Algorithmes itératifs et plates-formes distribuées hétérogènes

Frédéric Vivien

INRIA - LIP

27 avril 2004

Plan de l'exposé

Plan de l'exposé

Plan de l'exposé

Plan de l'exposé

Plan de l'exposé

Plan de l'exposé

Le contexte: plates-formes distribuées hétérogènes

Évolution des plates-formes considérées

- Abandon des ordinateurs parallèles (coût...)
- Souhait d'utiliser toute la puissance disponible

Modification des problématiques

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

Plates-formes distribuées hétérogènes
(networks-of-workstations, clusters, grille, etc.)

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 ?

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 ?

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 ?

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 ?

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 ?

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 ?

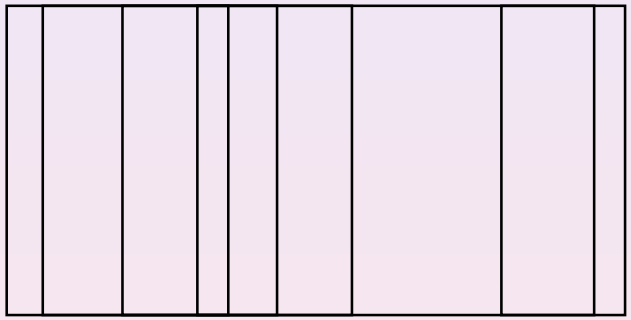
Simplification préalable: découpage par bande

- Données: un tableau 2D
- Découpage: par bandes verticales

- Conséquences:
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants: D_6
  3. Pas trop compliqué...

Simplification préalable: découpage par bande

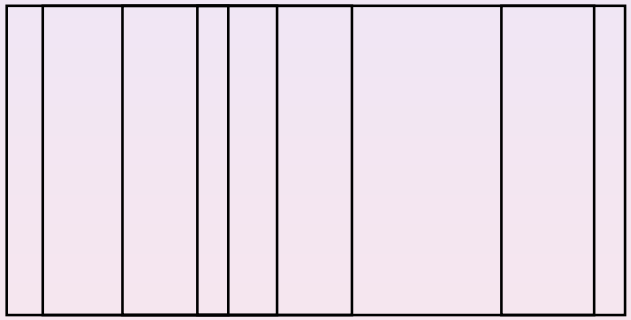
- Données: un tableau 2D
- Découpage: par bandes verticales



- Conséquences:
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants: D_c
  3. Pas trop compliqué...

Simplification préalable: découpage par bande

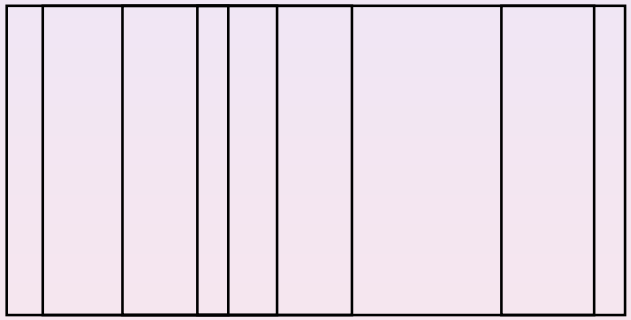
- Données: un tableau 2D
- Découpage: par bandes verticales



- Conséquences:
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants: D_c
  3. Pas trop compliqué...

Simplification préalable: découpage par bande

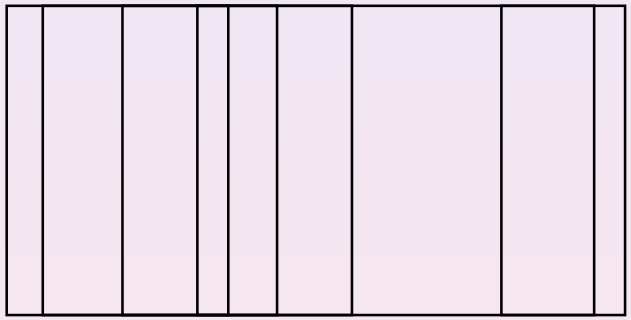
- Données: un tableau 2D
- Découpage: par bandes verticales



- Conséquences:
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants: D_c
  3. Pas trop compliqué...

Simplification préalable: découpage par bande

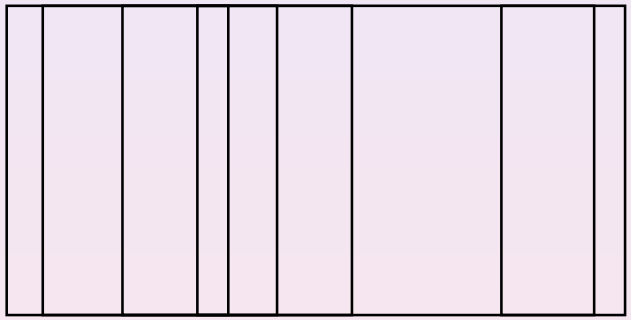
- Données: un tableau 2D
- Découpage: par bandes verticales



- Conséquences:
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants: D_c
  3. Pas trop compliqué...

Simplification préalable: découpage par bande

- Données: un tableau 2D
- Découpage: par bandes verticales



- Conséquences:
  1. Détermination simple des frontières et des voisins
  2. Volume de données à échanger entre voisins constants: D_c
  3. Pas trop compliqué...

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: $\alpha_i.D_w$
  $(\alpha_i \geq 0, \sum_j \alpha_j = 1)$
- Communications: modèle 1-port (envoi à un voisin à la fois ; réception d'un voisin à la fois ; envoi et réception simultanés)
- 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: $D_c.c_{i,\mathrm{succ}(i)}$

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: $\alpha_i.D_w$
  $(\alpha_i \geq 0, \sum_j \alpha_j = 1)$
- Communications: modèle 1-port (envoi à un voisin à la fois ; réception d'un voisin à la fois ; envoi et réception simultanés)
- 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: $D_c.c_{i,\text{succ}(i)}$

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: $\alpha_i.D_w$
  ($\alpha_i \geq 0$, $\sum_j \alpha_j = 1$)
- Communications: modèle 1-port (envoi à un voisin à la fois ; réception d'un voisin à la fois ; envoi et réception simultanés)
- 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: $D_c.c_{i,\text{succ}(i)}$

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: $\alpha_i.D_w$
  ($\alpha_i \geq 0$, $\sum_j \alpha_j = 1$)
- Communications: modèle 1-port (envoi à un voisin à la fois ; réception d'un voisin à la fois ; envoi et réception simultanés)
- 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: $D_c.c_{i,\text{succ}(i)}$

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: $\alpha_i.D_w$
  ($\alpha_i \geq 0$, $\sum_j \alpha_j = 1$)
- Communications: modèle 1-port (envoi à un voisin à la fois ; réception d'un voisin à la fois ; envoi et réception simultanés)
- 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: $D_c.c_{i,\text{succ}(i)}$

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: $\alpha_i.D_w$
  ($\alpha_i \geq 0$, $\sum_j \alpha_j = 1$)
- Communications: modèle 1-port (envoi à un voisin à la fois ; réception d'un voisin à la fois ; envoi et réception simultanés)
- 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: $D_c.c_{i,\mathsf{succ}(i)}$

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

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

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

Objectif

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

Afin de minimiser:

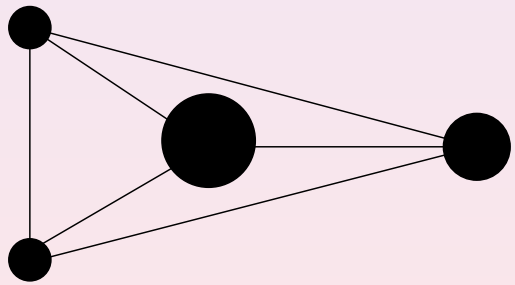
$$\max_{1 \leq i \leq p} \mathbb{I}\{i\}[\alpha_i . D_w . w_i + D_c . (c_{i,\text{pred}(i)} + c_{i,\text{succ}(i)})]$$

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
- Si tous les processeurs participent, ils finissent tous leur part de calcul en même temps
- Temps pour la solution optimale:

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:

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

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:

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

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:

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

Conséquences

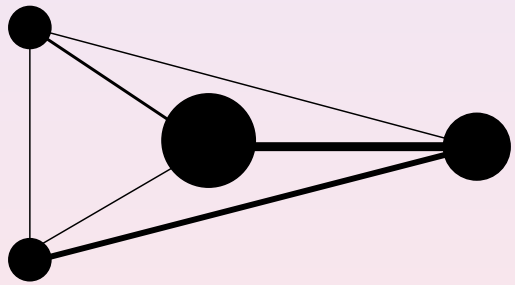
- Soit le processeur le plus rapide fait tout le travail, soit tous les processeurs participent
- Si tous les processeurs participent, ils finissent tous leur part de calcul en même temps $\alpha_i.D_w$ *rationnels ? ? ?*
  ($\exists \tau, \quad \alpha_i.D_w.w_i = \tau$, d'où $1 = \sum_i \frac{\tau}{D_w.w_i}$)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

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

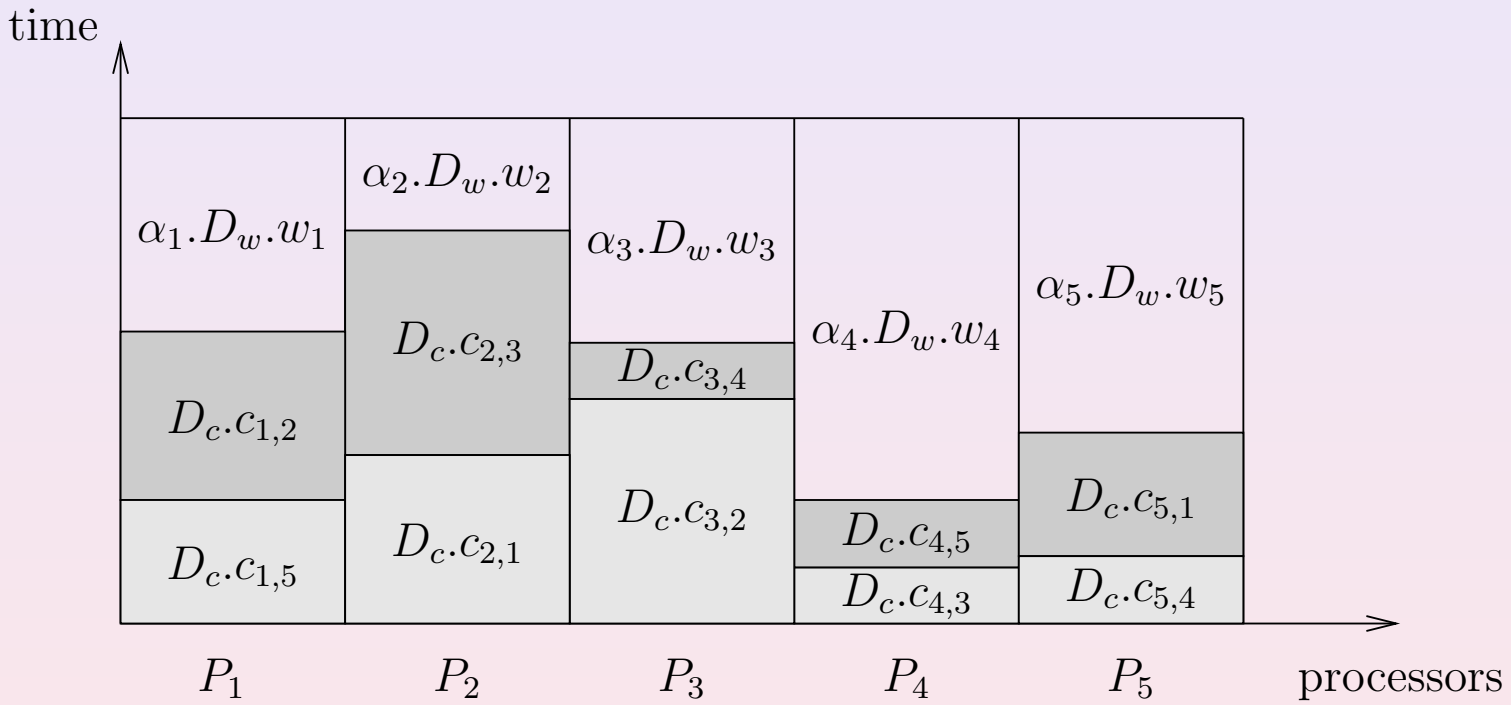
Plan de l'exposé

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:

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

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

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

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: 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: 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: 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

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

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

MINIMISER T SATISFAISANTS LES (IN)ÉQUATIONS

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

La programmation linéaire

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

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...

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 (e.g. 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...

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (e.g. 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...

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (e.g. 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...

Et en pratique ?

Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (e.g. 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...

Et en pratique ?

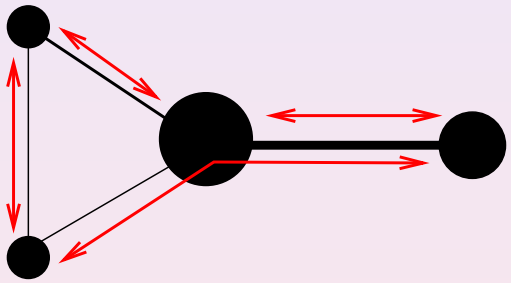
Tous les processeurs participent. On utilise une heuristique de résolution du problème de voyageur de commerce (e.g. Lin-Kernighan) Pas de garanties, mais en pratique les résultats sont excellents.

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 comunications



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 $D_i + \sum_{e_m \in \mathcal{S}_i} s_{i,m} L_{e_m}$ pour envoyer à son successeur un message de taille D_i
  Contraintes sur la bande passante de e_m : $\sum_{i \,|\, e_m \in \mathcal{S}_i} s_{i,m} \leq b_{e_m}$
- 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

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 $D_i \cdot \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ pour envoyer à son successeur un message de taille D_i
  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

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
  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 $P_{\mathsf{pred}(i)}$ dans le réseau, qui utilise une partie $p_{i,m}$ 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 $P_{\mathsf{succ}(i)}$ dans le réseau
  $\mathcal{S}_i$ utilise une partie $s_{i,m}$ de la bande passante b_{e_m} du lien e_m

  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 $P_{\mathsf{pred}(i)}$ dans le réseau, qui utilise une partie $p_{i,m}$ 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
  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

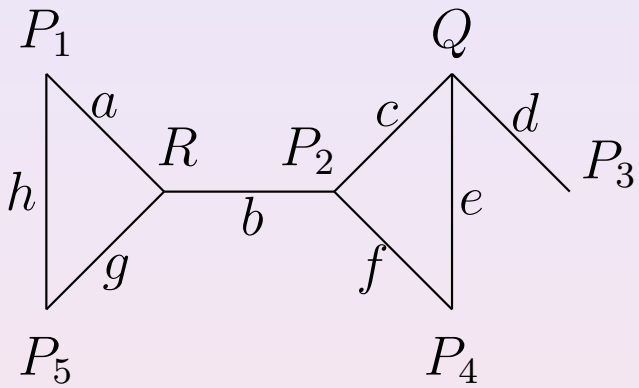
Nouvelles notations

- 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

Nouvelles notations

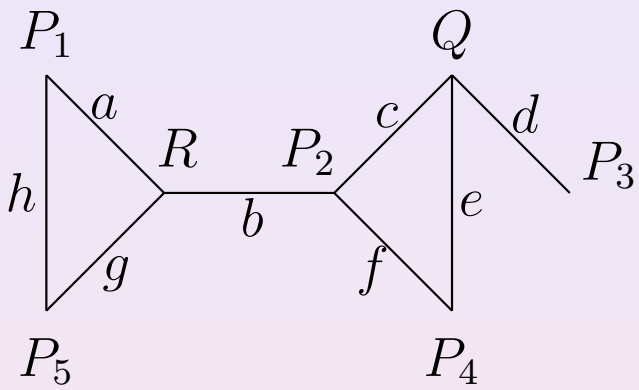
- Un ensemble de liens de communications: e_1, ..., e_n
- Bande passante de e_m: b_{e_m}
- Il y a un chemin $\mathcal{S}_i$ de P_i à $P_{\text{succ}(i)}$ dans le réseau
  $\mathcal{S}_i$ utilise une partie $s_{i,m}$ de la bande passante b_{e_m} du lien e_m
  P_i a besoin d'un temps $D_c.\frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ 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
- On choisit un anneau de 5 processeurs:
  $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5$ (on n'utilise ni Q, ni R)
- Les liens sont étiquetés de a à h
- On note b_x la bande passante du lien x

Exemple jouet : choix des chemins



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

Pour P_1: comme $\mathcal{S}_1 = \{a, b\}$, $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$.

Pour P_1: comme $\mathcal{P}_1 = \{h\}$, $c_{1,5} = \frac{1}{p_{1,h}}$.

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: partage des bandes passantes

Pour P_1: comme $\mathcal{S}_1 = \{a, b\}$, $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$.

Pour P_1: comme $\mathcal{P}_1 = \{h\}$, $c_{1,5} = \frac{1}{p_{1,h}}$.

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

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

Exemple jouet: moralité

Le problème est un système quadratique si

1. Les processeurs sont sélectionnés ;
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.

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.

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.

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.

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
- 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
- Résultats bien meilleurs si, dès le départ, on prend en compte les nécessaires partages de bandes passantes

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
- Résultats bien meilleurs si, dès le départ, on prend en compte les nécessaires partages de bandes passantes

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
- Résultats bien meilleurs si, dès le départ, on prend en compte les nécessaires partages de bandes passantes

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
- Résultats bien meilleurs si, dès le départ, on prend en compte les nécessaires partages de bandes passantes

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
- Résultats bien meilleurs si, dès le départ, on prend en compte les nécessaires partages de bandes passantes

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
- Résultats bien meilleurs si, dès le départ, on prend en compte les nécessaires partages de bandes passantes

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

  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 ?

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 ?

Conclusion

Le parallélisme « régulier » était déjà compliqué, maintennat 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

Solutions: des heuristiques non garanties à des algorithmes optimaux...