Produit de matrices Strassen, Winograd et parallélisme mixte en récursif ?!

Frédéric SUTER
Projet ReMaP
Laboratoire de l'Informatique du Parallélisme
UMR CNRS-INRIA-ENS Lyon 5668
Lyon, France

Plan de l'exposé

- ✓ Introduction
- ✓ Versions mixtes **non-récursive** des algorithmes de Strassen et Winograd
- ✓ Comparaisons théorique et expérimentale
- ✓ Un point sur la récursivité
- ✓ Conclusion et perspectives

Introduction : Contexte

- ✓ Ordonnancement des travaux dans Scilab//
  $$C = A \times B + D \times E$$
- ✓ Serveurs de calculs disponibles à travers un réseau
- ✓ Trouver de « bons » algorithmes candidats
  - Si possible parallèles
- ✓ Utilisation de bibliothèques numériques parallèles
- ✓ Développer de nouveaux paradigmes de programmation
- ✓ Automatiser !

http:www.ens-lyon.fr/~desprez/OURAGAN/

Introduction : parallélisme mixte (1)

- ✓ Parallélisme mixte :
  - Exploitation simultanée du parallélisme de données et du parallélisme de tâches
  - Graphes de tâches data-parallèles
  - Découpage de l'ensemble des processeurs en sous-ensembles
- ✓ Switched parallelism
  - Ordonnancement de Divide and Conquer Trees
  - Détermination d'un seuil
    - avant → Parallélisme de données (sur tous les processeurs)
    - après → Parallélisme de tâches (une tâche par processeur)

Introduction : parallélisme mixte (2)

- ✓ Au niveau langages et compilateurs
  - Du Task dans du Data (Fx, Paradigm, TwoL, ext. HPF, etc.)

    ajout de structures de contrôle et de communications explicites dans un langage data-parallèle
  - Du Data dans du Task (Data//-Orca, etc.)

    ajout de structures data-parallèles dans un langage à parallélisme de tâches
- ✓ Thèse de Ramaswamy
  - Extraire un graphe de tâches data-parallèles d'un code C, Fortran ou Matlab

    trouver découpages, ordonnancement et placement qui minimisent le temps d'exécution

Introduction : Strassen et Winograd

- ✓ Algorithmes de produit de matrices
- ✓ Réduction du nombre de multiplications dans le produit de matrices 2 x 2
  - Strassen : 7 multiplications et 18 additions/soustractions
  - Winograd : 7 multiplications et 15 additions/soustractions
  - Algorithme classique : 8 multiplications et 4 additions
  - Complexité en $O(2^{log(7)}) = O(2^{2.807})$ par récursivité
- ✓ Applicables à des matrices blocs 2 x 2

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} * \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Algorithme et graphe de tâches

Phase 1

T1 = A11+A22 T5 = B12-B21

T2 = T1-A11 T6 = B22-T5

T3 = A11-A21 T7 = B22-B12

T4 = A12-T2 T8 = B21-T6

Phase 2

Q1 = A11*B11 Q5 = T3*T7

Q2 = A12*B21 Q6 = T4*B22

Q3 = T1*T5 Q7 = A22*T8

Q4 = T2*T6

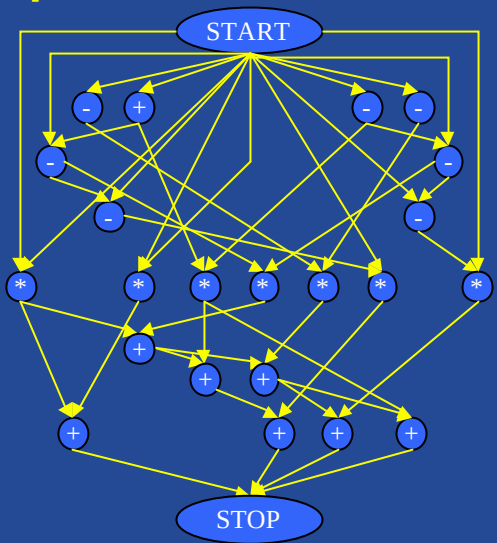
Phase 3

C11 = Q1+Q2

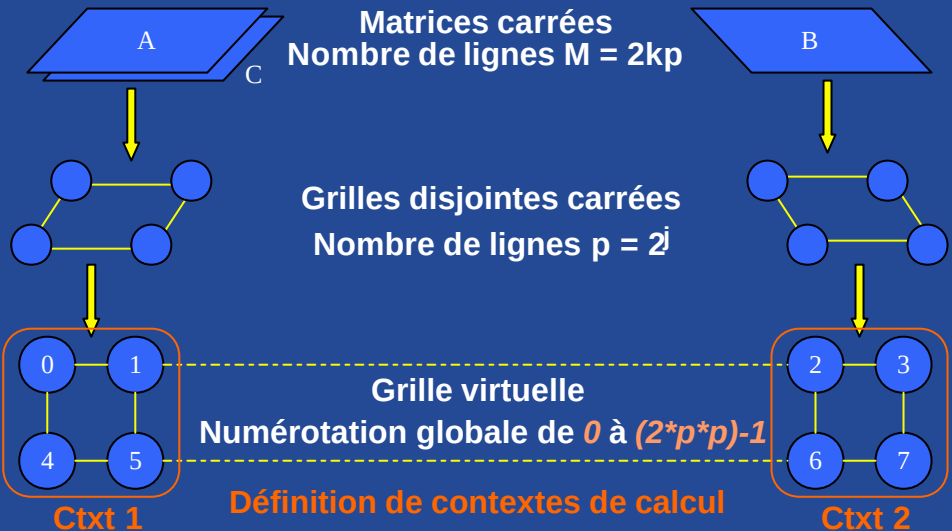
C12= Q1+Q3+Q4+Q6

C21 = Q1+Q4+Q5+Q7

C22=Q1+Q3+Q4+Q5



Introduction : cadre de travail



Qu'espère-t-on obtenir ?

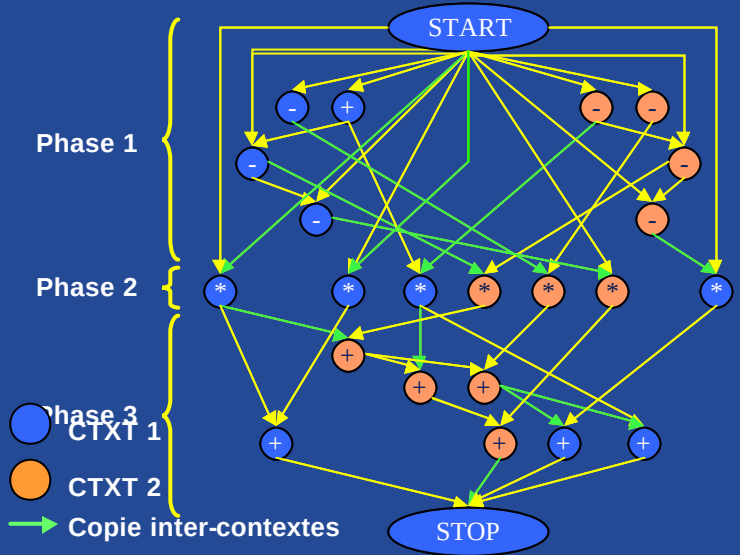
- ✓ On ne cherche pas à ridiculiser ScaLAPACK
- ✓ En revanche on espère s'en approcher
  - ☺ Validation du paradigme utilisé
- ✓ Si le surcoût est faible
  - ☺ Possibilité d'étendre le cas homogène au cas hétérogène
  - ☺ Passage d'algorithmes data-// en hétérogène difficile
- ✓ Autres extensions possibles
  - Placement automatique de tâches data-//
  - Utilisation par un ordonnanceur dans le cadre d'un environnement à base de serveurs de calculs

Distributions des données

- ✓ Intérêt de Strassen : additions en temps linéaire et sans communications
- ✓ Distribution telle que chaque processeur doit posséder un morceau de chaque matrice
  - ➲ Distribution *cyclique par blocs* bidimensionnelle
  - ➲ Taille des blocs : M/2p

Calcul de
A11 + A22

Placement des tâches : Winograd



Algorithmes comparés

- ✓ Strassen et Winograd mixte
- ✓ PDGEMM sur la moitié des processeurs, avec la redistribution de ScaLAPACK
- ✓ PDGEMM sur l'ensemble des processeurs, avec la redistribution de ScaLAPACK
- ✓ PDGEMM sur l'ensemble des processeurs, avec une redistribution optimisée

Utilisation

Algorithme de la chenille

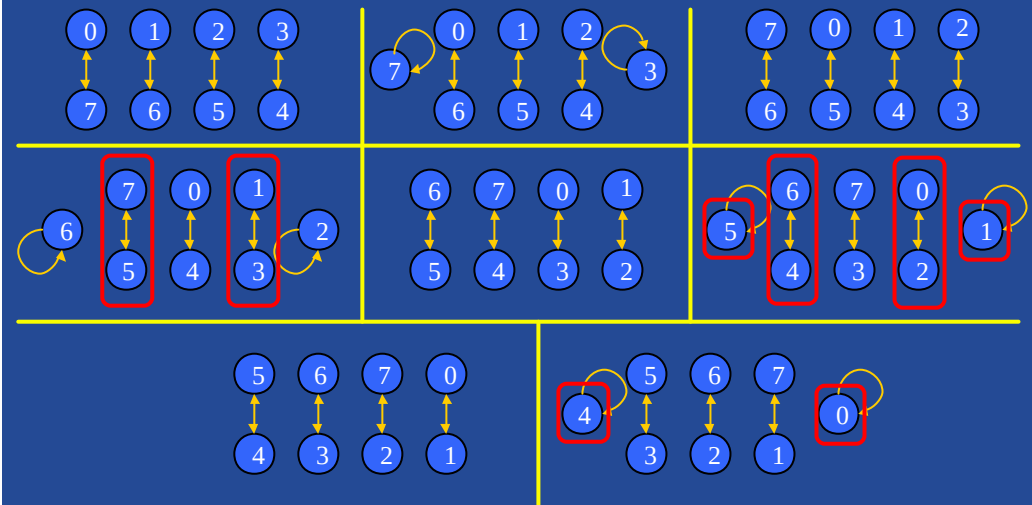
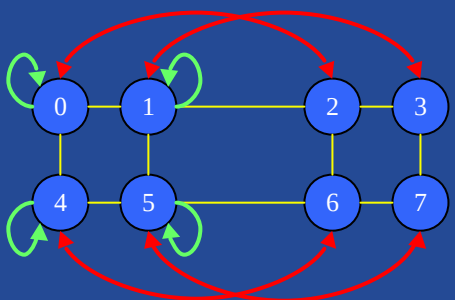


Schéma de communication effectif



Copie mémoire

Communication

Comparatif théorique (résumé)

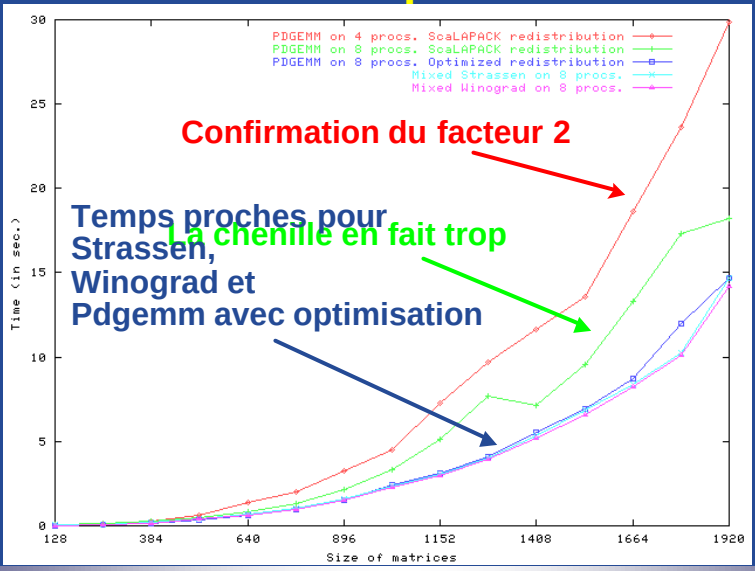
- ✓ Version «PDGEMM sur la moitié des processeurs»
  - 2x plus de produits/processeurs
- ✓ Autres algorithmes
  - Différences au niveau des latences
- ✓ Strassen vs Winograd
  - Plus de calcul mais moins de comm. pour Winograd
- ✓ Winograd vs «PDGEMM + redistribution optimisée»
  - Moins de comm. pour Winograd si $p \geq 2$

Plate-forme de test

- ✓ Grappe homogène

- ✓ Processeurs « lents »
  - 8 Pentium Pro 200 MHz
- ✓ Mais réseau rapide
  - Myrinet

Résultats expérimentaux



Conclusion sur le non-récursif

- ✓ Implémentations mixtes de Strassen et Winograd
  - ☺ Meilleures que des programmes « pur ScaLAPACK » (moitié des processeurs ou redistribution ScaLAPACK)
  - 😐 Au niveau de ScaLAPACK (si on optimise la redistribution)
- ✓ Validation de l'approche utilisée
  - Strassen et Winograd sont des algorithmes bien adaptés (tâches indépendantes et identifiées)
  - Essayer sur d'autres plates-formes (grappe HP Grenoble, SP3 CINES)
  - Extension vers la récursivité à étudier

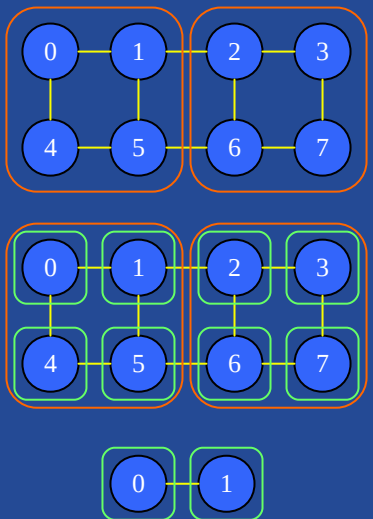
Récursivité ?

- ✓ Bonne complexité obtenue par récursivité
- ✓ Nos algorithmes peuvent-ils être appliqués récursivement ?

- ✓ Deux approches possibles
  - Même ratio (nombre de processeurs / taille des matrices)
  - Mêmes contextes de calcul et diminution des tailles

Première approche

- ✓ But : exécuter les produits en local
- ✓ Réduction et multiplication des contextes de calcul
- ✓ Problèmes
  - Chemin critique : 4 produits
  - Pas de diminution effective du nombre de produits
  - Plus de communications
- ✓ Approche abandonnée !



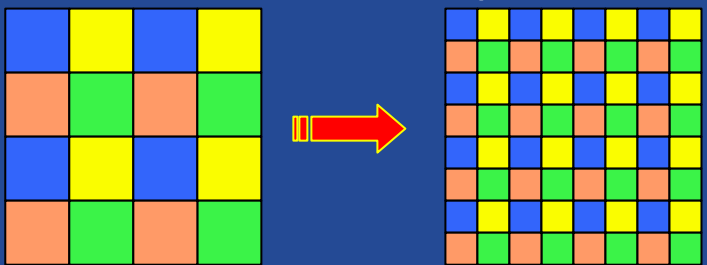
Deuxième approche

- ✓ Remarque sur les produits de la phase 2
  - Opérande 1 sur le contexte 1
  - Opérande 2 sur le contexte 2
- ✓ Mêmes conditions que pour le produit initial
- ✓ Application du même algorithme
  - Sur les mêmes contextes de calcul
  - Mais sur matrices 4x plus petites
- ✓ Dernier appel récursif
  - Utilisation de ScaLAPACK

Deuxième approche (suite)

- ✓ Taille des blocs fonction du nombre de pas

Sans récursivité **1 pas de récursivité**



- ✓ Plus de phases de comm. supplémentaires
- ✓ Réduction du nombre de produits effectués
- ✓ Étude et tests en cours

Perspectives

- ✓ A court terme : récursif
  - Vérifier l'approche récursive sur différentes plates-formes
- ✓ A moyen terme : hétérogène
  - Processeurs de différentes puissances
    - Déséquilibrer les calculs en fonction des puissances
  - Réseaux de différents débits
    - Optimiser les redistributions entre les contextes
    - But : maximiser le débit du lien inter-contextes
  - Combiner ces deux types
    - Grappe de grappes
- ✓ A long terme
  - Confirmer sur d'autres algorithmes d'algèbre linéaire


Questions ?