

- Ils finissent tous en même temps :

$$T_{\text{step}} = \alpha_i.D_w.w_i + D_c.(c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)})$$

- $\sum_{i=1}^{p} \alpha_i = 1 \Rightarrow \sum_{i=1}^{p} \frac{T_{\text{step}} - D_c.(c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)})}{D_w.w_i} = 1$. D'où

$$\frac{T_{\text{step}}}{D_w.w_{\text{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$$

où $w_{\text{cumul}} = \frac{1}{\sum_i \frac{1}{w_i}}$

Tous les processeurs participent : interprétation

$$\frac{T_{\mathsf{step}}}{D_w . w_{\mathsf{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\mathsf{succ}(i)} + c_{i,\mathsf{pred}(i)}}{w_i}$$

Tous les processeurs participent : interprétation

$$\frac{T_{\text{step}}}{D_w . w_{\text{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$$

T_{step} est minimal quand $\sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$ est minimal

Tous les processeurs participent : interprétation

$$\frac{T_{\text{step}}}{D_w . w_{\text{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$$

T_{step} est minimal quand $\sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$ est minimal

Recherche d'un cycle hamiltonien de poids minimal dans un graphe où l'arête de P_i à P_j est de poids $d_{i,j} = \frac{c_{i,j}}{w_i} + \frac{c_{j,i}}{w_j}$

Tous les processeurs participent : interprétation

$$\frac{T_{\text{step}}}{D_w . w_{\text{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$$

T_{step} est minimal quand $\sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$ est minimal

Recherche d'un cycle hamiltonien de poids minimal dans un graphe où l'arête de P_i à P_j est de poids $d_{i,j} = \frac{c_{i,j}}{w_i} + \frac{c_{j,i}}{w_j}$

Problème NP-complet

Tous les processeurs participent : programmation linéaire

$$\text{Minimiser } \sum_{i=1}^{p} \sum_{j=1}^{p} d_{i,j}.x_{i,j},$$

$$\text{satisfaisant les (in)équations}$$

$$\begin{cases} (1)\ \sum_{j=1}^{p} x_{i,j} = 1 & 1 \leq i \leq p \\ (2)\ \sum_{i=1}^{p} x_{i,j} = 1 & 1 \leq j \leq p \\ (3)\ x_{i,j} \in \{0,1\} & 1 \leq i,j \leq p \\ (4)\ u_i - u_j + p.x_{i,j} \leq p-1 & 2 \leq i,j \leq p, i \neq j \\ (5)\ u_i \text{ integer}, u_i \geq 0 & 2 \leq i \leq p \end{cases}$$

$x_{i,j} = 1$ si et seulement si l'arête de P_i à P_j est utilisée

Cas général : programmation linéaire

Meilleur anneau de q processeurs

MINIMIZE T SATISFYING THE (IN)EQUATIONS

$$
\begin{cases}
(1)\ x_{i,j} \in \{0,1\} & 1 \leq i,j \leq p \\
(2)\ \sum_{i=1}^{p} x_{i,j} \leq 1 & 1 \leq j \leq p \\
(3)\ \sum_{i=1}^{p} \sum_{j=1}^{p} x_{i,j} = q & \\
(4)\ \sum_{i=1}^{p} x_{i,j} = \sum_{i=1}^{p} x_{j,i} & 1 \leq j \leq p \\
 & \\
(5)\ \sum_{i=1}^{p} \alpha_i = 1 & \\
(6)\ \alpha_i \leq \sum_{j=1}^{p} x_{i,j} & 1 \leq i \leq p \\
(7)\ \alpha_i.w_i + \frac{D_c}{D_w} \sum_{j=1}^{p} (x_{i,j}c_{i,j} + x_{j,i}c_{j,i}) \leq T & 1 \leq i \leq p \\
 & \\
(8)\ \sum_{i=1}^{p} y_i = 1 & \\
(9)\ -p.y_i - p.y_j + u_i - u_j + q.x_{i,j} \leq q-1 & 1 \leq i,j \leq p, i \neq j \\
(10)\ y_i \in \{0,1\} & 1 \leq i \leq p \\
(11)\ u_i \text{ integer}, u_i \geq 0 & 1 \leq i \leq p
\end{cases}
$$

La programmation linéaire

- Problèmes en rationnels : résolution en temps polynomial (en la taille du problème).

La programmation linéaire

- Problèmes en rationnels : résolution en temps polynomial (en la taille du problème).
- Problèmes en entier : résolution en temps exponentiel dans le pire cas.

La programmation linéaire

- Problèmes en rationnels : résolution en temps polynomial (en la taille du problème).
- Problèmes en entier : résolution en temps exponentiel dans le pire cas.
- Pas de relaxation en rationnels possible...

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (Lin-Kernighan)

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (Lin-Kernighan)
Pas de garanties, mais en pratique les résultats sont excellents.

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (Lin-Kernighan)
Pas de garanties, mais en pratique les résultats sont excellents.

Cas général.

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (Lin-Kernighan) Pas de garanties, mais en pratique les résultats sont excellents.

Cas général.

1. Recherche exhaustive : faisable jusqu'à une douzaine de processeurs...

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (Lin-Kernighan) Pas de garanties, mais en pratique les résultats sont excellents.

Cas général.

1. Recherche exhaustive : faisable jusqu'à une douzaine de processeurs...
2. Heuristique gloutonne : initialement on prend la meilleure paire de processeurs ; pour un anneau donné on essaye d'insérer n'importe quel processeur non utilisé entre chaque paire de voisins de l'anneau...

Plan de l'exposé

Nouvelle difficulté : partage de liens de communications

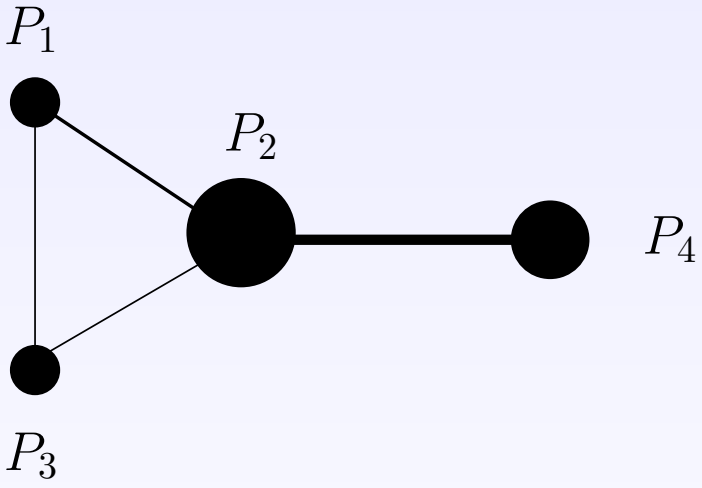
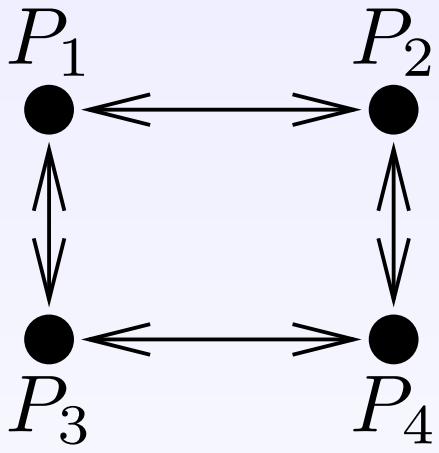


Plate-forme hétérogène



Anneau virtuel

Nouvelle difficulté : partage de liens de communications

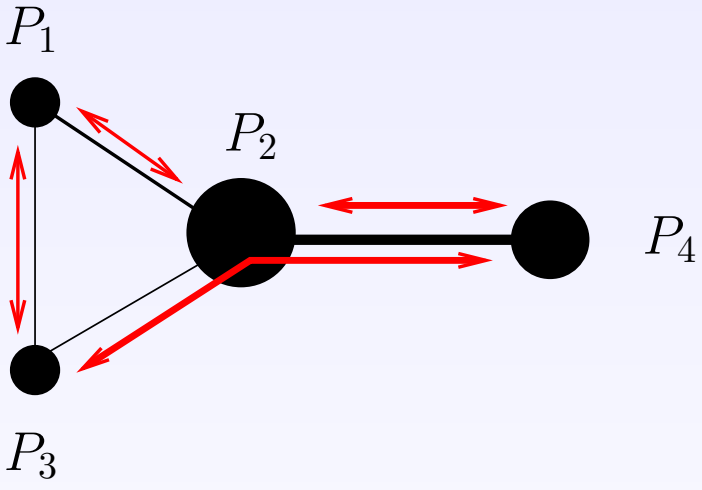
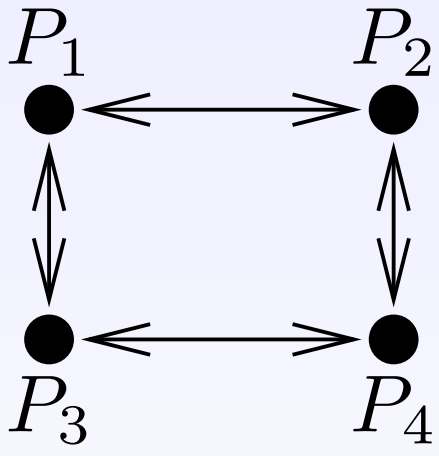


Plate-forme hétérogène



Anneau virtuel

Nouvelle difficulté : partage de liens de communications

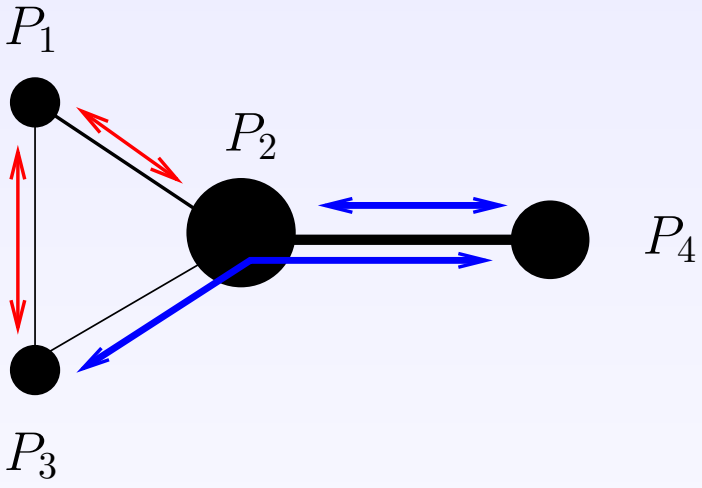
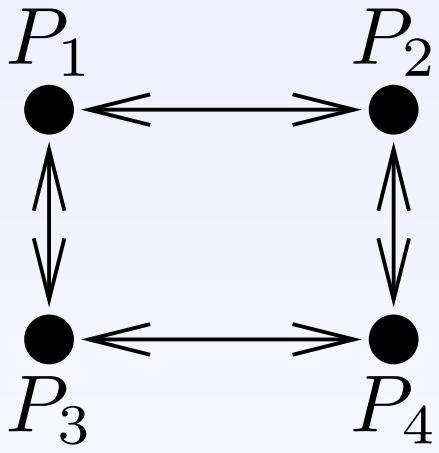


Plate-forme hétérogène



Anneau virtuel

Nouvelle difficulté : partage de liens de communications

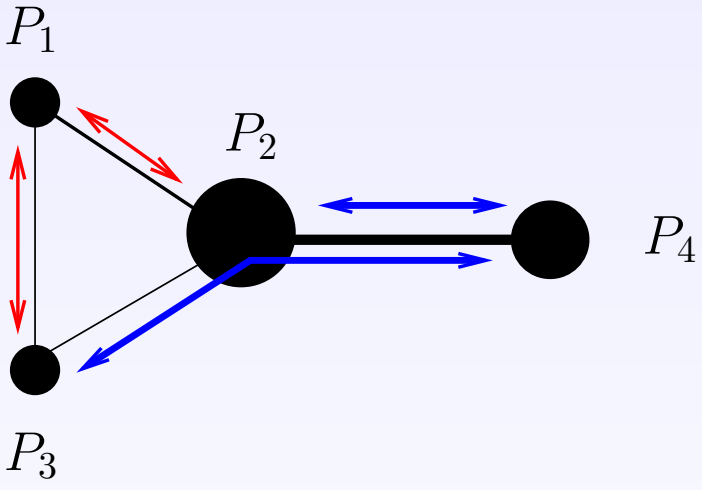
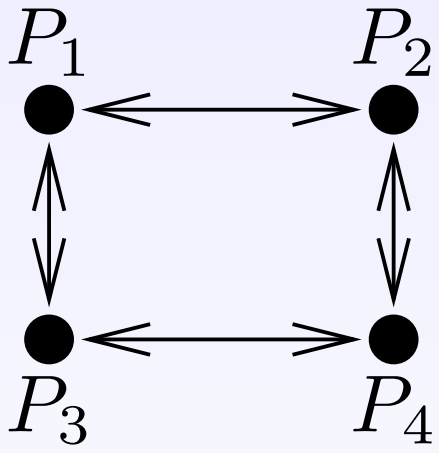


Plate-forme hétérogène



Anneau virtuel

If faut tenir compte du partage des liens de communications.

Nouvelles notations

- Un ensemble de liens de communications : e_1, ..., e_n

Nouvelles notations

- Un ensemble de liens de communications : e_1, ..., e_n
- Bande passante de e_m : b_{e_m}

Nouvelles notations

- Un ensemble de liens de communications : e_1, ..., e_n
- Bande passante de e_m : b_{e_m}
- Il y a un chemin $\mathcal{S}_i$ de P_i à $P_{\mathsf{succ}(i)}$ dans le réseau

Nouvelles notations

- Un ensemble de liens de communications : e_1, ..., e_n
- Bande passante de e_m : b_{e_m}
- Il y a un chemin $\mathcal{S}_i$ de P_i à $P_{\mathsf{succ}(i)}$ dans le réseau
  - $\mathcal{S}_i$ utilise une partie $s_{i,m}$ de la bande passante b_{e_m} du lien e_m

Nouvelles notations

- Un ensemble de liens de communications : e_1, ..., e_n
- Bande passante de e_m : b_{e_m}
- Il y a un chemin $\mathcal{S}_i$ de P_i à $P_{\text{succ}(i)}$ dans le réseau
  - $\mathcal{S}_i$ utilise une partie $s_{i,m}$ de la bande passante b_{e_m} du lien e_m
  - P_i a besoin d'un temps $D_c.\frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ pour envoyer à son successeur un message de taille D_c

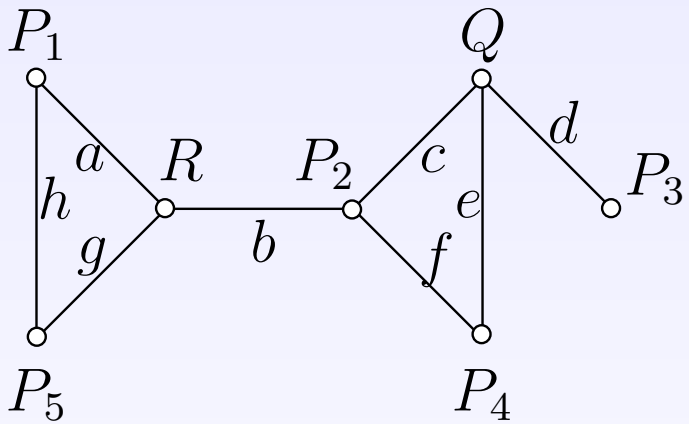
Nouvelles notations

- Un ensemble de liens de communications : e_1, ..., e_n
- Bande passante de e_m : b_{e_m}
- Il y a un chemin $\mathcal{S}_i$ de P_i à $P_{\text{succ}(i)}$ dans le réseau
  - $\mathcal{S}_i$ utilise une partie $s_{i,m}$ de la bande passante b_{e_m} du lien e_m
  - P_i a besoin d'un temps $D_c.\frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ pour envoyer à son successeur un message de taille D_c
  - Contraintes sur la bande passante de e_m : $\sum_{1 \leq i \leq p} s_{i,m} \leq b_{e_m}$

Nouvelles notations

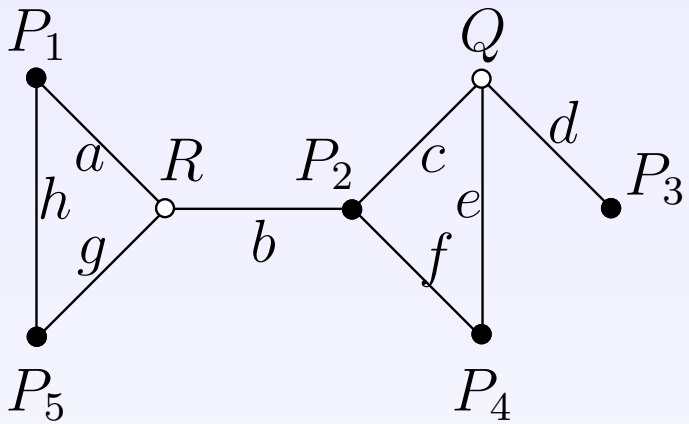
- Un ensemble de liens de communications : e_1, ..., e_n
- Bande passante de e_m : b_{e_m}
- Il y a un chemin $\mathcal{S}_i$ de P_i à $P_{\text{succ}(i)}$ dans le réseau
  - $\mathcal{S}_i$ utilise une partie $s_{i,m}$ de la bande passante b_{e_m} du lien e_m
  - P_i a besoin d'un temps $D_c.\frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ pour envoyer à son successeur un message de taille D_c
  - Contraintes sur la bande passante de e_m : $\sum_{1 \leq i \leq p} s_{i,m} \leq b_{e_m}$
- De même il y a un chemin $\mathcal{P}_i$ de P_i à $P_{\text{pred}(i)}$ dans le réseau, qui utilise une partie $p_{i,m}$ de la bande passante b_{e_m} du lien e_m

Exemple jouet : choix de l'anneau



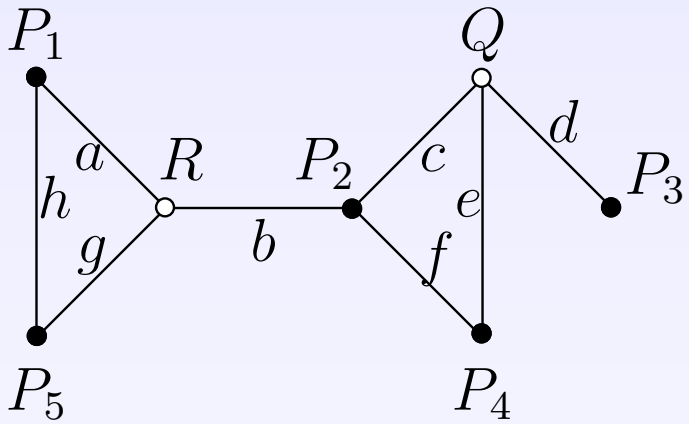
- 7 processeurs et 8 liens de communications bidirectionnels

Exemple jouet : choix de l'anneau

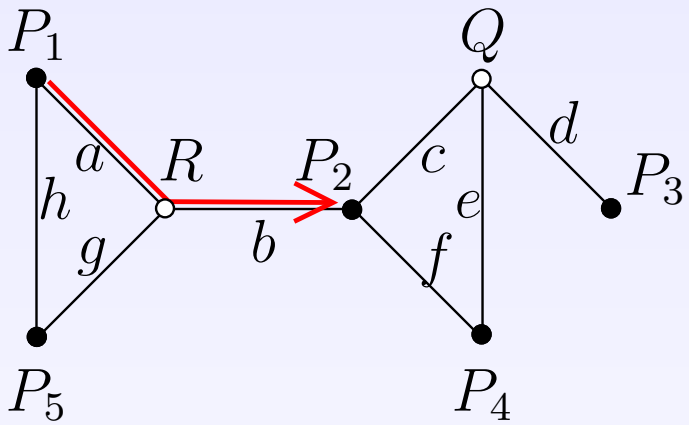


- 7 processeurs et 8 liens de communications bidirectionnels
- On choisit un anneau de 5 processeurs :
  $P_1 \to P_2 \to P_3 \to P_4 \to P_5$ (on n'utilise ni Q, ni R)

Exemple jouet : choix des chemins

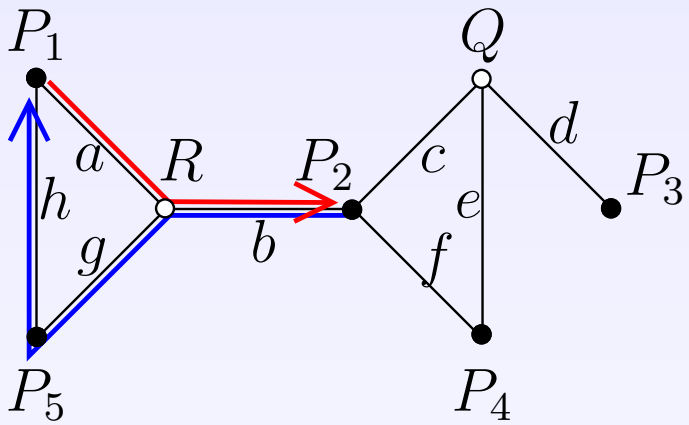


Exemple jouet : choix des chemins



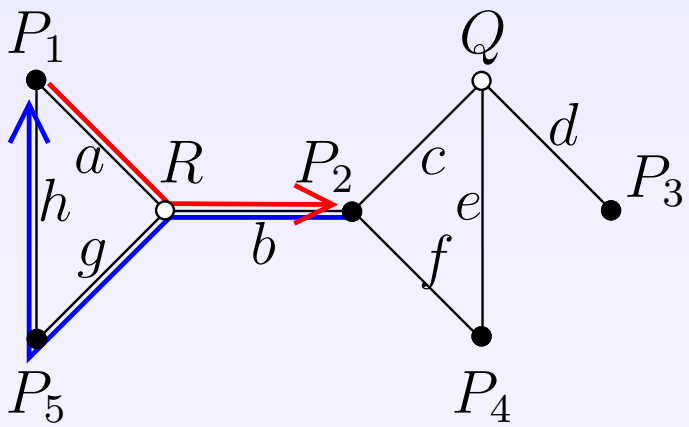
De P_1 à P_2, on utilise les liens a et b : $\mathcal{S}_1 = \{a, b\}$.

Exemple jouet : choix des chemins



De P_1 à P_2, on utilise les liens a et b : $\mathcal{S}_1 = \{a, b\}$.
De P_2 à P_1, on utilise les liens b, g et h : $\mathcal{P}_2 = \{b, g, h\}$.

Exemple jouet : choix des chemins



De P_1 à P_2, on utilise les liens a et b : $\mathcal{S}_1 = \{a, b\}$.
De P_2 à P_1, on utilise les liens b, g et h : $\mathcal{P}_2 = \{b, g, h\}$.

De P_1 : à P_2, $\mathcal{S}_1 = \{a, b\}$ et à P_5, $\mathcal{P}_1 = \{h\}$
De P_2 : à P_3, $\mathcal{S}_2 = \{c, d\}$ et à P_1, $\mathcal{P}_2 = \{b, g, h\}$
De P_3 : à P_4, $\mathcal{S}_3 = \{d, e\}$ et à P_2, $\mathcal{P}_3 = \{d, e, f\}$
De P_4 : à P_5, $\mathcal{S}_4 = \{f, b, g\}$ et à P_3, $\mathcal{P}_4 = \{e, d\}$
De P_5 : à P_1, $\mathcal{S}_5 = \{h\}$ et à P_4, $\mathcal{P}_5 = \{g, b, f\}$

Exemple jouet : partage des bandes passantes

De P_1 à P_2 on utilise les liens a et b : $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$.

De P_1 à P_5 on utilise le lien h : $c_{1,5} = \frac{1}{p_{1,h}}$.

Exemple jouet : partage des bandes passantes

De P_1 à P_2 on utilise les liens a et b : $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$.

De P_1 à P_5 on utilise le lien h : $c_{1,5} = \frac{1}{p_{1,h}}$.

Ensemble de toutes les contraintes de partage :

Lien a : $s_{1,a} \leq b_a$

Lien b : $s_{1,b} + s_{4,b} + p_{2,b} + p_{5,b} \leq b_b$

Lien c : $s_{2,c} \leq b_c$

Lien d : $s_{2,d} + s_{3,d} + p_{3,d} + p_{4,d} \leq b_d$

Lien e : $s_{3,e} + p_{3,e} + p_{4,e} \leq b_e$

Lien f : $s_{4,f} + p_{3,f} + p_{5,f} \leq b_f$

Lien g : $s_{4,g} + p_{2,g} + p_{5,g} \leq b_g$

Lien h : $s_{5,h} + p_{1,h} + p_{2,h} \leq b_h$

Exemple jouet : système quadratique final

MINIMISER $\max_{1\leq i\leq 5}\ (\alpha_i.D_w.w_i + D_c.(c_{i,i-1} + c_{i,i+1}))$ SOUS LES CONTRAINTES

$$
\begin{cases}
\sum_{i=1}^{5} \alpha_i = 1 & & \\
s_{1,a} \leq b_a & s_{1,b} + s_{4,b} + p_{2,b} + p_{5,b} \leq b_b & s_{2,c} \leq b_c \\
s_{2,d} + s_{3,d} + p_{3,d} + p_{4,d} \leq b_d & s_{3,e} + p_{3,e} + p_{4,e} \leq b_e & s_{4,f} + p_{3,f} + p_{5,f} \leq b_f \\
s_{4,g} + p_{2,g} + p_{5,g} \leq b_g & s_{5,h} + p_{1,h} + p_{2,h} \leq b_h & \\
s_{1,a}.c_{1,2} \geq 1 & s_{1,b}.c_{1,2} \geq 1 & p_{1,h}.c_{1,5} \geq 1 \\
s_{2,c}.c_{2,3} \geq 1 & s_{2,d}.c_{2,3} \geq 1 & p_{2,b}.c_{2,1} \geq 1 \\
p_{2,g}.c_{2,1} \geq 1 & p_{2,h}.c_{2,1} \geq 1 & s_{3,d}.c_{3,4} \geq 1 \\
s_{3,e}.c_{3,4} \geq 1 & p_{3,d}.c_{3,2} \geq 1 & p_{3,e}.c_{3,2} \geq 1 \\
p_{3,f}.c_{3,2} \geq 1 & s_{4,f}.c_{4,5} \geq 1 & s_{4,b}.c_{4,5} \geq 1 \\
s_{4,g}.c_{4,5} \geq 1 & p_{4,e}.c_{4,3} \geq 1 & p_{4,d}.c_{4,3} \geq 1 \\
s_{5,h}.c_{5,1} \geq 1 & p_{5,g}.c_{5,4} \geq 1 & p_{5,b}.c_{5,4} \geq 1 \\
p_{5,f}.c_{5,4} \geq 1 &
\end{cases}
$$

Exemple jouet : moralité

Le problème est un système quadratique si

1. Les processeurs sont sélectionnés ;

Exemple jouet : moralité

Le problème est un système quadratique si

1. Les processeurs sont sélectionnés ;
2. Les processeurs sont ordonnés en anneau ;

Exemple jouet : moralité

Le problème est un système quadratique si

1. Les processeurs sont sélectionnés ;
2. Les processeurs sont ordonnés en anneau ;
3. Les chemins de communications entre les processeurs sont connus.

Exemple jouet : moralité

Le problème est un système quadratique si

1. Les processeurs sont sélectionnés ;
2. Les processeurs sont ordonnés en anneau ;
3. Les chemins de communications entre les processeurs sont connus.

Autrement dit : système quadratique si l'anneau est connu.

Exemple jouet : moralité

Le problème est un système quadratique si

1. Les processeurs sont sélectionnés ;
2. Les processeurs sont ordonnés en anneau ;
3. Les chemins de communications entre les processeurs sont connus.

Autrement dit : système quadratique si l'anneau est connu.

Si l'anneau est connu :

- Graphe complet : formule.
- Graphe quelconque : système quadratique.

Et en pratique ?

On adapte notre heuristique gloutonne :

1. Initialement : meilleure paire de processeurs
2. Pour chaque processeur P_k (non encore dans l'anneau)
   - Pour chaque paire (P_i, P_j) de voisins dans l'anneau
     1. On construit le graphe des bandes passantes non utilisées (Sans considérer les chemins entre P_i et P_j)
     2. On calcule des plus courts chemins (en terme de bandes passantes) entre P_k et P_i et P_j
     3. On évalue la solution
3. On garde la meilleure solution trouvée à l'étape 2 et on recommence

+ raffinements (*max-min fairness*, résolution quadratique)

Est-ce bien raisonnable ?

- Aucune garantie, ni théorique, ni pratique

Est-ce bien raisonnable ?

- Aucune garantie, ni théorique, ni pratique
- Solution simple :

Est-ce bien raisonnable ?

- Aucune garantie, ni théorique, ni pratique
- Solution simple :
  1. on construit le graphe complet dont les arêtes sont étiquetées par les bandes passantes des meilleurs chemins

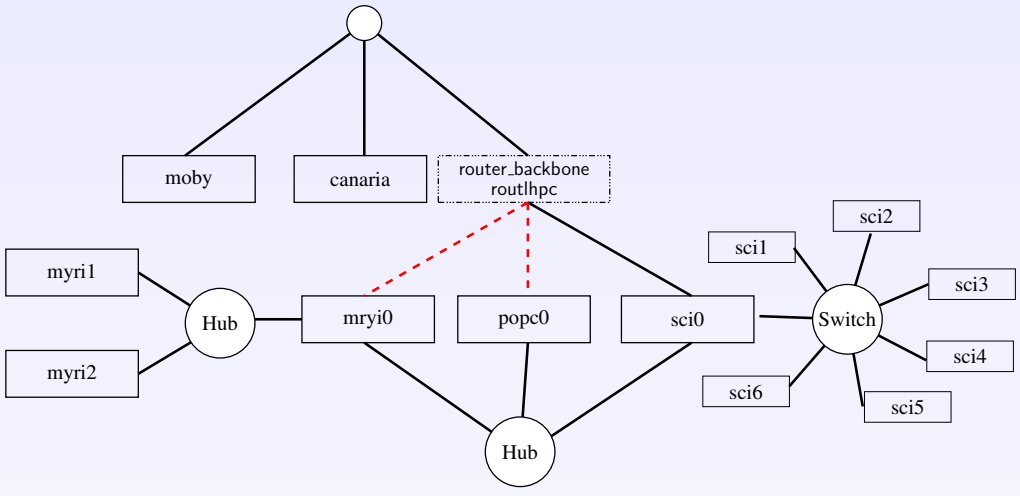
Est-ce bien raisonnable ?

- Aucune garantie, ni théorique, ni pratique
- Solution simple :
  1. on construit le graphe complet dont les arêtes sont étiquetées par les bandes passantes des meilleurs chemins
  2. on applique l'heuristique pour graphe complet

Est-ce bien raisonnable ?

- Aucune garantie, ni théorique, ni pratique
- Solution simple :
  1. on construit le graphe complet dont les arêtes sont étiquetées par les bandes passantes des meilleurs chemins
  2. on applique l'heuristique pour graphe complet
  3. on alloue les bandes passantes

Description d'une plate-forme(Lyon)



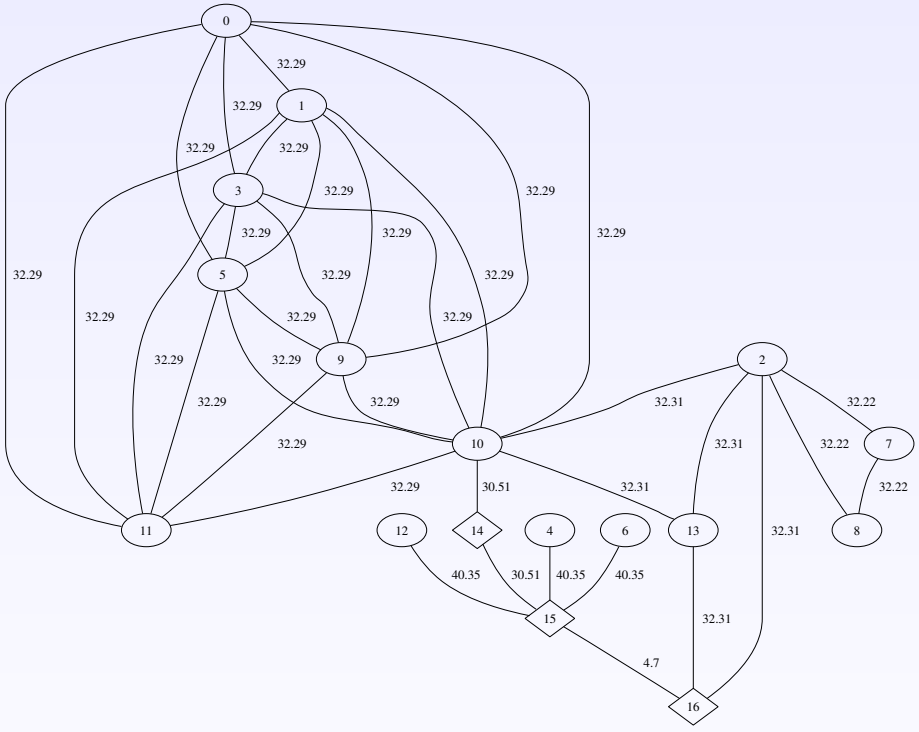
Topologie

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
0.0206	0.0206	0.0206	0.0206	0.0291	0.0206	0.0087	0.0206	0.0206

P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}
0.0206	0.0206	0.0206	0.0291	0.0451	0	0	0

Temps de traitement des processeurs (en secondes par megaflop)

Description de la plate-forme de Lyon



Abstraction de la plate-forme de Lyon.

Résultats

Première heuristique construction de l'anneau sans tenir compte du partage des liens

Seconde heuristique en tenant compte du partage de liens (et avec programmation quadratique)

Ratio D_c/D_w	H1	H2	Gain
0.64	0.008738 (1)	0.008738 (1)	0%
0.064	0.018837 (13)	0.006639 (14)	64.75%
0.0064	0.003819 (13)	0.001975 (14)	48.28%

Ratio D_c/D_w	H1	H2	Gain
0.64	0.005825 (1)	0.005825 (1)	0 %
0.064	0.027919 (8)	0.004865 (6)	82.57%
0.0064	0.007218 (13)	0.001608 (8)	77.72%

TAB.: T_{step}/D_w pour chaque heuristique sur les plates-formes de Lyon et Strasbourg (les nombres entre parenthèses indiquent les tailles des anneaux construits).

Plan de l'exposé

Nouvelles difficultés

La puissance de calcul disponible pour chaque processeur varie au cours du temps

La bande passante disponible de chaque lien réseau varie au cours du temps

$\Rightarrow$ Nécessité de remettre en cause les allocations effectuées

$\Rightarrow$ Introduction de dynamicité dans une approche statique

Une approche possible

- Si les performances constatées diffèrent « trop » des caractéristiques utilisées pour déterminer l'allocation
  - Si les performances diffèrent « beaucoup »
    - On calcule un nouvel anneau
    - On redistribue les données d'un anneau à l'autre
  - Si les performances diffèrent « peu »
    - On calcule un nouvel équilibrage de la charge dans l'anneau existant
    - On redistribue les données dans l'anneau

Une approche possible

- Si les performances constatées diffèrent « trop » des caractéristiques utilisées pour déterminer l'allocation

  Critère définissant « trop » ?

  - Si les performances diffèrent « beaucoup »
    - On calcule un nouvel anneau
    - On redistribue les données d'un anneau à l'autre
      Critère définissant « beaucoup » ?
      Coût de la redistribution ?
  - Si les performances diffèrent « peu »
    - On calcule un nouvel équilibrage de la charge dans l'anneau existant
    - On redistribue les données dans l'anneau
      Comment effectuer efficacement la redistribution ?

Principe du rééquilibrage

Principe : on ne modifie l'anneau que si c'est profitable.

- T_{step} : durée d'une itération *avant* rééquilibrage ;
- T'_{step} : durée d'une itération *après* rééquilibrage ;
- $T_{\text{redistribution}}$: coût de la redistribution ;
- n_{iter} : nombre d'itérations

Condition : $$T_{\text{redistribution}} + n_{\text{iter}} \times T'_{\text{step}} \leq n_{\text{iter}} \times T_{\text{step}}$$

Rééquilibrage sur un anneau

- Anneau unidirectionnel homogène
- Anneau unidirectionnel hétérogène
- Anneau bidirectionnel homogène
- Anneau bidirectionnel hétérogène

Notations

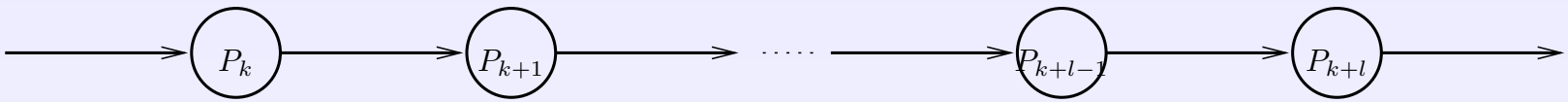
- $C_{k,l}$ l'ensemble des processeurs de P_k à P_l :

$$C_{k,l} = P_k, P_{k+1}, ..., P_l$$

- $c_{i,i+1}$: temps nécessaire au processeur P_i pour envoyer une donnée au processeur P_{i+1} (suivant dans l'anneau).

- Initialement, le processeur P_i détient L_i données (atomiques).
  Après redistribution, P_i détiendra $L_i - \delta_i$ données.
  δ_i est le déséquilibre du processeur P_i.
  $\delta_{k,l}$: déséquilibre de l'ensemble $C_{k,l}$: $\delta_{k,l} = \sum_{i=k}^{l} \delta_i$.
  Loi de conservation des données : $\sum_i \delta_i = 0$
  On suppose que chaque processeur contient au moins une donnée avant et après la redistribution : $L_i \geq 1$ et $L_i \geq 1 + \delta_i$.

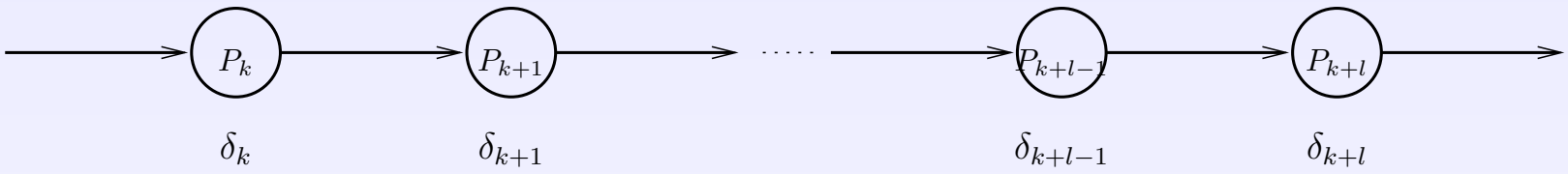
Cadre de travail



Temps de communication homogène : c.

P_k peut seulement envoyer des messages à P_{k+1}.

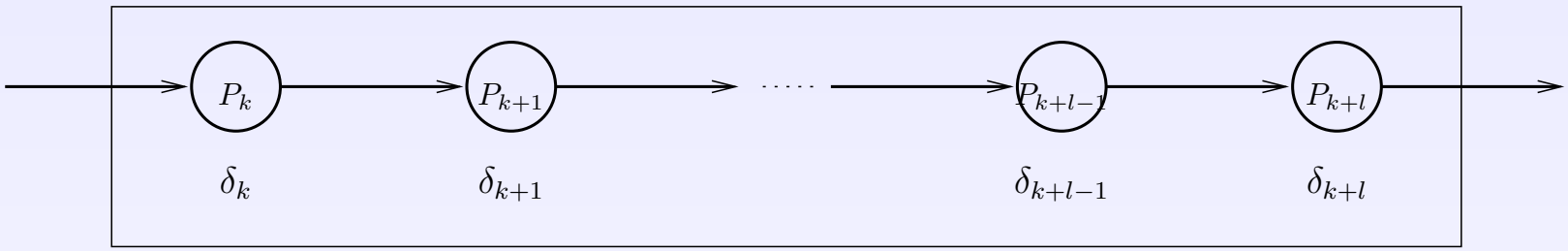
Borne inférieure sur la durée de la redistribution



Temps de communication homogène : c.

P_k peut seulement envoyer des messages à P_{k+1}.

Borne inférieure sur la durée de la redistribution

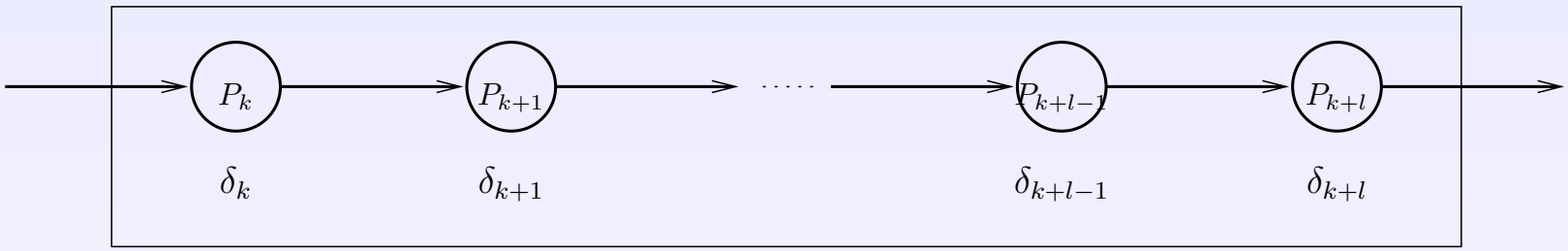


$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Temps de communication homogène : c.

P_k peut seulement envoyer des messages à P_{k+1}.

Borne inférieure sur la durée de la redistribution



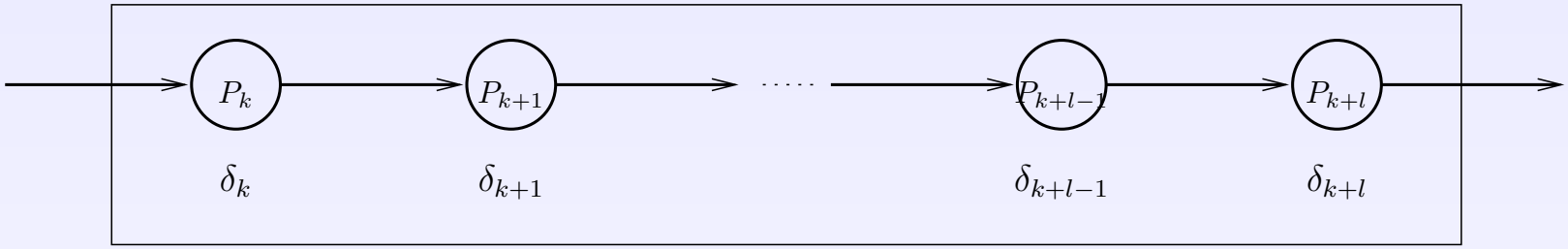
$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Temps de communication homogène : c.

P_k peut seulement envoyer des messages à P_{k+1}.

P_l a besoin d'un temps $\delta_{k,l} \times c$ pour envoyer $\delta_{k,l}$ données (si $\delta_{k,l} > 0$).

Borne inférieure sur la durée de la redistribution



$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

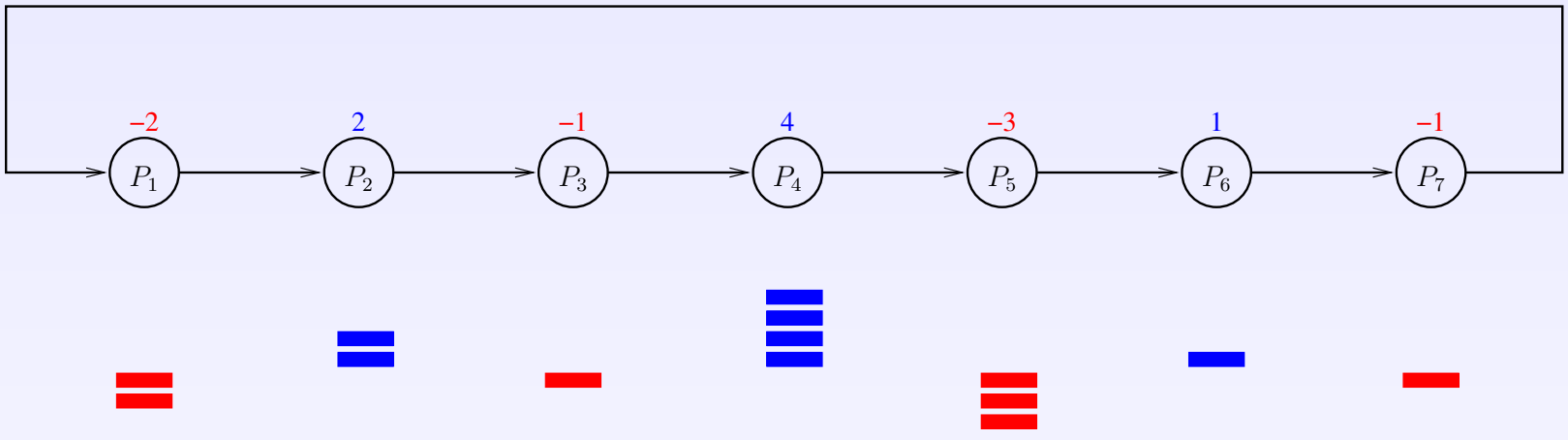
Temps de communication homogène : c.

P_k peut seulement envoyer des messages à P_{k+1}.

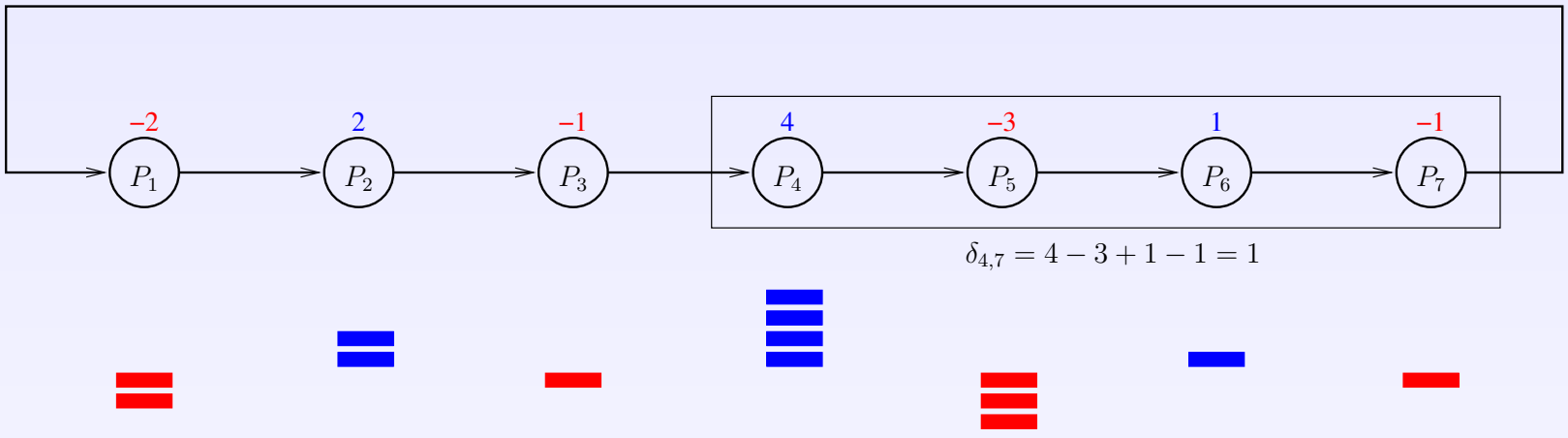
P_l a besoin d'un temps $\delta_{k,l} \times c$ pour envoyer $\delta_{k,l}$ données (si $\delta_{k,l} > 0$).

Borne inférieure : $$\left(\max_{1 \leq k \leq n,\ 0 \leq l \leq n-1} \delta_{k,k+l} \right) \times c$$

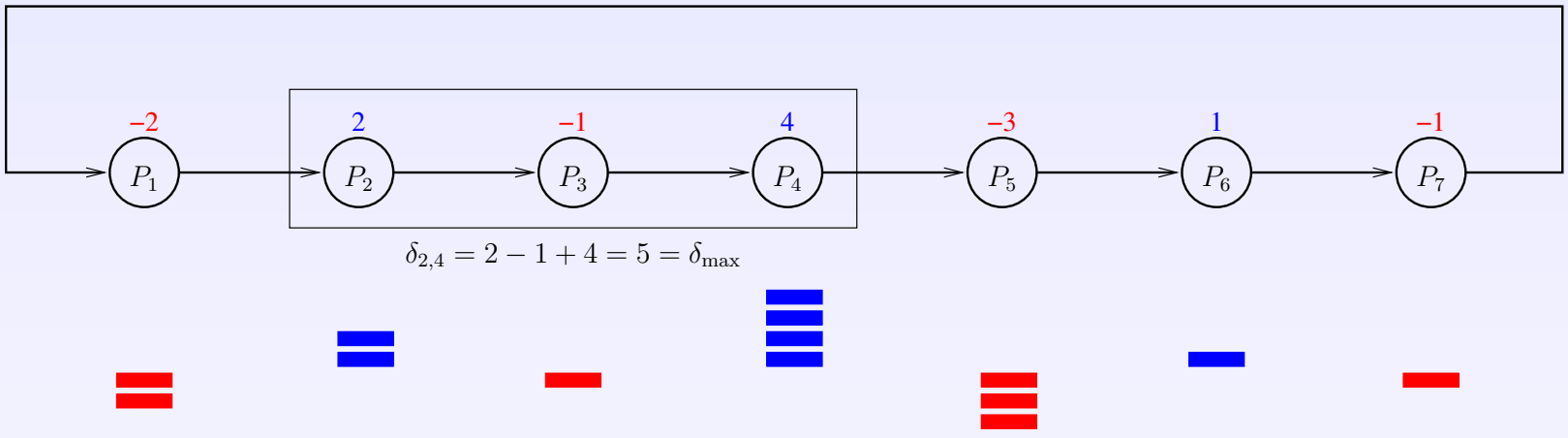
Algorithme de redistribution



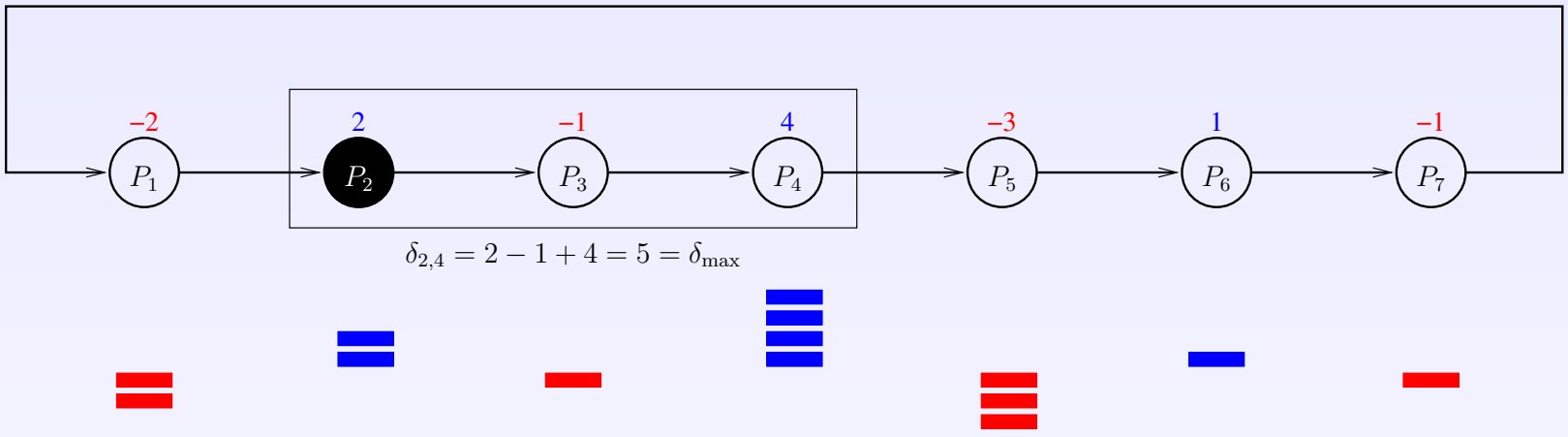
Algorithme de redistribution



Algorithme de redistribution



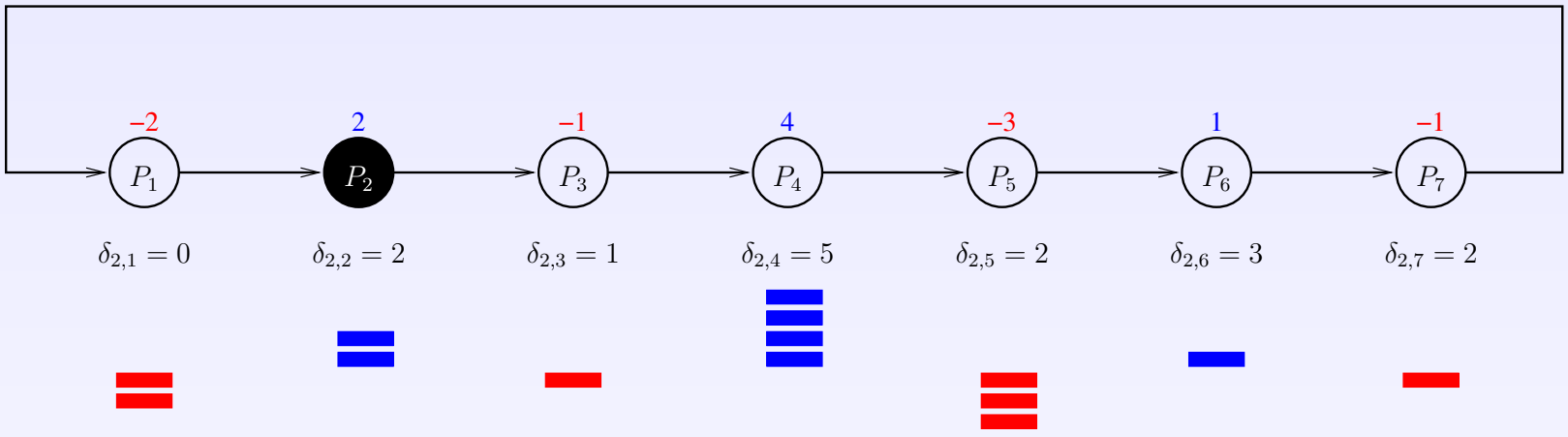
Algorithme de redistribution



$\delta_{\max} = 5$

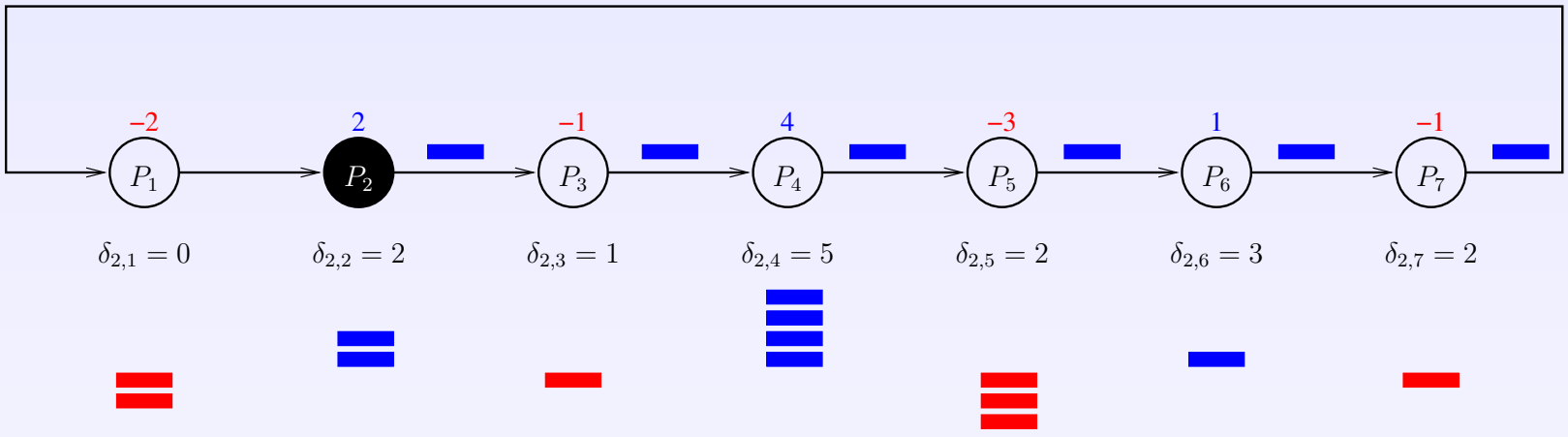
L'algorithme de redistribution est défini par le premier processeur d'une « chaîne » de processeurs de déséquilibre maximal.

Algorithme de redistribution



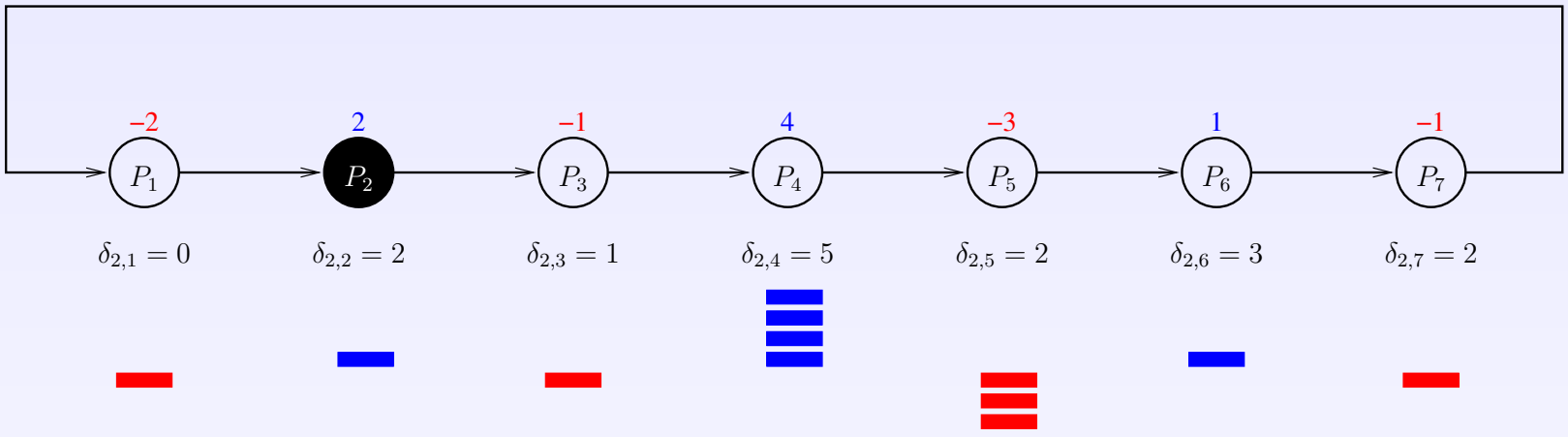
Pendant l'algorithme, le processeur P_i envoie $\delta_{2,i}$ données.

Algorithme de redistribution



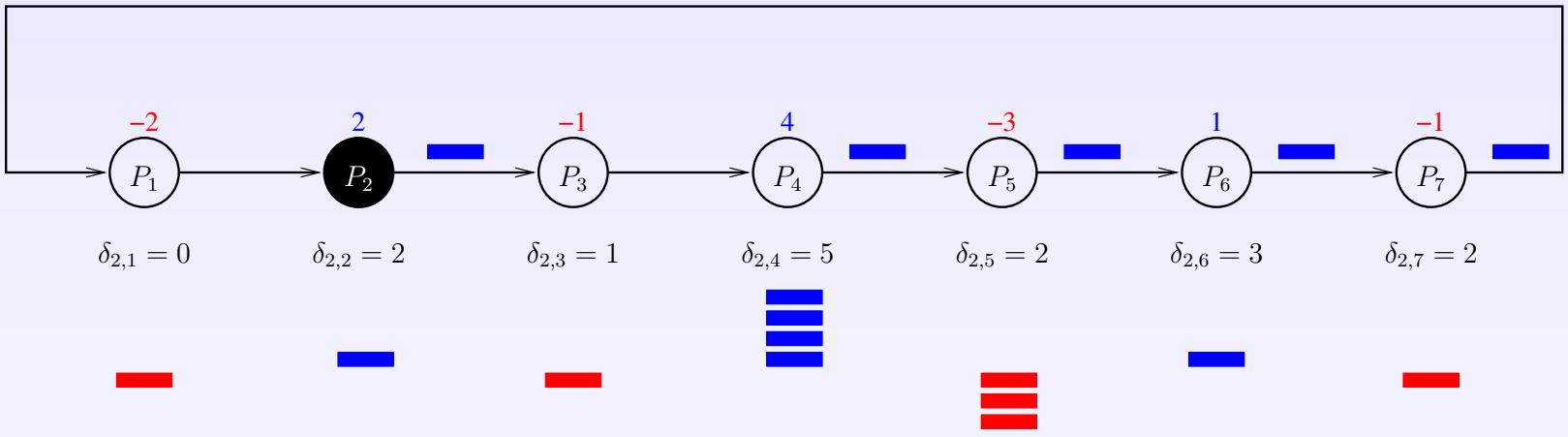
À l'étape 1, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 1$

Algorithme de redistribution



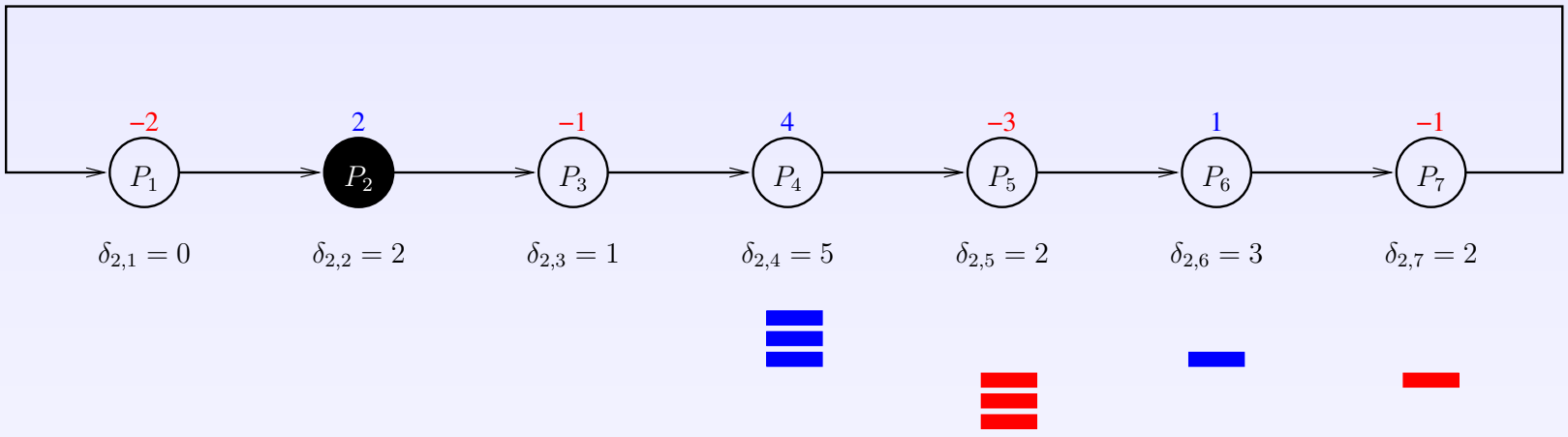
À l'étape 1, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 1$

Algorithme de redistribution



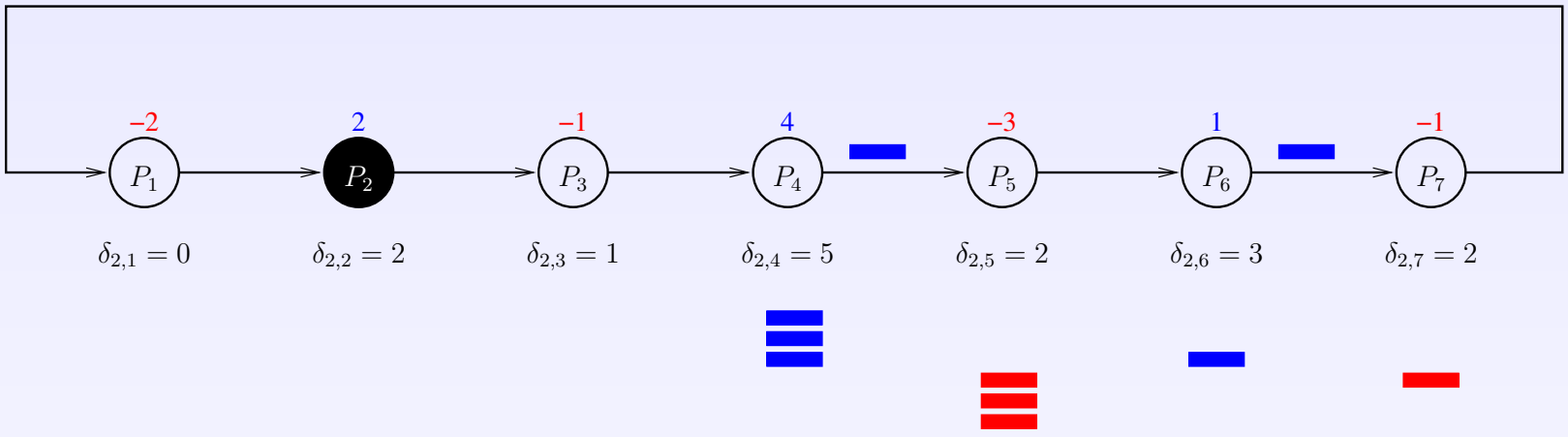
À l'étape 2, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 2$

Algorithme de redistribution



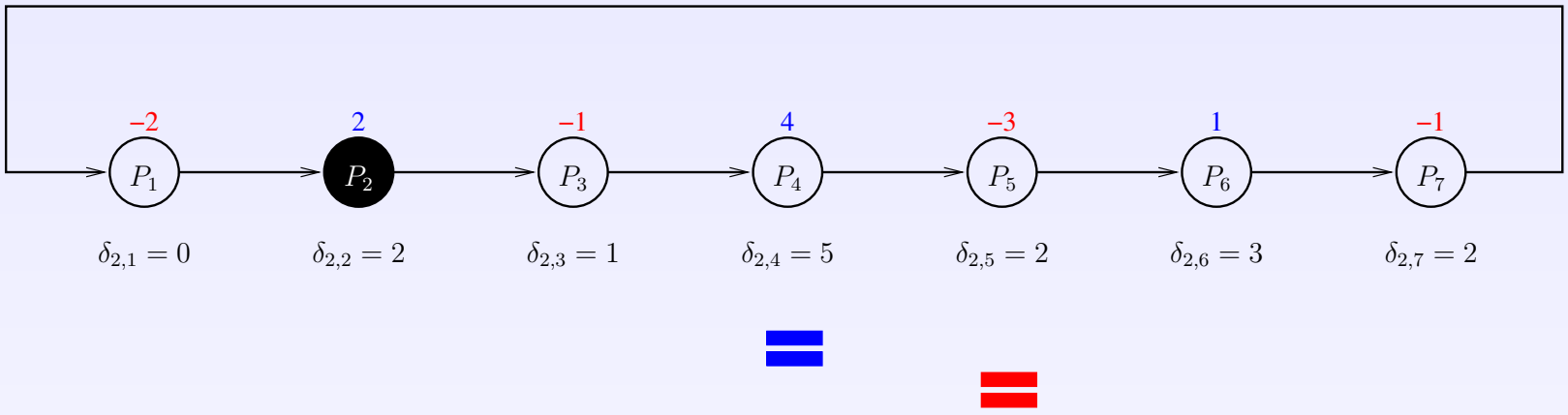
À l'étape 2, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 2$

Algorithme de redistribution



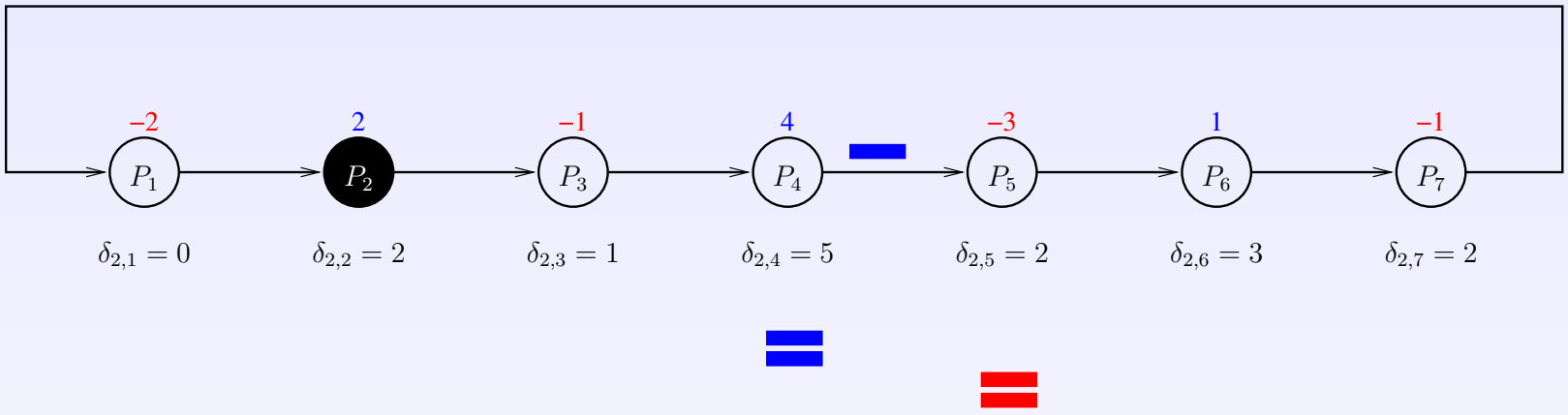
À l'étape 3, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 3$

Algorithme de redistribution



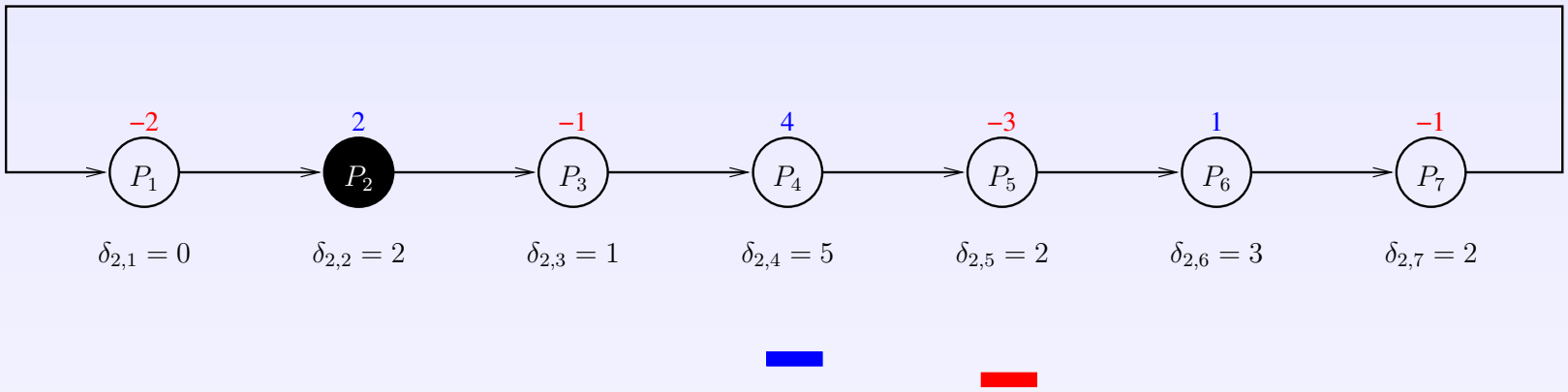
À l'étape 3, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 3$

Algorithme de redistribution



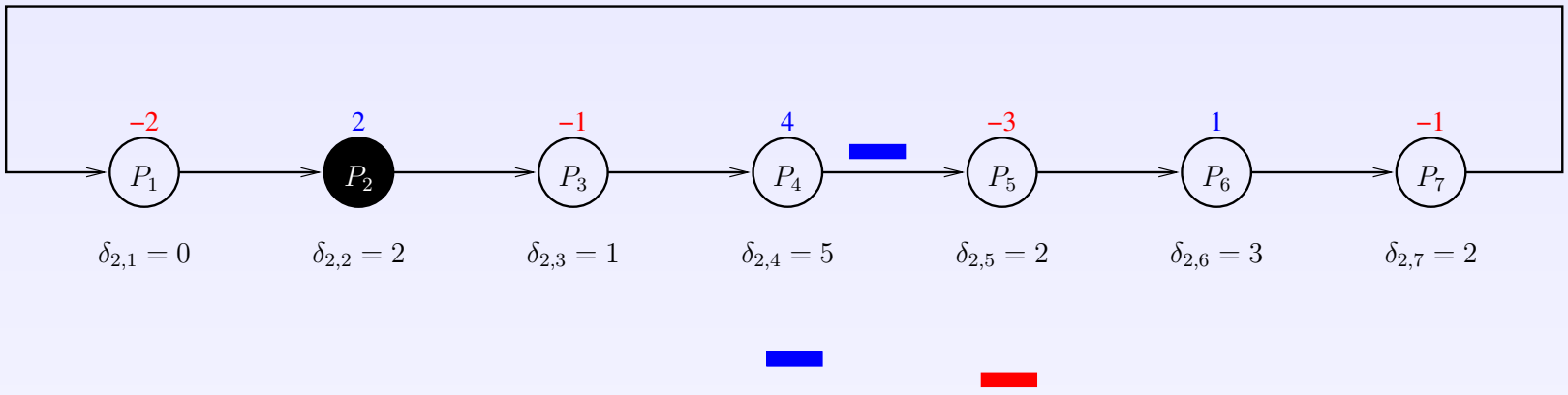
À l'étape 4, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 4$

Algorithme de redistribution



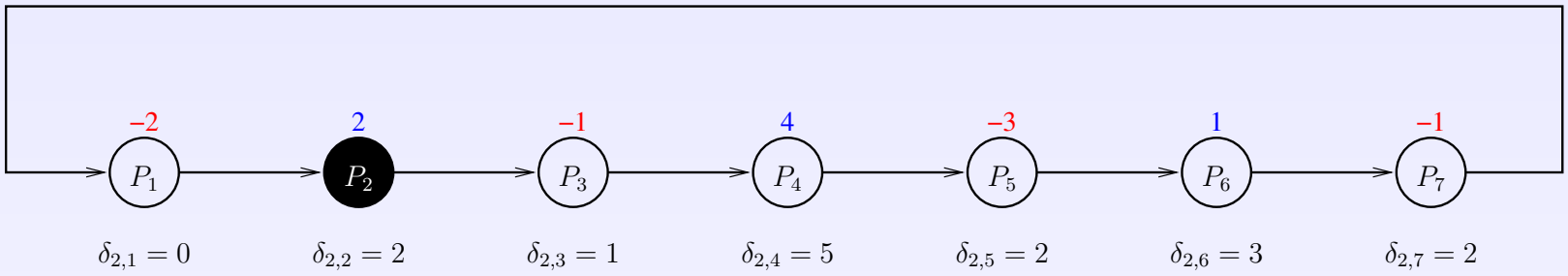
À l'étape 4, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 4$

Algorithme de redistribution



À l'étape 5, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 5$

Algorithme de redistribution



À l'étape 5, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 5$

Unidirectionnel homogène : algorithme formel

1: Let $\delta_{\text{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$
2: Let `start` and `end` be two indices such that the slice $C_{\texttt{start,end}}$ is of maximal imbalance : $\delta_{\texttt{start,end}} = \delta_{\text{max}}$.
3: **for** $s = 1$ to δ_{max} **do**
4: **for all** $l = 0$ to $n - 1$ **do**
5: **if** $\delta_{\texttt{start,start}+l} \geq s$ **then**
6: $P_{\texttt{start}+l}$ sends to $P_{\texttt{start}+l+1}$ a data item during the time interval $[(s-1) \times c, s \times c[$

Unidirectionnel homogène : algorithme formel

1: Let $\delta_{\text{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$
2: Let `start` and `end` be two indices such that the slice $C_{\texttt{start,end}}$ is of maximal imbalance : $\delta_{\texttt{start,end}} = \delta_{\text{max}}$.
3: **for** $s = 1$ to δ_{max} **do**
4: **for all** $l = 0$ to $n - 1$ **do**
5: **if** $\delta_{\texttt{start,start}+l} \geq s$ **then**
6: $P_{\texttt{start}+l}$ sends to $P_{\texttt{start}+l+1}$ a data item during the time interval $[(s-1) \times c, s \times c[$

Théorème

Cet algorithme de redistribution est optimal.

Unidirectionnel hétérogène : borne inférieure

Processeur P_i a besoin d'un temps $c_{i,i+1}$ pour envoyer une donnée au processeur P_{i+1}.

Unidirectionnel hétérogène : borne inférieure

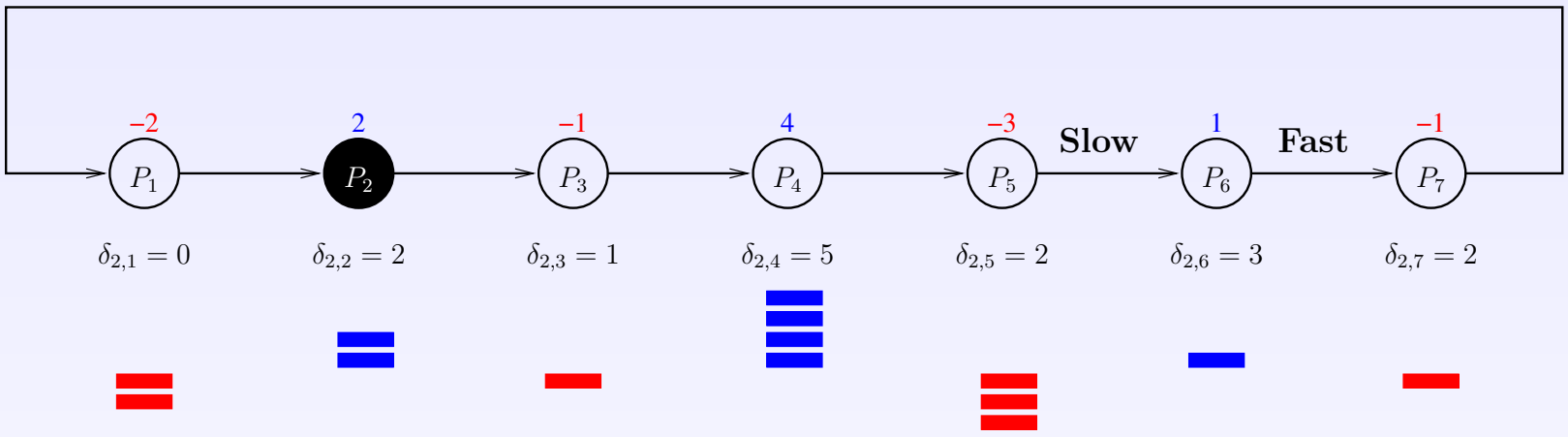
Processeur P_i a besoin d'un temps $c_{i,i+1}$ pour envoyer une donnée au processeur P_{i+1}.

Principe de la borne inférieure : comme pour le cas homogène.

P_l a besoin d'un temps $\delta_{k,l} \times c_{l,l+1}$ pour envoyer $\delta_{k,l}$ données à P_{l+1} (si $\delta_{k,l} > 0$).

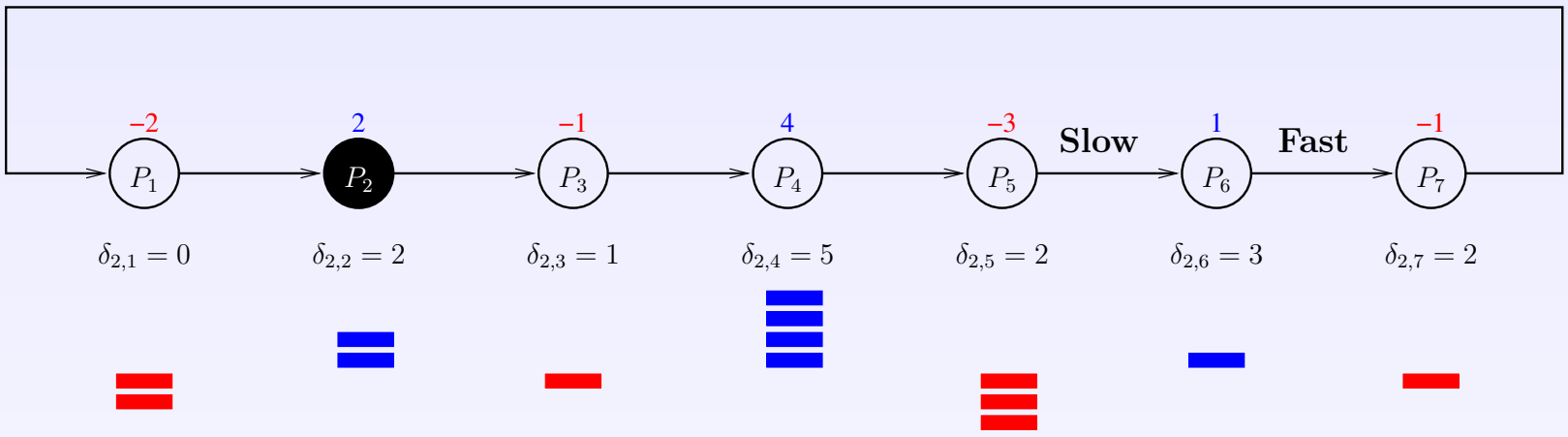
Borne inférieure : $$\max_{1 \leq k \leq n,\ 0 \leq l \leq n-1} \delta_{k,k+l} \times c_{k+l,k+l+1}$$

Conséquences de l'hétérogénéité des communications



P_6 peut avoir à attendre de recevoir des données de P_5 pour pouvoir finir d'envoyer toutes les données nécessaires à P_7.

Conséquences de l'hétérogénéité des communications



P_6 peut avoir à attendre de recevoir des données de P_5 pour pouvoir finir d'envoyer toutes les données nécessaires à P_7.

On ne peut pas exprimer avec une formule simple le temps nécessaire à P_6 pour effectuer sa part de travail.

L'algorithme de redistribution est asynchrone.

L'algorithme de redistribution

C'est simplement une version asynchrone de l'algorithme précédent.

1: Let $\delta_{\mathsf{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$

2: Let `start` and `end` be two indices such that the slice $C_{\mathtt{start},\mathtt{end}}$ is of maximal unbalance : $\delta_{\mathtt{start},\mathtt{end}} = \delta_{\mathsf{max}}$.

3: **for all** $l = 0$ to $n - 1$ **do**

4: $P_{\mathtt{start}+l}$ sends $\delta_{\mathtt{start},\mathtt{start}+l}$ data items one by one and as soon as possible to processeur $P_{\mathtt{start}+l+1}$

Optimalité

Évidente par construction

Optimalité

Évidente par construction

Lemme

Le temps d'exécution de l'algorithme de redistribution est

$$\max_{0 \leq l \leq n-1} \delta_{start,start+l} \times c_{start+l,start+l+1}.$$

Optimalité

Évidente par construction

Lemme

Le temps d'exécution de l'algorithme de redistribution est

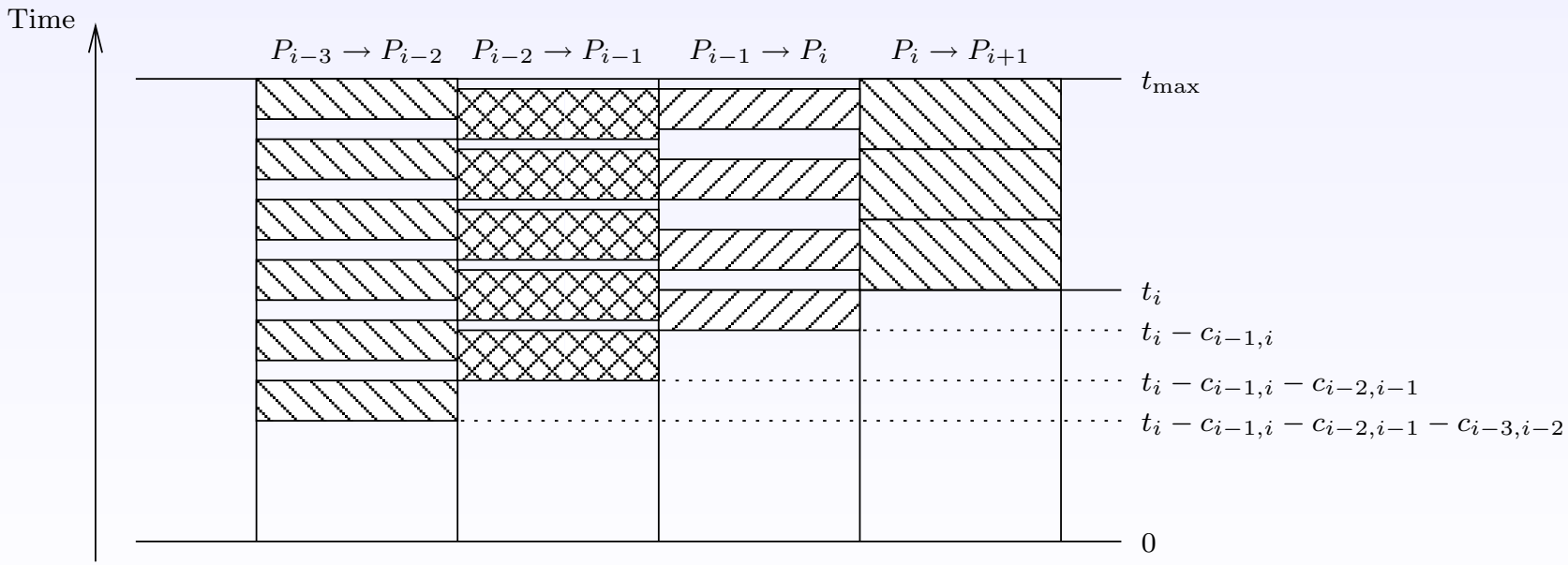
$$\max_{0 \leq l \leq n-1} \delta_{start,start+l} \times c_{start+l,start+l+1}.$$

En d'autres termes, il n'y a pas de délai de propagation, quelle que soit la distribution initiale des données, et quelles que soient les vitesses de communications...

Optimalité : principe de la preuve

Le temps d'exécution de l'algorithme de redistribution est

$$\max_{0 \leq l \leq n-1} \delta_{\texttt{start},\texttt{start}+l} \times c_{\texttt{start}+l,\texttt{start}+l+1}.$$

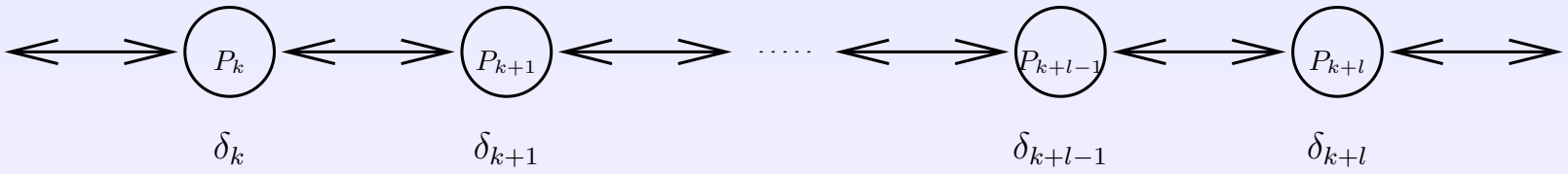


Bidirectionnel homogène : contexte

Temps de communication homogène : c.

Communications bi-directionnelles

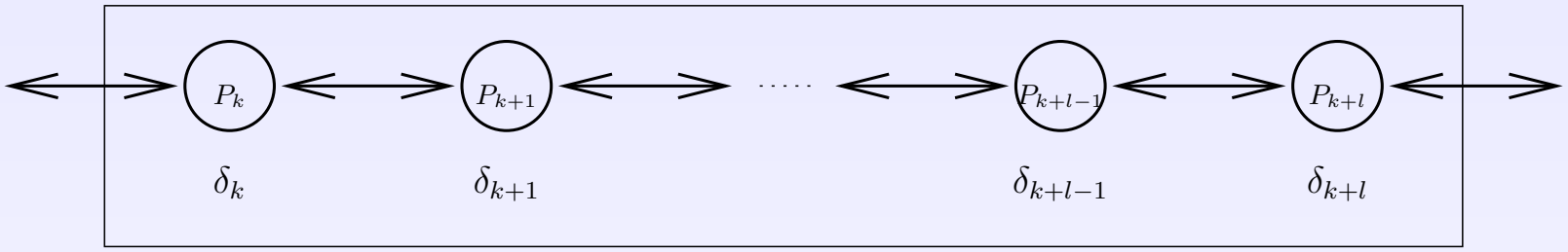
Bidirectionnel homogène : borne inférieure



Temps de communication homogène : c.

Communications bi-directionnelles

Bidirectionnel homogène : borne inférieure

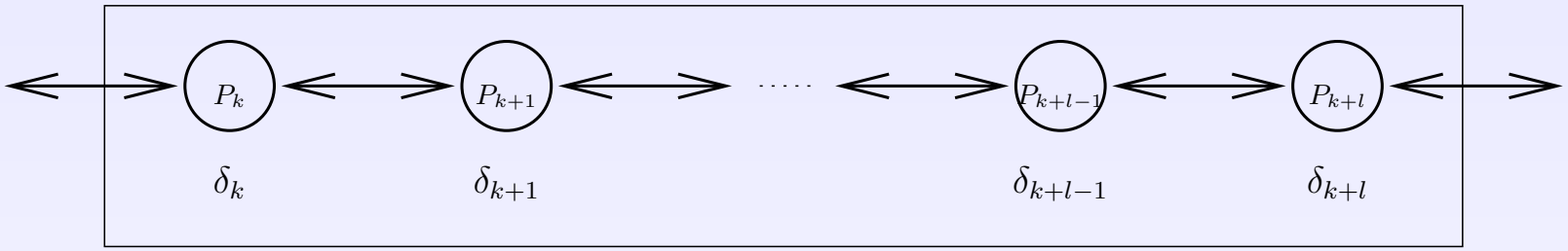


$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Temps de communication homogène : c.

Communications bi-directionnelles

Bidirectionnel homogène : borne inférieure

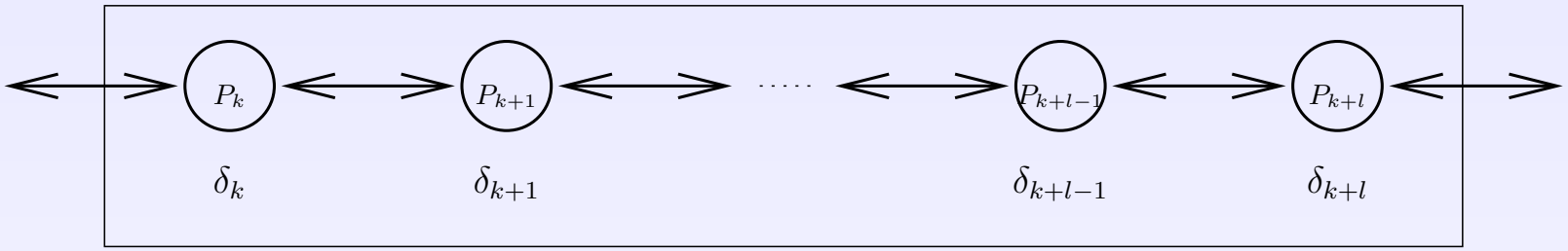


$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Temps de communication homogène : c.

On a besoin d'un temps $\left\lceil \frac{\delta_{k,k+l}}{2} \right\rceil \times c$ pour envoyer $\delta_{k,k+l}$ données de la « chaîne » de processeurs $P_k, ..., P_{k+l}$(si $\delta_{k,l} > 0$).

Bidirectionnel homogène : borne inférieure



$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Temps de communication homogène : c.

On a besoin d'un temps $\left\lceil \frac{\delta_{k,k+l}}{2} \right\rceil \times c$ pour envoyer $\delta_{k,k+l}$ données de la « chaîne » de processeurs $P_k, ..., P_{k+l}$(si $\delta_{k,l} > 0$).

Borne inférieure : $$\max \left\{ \max_{1 \leq i \leq n} |\delta_i|, \max_{1 \leq i \leq n, 1 \leq l \leq n-1} \left\lceil \frac{|\delta_{i,i+l}|}{2} \right\rceil \right\} \times c$$

Bidirectionnel homogène : principe de l'algorithme

1. Tout ensemble non trivial $C_{k,l}$ tel que $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{k,l} \geq 0$ doit envoyer deux données à chaque étape, une par chacune de ses extrémités.

Bidirectionnel homogène : principe de l'algorithme

1. Tout ensemble non trivial $C_{k,l}$ tel que $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{k,l} \geq 0$ doit envoyer deux données à chaque étape, une par chacune de ses extrémités.
2. Tout ensemble non trivial $C_{k,l}$ tel que $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{k,l} \leq 0$ doit recevoir deux données à chaque étape, une par chacune de ses extrémités.

Bidirectionnel homogène : principe de l'algorithme

1. Tout ensemble non trivial $C_{k,l}$ tel que $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{k,l} \geq 0$ doit envoyer deux données à chaque étape, une par chacune de ses extrémités.
2. Tout ensemble non trivial $C_{k,l}$ tel que $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{k,l} \leq 0$ doit recevoir deux données à chaque étape, une par chacune de ses extrémités.
3. Une fois définies les communications requises par les deux précédents cas, on s'occupe de P_i tel que $|\delta_i| = \delta_{\max}$.
   Si P_i est déjà impliqué dans une communication : tout est réglé.
   Autrement, on a le choix du processeur à qui P_i envoie (cas $\delta_i \geq 0$) ou de qui P_i reçoit (cas $\delta_i \leq 0$) une donnée.
   Pour simplifier : toutes ces communications ont lieu dans le sens « P_i vers P_{i+1} ».

Bidirectionnel homogène : optimalité

Difficultés :

- Cas particuliers (cas limites)
- Preuve de la correction de l'algorithme (l'optimalité est alors immédiate)

Bidirectionnel hétérogène : borne

La durée τ de toute redistribution vérifie :

$$\tau \geq \max \left\{ \max_{1 \leq k \leq n,\ \delta_k > 0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \right.$$

Bidirectionnel hétérogène : borne

La durée τ de toute redistribution vérifie :

$$\tau \geq \max \begin{cases} \max\limits_{1 \leq k \leq n,\ \delta_k > 0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \max\limits_{1 \leq k \leq n,\ \delta_k < 0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \end{cases}$$

Bidirectionnel hétérogène : borne

La durée τ de toute redistribution vérifie :

$$\tau \geq \max \begin{cases} \max\limits_{1\leq k\leq n,\ \delta_k>0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \max\limits_{1\leq k\leq n,\ \delta_k<0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \max\limits_{\substack{1\leq k\leq n,\\ 1\leq l\leq n-2,\\ \delta_{k,k+l}>0}} \ \min\limits_{0\leq i\leq \delta_{k,k+l}} \max\{i\cdot c_{k,k-1}, (\delta_{k,k+l}-i)\cdot c_{k+l,k+l+1}\} \end{cases}$$

Bidirectionnel hétérogène : borne

La durée τ de toute redistribution vérifie :

$$\tau \geq \max \begin{cases} \max\limits_{1\leq k\leq n,\ \delta_k>0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \max\limits_{1\leq k\leq n,\ \delta_k<0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \max\limits_{\substack{1\leq k\leq n,\\ 1\leq l\leq n-2,\\ \delta_{k,k+l}>0}} \min\limits_{0\leq i\leq \delta_{k,k+l}} \max\{i\cdot c_{k,k-1}, (\delta_{k,k+l}-i)\cdot c_{k+l,k+l+1}\} \\ \max\limits_{\substack{1\leq k\leq n,\\ 1\leq l\leq n-2,\\ \delta_{k,k+l}<0}} \min\limits_{0\leq i\leq -\delta_{k,k+l}} \max\{i\cdot c_{k-1,k}, -(\delta_{k,k+l}+i)\cdot c_{k+l+1,k+l}\} \end{cases}$$

Bidirectionnel hétérogène : redistributions « légères » (1)

Définition : on dit qu'une redistribution est « légère » si tout processeur détient initialement toutes les donnés qu'il doit envoyer au cours de l'exécution de l'algorithme.

$\mathcal{S}_{i,j}$: quantité de données envoyées par P_i à son voisin P_j.

$$\text{Minimize } \tau, \text{ subject to}$$

$$\begin{cases} \mathcal{S}_{i,i+1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i-1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} + \mathcal{S}_{i,i-1} - \mathcal{S}_{i+1,i} - \mathcal{S}_{i-1,i} = \delta_i & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} c_{i,i+1} + \mathcal{S}_{i,i-1} c_{i,i-1} \leq \tau & 1 \leq i \leq n \\ \mathcal{S}_{i+1,i} c_{i+1,i} + \mathcal{S}_{i-1,i} c_{i-1,i} \leq \tau & 1 \leq i \leq n \end{cases}$$

Bidirectionnel hétérogène : redistributions « légères » (2)

1. Toute solution entière est réalisable.

   Ex. : P_i envoie ses $\mathcal{S}_{i,i+1}$ données à P_{i+1} à partir du temps 0. Une fois cet envoi terminé, P_i envoie $\mathcal{S}_{i,i-1}$ données à P_{i-1} dès que possible sous le modèle un-port.

2. Si on résout le système en rationnel, un des deux arrondis naturels en entiers défini une solution entière optimale.

Bidirectionnel hétérogène : cas général

Quelqu'un aurait-il une idée ?

Plan de l'exposé

Conclusion

Le parallélisme « régulier » était déjà compliqué, maintenant on a

- Des processeurs de caractéristiques différentes
- Des liens de communications de caractéristiques différentes
- Des réseaux irréguliers... voire de topologie inconnue
- Des ressources dont les caractéristiques évoluent au cours du temps

Il est nécessaire d'avoir une modélisation réseau réaliste... mais une modélisation plus réaliste (peut) rend(re) le problème plus compliqué