Faculté sciences & sciences de l'ingénieur
Université Bretagne Sud

Master d'Informatique

UE INF2245

Calcul haute performance pour le «Big Data»

Sujet d'examen terminal

(première session)

Durée : 2 heures
Copie à remettre sur l'ENT

F. Raimbault, N. Courty

mai 2020

Avertissement : Le temps conseillé à consacrer à chaque question est précisé pour chaque exercice. Il sera tenu compte dans la notation de la qualité de rédaction, de votre capacité à abstraire une solution et de la rigueur avec laquelle vous la décrivez.

Exercice 1 : Complexité : transport optimal (30 mn)

Le problème dit du transport optimal est un vieux problème mathématique qui est dû à Gaspard Monge (18ème siècle). Le but de ce problème est de transporter de manière optimale (i.e. relativement à un coût donné) un tas de matière d'un endroit à un autre.

On propose la formulation suivante du problème : soit x_i (i de 1 à N) les endroits d'où l'on part. Chacun de ces endroits comportent une masse de matière p_i (i de 1 à N). De la même manière, on cherche à déplacer ces tas vers M endroits y_j (j de 1 à M), et la masse associée à ces tas est notée q_j (j de 1 à M). On suppose dans toute la suite que la quantité de matière à déplacer est bien constante, i.e. $\sum_i p_i = \sum_j q_j$. Déplacer de la matière du tas i au tas j coûte C_{ij}.

Un exemple de coût est de prendre la distance euclidienne, i.e. $C_{ij} = ||x_i - y_j||_2$. Le problème revient à déterminer quelle quantité de matière on envoie de chaque tas i à chaque tas j. Cette quantité est notée T_{ij}. On précise alors que $\sum_i T_{ij} = q_j$ et que $\sum_j T_{ij} = p_i$. Avec ces notations, T et C sont des matrices réelles de taille $N \times M$.

1. Que vaut $\sum_{ij} T_{ij}$?

2. Donnez un algorithme prenant en paramètre les vecteurs p et q, et une matrice T en entrée, et vérifiant que cette matrice est bien solution du problème de transport. Donnez sa complexité en fonction de M et N.

Le problème de transport optimal est de trouver une telle matrice de transport T **vérifiant les contraintes** et minimisant le coût total de déplacement

$$\min_T \sum_{ij} T_{ij} C_{ij}$$

3. Quelle est la nature de problème ? Est ce un problème combinatoire ?

On se place maintenant dans le cas où $M = N$ et $p_i = q_i = 1$ pour tout i de 1 à $N = M$. On admet que le problème se transforme alors en un problème d'affectation, ou on apparie exactement un tas de départ à un tas d'arrivée. La matrice T est alors pleine de zéros, avec exactement N 1 situés en (i, j) quand les tas i et j sont appariés.

4. Du point de vue de la complexité de la vérification, est il plus simple par rapport au cas général précédent de vérifier qu'une matrice T donnée vérifie les contraintes ? Justifiez votre réponse.
5. Soit une matrice T vérifiant les contraintes. Quelle est la complexité du calcul du coût total ? Justifiez votre réponse.
6. Si l'on devait résoudre ce problème d'optimisation par force brute, quelle serait la complexité de l'algorithme ?

Heureusement il existe des méthodes plus rapides pour résoudre ce problème d'optimisation, mais ce sera pour un autre contrôle terminal...

Exercice 2 : CUDA (30 mn)

On s'intéresse dans cet exercice au codage de la transformation DCT d'une image en niveau de gris en CUDA. On rappelle qu'une bonne manière d'implémenter la DCT par blocs de taille 8×8 serait de créer une matrice de la transformée de cosinus, C.

$$C_{i,j} = \begin{bmatrix} \frac{1}{\sqrt{N}} & si\ i = 0 \\ \sqrt{\frac{2}{N}} cos(\frac{(2j+1)i\pi}{2N}) & si\ i > 0 \end{bmatrix}$$

où $N = 8$. On note alors C^T la transposée de C. La transformée par bloc 8x8 se réduit à deux multiplications matricielles $DCT(P) = C \times P \times C^T$ où P est le bloc de pixels exprimé sous la forme d'une matrice.

On suppose que l'image est accessible dans la mémoire du GPU au travers d'un pointeur `image` de type `char8 **`. On s'intéresse à l'implémentation d'un noyau CUDA prenant cette image en entrée et donnant une image de float de la DCT en sortie. La mémoire est allouée dans le GPU et correspond à un pointeur sur un `float ** imDCT`. La taille de l'image d'origine est (N_X, N_Y) où les 2 valeurs sont des multiples de 8.

1. Quelle stratégie de découpage en blocs (taille de grille) vous semble adaptée à ce problème ? Donnez les dimensions de la grille.
2. Quelle gestion de la mémoire devrait on adopter pour ce problème (mémoire locale/globale) ?

3. Donnez la formule permettant de passer d'un pixel d'un bloc au pixel correspondant dans l'image d'origine (en utilisant la syntaxe CUDA).
4. On dispose d'une fonction `calcDCT(int i, int j, float **C, float **P)` qui calcule la valeur de la DCT de l'élément (i, j) d'un bloc. Ecrivez un noyau CUDA implémentant la transformée de l'image.

Le cas échéant, vous préciserez utilement pour cet exercice les différentes hypothèses que vous pourriez être amenés à faire pour la résolution du problème.

Exercice 3 : Jeu d'instructions SIMD (30 mn)

Complétez le programme C des figures 1 et 2, en écrivant le contenu de la fonction *simd_matvec* pour accélérer le calcul du produit d'un vecteur par une matrice. Vous pouvez utiliser la fonction *horizontal_add* pour terminer le calcul d'une composante du vecteur *vec_c*[].

Exercice 4 : Programmation MapReduce (30 mn)

On utilise comme jeu de données des algorithmes demandés ci dessous la base *StackOverflow* étudiée en CM et en TP.

Les solutions à cet exercice doivent être exprimées selon le modèle donné ci-dessous pour l'algorithme du minmaxcount. *Aucun programme Java ne sera pris en compte.*

Algorithme du minmaxcount (modèle)

Le problème consiste à déterminer, à partir du fichier `Comments.xml`, la date du premier commentaire, la date du dernier commentaire et le nombre de commentaires écrit par chaque utilisateur.

Rappel : Le fichier `Comments.xml` est organisé en lignes de commentaires XML, vues par la fonction de lecture des tâches map comme des couples (x=adresse de la ligne,xml=contenu textuel de la ligne). Chaque commentaire possède un champ `UserId` qui identifie de manière unique le rédacteur d'un message, un champ `CreationDate` qui contient la date d'écriture du commentaire et un champ `Text` qui contient le texte du commentaire.

Algorithme *MapReduce* du *minmaxcount* :

— Fonction $map(x, xml) \rightarrow list(userid, (date, date, 1))$
Génère un triplet pour chaque commentaire créé à la date *date* par un utilisateur *userid* en extrayant les valeurs des champs `UserId` et `CreationDate` à l'aide de la fonction utilitaire `getAttributeValue(String key, String xml)` qui renvoie la valeur associée à l'attribut `key` recherchée dans le texte `xml`).

— Fonction *reduce* : $(userid, list((date1, date2, n)) \rightarrow (userid, (mindate, maxdate, sum))$
Parcours la liste des couples d'entrées en retenant le minimum des valeurs *date*1, le maximum des valeurs *date*2 et en sommant le nombre de commentaires n de l'utilisateur *userid*.

— Optimisation possible : utiliser la fonction *reduce* comme fonction *combine*.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <limits.h>

#include <x86intrin.h> // gcc specific
#include <cpuid.h>

#define N (16*100)

/************************************************************************
                        array utilities
************************************************************************/

// randomly initialize a vector of short
void rnd_float_vector(float A[N]){
    for(int i=0; i<N; i++){
        A[i]= drand48();
    }
}

// randomly initialize a matrix of short
void rnd_float_matrix(float M[N][N]){
    for(int i=0; i<N; i++){
        for(int j=0; j<N; j++){
           M[i][j] = drand48();
        }
    }
}

// compare two vectors of float
int compare_float_vector(float A1[N],float A2[N]){
    for(int i=0; i<N; i++){
        if (fabsf(A1[i]-A2[i])>1e-3f){
            printf("*** error: A1[%i]=%e and A2[%i]=%e differ by %e\n",
                   i,A1[i],i,A2[i],fabsf(A1[i]-A2[i]));
            return 1;
        }
    }
    return 0;
```

FIGURE 1 – Programme de multiplication matrice-vecteur

```
/*************************************************************************
                        scalar function
*************************************************************************/

// C[N]= A[N][N] x B[N]
void scalar_matvec(float vec_c[N], float mat_a[N][N], float vec_b[N]) {
    for (int i = 0; i < N; i++) {
      vec_c[i]= 0;
      for (int j = 0; j < N; j++) {
        vec_c[i] += mat_a[i][j]*vec_b[j];
      }
    }
}
/*************************************************************************
                        intrinsics function
*************************************************************************/

// sum the 8 floats contained in a __m256 word
static inline float horizontal_add(__m256 a) {
    float acc= 0;
    float *p= (float *) &a;
    for (int i=0; i<8; i++){
        acc += p[i];
    }
    return acc;
}

// C[N]= A[N][N] x B[N]
void simd_matvec(float vec_c[N], float mat_a[N][N], float vec_b[N]) {
    // to complete
}
/*************************************************************************
                        main function
*************************************************************************/

float vector_C1[N] __attribute__((aligned(32)));
float vector_C2[N] __attribute__((aligned(32)));
float matrix_A[N][N] __attribute__((aligned(32)));
float vector_B[N] __attribute__((aligned(32)));

int main(int argc,char *argv[]){
    srand48(time (0));
    rnd_float_vector(vector_B);
    rnd_float_matrix(matrix_A);
    scalar_matvec(vector_C1,matrix_A,vector_B);
    simd_matvec(vector_C2,matrix_A,vector_B);
    return compare_float_vector(vector_C2,vector_C1);
}
```

FIGURE 2 – Programme de multiplication matrice-vecteur (suite)

Questions

1. Calcul d'une moyenne

   Le problème consiste à calculer la taille moyenne des commentaires par heure de la journée. On ne cherchera pas dans cette question à utiliser de fonction *combine*.

   Exprimez la fonction *map* et la fonction *reduce* suivant le modèle ci dessus.

2. Calcul optimisé d'une moyenne

   Modifiez votre solution à la question précédente afin de pouvoir utiliser votre fonction *reduce* comme fonction *combine*.