Projet 1: Multiplication de matrices creuses

Release 1.1

Fabien Duchêne

September 19, 2013

CONTENTS

CHAPTER

ONE

INTRODUCTION

Depuis le début de l'informatique, les ordinateurs ont très souvent été utilisés pour effectuer des opérations sur les matrices dont la complexité rend leur calcul impossible à la main. Parmi ces opérations, le produit matriciel est l'un des plus important car de son efficacité découle la résolution de nombreux problèmes d'algèbre linéaire, tels que l'inversion de matrices, la résolution de système d'équations ou le calcul de déterminants.

Les matrices creuses, qui font l'objet de cet énoncé, sont des matrices qui contiennent majoritairement des éléments nuls. Dans cet exercice, nous considérerons que plus de 90% sont nuls. Dans ce cas, il est plus efficace, que ce soit pour l'utilisation de la mémoire ou pour le calcul, que seuls les éléments non nuls soient stockés en mémoire.

CHAPTER

TWO

MISSION

Cette section décrit plus en détails ce que vous devez faire pour ce projet et comment.

2.1 Objectif

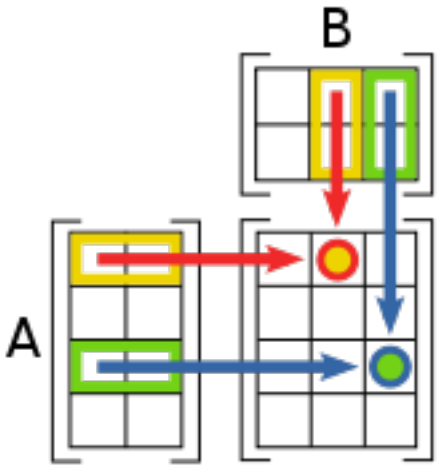
L'objectif de ce projet est que vous démontriez que vous êtes capable de concevoir et d'implémenter en C un programme UNIX non-trivial utilisant des pointeurs et des structures de données complexes. Vous devez en outre pouvoir justifier vos choix, et être capable d'expliquer les résultats obtenus et les limites de votre implémentation.

2.2 Rappels importants sur le produit matriciel

Soit deux matrices, A et B, respectivement de taille $n \times m$ (c'est-à-dire, n lignes et m colonnes) et $m \times p$. La matrice résultante du produit de A et de B, $C = AB$, est une matrice de taille $n \times p$ dont chacune des cellules, indicée en (i, j), est calculée comme suit :

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} bk, j = a_{i,1} b_{1,j} + a_{i,2} b_{2,j} + \ldots + a_{i,n} b_{n,j}$$

La figure suivante montre comment calculer les coefficients $c_{1,2}$ et $c_{3,3}$ de la matrice produit AB si A est une matrice 4×2, et B est une matrice 2×3. ($c_{1,2} = a_{1,1} b_{1,2} + a_{1,2} b_{2,2}$ et $c_{3,3} = a_{3,1} b_{1,3} + a_{3,2} b_{2,3}$)



Dans le problème qui nous intéresse, vous devez effectuer le produit sur un grand nombre de matrices, c'est-à-dire calculer X:

$$X = M_1\, M_2\, M_3\, M_4\, M_5\, M_6\, M_7\, M_8\, \ldots\, M_N$$

Notez que pour toute paire de matrices qui se suivent, le nombre de colonnes de la première matrice est toujours égale au nombre de ligne de la suivante. La taille de X sera donc égale au nombre de ligne de M_1 et le nombre de lignes de M_N.

Par ailleurs, le produit matriciel n'est pas commutatif mais est associatif. Vous ne pouvez donc multiplier que deux matrices voisines ensemble, mais vous pouvez aussi bien commencer par calculer $M_4\, M_5$ que par $M_5\, M_6$.

2.3 Programme attendu

Votre programme devra, en entrée, lire un fichier contenant une liste ordonnée de matrices dans le format de fichier de matrices défini et devra écrire la matrice résultante de la multiplication de toutes les matrices sous ce même format. Le nombre de colonnes ou de lignes des matrices devrait être grand (> 3000) et les valeurs de chaque cellule de ces matrices ne seront exclusivement que des entiers de valeur 0 ou 1.

Votre programme doit utiliser des matrices creuses uniquement.

Votre programme doit être **portable** sous UNIX, il doit au minimum fonctionner de façon transparente aussi bien sur les Linux CentOS des salles didactiques (en l'occurence la salle Intel nouvellement équipée) que sur vos machines personelles.

2.4 Implémentation des matrices creuses

Comme décrit précédemment, les matrices creuses sont des matrices dont seuls les éléments non-nuls sont stockés en mémoire, et ce, par opposition aux matrices dites "pleines" où l'ensemble des éléments est stocké en mémoire.

Pour implémenter vos matrices creuses vous devez utiliser une (des) liste(s) chaînée(s) ou la/les structures de votre choix

2.5 Mesures de performance et interprétations

Une fois que votre programme est implémenté et fonctionnel, nous vous demandons d'évaluer les performances de celui-ci depuis les points de vues suivants :

1. performances de votre programme en comparaison avec une implémentation triviale des matrices pleines (tableau à deux dimensions);
2. types de matrices pour lesquels l'utilisation d'une matrice creuse apporte un gain important;

Pour chacun de ces scénarios, nous vous demandons :

1. d'effectuer les mesures nécessaires,
2. d'expliquer clairement les mesures effectuées et leurs contextes (spécification de la machine, paramètres, ...)
3. de tracer le graphe (1 seul!) le plus intéressant (pour l'interprétation),
4. d'interpréter les résultats.

2.6 Calendrier

- 19 septembre : TP - remise de l'énnoncé, discussions éventuelles.
- 26-27 septembre : TP - discussion avec les assistants sur l'architecture de votre programme.
- 3-4 octobre : TP - séance intermédiaire avec les assistants.
- 6 octobre : rendu du projet.
- 10-11 octobre : Feedback des assistants sur les projets rendus.

2.7 Bonus

La méthode classique pour réaliser un produit matriciel est en $O(n^3)$ (n étant le nombre de lignes ou colonnes de la matrice) et peu d'optimisations sont possibles.

En bonus, nous vous proposons de modifier votre programme afin de lui permettre de tirer parti de la parallélisation et donc d'accélérer le calcul du produit d'un grand nombre de matrices.

2.8 Demande de création d'un dépôt SVN

Pour ce projet, vous devez avoir accès un dépôt SVN. Veuillez vous inscrire par groupes à l'adresse https://scm.info.ucl.ac.be/cgi-bin/inscription.sh. Cochez la case "*Intégration dans un repository monitoré de travaux de groupes.*" et inscrivez-vous au cours INGI1113_2013 en indiquant les adresses e-mail de chaque membre du groupe.

N'hésitez pas à consulter le wiki étudiant pour plus d'info sur SVN.

2.9 Syntaxe

Votre programme s'utilisera de la façon suivante (pensez à utiliser *getopt()* pour gérer les paramètres dans votre programme):

```
matrixprod [-o OUTPUT_FILE] INPUT_FILE
```

-o OUTPUT_FILE	spécifie le fichier de sortie dans lequel le résultat du produit doit être enregistré, si cet argument n'est pas définit, affiche à la sortie standard
INPUT_FILE	le fichier d'entrée qui défini une liste ordonnée de matrices

Les fichiers de matrice en entrée **et** en sortie ont le format suivant: une ligne spécifiant la taille de la matrice, par exemple `432x574`, suivi de la matrice représentée sous forme de suite de nombre séparés par des espaces et des sauts de ligne pour délimiter les lignes de matrice. Plusieurs matrices peuvent bien entendu être définie en chainant les différentes définition de matrices.

Un exemple d'un tel fichier:

```
2x3
0 1 0
1 0 1
3x1
0
```

```
0
1
```

Vous pouvez considérer que le fichier d'entrée donné à votre programme est dans un format valide. Vous ne pouvez par contre pas considérer que les matrices sont bien multipliables entre elles (vous devez afficher un message adéquat si nécessaire).