

CentraleSupélec

Mineure CalHau1

MPI : produit de matrices denses en anneau

Stéphane Vialle

Stephane.Vialle@centralesupelec.fr
http://www.metz.supelec.fr/~vialle

CentraleSupélec

Produit de matrices denses en anneau

- **Algorithme distribué en « anneau de processus »**
- Modélisation de performances sur un anneau théorique
- Topologie de processus vs réseau réels
- Modélisation des comms. selon déploiement

CentraleSupélec

Produit de matrices denses sur anneau

Algorithme distribué

Problème à résoudre :
A, B, C : $n \times n = N$ éléments

$$C = A \cdot B \quad c_{ij} = \sum_{k=1}^n (a_{ik} \cdot b_{kj}) \quad O(\text{Nbr d'opérations}) = O(N^3/2)$$

Comment répartir les données ?

Pb du **partitionnement**

Calculs déterministes localisés :
• Partitionnement statique

Calculs déterministes NON localisés :
• Duplication ?
• **Circulation** ?

CentraleSupélec

Produit de matrices denses sur anneau

Algorithme distribué

Partitionnement sur un anneau de processeurs :

- A partitionnée en blocs de lignes
- B et C partitionnées en blocs de colonnes
- **Circulation de A**
- B et C statiques

Topologie

Partitionnement et circulation de A

Partitionnement statique de B

Partitionnement statique de C

Etape 0 (état initial)

CentraleSupélec

Produit de matrices denses sur anneau

Algorithme distribué

Partitionnement sur un anneau de processeurs :

- A partitionnée en blocs de lignes
- B et C partitionnées en blocs de colonnes
- **Circulation de A**
- B et C statiques

Topologie

Partitionnement et circulation de A

Partitionnement statique de B

Partitionnement statique de C

Etape 1

CentraleSupélec

Produit de matrices denses sur anneau

Algorithme distribué

Partitionnement sur un anneau de processeurs :

- A partitionnée en blocs de lignes
- B et C partitionnées en blocs de colonnes
- **Circulation de A**
- B et C statiques

Topologie

Partitionnement et circulation de A

Partitionnement statique de B

Partitionnement statique de C

Etape 2

Produit de matrices denses sur anneau
Algorithme distribué

Partitionnement sur un anneau de processeurs :

- A partitionnée en blocs de lignes
- B et C partitionnées en blocs de colonnes
- **Circulation de A**
- B et C statiques

Résultats à la fin des P étapes :

Topologie
Partitionnement statique de C

Bilan :

- Chaque PC a calculé un bloc de colonnes de C
- Les P PC ont travaillé en parallèle
- **Calcul de tous les blocs de colonnes en parallèle**, en P étapes

Produit de matrices denses sur anneau
Algorithme distribué

Partitionnement sur un anneau de processeurs :

- A partitionnée en blocs de lignes
- B et C partitionnées en blocs de colonnes
- **Circulation de A**
- B et C statiques

Déroulement de l'algorithme sur PE-2, avec P = 4 :

Étape 0 C Étape 1 C Étape 2 C Étape 3 C

Produit de matrices denses sur anneau
Algorithme distribué

Stratégies d'implantation sur un anneau de P processeurs :

<pre>// Sans recouvrement for (step=0; step<P; step++) calcul(); barrier(); // si besoin circulation(); barrier(); // si besoin }</pre>	<pre>// Avec recouvrement for (step=0; step<P; step++) thread{ calcul(); } thread{ circulation(); } joinThreads(); permutBuff(); }</pre>
--	---

↓

- Concevoir l'algorithme avec des barrières de (re)synchronisation potentielles
- Selon le mécanisme de communication utilisé :
 - Implanter des barrières explicites : synchronisation forte ou bien
 - Se contenter de la synchro des comms : synchronisation relaxée

Produit de matrices denses sur anneau
Algorithme distribué

Stratégies d'implantation sur un anneau de P processeurs :

<pre>// Sans recouvrement for (step=0; step<P; step++) calcul(); barrier(); // si besoin circulation(); barrier(); // si besoin }</pre>	<pre>// Avec recouvrement for (step=0; step<P; step++) thread{ calcul(); } thread{ circulation(); } joinThreads(); // barrier permutBuff(); }</pre>
--	--

↓

- Concevoir l'algorithme avec une barrière de (re)synchronisation et implanter la barrière de (re)synchronisation !
- Permuter les buffers de calcul et de communication (quasi-obligatoire)

→ Voir chapitre de [programmation asynchrone](#)

Produit de matrices denses sur anneau
Algorithme distribué

→ TP

Produit de matrices denses en anneau

- Algorithme distribué en « anneau de processus »
- **Modélisation de performances sur un anneau théorique**
- Topologie de processus vs réseau réels
- Modélisation des comms. selon déploiement

Produit de matrices denses sur anneau

Modélisation de perfs sur anneau théorique

On suppose que l'on a un « anneau de processeurs »

Anneau de processus

Anneau théorique de machines & 1 processeur par machine

Déploiement *bijectif* :

- 1 process/machine
- processus k voisin des processus k-1 et k+1

Produit de matrices denses sur anneau

Modélisation de perfs sur anneau théorique

Performances **sans** recouvrement :

- Temps séquentiel : $T_{seq} = N \cdot (2\sqrt{N}-1) t_{flop}$
 $T_{seq} \approx 2 \cdot N \cdot \sqrt{N} t_{flop}$
- Temps parallèle :

Modèle de comm :

- $t_{com}(q) = t_s + q \cdot t_w$
- toutes les comms en parallèles

Speed up :

$$S^{no} = \frac{T_{seq}^{no}}{T_{par}^{no}} \approx P \cdot \frac{1}{1 + \frac{P \cdot t_w}{2 \cdot \sqrt{N} t_{flop}}}$$

$T_{1-calc} = \frac{\sqrt{N}}{P} \cdot \frac{\sqrt{N}}{P} \cdot (2\sqrt{N}-1) t_{flop}$
 $T_{1-calc} \approx 2 \cdot \frac{N \cdot \sqrt{N}}{P^2} t_{flop}$
 $T_{1-circ} = t_s + \sqrt{N} \cdot \frac{\sqrt{N}}{P} t_w$
 $T_{1-circ} \approx \frac{N}{P} t_w$

$T_{par}^{no} \approx P \cdot (t_{1-calc} + t_{1-circ})$
 $T_{par}^{no} \approx 2 \cdot \frac{N \cdot \sqrt{N}}{P} t_{flop} + N t_w$

Produit de matrices denses sur anneau

Modélisation de perfs sur anneau théorique

Comparaison calculs-communications (**sans** recouvrement) :

- Calculs : $T_{par}^{calc} \approx 2 \cdot \frac{N^{3/2}}{P} t_{flop}$
P fixé $\Rightarrow O(T_{par}^{calc}) = O(N^{3/2})$
- Circulation : $T_{par}^{circ} \approx N t_w$
 $O(T_{par}^{circ}) = O(N)$

$O(T_{par}^{calc}) > O(T_{par}^{circ})$

Avec P fixé, quand N ↑ : les calculs deviennent prépondérants
 → le surcoût des comms devient négligeable

Et le speedup devient parfait :

$$S^{no} \approx P \cdot \frac{1}{1 + \frac{P \cdot t_w}{2 \cdot \sqrt{N} t_{flop}}} \xrightarrow{N \rightarrow \infty} P$$

« Bon » problème pour un TP !

Produit de matrices denses sur anneau

Ex. de performances expérimentales

• 12 PC à 1 CPU à 1 cœur (P4- 3GHz)
 • 1 Gbit Eth - 3 switches
 (pas un vrai anneau)

Communications bloquantes :

- Peu de différences
- La plus simple est aussi la plus efficace! (MPI_Sendrecv_replace)

Texec(P)

Mflops(P)

Produit de matrices denses sur anneau

Ex. de performances expérimentales

• 12 PC à 1 CPU à 1 cœur (P4- 3GHz)
 • 1 Gbit Eth - 3 switches
 (pas un vrai anneau)

Speed Up (P)

On s'éloigne lentement de l'asymptote :

- Coût des communications,
- Loi de Amdahl,
- Calculs locaux de – en – rentables.

Efficacité (P)

Produit de matrices denses en anneau

- Algorithme distribué en « anneau de processus »
- Modélisation de performances sur un anneau théorique
- Topologie de processus vs réseau réels
- Modélisation des comms. selon déploiement

Produit de matrices denses sur anneau

Topologie de processus vs réseau réels

Principe du produit de matrices denses sur cluster :

1. Partitionnement initial des matrices A et B
2. Boucle de calcul et de circulation des données :
 - **Sans** recouvrement des calculs et des communications :
`Loop {calcul ; circulation}`
 - **Avec** recouvrement des calculs et des communications :
`Loop {par{calcul ; circulation}}`
3. Sauvegarde des résultats ou suite des calculs

Différents algorithmes de circulation de données existent :

Ex : circulation de données sur un anneau
 circulation de données sur un tore

→ Quelle topologie virtuelle est supportable par le cluster réel ?

Produit de matrices denses sur anneau

Topologie de processus vs réseau réels

Réalité matérielle :

Topologie virtuelle des processus :

- Anneau :**
- Tore 2D :**

Supporté par le réseau réel, avec des comms. en 1 phase (for the ring)

Peut saturer le switch, ou nécessiter des comms. en 2 phases (for the 2D torus)

Produit de matrices denses sur anneau

Topologie de processus vs réseau réels

Circulation en anneau :

Implantation `Bsend/Recv` ou `Sendrecv_replace` :

- **Comms. en une seule phase** : toutes les comms lancées en parallèle
- **Supportable par le switch ethernet** : toutes les comms faites en parallèle

Circulation en tore 2D :

Implantation `Bsend/Recv` ou `Sendrecv_replace` :

- **Comms. en 2 phases** : comms horizontales lancées en parallèle
 PUIS comms verticales lancées en parallèles
- **Supportable par le switch ethernet** : comms faites en 2 phases

Implantation en **comm. asynchrones** :

- **Comms. en une seule phase** : toutes les comms lancées en parallèle
- **Non supportable par le switch ethernet** : comms étalées dans le temps

→ Quelle sera l'algorithme de circulation le plus efficace sur la plateforme réelle ?

Produit de matrices denses en anneau

- Algorithme distribué en « anneau de processus »
- Modélisation de performances sur un anneau théorique
- Topologie de processus vs réseau réels
- **Modélisation des comms. selon déploiement**

Produit de matrices denses sur anneau

Modélisation des comms. selon déploiement

Algorithme distribué sur un anneau de processus

↓

Implantation MPI avec organisation des comms.

↓

Déploiement MPI sur un cluster de PC multi-cœurs

nodes × sockets × cores

Produit de matrices denses sur anneau

Modélisation des comms. selon déploiement

2 types de comms. :

- Comm sur le réseau : LENT** (shown with a red box)
- Comm. en mémoire : TRES RAPIDE** (shown with a blue box)

Modélisation :

Comm sur le réseau :

- modélisée en : $t_{comm}(q) = (t_s) + q.t_w$
- exécutées en parallèle : - si le réseau réel n'est pas saturé
 - et si elles ont été lancées en parallèle

Comm au sein d'un nœud :

- considérées instantanées : $t_{comm}(q) = 0$

→ comparer les déploiements selon leurs comms. sur le réseau

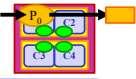
Produit de matrices denses sur anneau

Modélisation des comms. selon déploiement

Volume de communications sur le réseau d'interconnexion :
Cluster : $N_{nodes} \times N_{socket/node} \times N_{coeurs/socket} = N_n \times N_s \times N_c$
Déploiement : 1 processus/nœud et $N_s \times N_c$ threads par processus

- N_n processus et étapes
- Volume d'une tranche de matrice A : $Q_{slice} = Q_s = Q_A / N_n$

A chaque étape : - chaque processus émet Q_s data
- chaque nœud émet Q_s data vers un autre nœud



Au final : un autre nœud

Volume total de données traversant le réseau durant tout le calcul :
 $N_n \times (N_n \times Q_s) = N_n \times Q_A$

Taille de chaque message :
 Q_A / N_n

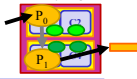
Produit de matrices denses sur anneau

Modélisation des comms. selon déploiement

Volume de communications sur le réseau d'interconnexion :
Cluster : $N_{nodes} \times N_{socket/node} \times N_{coeurs/socket} = N_n \times N_s \times N_c$
Déploiement : 1 processus/socket et N_c threads par processus et numérotation selon les sockets

- $N_n \times N_s$ processus et étapes
- Volume d'une tranche de matrice A : $Q_{slice} = Q_s = Q_A / (N_n \times N_s)$

A chaque étape : - chaque processus émet Q_s data
- chaque nœud émet Q_s data vers un autre nœud



Au final : un autre nœud

Volume total de données traversant le réseau durant tout le calcul :
 $(N_n \times N_s) \times (N_n \times Q_s) = N_n \times Q_A$

Taille de chaque message :
 $Q_A / (N_n \times N_s)$

Plus de plus petits msg. mais même volume total

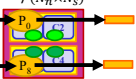
Produit de matrices denses sur anneau

Modélisation des comms. selon déploiement

Volume de communications sur le réseau d'interconnexion :
Cluster : $N_{nodes} \times N_{socket/node} \times N_{coeurs/socket} = N_n \times N_s \times N_c$
Déploiement : 1 processus/socket et N_c threads par processus et numérotation selon les nœuds

- $N_n \times N_s$ processus et étapes
- Volume d'une tranche de matrice A : $Q_{slice} = Q_s = Q_A / (N_n \times N_s)$

A chaque étape : - chaque processus émet Q_s data
- chaque nœud émet $N_s \times Q_s$ data vers d'autres nœuds



Au final : un autre nœud

Volume total de données traversant le réseau durant tout le calcul :
 $(N_n \times N_s) \times (N_n \times (N_s \times Q_s)) = N_n \times N_s \times Q_A$

Taille de chaque message :
 $Q_A / (N_n \times N_s)$

Bcp plus de plus petits msg. et volume total plus important !
→ Plus lent !

L'essentiel de MPI-1

Questions ?