Équilibrage de charge et redistribution de données sur plates-formes hétérogènes

Soutenance de thèse :

HÉLÈNE RENARD

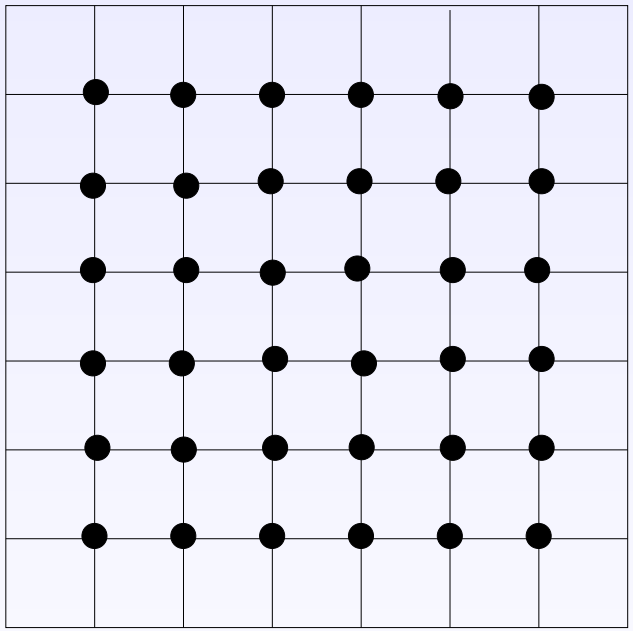
Devant la commission d'examen formée de :

Monsieur	Jacques	BAHI, Rapporteur
Monsieur	Michel	DAYDÉ, Rapporteur
Monsieur	Serge	MIGUET, Membre extérieur
Monsieur	Yves	ROBERT, Directeur de Thèse
Monsieur	Denis	TRYSTRAM, Membre extérieur
Monsieur	Frédéric	VIVIEN, Directeur de Thèse

De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?



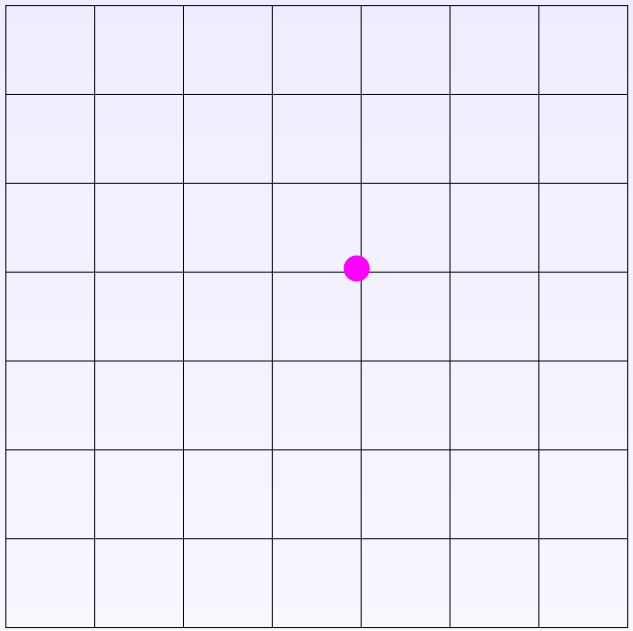
- Point de la grille 2D

  36 points au total

De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

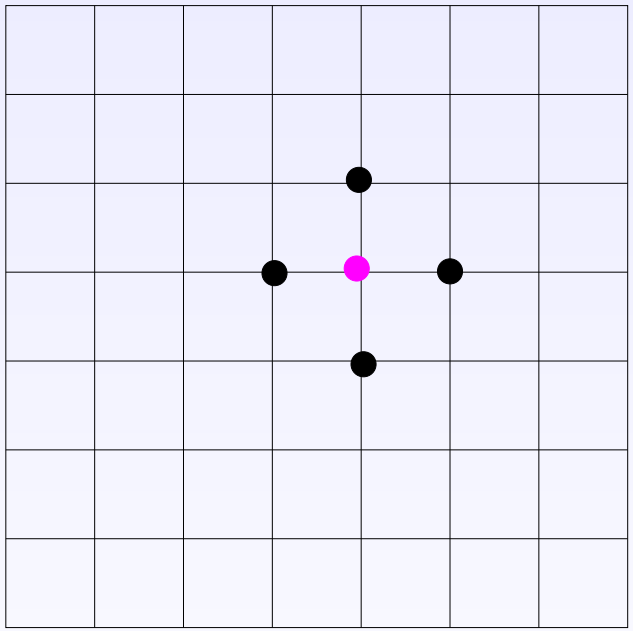
Qu'est-ce que la méthode itérative de Jacobi ?



De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

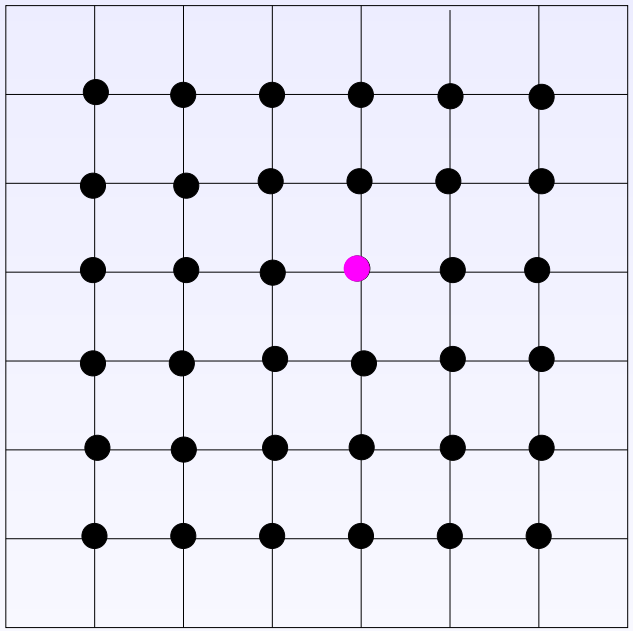
Qu'est-ce que la méthode itérative de Jacobi ?

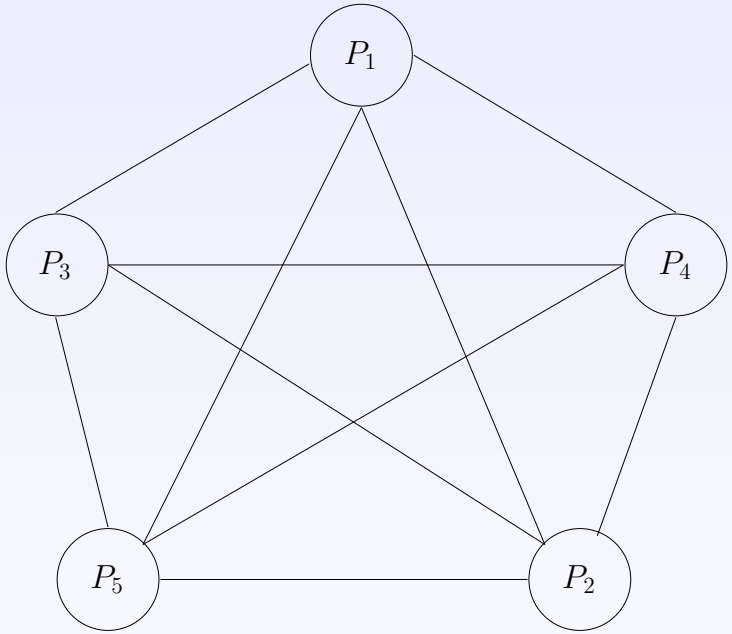


De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?

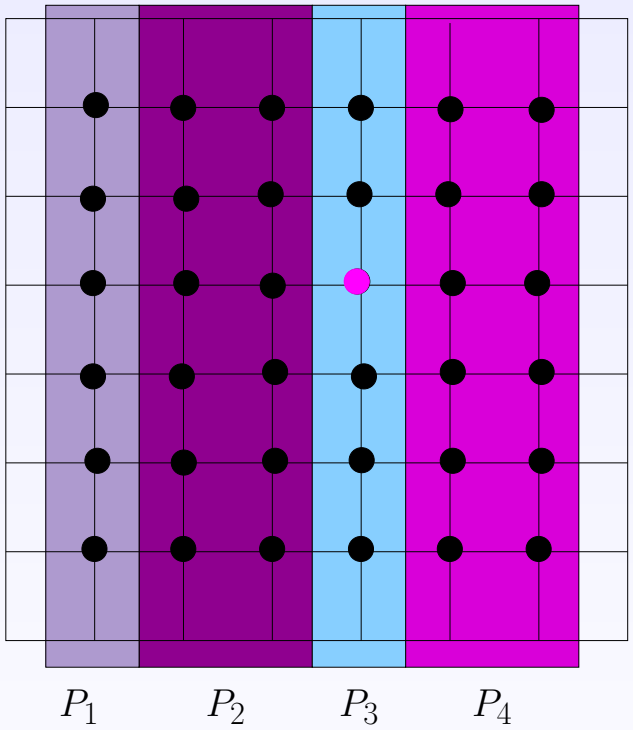




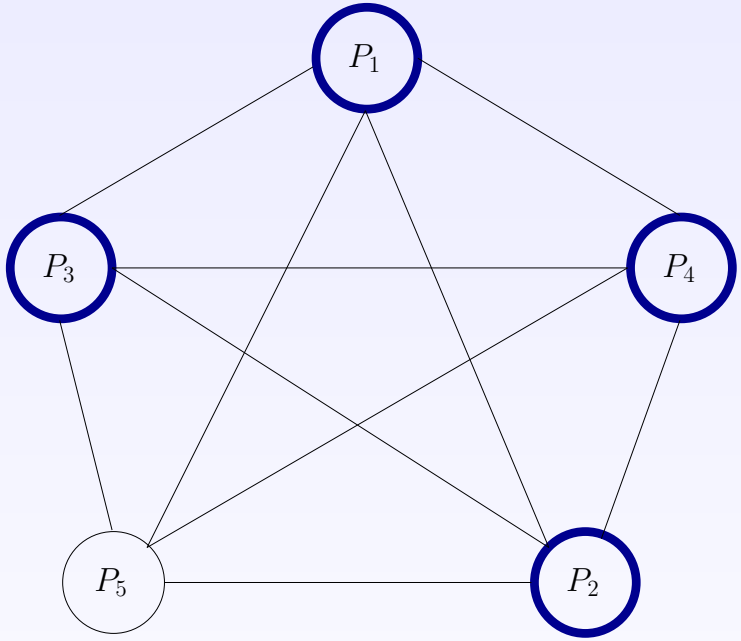
De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?



Répartition des données

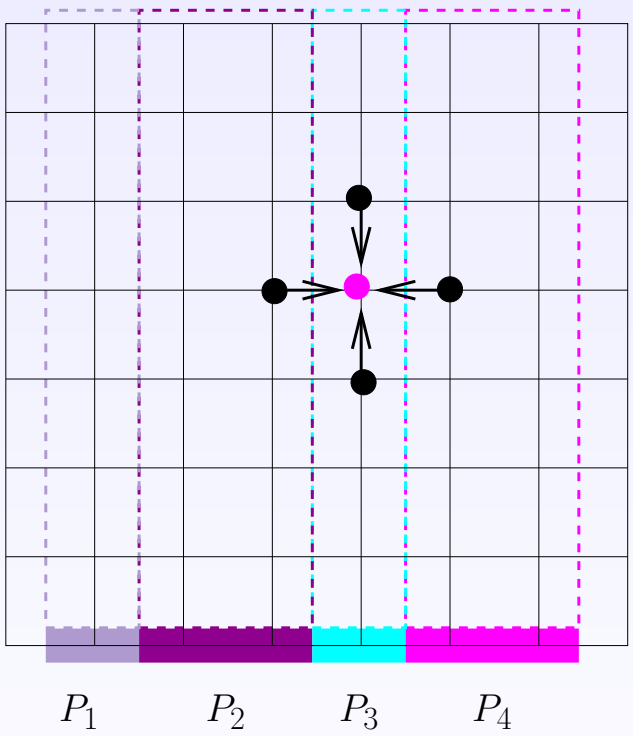


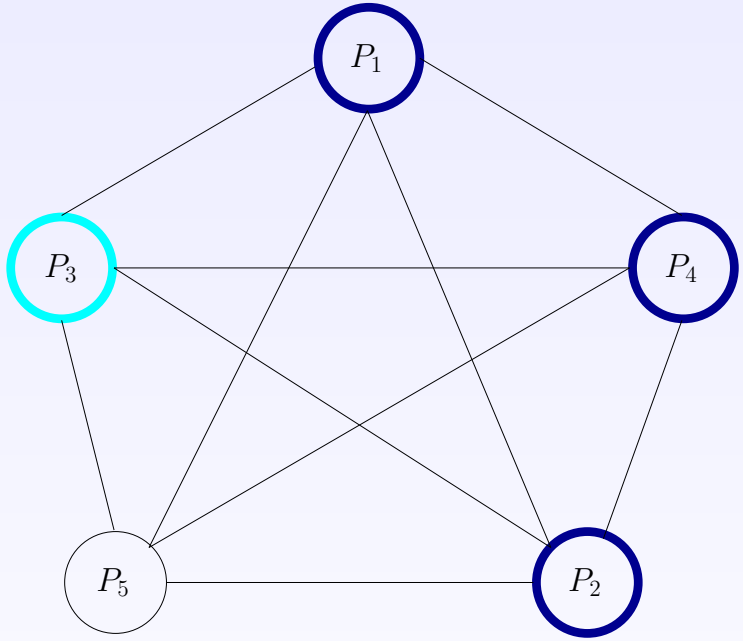
Sélection des processeurs

De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?

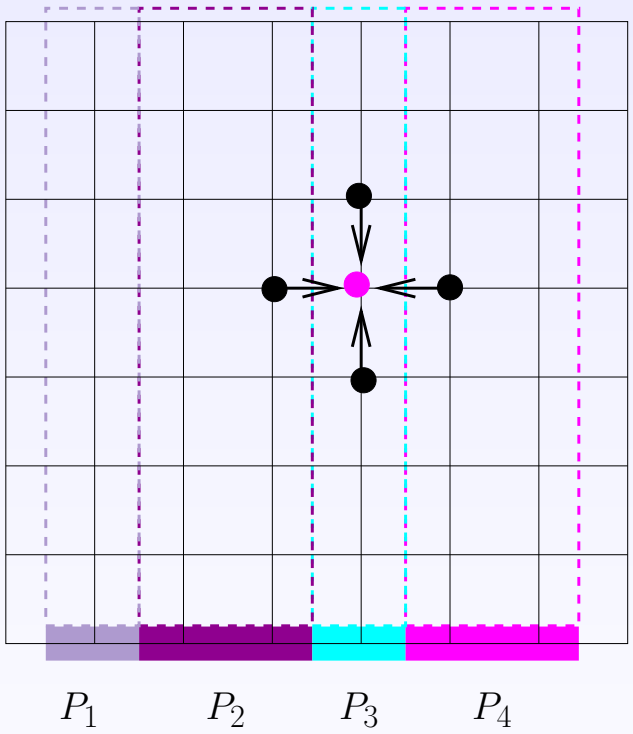


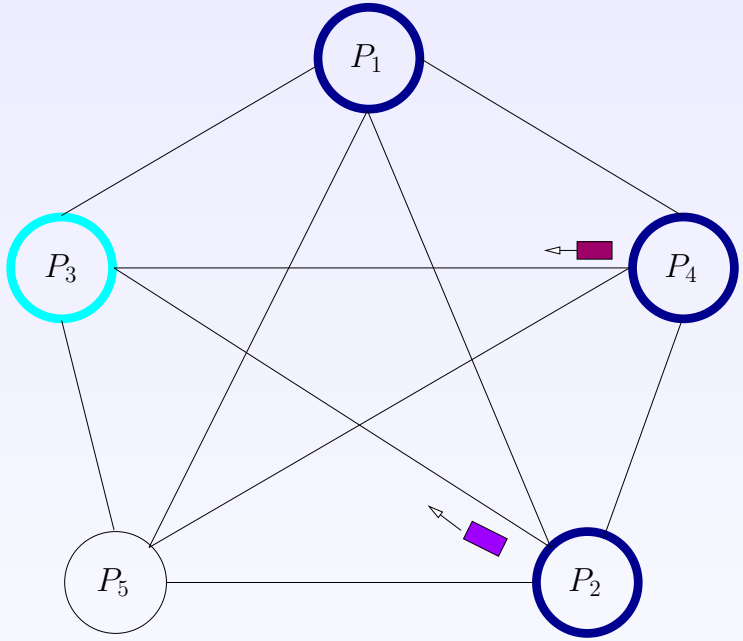


De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?



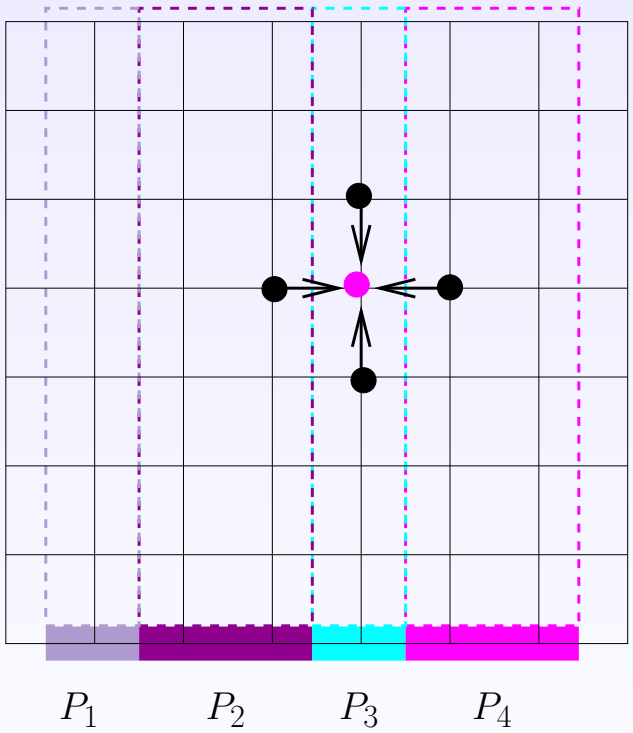


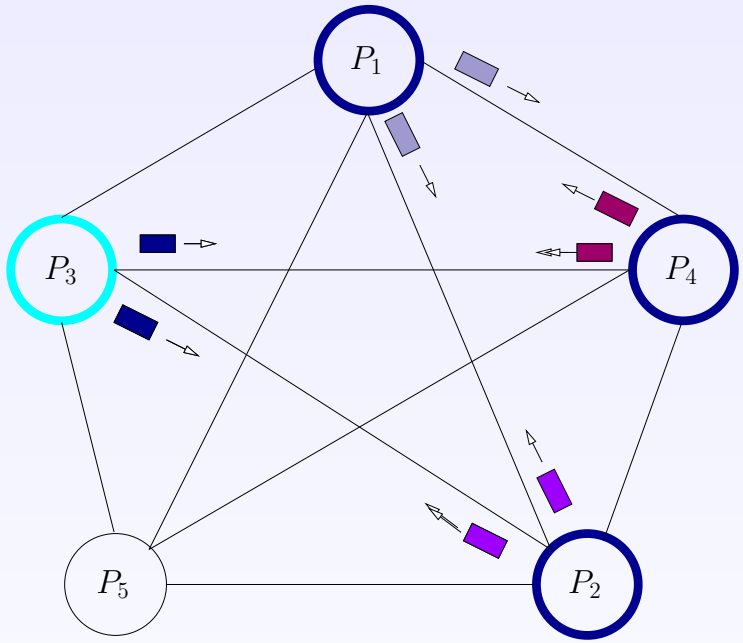
Exemple d'échange des données frontières

De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?



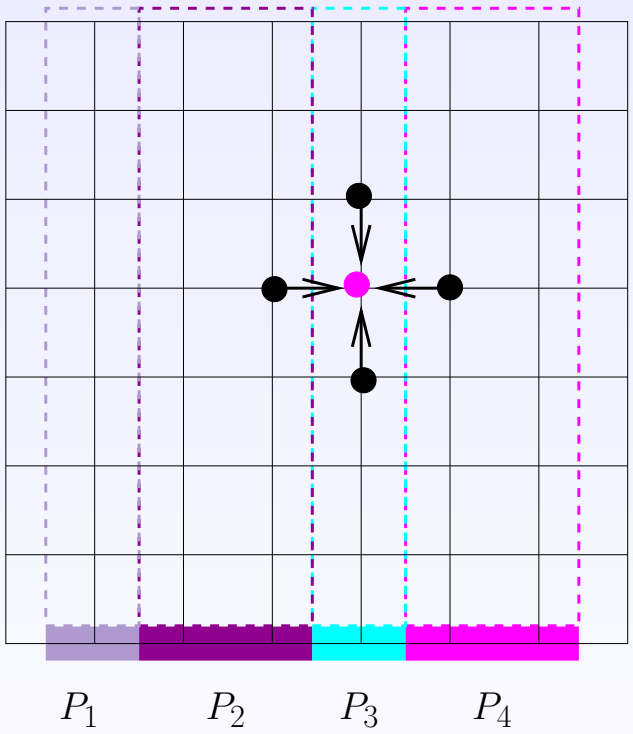


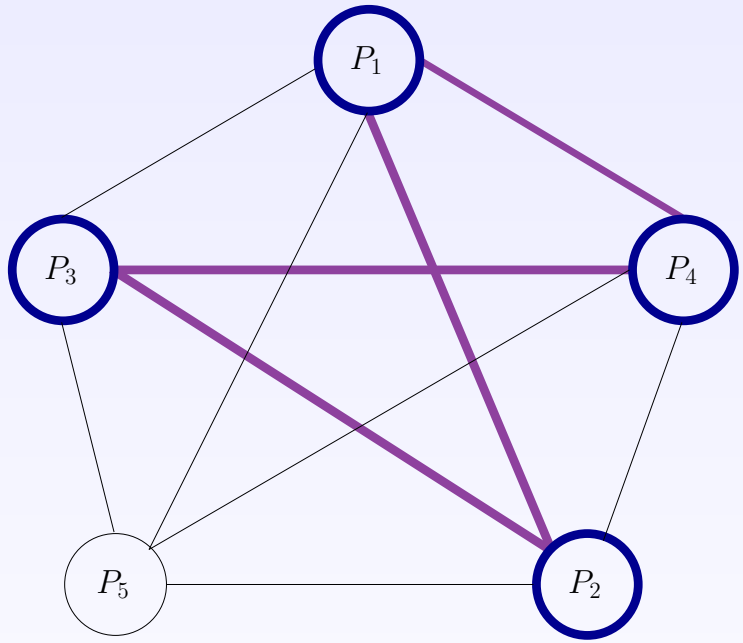
Échange des données frontières

De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?



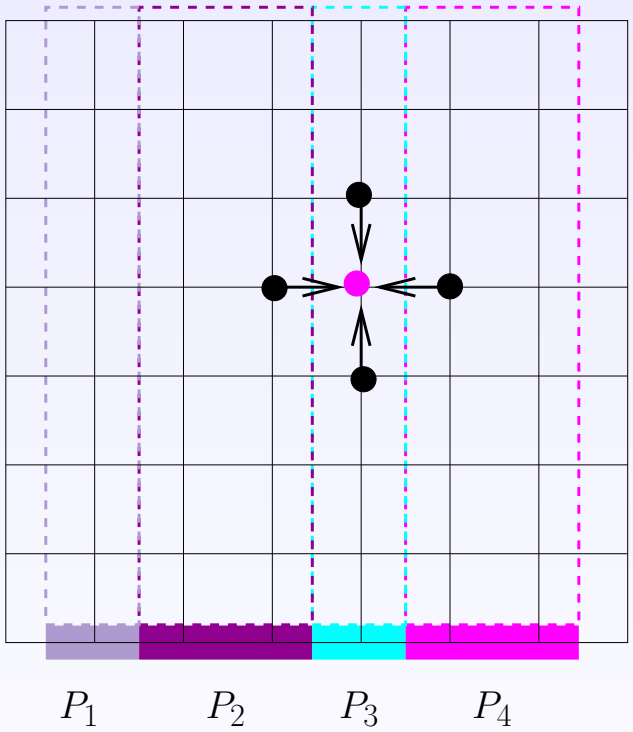


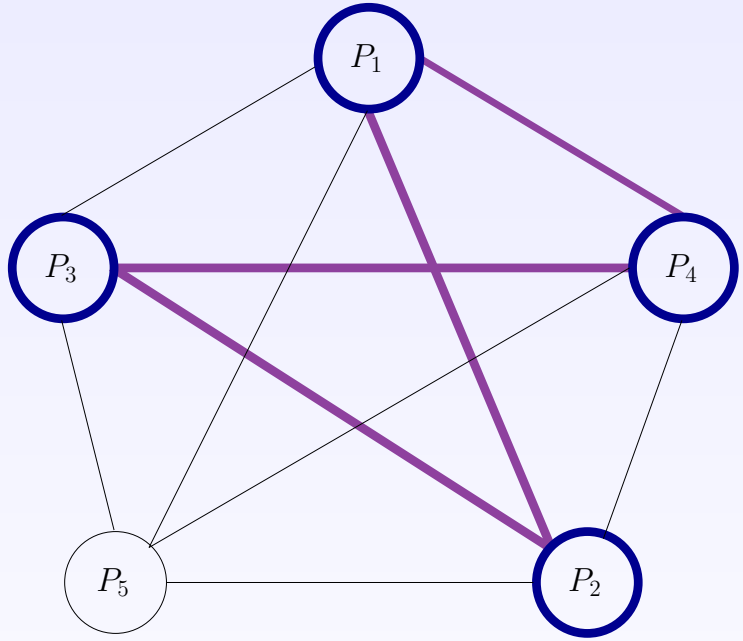
Répéter « phase de calcul / phase d'échange »

De nombreux problèmes à résoudre

Prenons un exemple : Jacobi

Qu'est-ce que la méthode itérative de Jacobi ?





Répéter « phase de calcul / phase d'échange »

Une distribution uni-dimensionnelle des données induit un arrangement uni-dimensionnel des processeurs.

Contexte de la thèse

Distribution des calculs : volume important de données (calcul de matrices, traitement d'images), optimisation du temps d'exécution.

Plates-formes ciblées : plates-formes distribuées hétérogènes (réseau de stations de travail, clusters de clusters, grilles, etc.).

Équilibrage de charge :

- Statique.

Redistribution de données :

1. Différentes variations : ressources ou besoins de l'application.
2. Les données doivent être redistribuées afin d'obtenir un meilleur équilibrage de charge.

Plan de l'exposé

Plan de l'exposé

Plan de l'exposé

SliceRing *vs.* SharedRing

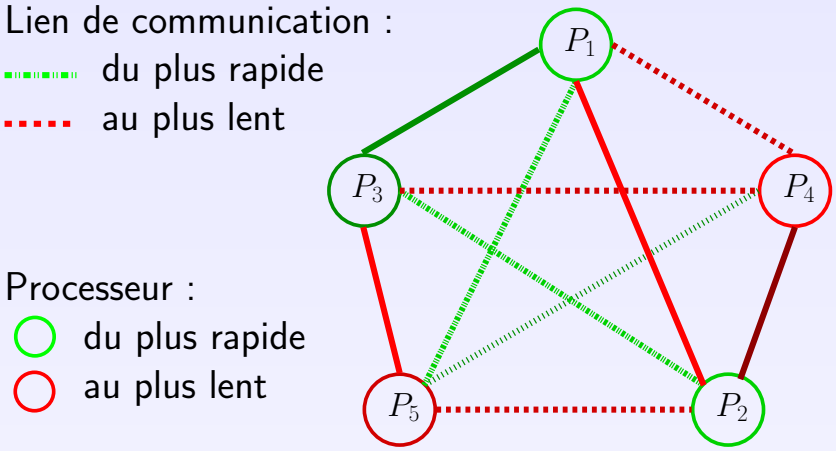


FIG.: Exemple de plate-forme hétérogène complète pour SliceRing

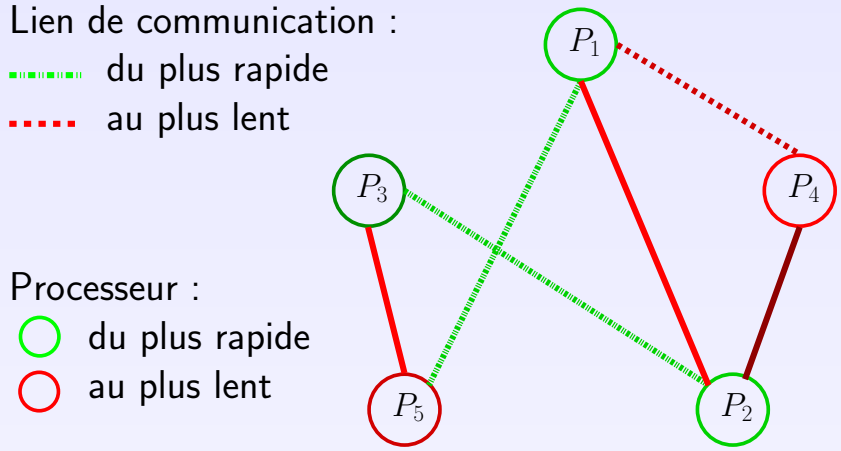


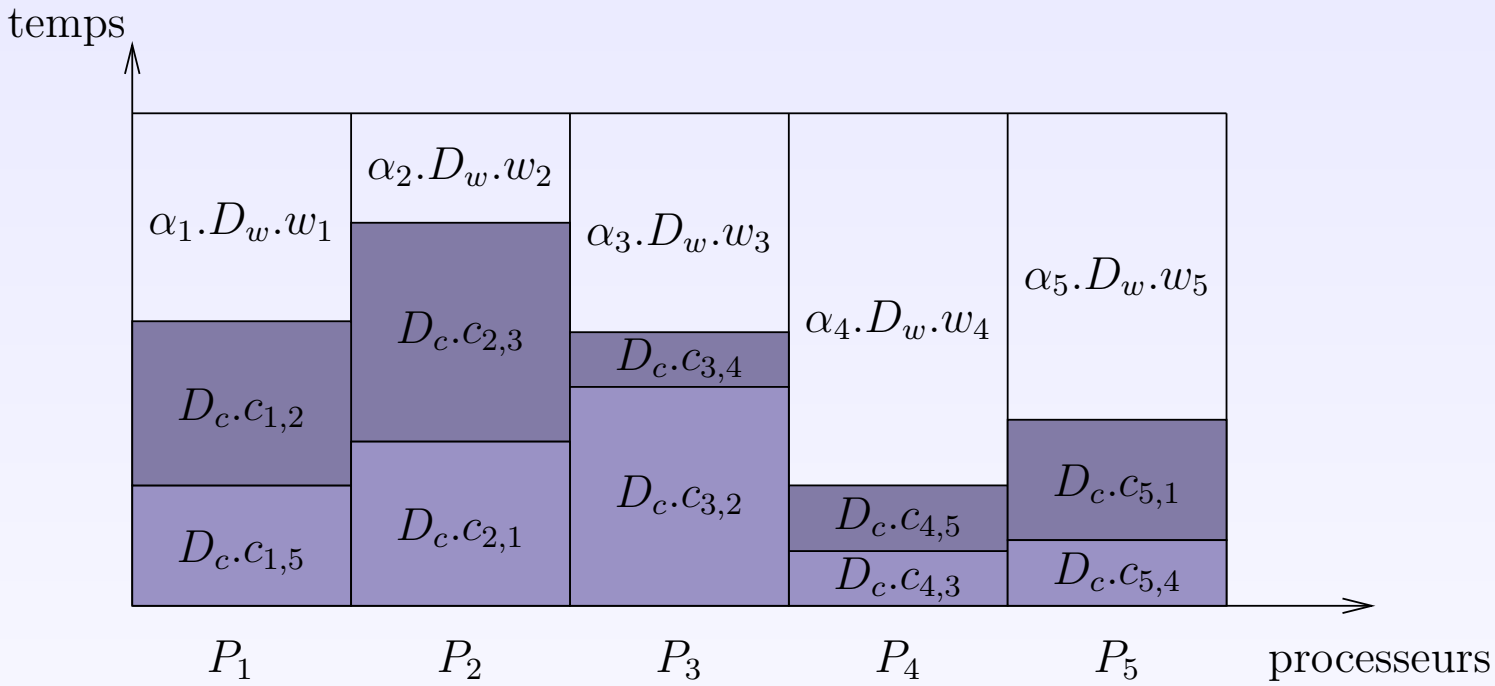
FIG.: Exemple de plate-forme hétérogène pour SharedRing

SliceRing	SharedRing
Graphe complet	Graphe quelconque
Lien direct pour aller de P_i à P_j	Recherche du routage
Pas de partage de liens	Partage de liens

Plan de l'exposé

Notre objectif

« Répartir les données au mieux afin d'optimiser le temps d'exécution »



avec :

- 5 processeurs : $P_1, \cdots, P_5$
- w_i = temps de cycle de P_i
- $c_{i,j}$ = capacité du lien de communication pour aller de P_i à P_j
- D_w = charge de travail totale
- D_c = volume de communication à chaque étape

Plan d'attaque : formalisation

Définition

Étant donnés :

- p *processeurs de temps de cycle* w_i, $p(p-1)$ *liens de communication de capacité* $c_{i,j}$, D_w *la charge de travail totale,* D_c *le volume de communication à chaque étape, et une borne de temps* K.

Est-il possible de trouver :

- *un entier* $q \leq p$,
- *une application* $\sigma : [1..q] \rightarrow [1..p]$,
- *des nombres rationnels positifs* α_i *avec* $\sum_{i=1}^{q} \alpha_{\sigma(i)} = 1$,

tels que :

$$T_{step} = \min_{\substack{1 \leq q \leq p, \\ \sigma \in \Theta_{q,p},}} \left\{ \max_{1 \leq i \leq q} \left(\alpha_{\sigma(i)}.D_w.w_{\sigma(i)} + D_c.\left(c_{\sigma(i),\sigma(i-1)} + c_{\sigma(i),\sigma(i+1)} \right) \right) \right\} \leq K?$$

$$\sum_{i=1}^{q} \alpha_{\sigma(i)} = 1$$

Complexité

Théorème

Le problème de décision associé au problème d'optimisatation SliceRing *est NP-complet.*

Démonstration

Idée générale :

- *Réduction à un cycle Hamiltonien*

Solutions pratiques

- *Modélisation sous forme de programmation linéaire en entiers (PLE)*

Heuristiques

- *Lin-Kernighan*
- *Gloutonne*

Deux heuristiques

- Heuristique basée sur le principe du voyageur de commerce
  - solution optimale (tous les processeurs sont impliqués) = plus court cycle Hamiltonien dans le graphe.
  - heuristique de Lin-Kernighan afin d'approximer le plus court chemin
  - inconvénient : peu de processeurs dans la solution optimale, solution de mauvaise qualité (temps d'exécution plus élevé).

- Heuristique gloutonne
  - sélectionne le processeur le plus rapide,
  - insère itérativement un nouveau processeur dans l'anneau solution,
  - retourne le nombre de processeurs optimal dans l'anneau solution.

Simulations : description de la plate-forme

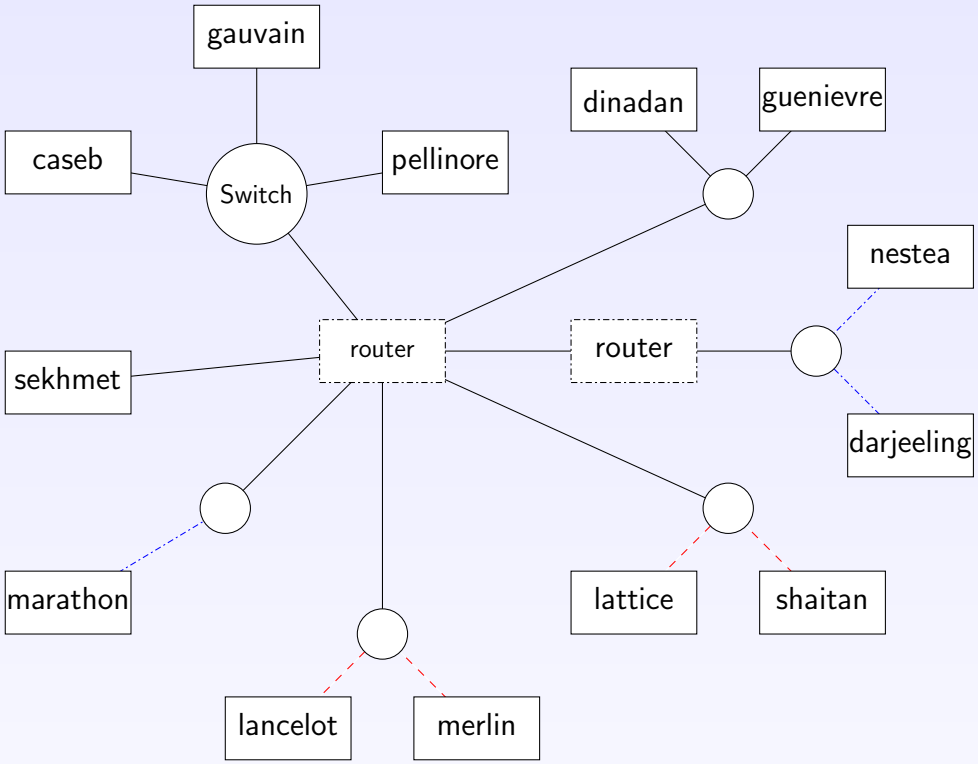


FIG.: Topologie de la plate-forme de Strasbourg.

- 13 unités de calcul,
- 6 routeurs,
- Caractéristiques mesurées $\Rightarrow$ simulations basées sur la « réalité ».

Simulations : description des heuristiques

- Heuristique gloutonne
- Solution optimale obtenue par :
  - LP_SOLVE
  - PIP

Heuristiques \ Processeurs	3	4	5	6	7	8	9	10	11	12	13
Gloutonne	1520	2112	3144	3736	4958	5668	7353	8505	10195	12490	15759
Solution optimale	1517	2109	3141	3733	4955	5660	7348	8500	10188	12235	14757

Tab.: Comparaison entre l'heuristique gloutonne et la solution optimale.

- Heuristique gloutonne : rapide et efficace, à $6,8\%$ de l'optimal.
- Logiciels de programmation linéaire en entiers : échouent rapidement sur le calcul de la solution optimale.

Plan de l'exposé

Rappels

Lien de communication :
- du plus rapide
- au plus lent

Processeur :
- du plus rapide
- au plus lent

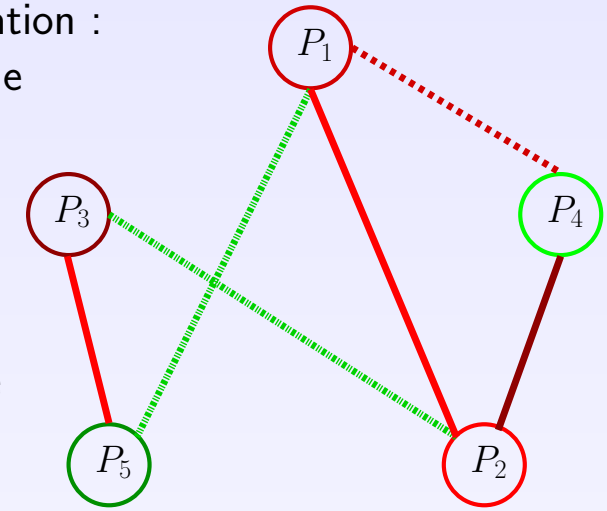
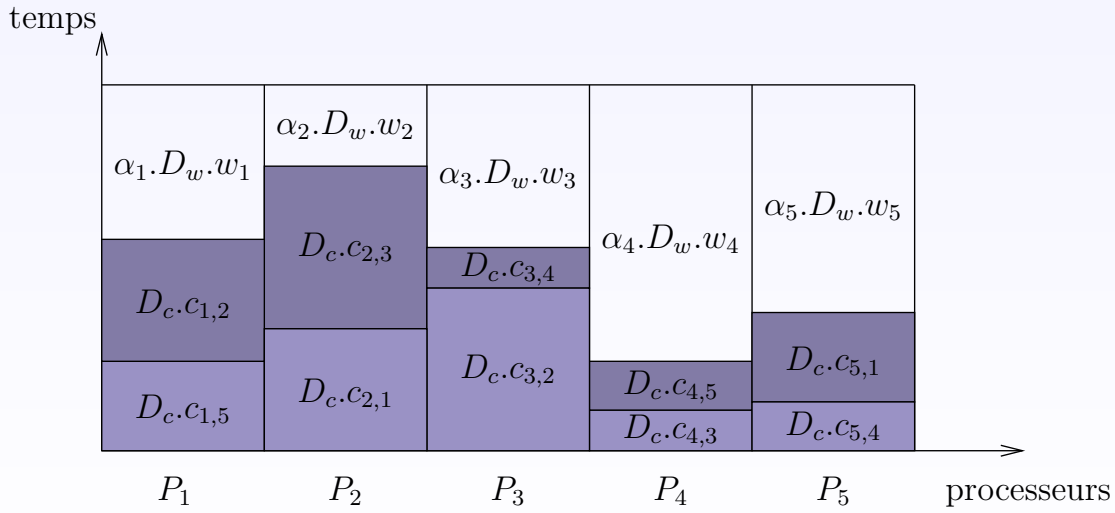
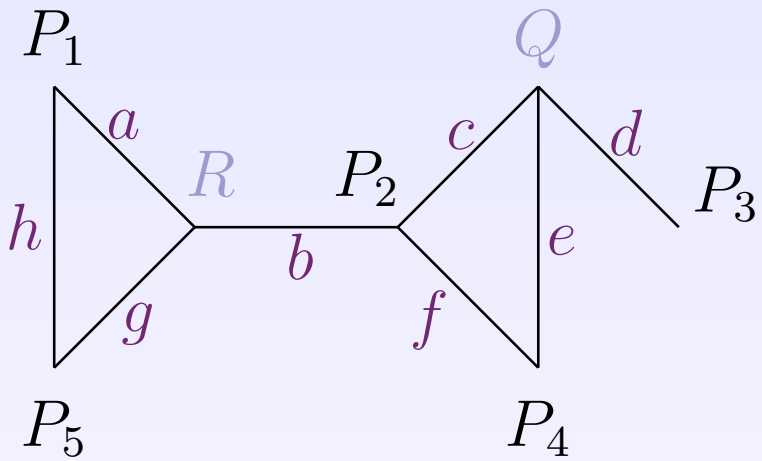


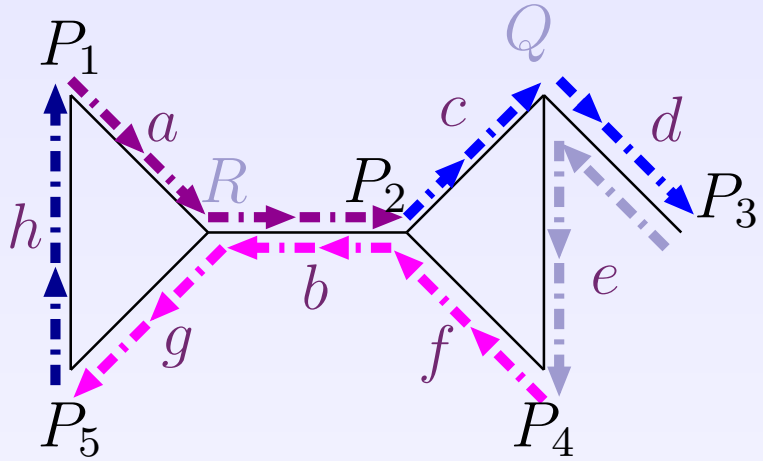
Plate-forme hétérogène quelconque
Recherche du routage
$\Rightarrow$ partage de certains liens de communication

Minimiser le temps total d'exécution
$\Rightarrow$ tous les processeurs terminent en même temps



Un exemple pour comprendre (1/3)





De P_1 à P_2, nous utilisons les liens a et b d'où $\mathcal{S}_1 = \{a, b\}$.
De P_2 à P_1, nous utilisons les liens b, g et h, d'où $\mathcal{P}_2 = \{b, g, h\}$.

- 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, $\mathcal{P}_5 = \{g, b, f\}$

Un exemple pour comprendre (2/3)

Pour P_1, puisque $\mathcal{S}_1 = \{a, b\}$, nous avons $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$;
Et puisque $\mathcal{P}_1 = \{h\}$, nous avons $c_{1,5} = \frac{1}{p_{1,h}}$.

La liste de toutes les équations devant être satisfaites :

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

Un exemple pour comprendre (3/3)

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

AVEC LES CONTRAINTES SUIVANTES :

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

Formalisation

Définition

Étant donnés :

- p *processeurs* P_i *de temps de cycle* w_i, $|E|$ *liens de communication* e_m *de bande passante* b_{e_m}, D_w *la charge de travail totale,* D_c *le volume de communication et une borne de temps* K.

Est-il possible de trouver :

- *un sous-ensemble de* $q \leq p$ *processeurs,*
- *une application* $\sigma : [1..q] \rightarrow [1..p]$,
- $2q$ *chemins de communication* $\mathcal{S}_i$ *et* $\mathcal{P}_i$

tels que :

- *aucune bande passante totale de lien ne soit pas dépassée,*
- *des nombres rationnels strictement positifs* α_i *avec* $\sum_{i=1}^{q} \alpha_{\sigma(i)} = 1$,

$$T_{step} = \min_{\substack{1 \leq q \leq p \\ \sigma \in \Theta_{q,p}}} \left\{ \max_{1 \leq i \leq q} \left(\alpha_{\sigma(i)}.D_w.w_{\sigma(i)} + D_c.\left(c_{\sigma(i),\sigma(i-1)} + c_{\sigma(i),\sigma(i+1)} \right) \right) \right\} \leq K$$

$$\sum_{i=1}^{q} \alpha_{\sigma(i)} = 1$$

Formalisation

Pour chaque anneau candidat, il existe des contraintes cachées :

$$\begin{cases} s_{\sigma(i),m} \geq 0,\ p_{\sigma(i),m} \geq 0 & 1 \leq i \leq q, 1 \leq m \leq E \\ c_{\sigma(i),\mathsf{succ}(\sigma(i))} = \frac{1}{\min_{e_m \in \mathcal{S}_{\sigma(i)}} s_{\sigma(i),m}} & 1 \leq i \leq q \\ c_{\sigma(i),\mathsf{pred}(\sigma(i))} = \frac{1}{\min_{e_m \in \mathcal{P}_{\sigma(i)}} p_{\sigma(i),m}} & 1 \leq i \leq q \\ \sum_{i=1}^{q} (s_{\sigma(i),m} + p_{\sigma(i),m}) \leq b_{e_m} & 1 \leq m \leq E \end{cases}$$

Complexité

Théorème

Le problème de décision associé au problème d'optimisation SharedRing *est NP-complet.*

Démonstration

Idée générale :

- *réduction à un cycle Hamiltonien*

Heuristique et impact du partage des liens

- Description de l'heuristique
  - ▷ construction de l'anneau avec un algorithme glouton,
  - ▷ assignation des fractions de bandes passantes pendant la construction de l'anneau solution,
  - ▷ optimisations finales
- Impact du partage des liens de communication
  - ▷ première heuristique :
    - ⋆ construit un anneau solution sans prendre en compte le partage de liens à la construction de l'anneau.
    - ⋆ solution pas forcément réalisable (réseau réel $\neq$ graphe complet).
    - ⋆ évaluation solution = optimisation de programmation quadratique.
  - ▷ seconde heuristique :
    - ⋆ heuristique gloutonne du problème SHAREDRING en utilisant l'optimisation de programmation quadratique.
    - ⋆ prise en compte de la contention des liens à la construction.

  $\Rightarrow$ coût des communications faible = heuristiques équivalentes.

  $\Rightarrow$ coût des communications élevé = 2^{nde} heuristique meilleure

Une modélisation précise des communications a un impact important sur la performance des stratégies d'équilibrage de charge.

Heuristique et impact du partage des liens

- Description de l'heuristique
  - ▷ construction de l'anneau avec un algorithme glouton,
  - ▷ assignation des fractions de bandes passantes pendant la construction de l'anneau solution,
  - ▷ optimisations finales
- Impact du partage des liens de communication
  - ▷ première heuristique :
    - ⋆ construit un anneau solution sans prendre en compte le partage de liens à la construction de l'anneau.
    - ⋆ solution pas forcément réalisable (réseau réel $\neq$ graphe complet).
    - ⋆ évaluation solution = optimisation de programmation quadratique.
  - ▷ seconde heuristique :
    - ⋆ heuristique gloutonne du problème SharedRing en utilisant l'optimisation de programmation quadratique.
    - ⋆ prise en compte de la contention des liens à la construction.

$\Rightarrow$ coût des communications faible = heuristiques équivalentes.

$\Rightarrow$ coût des communications élevé = 2^{nde} heuristique meilleure

Une modélisation précise des communications a un impact important sur la performance des stratégies d'équilibrage de charge.

Heuristique et impact du partage des liens

- Description de l'heuristique
  - ▷ construction de l'anneau avec un algorithme glouton,
  - ▷ assignation des fractions de bandes passantes pendant la construction de l'anneau solution,
  - ▷ optimisations finales
- Impact du partage des liens de communication
  - ▷ première heuristique :
    - ⋆ construit un anneau solution sans prendre en compte le partage de liens à la construction de l'anneau.
    - ⋆ solution pas forcément réalisable (réseau réel $\neq$ graphe complet).
    - ⋆ évaluation solution = optimisation de programmation quadratique.
  - ▷ seconde heuristique :
    - ⋆ heuristique gloutonne du problème SharedRing en utilisant l'optimisation de programmation quadratique.
    - ⋆ prise en compte de la contention des liens à la construction.

$\Rightarrow$ coût des communications faible = heuristiques équivalentes.

$\Rightarrow$ coût des communications élevé = 2^{nde} heuristique meilleure

Une modélisation précise des communications a un impact important sur la performance des stratégies d'équilibrage de charge.

Heuristique et impact du partage des liens

- Description de l'heuristique
  - ▷ construction de l'anneau avec un algorithme glouton,
  - ▷ assignation des fractions de bandes passantes pendant la construction de l'anneau solution,
  - ▷ optimisations finales
- Impact du partage des liens de communication
  - ▷ première heuristique :
    - ⋆ construit un anneau solution sans prendre en compte le partage de liens à la construction de l'anneau.
    - ⋆ solution pas forcément réalisable (réseau réel $\neq$ graphe complet).
    - ⋆ évaluation solution $=$ optimisation de programmation quadratique.
  - ▷ seconde heuristique :
    - ⋆ heuristique gloutonne du problème SharedRing en utilisant l'optimisation de programmation quadratique.
    - ⋆ prise en compte de la contention des liens à la construction.

  $\Rightarrow$ coût des communications faible $=$ heuristiques équivalentes.

  $\Rightarrow$ coût des communications élevé $= 2^{nde}$ heuristique meilleure

Une modélisation précise des communications a un impact important sur la performance des stratégies d'équilibrage de charge.

Heuristique et impact du partage des liens

- Description de l'heuristique
  - ▷ construction de l'anneau avec un algorithme glouton,
  - ▷ assignation des fractions de bandes passantes pendant la construction de l'anneau solution,
  - ▷ optimisations finales
- Impact du partage des liens de communication
  - ▷ première heuristique :
    - ⋆ construit un anneau solution sans prendre en compte le partage de liens à la construction de l'anneau.
    - ⋆ solution pas forcément réalisable (réseau réel $\neq$ graphe complet).
    - ⋆ évaluation solution $=$ optimisation de programmation quadratique.
  - ▷ seconde heuristique :
    - ⋆ heuristique gloutonne du problème SharedRing en utilisant l'optimisation de programmation quadratique.
    - ⋆ prise en compte de la contention des liens à la construction.

  $\Rightarrow$ coût des communications faible $=$ heuristiques équivalentes.

  $\Rightarrow$ coût des communications élevé $= 2^{nde}$ heuristique meilleure

Une modélisation précise des communications a un impact important sur la performance des stratégies d'équilibrage de charge.

Impact du partage des liens : simulations

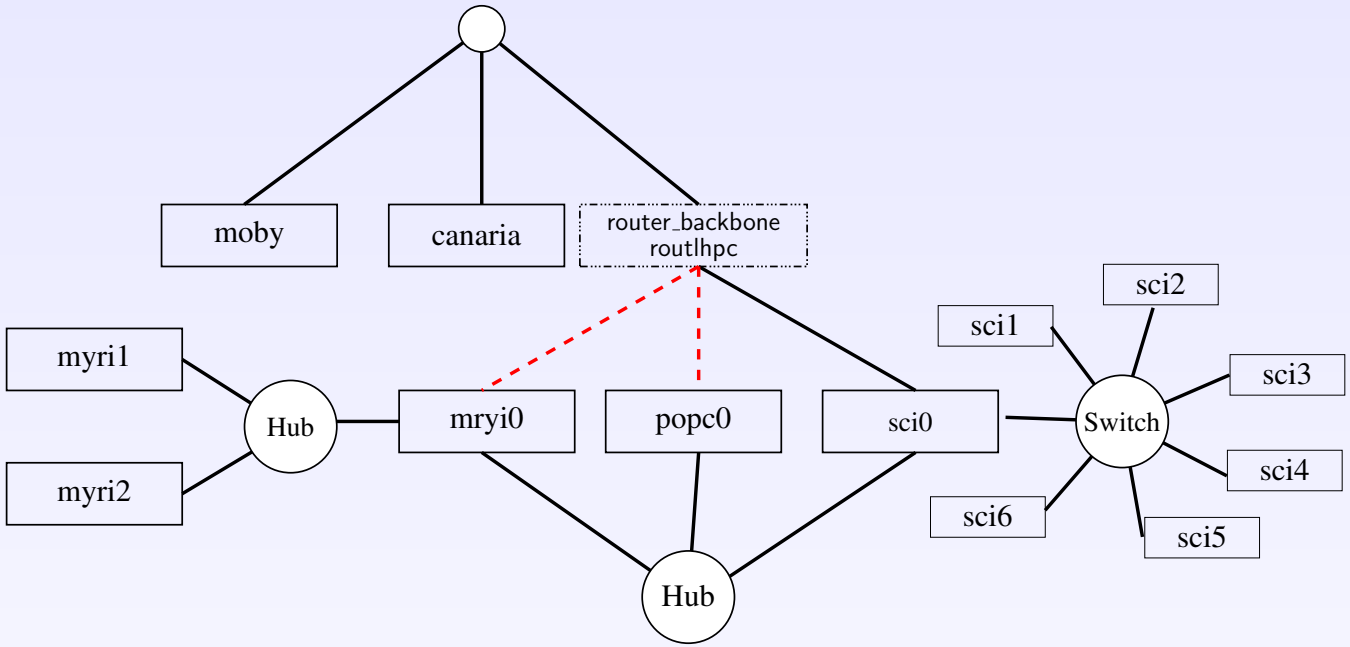


FIG.: Topologie de la plate-forme de Lyon.

Ratio D_c/D_w	H1 : slice-ring	H2 : shared-ring	Amélioration
0,1	3,17	3,15	0,63%
1	3,46	3,22	6,94%
10	10,39	3,9	62,46%

TAB.: T_{step}/D_w pour chaque heuristique sur la plate-forme de Lyon.

Plan de l'exposé

Contexte de redistribution

Plates-formes ciblées : plates-formes distribuées hétérogènes (réseau de stations de travail, clusters de clusters, grilles, etc.)

1. Différentes variations : ressources ou besoins de l'application.
2. Les données doivent être redistribuées afin d'obtenir un meilleur équilibrage de charge.
3. Pas de discussion du mécanisme d'équilibrage de charge : nous le considérons comme extérieur.

Notations et hypothèses

- Processeurs : P_1, ..., P_n.
- Initialement, le processeur P_i possède L_i données.

  δ_i est le déséquilibre de P_i : après redistribution, P_i possédera $L_i - \delta_i$ données.

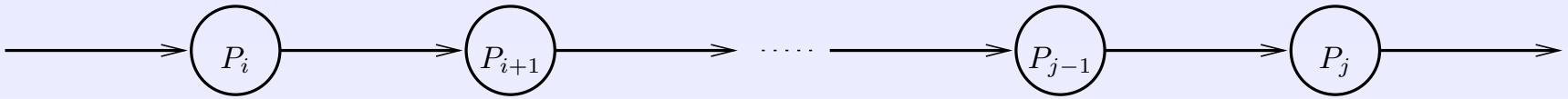
  Loi de conservation des données : $\sum_i \delta_i = 0$

  Hypothèses concernant le déséquilibre :
  - Un processeur possède au moins une donnée avant la redistribution : $L_i \geq 1$.
  - Un processeur possède au moins une donnée après la redistribution : $L_i \geq 1 + \delta_i$.

- $c_{i,i+1}$: temps nécessaire pour l'envoi d'une donnée de P_i à P_{i+1}.
- Émission / Réception simultanées, mais un processeur ne peut envoyer qu'à un seul processeur à la fois. (De même en réception)

Plan de l'exposé

Cadre de travail



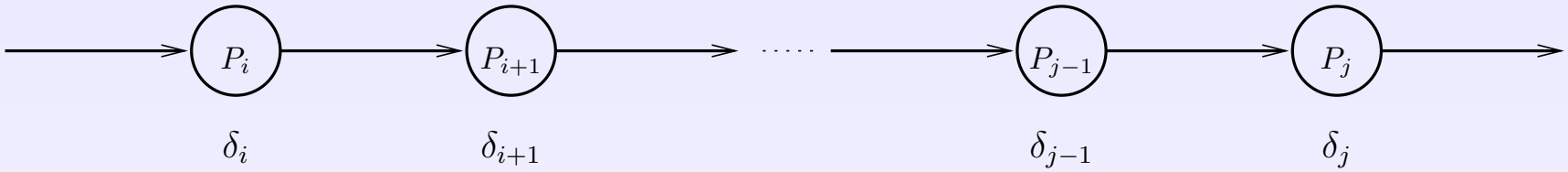
Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

Cela prend à P_i au moins $\delta_{i,j} \times c$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure : $\left(\max_{1 \leq i \leq n,\, 1 \leq j \leq n} \delta_{i,j} \right) \times c$

Borne inférieure du temps de redistribution



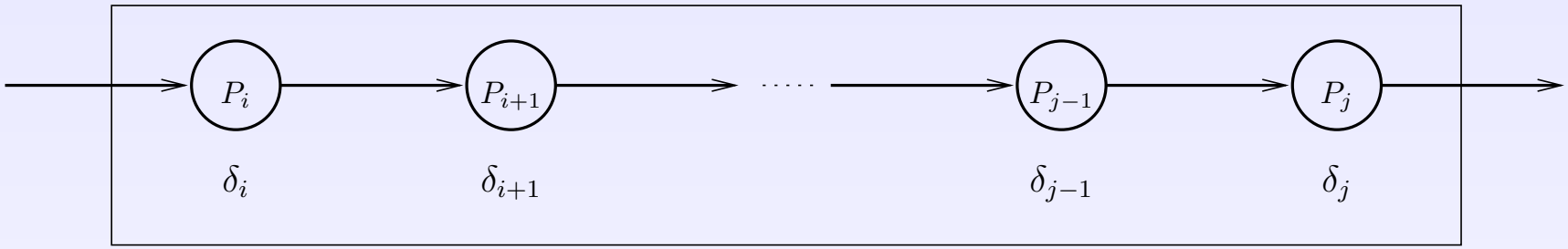
Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

Cela prend à P_i au moins $\delta_{i,j} \times c$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure : $\left(\max_{1 \leq i \leq n,\, 1 \leq j \leq n} \delta_{i,j} \right) \times c$

Borne inférieure du temps de redistribution



$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

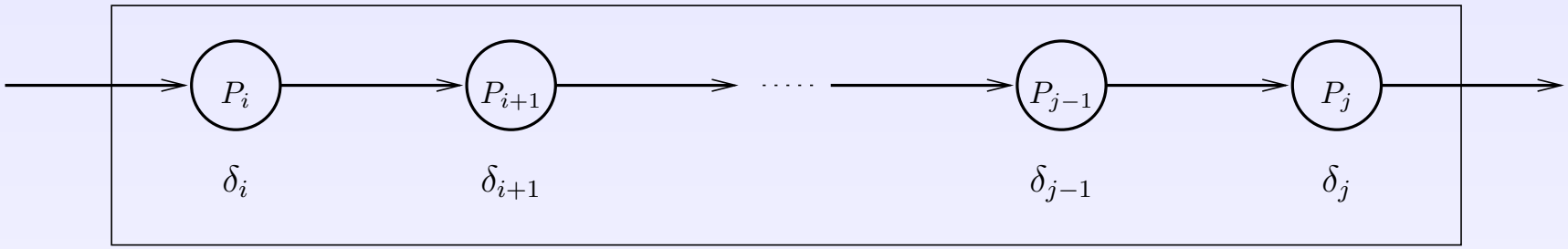
Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

Cela prend à P_i au moins $\delta_{i,j} \times c$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure : $\left(\max_{1 \leq i \leq n,\, 1 \leq j \leq n} \delta_{i,j} \right) \times c$

Borne inférieure du temps de redistribution



$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

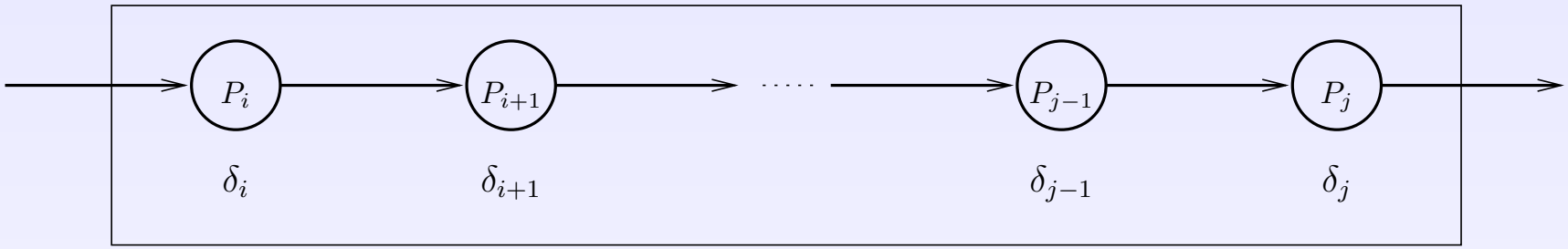
Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

Cela prend à P_i au moins $\delta_{i,j} \times c$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure : $\left(\max_{1 \leq i \leq n,\, 1 \leq j \leq n} \delta_{i,j} \right) \times c$

Borne inférieure du temps de redistribution



$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

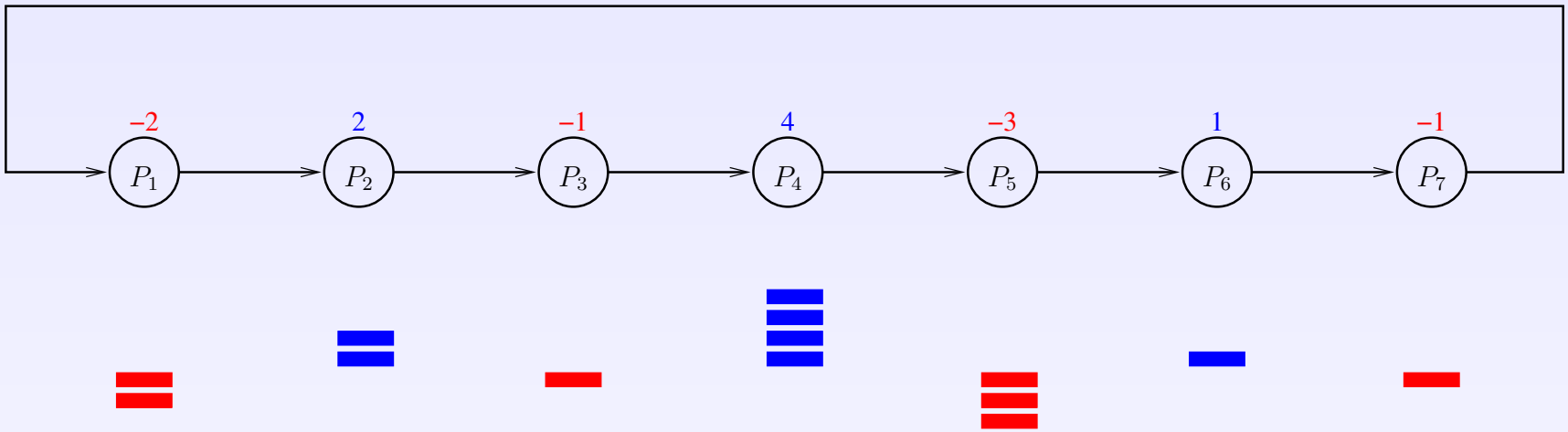
Capacité homogène des liens de communication : c.

P_i ne peut envoyer des données qu'à P_{i+1}.

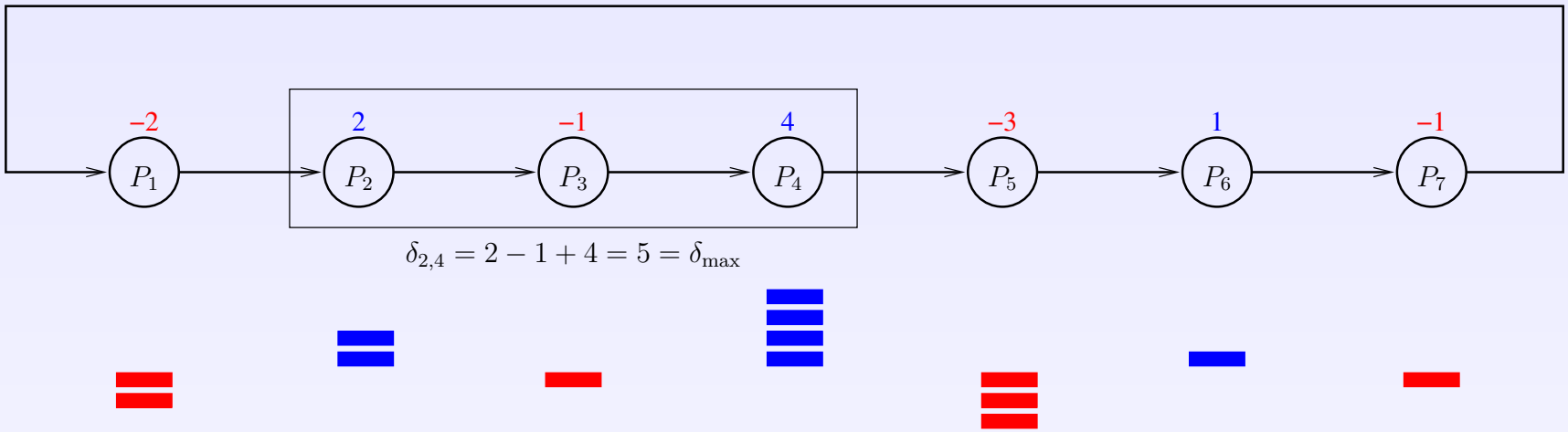
Cela prend à P_i au moins $\delta_{i,j} \times c$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure : $$\left(\max_{1 \leq i \leq n,\ 1 \leq j \leq n} \delta_{i,j} \right) \times c$$

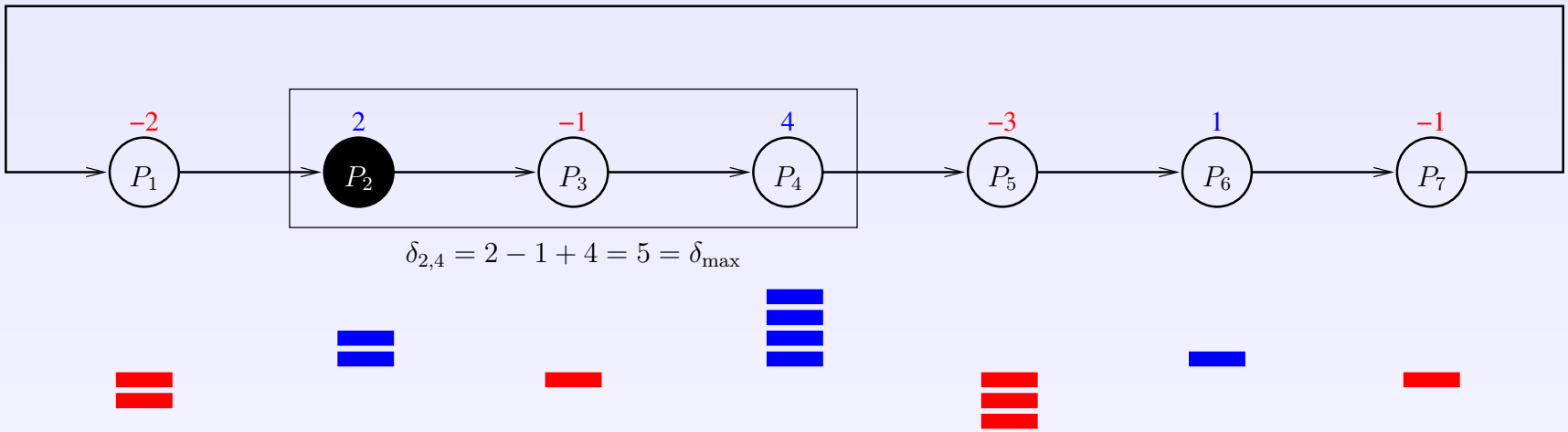
Algorithme de redistribution : exemple



Algorithme de redistribution : exemple



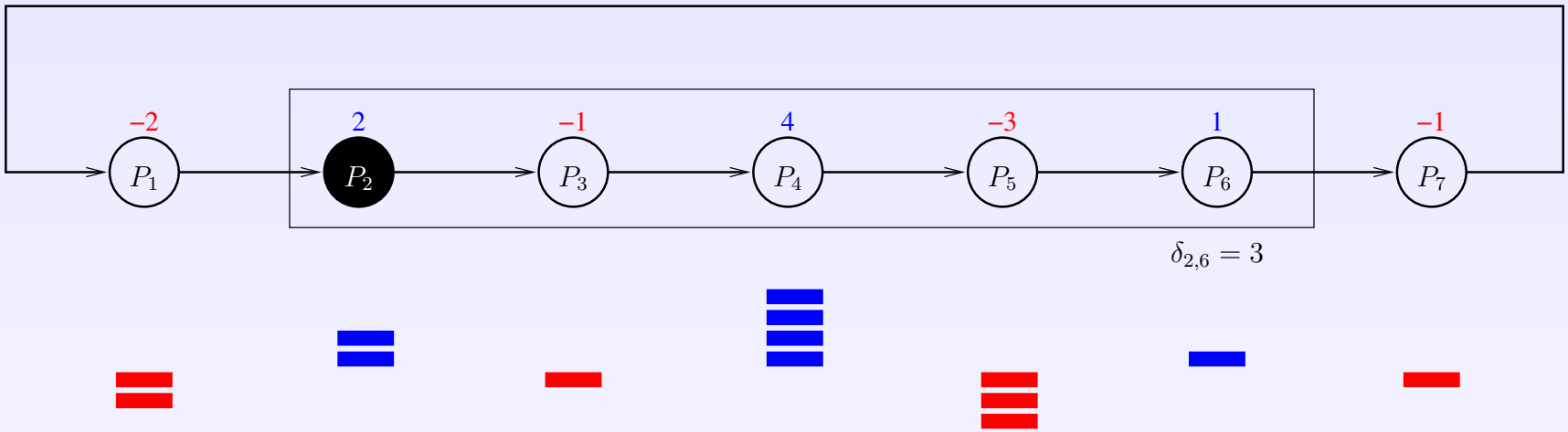
Algorithme de redistribution : exemple



$\delta_{\max} = 5$

L'algorithme complet de redistribution est défini par le premier processeur de déséquilibre maximal.

Algorithme de redistribution : exemple

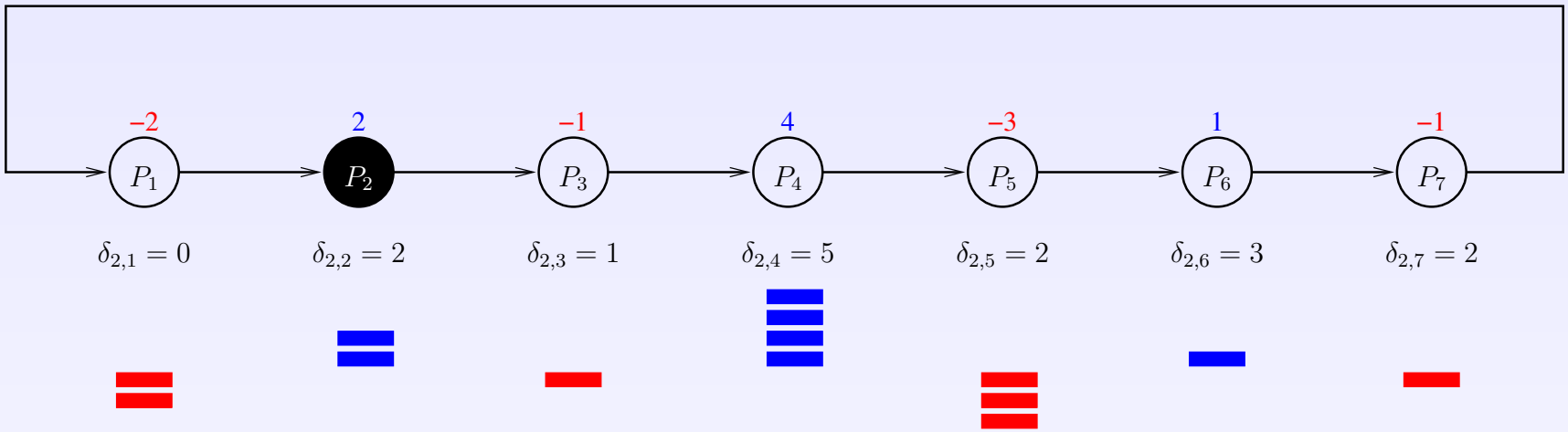


$\delta_{\text{max}} = 5$

L'algorithme complet de redistribution est défini par le premier processeur de déséquilibre maximal.

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

Algorithme de redistribution : exemple

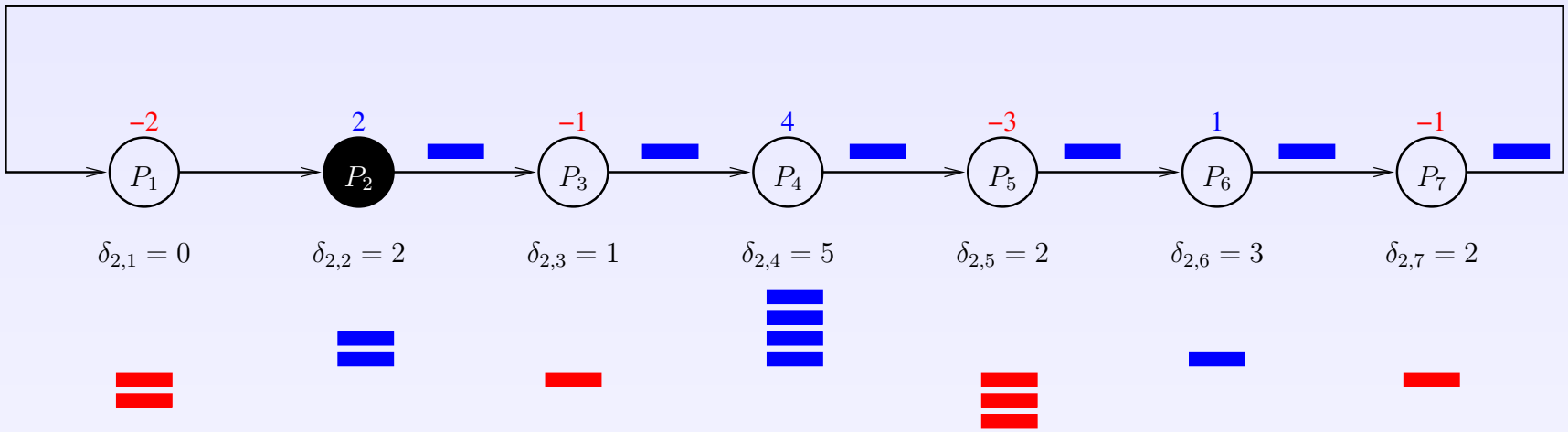


$\delta_{\text{max}} = 5$

L'algorithme complet de redistribution est défini par le premier processeur de déséquilibre maximal.

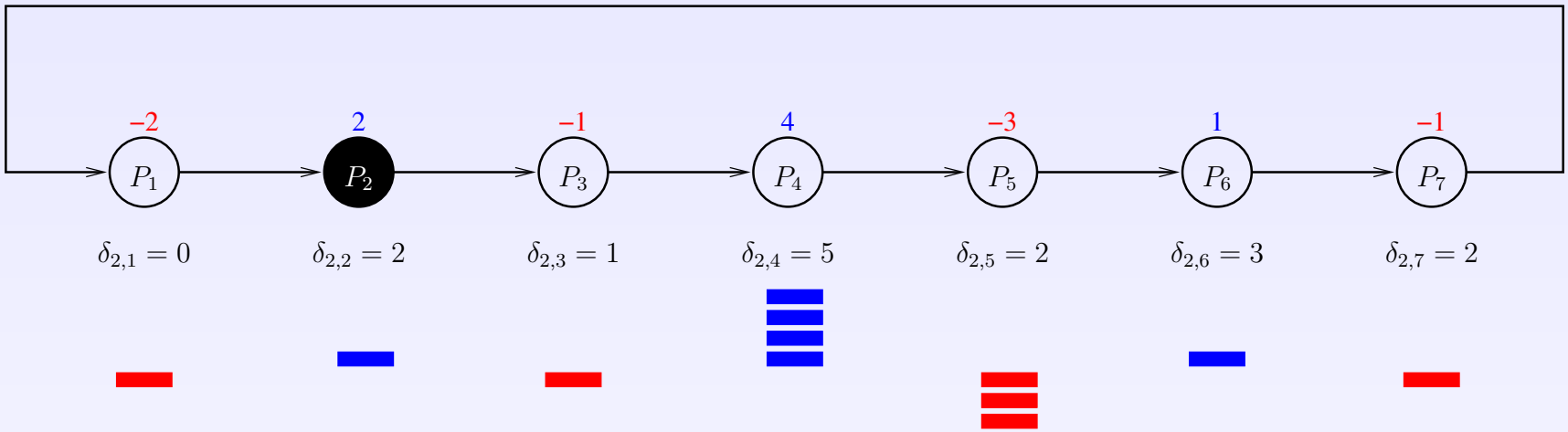
Pendant l'exécution de l'algorithme, le processeur P_i envoie $\delta_{2,i}$ données.

Algorithme de redistribution : exemple



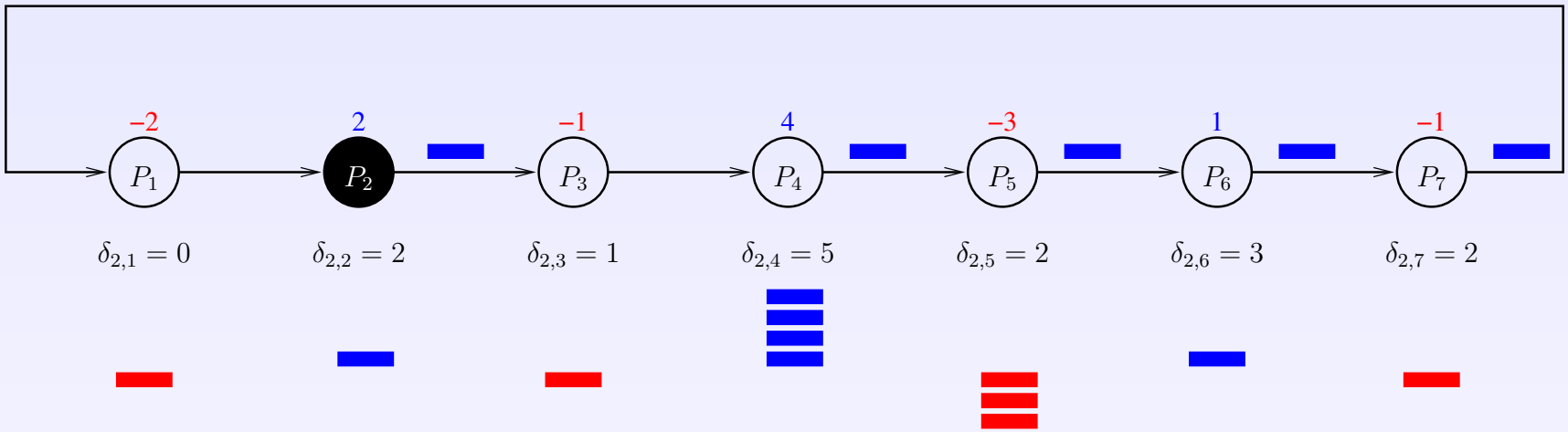
À l'étape 1, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 1$

Algorithme de redistribution : exemple



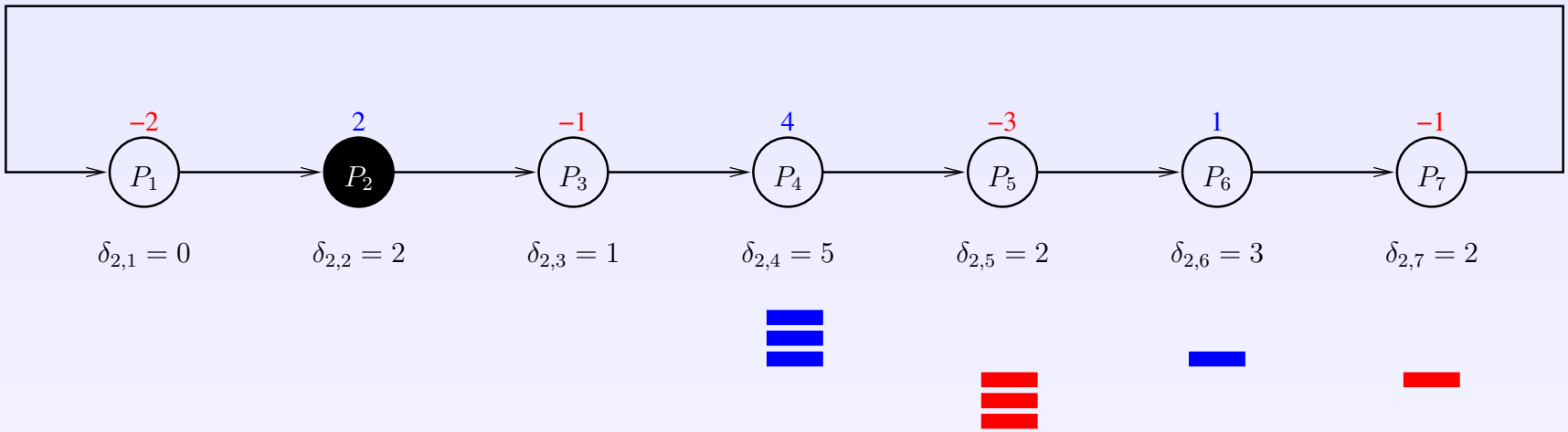
À l'étape 1, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 1$

Algorithme de redistribution : exemple



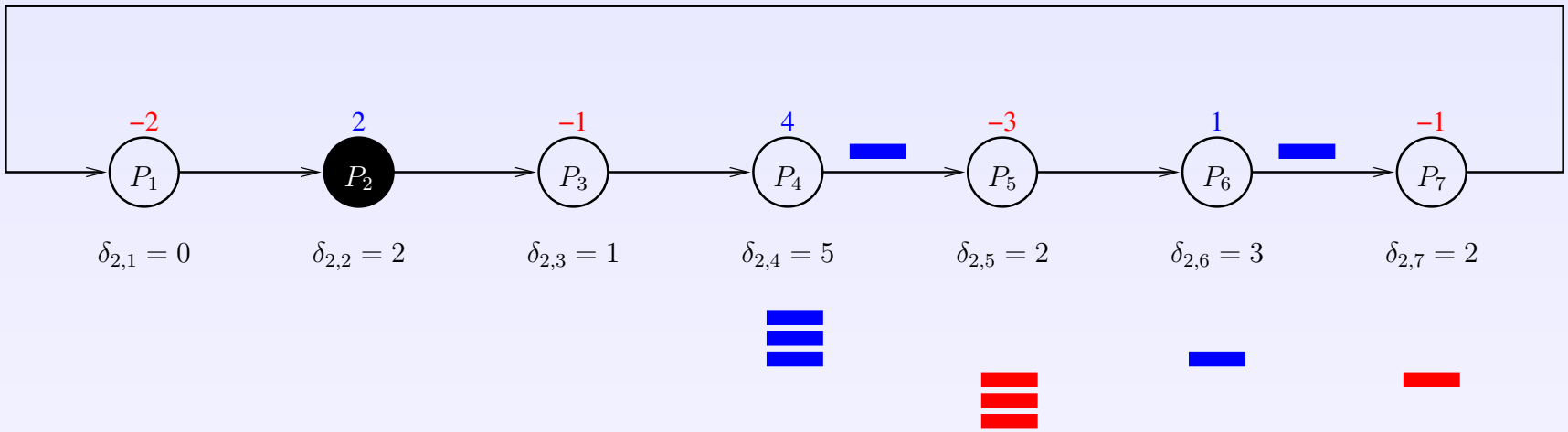
À l'étape 2, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 2$

Algorithme de redistribution : exemple



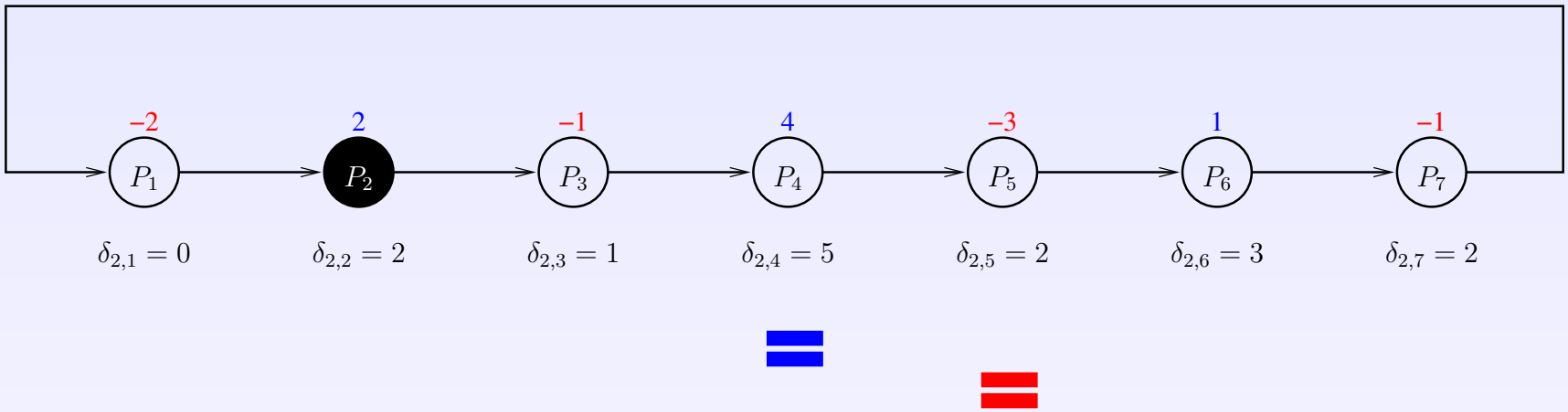
À l'étape 2, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 2$

Algorithme de redistribution : exemple



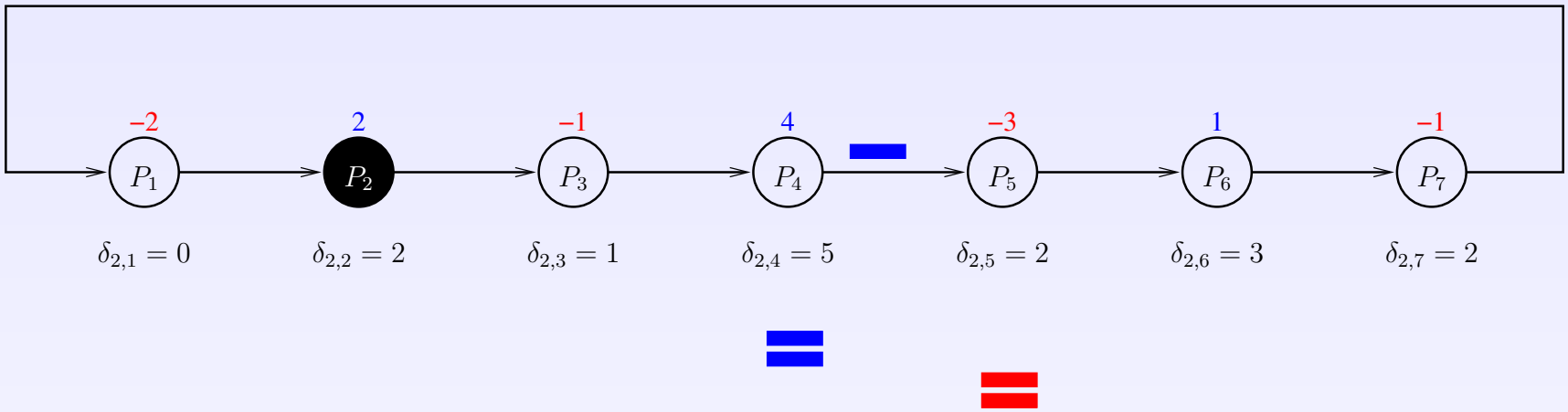
À l'étape 3, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 3$

Algorithme de redistribution : exemple



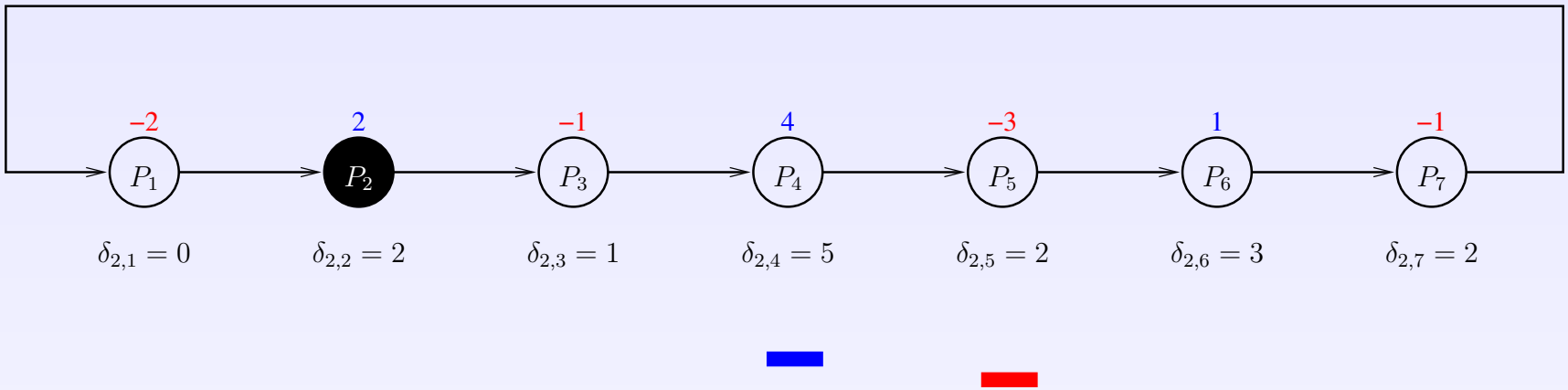
À l'étape 3, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 3$

Algorithme de redistribution : exemple



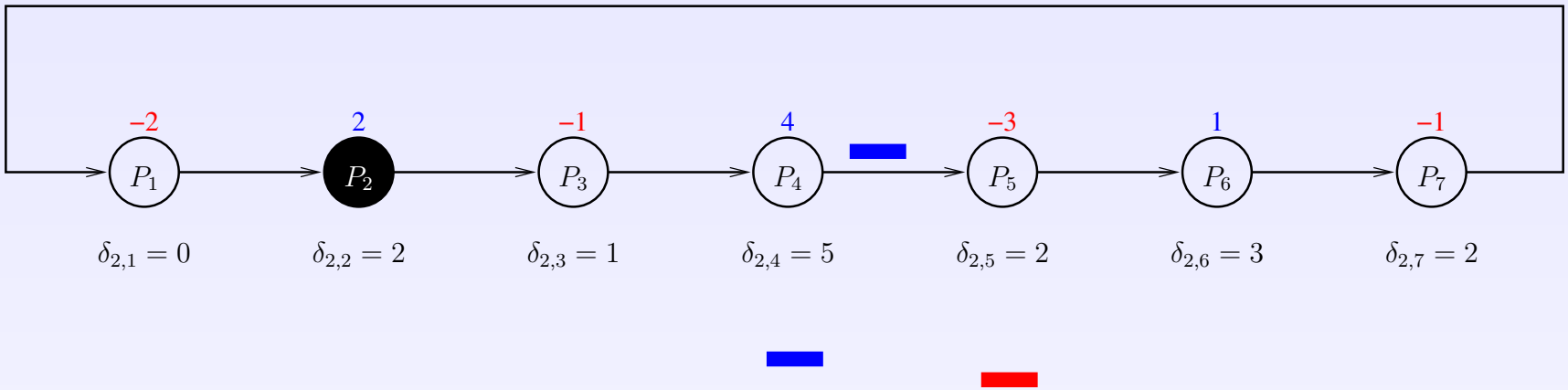
À l'étape 4, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 4$

Algorithme de redistribution : exemple



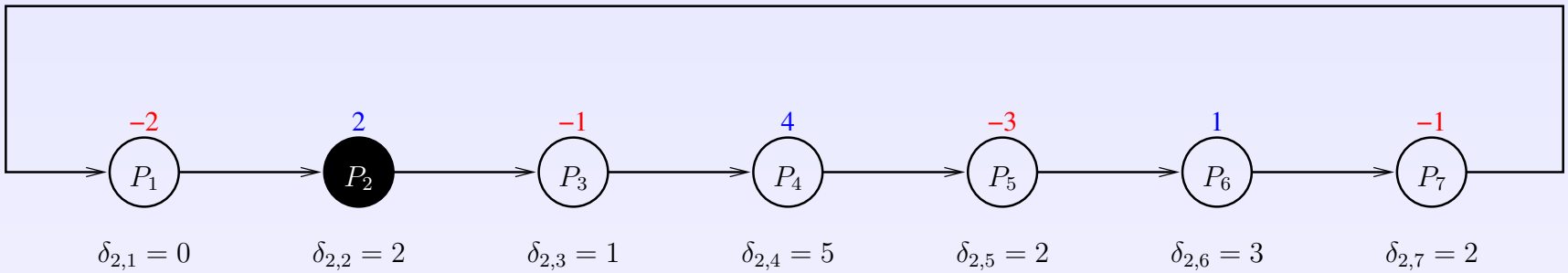
À l'étape 4, P_i envoie une donnée si et seulement si $\delta_{2,i} \geq 4$

Algorithme de redistribution : exemple



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

Algorithme de redistribution : exemple



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

L'algorithme

1: Soit $\delta_{\text{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$
2: Soient `start` et `end` deux indices tels que la tranche $C_{\texttt{start,end}}$ est de déséquilibre maximal : $\delta_{\texttt{start,end}} = \delta_{\text{max}}$.
3: **Pour** $s = 1$ à δ_{max} **:**
4: **Pour tout** $l = 0$ à $n - 1$ **:**
5: **Si** $\delta_{\texttt{start,start}+l} \geq s$ **Alors**
6: $P_{\texttt{start}+l}$ envoie à $P_{\texttt{start}+l+1}$ une donnée pendant l'intervalle de temps $[(s-1) \times c, s \times c[$

Théorème

Cet algorithme de redistribution est optimal.

Plan de l'exposé

Borne inférieure du temps de redistribution

Le processeur P_i a besoin de $c_{i,i+1}$ unités de temps pour envoyer une donnée au processeur P_{i+1}.

La borne inférieure est définie comme dans le cas de l'anneau homogène unidirectionnel.

Cela prend à P_i au moins $\delta_{i,j} \times c_{j,j+1}$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure :

$$\max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \delta_{i,j} \times c_{j,j+1}$$

Borne inférieure du temps de redistribution

Le processeur P_i a besoin de $c_{i,i+1}$ unités de temps pour envoyer une donnée au processeur P_{i+1}.

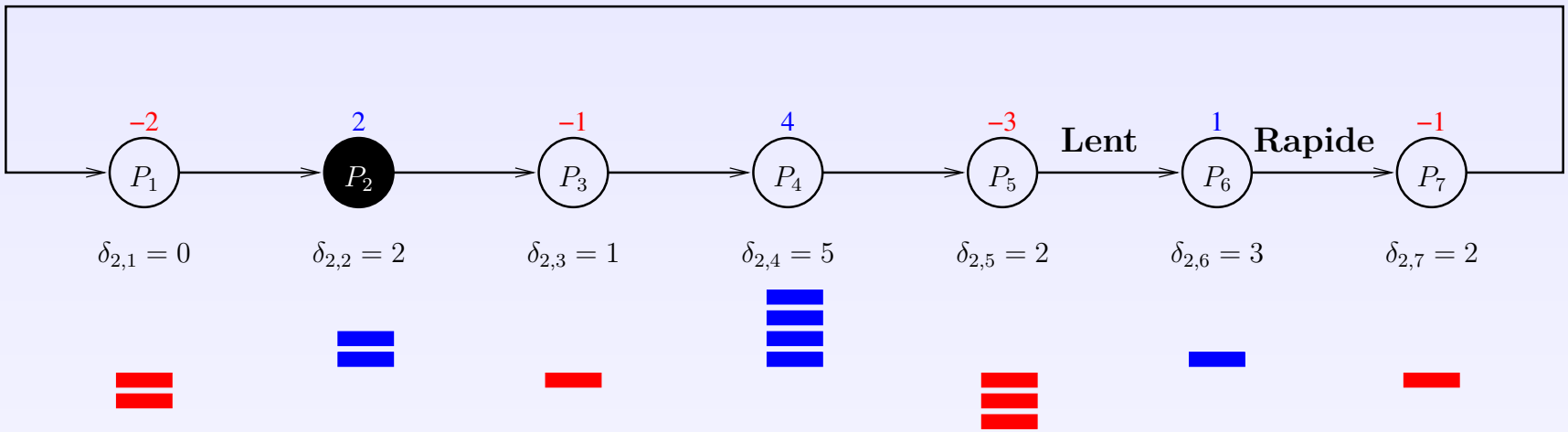
La borne inférieure est définie comme dans le cas de l'anneau homogène unidirectionnel.

Cela prend à P_i au moins $\delta_{i,j} \times c_{j,j+1}$ unités de temps pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure :

$$\max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \delta_{i,j} \times c_{j,j+1}$$

Conséquences de l'hétérogénéité des bandes passantes

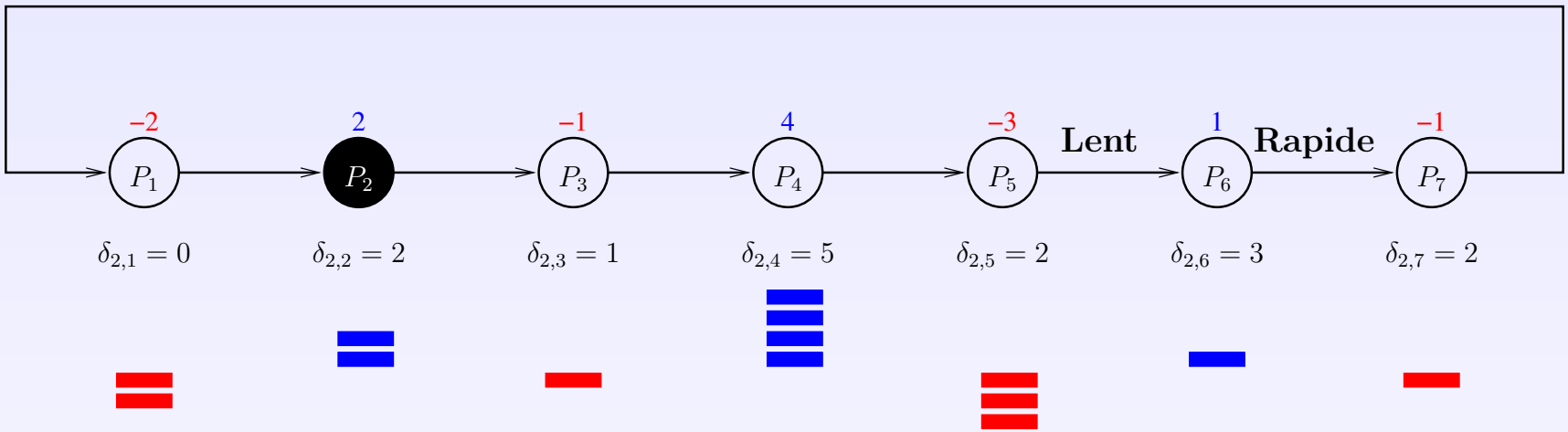


P_6 peut avoir à attendre des données de P_5 pour pouvoir envoyer toutes les données nécessaires à P_7.

Le temps d'exécution de P_6 ne peut pas être exprimé avec une formule simple.

L'algorithme de redistribution est asynchrone.

Conséquences de l'hétérogénéité des bandes passantes



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

Le temps d'exécution de P_6 ne peut pas être exprimé avec une formule simple.

L'algorithme de redistribution est asynchrone.

L'algorithme de redistribution

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

1: Soit $\delta_{\text{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$
2: Soient `start` et `end` deux indices tels que la tranche $C_{\texttt{start},\texttt{end}}$ est de déséquilibre maximal : $\delta_{\texttt{start},\texttt{end}} = \delta_{\text{max}}$.
3: **Pour tout** $l = 0$ à $n - 1$ **:**
4: $P_{\texttt{start}+l}$ envoie $\delta_{\texttt{start},\texttt{start}+l}$ données une par une et dès que possible au processeur $P_{\texttt{start}+l+1}$

Optimalité

Lemme

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

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

En d'autres termes, il n'y a aucun délai dû à l'attente, par un processeur, de la réception des données qu'il doit retransmettre.

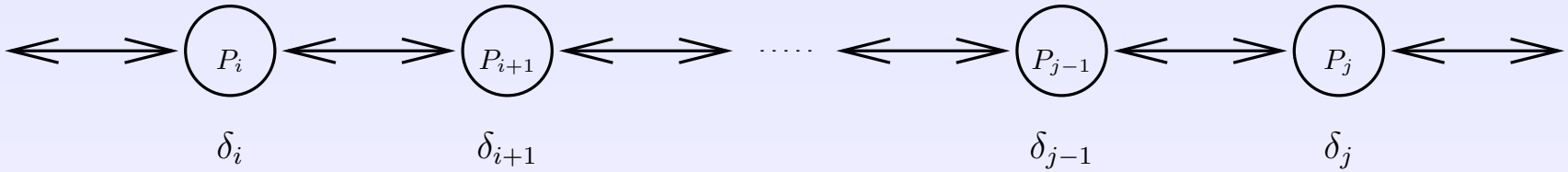
Plan de l'exposé

Cadre de travail

Capacité des liens de communication homogène : c.

Communications bi-directionnelles

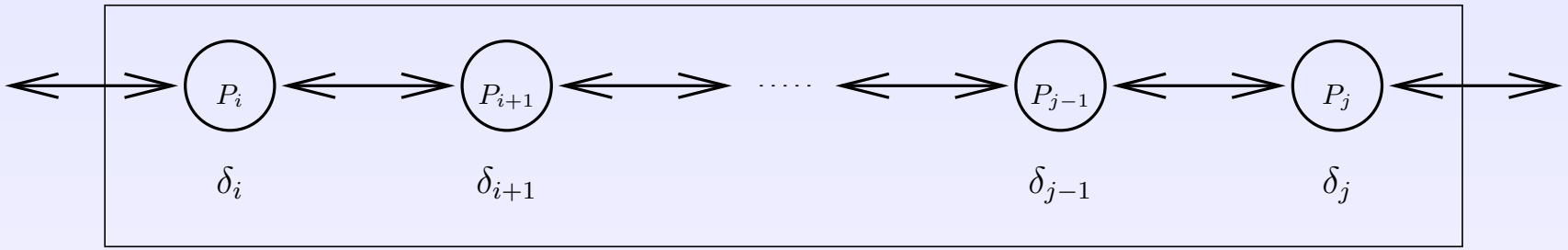
Borne inférieure du temps de redistribution



Capacité des liens de communication homogène : c.

Communications bi-directionnelles

Borne inférieure du temps de redistribution

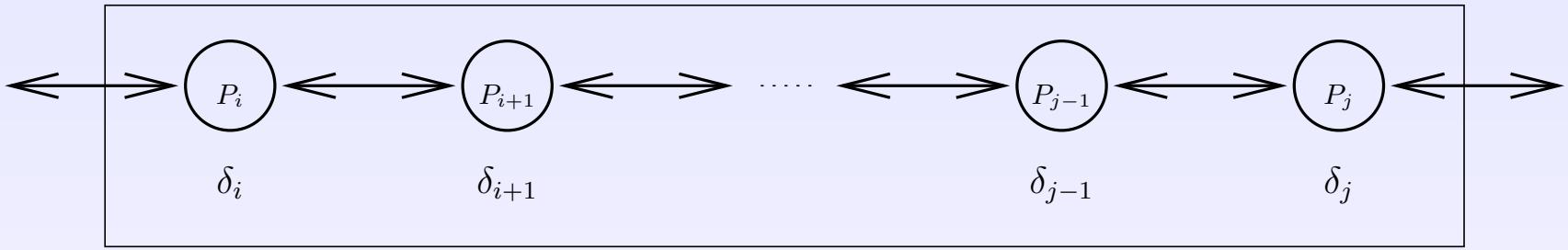


$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité des liens de communication homogène : c.

Communications bi-directionnelles

Borne inférieure du temps de redistribution



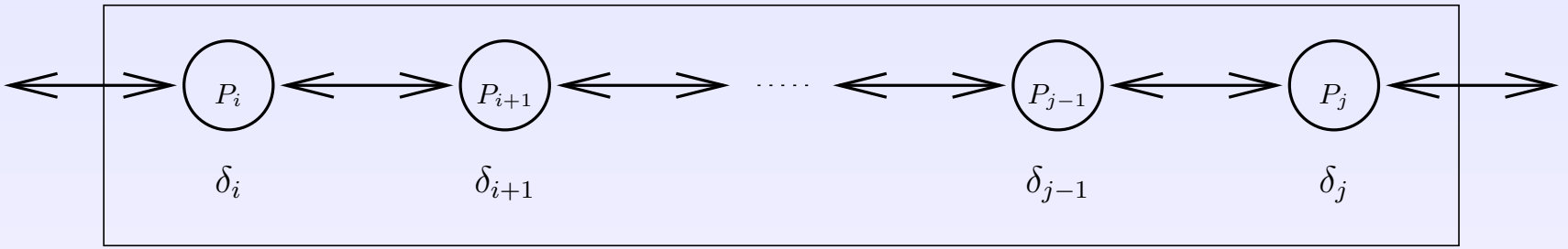
$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité des liens de communication homogène : c.

Communications bi-directionnelles

Cela prend au moins $\left\lceil \frac{\delta_{i,j}}{2} \right\rceil \times c$ unités de temps à un ensemble de processeurs $P_i, ..., P_j$ pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

Borne inférieure du temps de redistribution



$$\delta_{i,j} = \delta_i + \delta_{i+1} + ... + \delta_{j-1} + \delta_j$$

Capacité des liens de communication homogène : c.

Communications bi-directionnelles

Cela prend au moins $\left\lceil \frac{\delta_{i,j}}{2} \right\rceil \times c$ unités de temps à un ensemble de processeurs $P_i, ..., P_j$ pour envoyer $\delta_{i,j}$ données (si $\delta_{i,j} > 0$).

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

Intuitions de l'algorithme

1. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \geq 0$ doit envoyer deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.
2. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \leq 0$ doit recevoir deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.
3. Soit P_i tel que $|\delta_i| = \delta_{\max}$.
   Si P_i est déjà impliqué dans une communication (due aux cas précédents) : tout est arrangé.
   Sinon, nous avons la liberté de qui P_i recevra une donnée (cas $\delta_i \leq 0$), ou à qui P_i enverra une donnée (cas $\delta_i \geq 0$).
   De manière à simplifier l'algorithme, toutes les communications ont lieu dans le sens des indices croissants « de P_i vers P_{i+1} ».

Intuitions de l'algorithme

1. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \geq 0$ doit envoyer deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.

2. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \leq 0$ doit recevoir deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.

3. Soit P_i tel que $|\delta_i| = \delta_{\max}$.
   Si P_i est déjà impliqué dans une communication (due aux cas précédents) : tout est arrangé.
   Sinon, nous avons la liberté de qui P_i recevra une donnée (cas $\delta_i \leq 0$), ou à qui P_i enverra une donnée (cas $\delta_i \geq 0$).
   De manière à simplifier l'algorithme, toutes les communications ont lieu dans le sens des indices croissants « de P_i vers P_{i+1} ».

Intuitions de l'algorithme

1. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \geq 0$ doit envoyer deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.

2. Tout ensemble de processeurs contigus $P_i, ..., P_j$ tels que $\left\lceil \frac{|\delta_{i,j}|}{2} \right\rceil = \delta_{\text{max}}$ et $\delta_{i,j} \leq 0$ doit recevoir deux données à chaque étape de l'algorithme, une par chacune de ses extrémités.

3. Soit P_i tel que $|\delta_i| = \delta_{\max}$.
   Si P_i est déjà impliqué dans une communication (due aux cas précédents) : tout est arrangé.
   Sinon, nous avons la liberté de qui P_i recevra une donnée (cas $\delta_i \leq 0$), ou à qui P_i enverra une donnée (cas $\delta_i \geq 0$).
   De manière à simplifier l'algorithme, toutes les communications ont lieu dans le sens des indices croissants « de P_i vers P_{i+1} ».

Algorithme de redistribution optimal

1: Soit $\delta_{\max} = \max\{\max_{1\leq i\leq n} |\delta_i|, \max_{1\leq i\leq n, 1\leq l\leq n-1} \left\lceil \frac{|\delta_{i,i+l}|}{2} \right\rceil\}$
2: **Si** $\delta_{\max} \geq 1$ **Alors**
3: **Si** $\delta_{\max} \neq 2$ **Alors**
4: **Pour tout** tranche $C_{k,l}$ telle que $\delta_{k,l} > 1$ et $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\max}$:
5: P_k envoie une donnée à P_{k-1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
6: P_l envoie une donnée à P_{l+1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
7: **Pour tout** tranche $C_{k,l}$ telle que $\delta_{k,l} < -1$ et $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\max}$:
8: P_{k-1} envoie une donnée à P_k pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
9: P_{l+1} envoie une donnée à P_l pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
10: **Sinon Si** $\delta_{\max} = 2$ **Alors**
11: **Pour tout** tranche $C_{k,l}$ telle que $\delta_{k,l} \geq 3$:
12: P_l envoie une donnée à P_{l+1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
13: **Pour tout** tranche $C_{k,l}$ telle que $\delta_{k,l} = 4$:
14: P_k envoie une donnée à P_{k-1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
15: **Pour tout** tranche $C_{k,l}$ telle que $\delta_{k,l} \leq -3$:
16: P_{k-1} envoie une donnée à P_k pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
17: **Pour tout** tranche $C_{k,l}$ telle que $\delta_{k,l} = -4$:
18: P_{l+1} envoie une donnée à P_l pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.

2: **Si** $\delta_{\max} \geq 1$ **Alors**
19: **Pour tout** processeur P_i tel que $\delta_i = \delta_{\max}$:
20: **Si** P_i n'est pas déjà en train d'envoyer une donnée pendant l'intervalle de temps $[(s-1)\times c, s\times c[$, à cause d'un des cas précédents **Alors**
21: P_i envoie une donnée à P_{i+1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
22: **Pour tout** processeur P_i tel que $\delta_i = -(\delta_{\max})$:
23: **Si** P_i n'est pas déjà en train de recevoir une donnée pendant l'intervalle de temps $[(s-1)\times c, s\times c[$ à cause d'un des cas précédents **Alors**
24: P_i reçoit une donnée de P_{i-1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
25: **Si** $\delta_{\max} = 1$ **Alors**
26: **Pour tout** processeur P_i tel que $\delta_i = 0$:
27: **Si** P_{i-1} envoie une donnée à P_i pendant l'intervalle de temps $[(s-1)\times c, s\times c[$ **Alors**
28: P_i envoie une donnée à P_{i+1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
29: **Si** P_{i+1} envoie une donnée à P_i pendant l'intervalle de temps $[(s-1)\times c, s\times c[$ **Alors**
30: P_i envoie une donnée à P_{i-1} pendant l'intervalle de temps $[(s-1)\times c, s\times c[$.
31: Appel récursif de l'algorithme $(s+1)$

Optimalité

Théorème

L'algorithme précédent est optimal.

La principale difficulté : manipulation de tous les cas particuliers. (effets de bord).

Plan de l'exposé

Borne inférieure

Même type de démonstration que précédemment mais en tenant compte de l'hétérogénéité des liens de communication.

$$\max \left\{ \begin{array}{l} \displaystyle\max_{1\leq k\leq n,\ \delta_k>0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \\ \displaystyle\max_{1\leq k\leq n,\ \delta_k<0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \\ \displaystyle\max_{\substack{1\leq k\leq n,\\ 1\leq l\leq n-2,\\ \delta_{k,k+l}>0}} \ \min_{0\leq i\leq \delta_{k,k+l}} \max\{i\cdot c_{k,k-1}, (\delta_{k,k+l}-i)\cdot c_{k+l,k+l+1}\} \\ \displaystyle\max_{\substack{1\leq k\leq n,\\ 1\leq l\leq n-2,\\ \delta_{k,k+l}<0}} \ \min_{0\leq i\leq -\delta_{k,k+l}} \max\{i\cdot c_{k-1,k}, -(\delta_{k,k+l}+i)\cdot c_{k+l+1,k+l}\} \end{array} \right.$$

Redistribution légère : définition

Définition

Une redistribution est légère si chaque processeur possède initialement toutes les données qu'il devra envoyer pendant l'exécution de l'algorithme.

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

Condition mathématique de la redistribution légère :

$$\mathcal{S}_{i,i+1} + \mathcal{S}_{i,i-1} \leq L_i.$$

Redistribution légère : résolution

Minimiser τ, avec les conditions :

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

Lemme

N'importe quelle solution entière du système précédent est faisable.

Lemme

L'une des deux approximations entières évidentes de la solution rationnelle optimale du système précédent est une solution optimale entière.

Redistribution légère : résolution

$$\text{Minimiser } \tau, \text{ avec les conditions :}$$

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

Lemme

N'importe quelle solution entière du système précédent est faisable.

Lemme

L'une des deux approximations entières évidentes de la solution rationnelle optimale du système précédent est une solution optimale entière.

Redistribution légère : résolution

MINIMISER τ, AVEC LES CONDITIONS :

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

Lemme

N'importe quelle solution entière du système précédent est faisable.

Lemme

L'une des deux approximations entières évidentes de la solution rationnelle optimale du système précédent est une solution optimale entière.

Cas général

Ouvert.

Simulations : description de la plate-forme

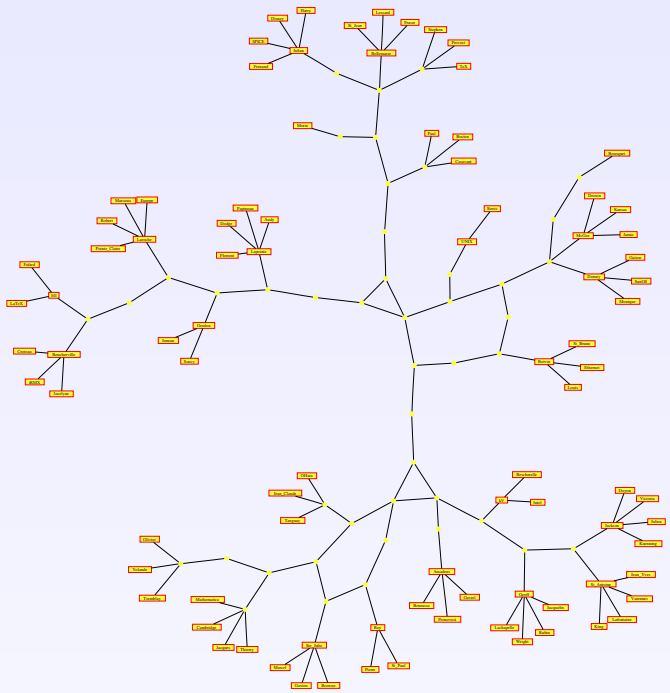


FIG.: Plate-forme générée par Tiers, composée de 90 nœuds, connectés par 192 liens de communication.

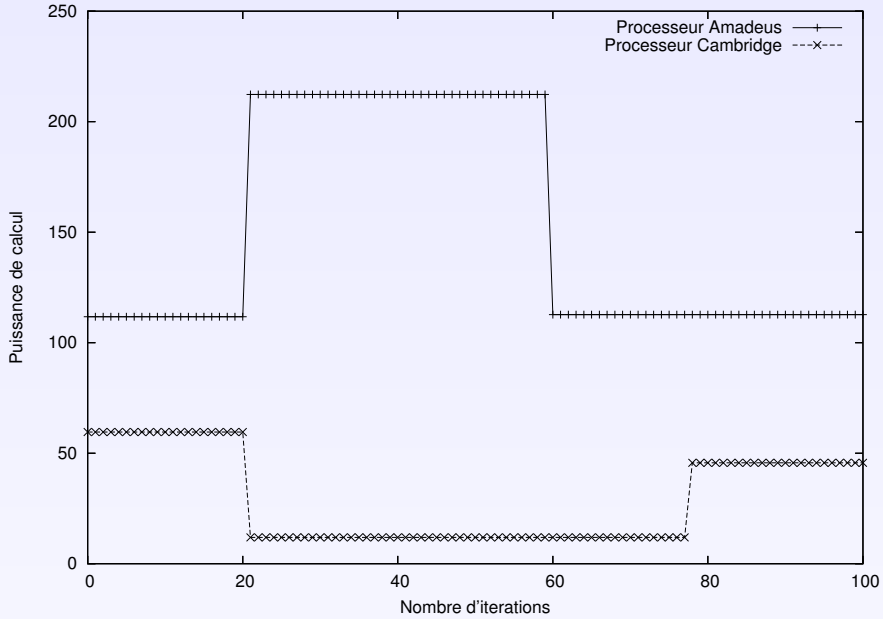


FIG.: Puissance de calcul de deux machines.

Utilisation du simulateur SIMGRID pour modéliser une application itérative.

Simulations : description

- Algorithme hétérogène bidirectionnel de la redistribution légère.
- Variation de la vitesse des processeurs 2 fois au cours du temps.
- 5 schémas de redistribution différents sur 100 itérations.
- Les phases de redistributions ont eu lieu de la manière suivante :
  - pas de redistribution ;
  - une seule redistribution à la 50^{e} itération ;
  - 4 redistributions, aux 20^{e}, 40^{e}, 60^{e} et 80^{e} itérations ;
  - 9 redistributions (une toutes les 10 itérations) ;
  - 19 redistributions (une toutes les 5 itérations).
- Prise en compte du ratio (puissance de calcul)/(vitesse de communication)

Plus ce paramètre décroît, plus les itérations prennent de temps et plus la redistribution de données devient nécessaire.

Simulations : description

- Algorithme hétérogène bidirectionnel de la redistribution légère.
- Variation de la vitesse des processeurs 2 fois au cours du temps.
- 5 schémas de redistribution différents sur 100 itérations.
- Les phases de redistributions ont eu lieu de la manière suivante :
  - pas de redistribution ;
  - une seule redistribution à la 50^{e} itération ;
  - 4 redistributions, aux 20^{e}, 40^{e}, 60^{e} et 80^{e} itérations ;
  - 9 redistributions (une toutes les 10 itérations) ;
  - 19 redistributions (une toutes les 5 itérations).
- Prise en compte du ratio (puissance de calcul)/(vitesse de communication)

Plus ce paramètre décroît, plus les itérations prennent de temps et plus la redistribution de données devient nécessaire.

Simulations

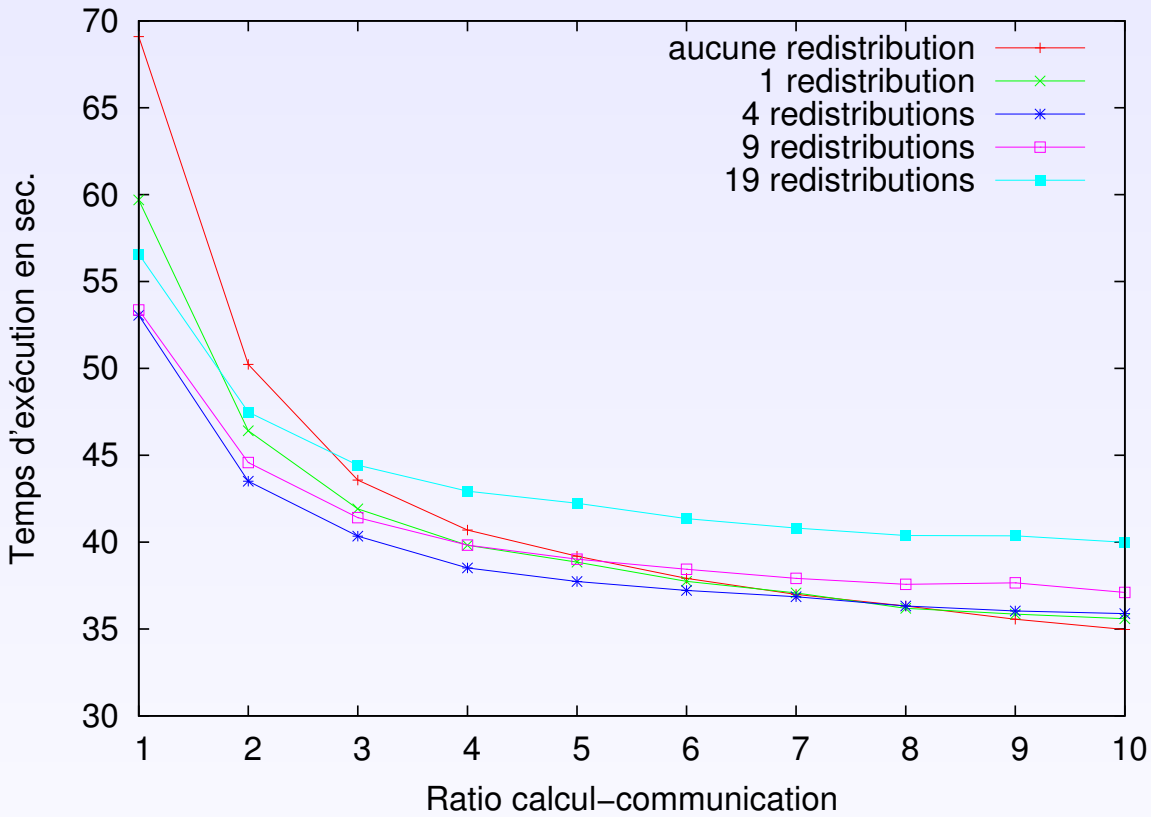


FIG.: Temps d'exécution en fonction du ratio calcul-communication, pour un anneau de 8 processeurs.

Simulations

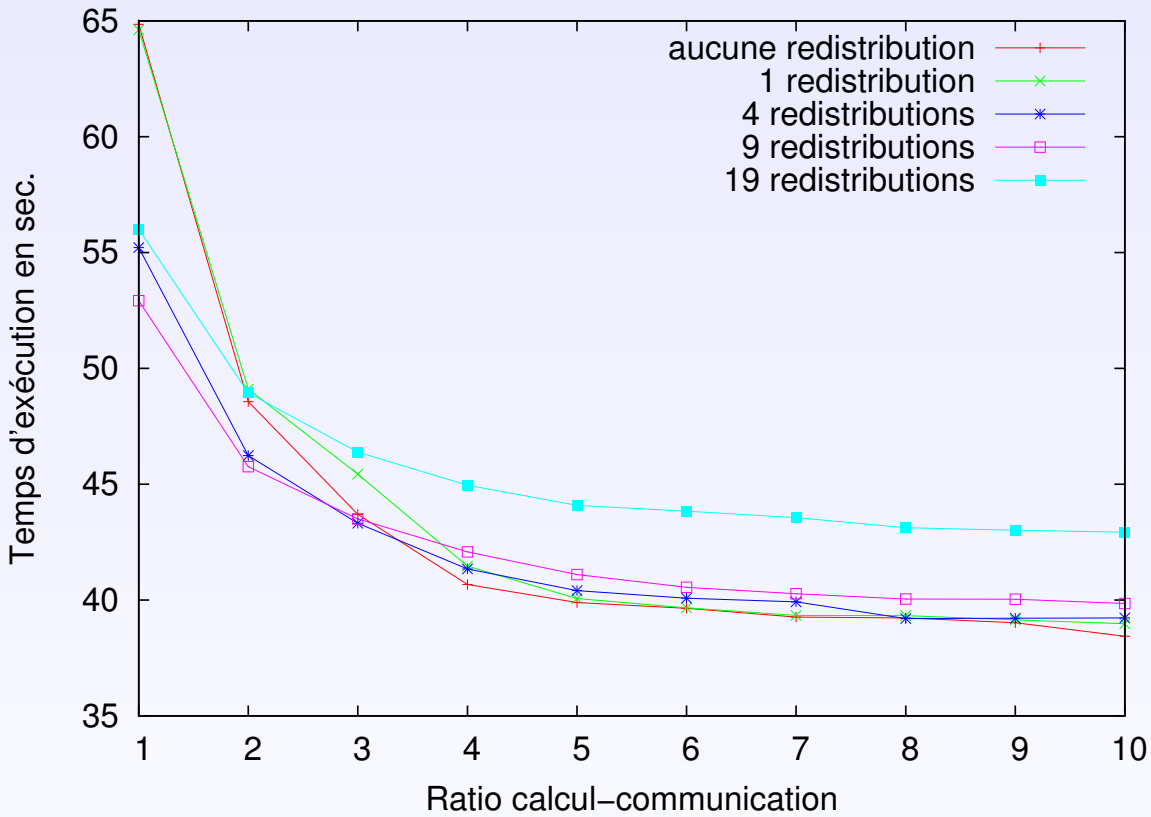


FIG.: Temps d'exécution en fonction du ratio calcul-communication, pour un anneau de 32 processeurs.

Plan de l'exposé

Plan de l'exposé

Conclusion

Difficultés rencontrées :

- Hétérogénéité des plates-formes
  $\Rightarrow$ Difficultés algorithmiques

1. **Placement et équilibrage de charge**
   - NP-complétude du problème d'optimisation SliceRing.
   - NP-complétude du problème d'optimisation SharedRing.
   - Heuristique efficace en temps polynomial pour résoudre ce problème.
   - Impact important du partage des liens de communication sur les stratégies d'équilibrage de charge.

2. **Redistribution de données**
   - Algorithme de redistribution optimal pour les anneaux :
     - homogènes unidirectionnels,
     - hétérogènes unidirectionnels,
     - homogènes bidirectionnels,
     - hétérogènes bidirectionnels avec l'hypothèse de redistribution légère.

Plan de l'exposé

Variation de performances

1. Variation significative des performances de la plate-forme cible
   - vitesse de calcul des processeurs
   - capacités des bandes passantes
2. Variation du nombre de nœuds de l'anneau solution

Objectif : étudier un problème d'optimisation en tenant compte des différentes variations.

Problème : décider *quand* et *comment* changer la solution courante.

Plusieurs stratégies de transition :

- Redistribution de données
- Mises-à-jour locales
- Changement global

Vue plus générale

1. Partitionnement multi-dimensionnel des données
   - ▷ difficile au niveau complexité
   - ▷ heuristiques possibles

2. Modèles algorithmiques plus complexes
   - ▷ problèmes irréguliers (*ex : matrices creuses*)
     - ⋆ décomposition de domaines
     - ⋆ non linéaire en la taille des données
     - ⋆ variation des communications en fonction du séparateur

Questions

« Le pire, quand on fait un discours, ce n'est pas de prendre conscience que vos auditeurs regardent leur montre, c'est le moment où ils se mettent à la secouer pour voir si elle n'est pas arrêtée. »

C'est à vous !