IN203 – Examen machine

Juvigny Xavier

Janvier 2020

Pour toutes les questions demandant une réponse écrite autre que le programme, écrire un fichier dans le format de votre choix (latex, word, texte, markdown, etc.) dans lequel vous mettrez vos réponses.

Introduction

Vous aurez à modifier les codes sources fournis pour les paralléliser. Certaines questions demandent des argumentations et discussions des résultats. Vous noterez vos réponses dans un fichier `reponses.md` (ou `.doc`, `.txt`, `.tex`, …).

En tout premier lieu, notez dans ce fichier le résultat de la commande `lscpu`.

Les exercices 1 et 3 contiennent une boucle sur `nb_repet` qui ne sert qu'à avoir des mesures de chrono plus précises. Ce ne sont pas elles qu'il faut paralléliser.

À la fin de l'examen, faites une archive contenant ces codes sources et ce fichier réponses :

```
tar -zcvf ExamenMachine.tgz ExamenMachine
```

puis envoyez-la par mail à `xavier.juvigny@onera.fr`, `jean-didier.garaud@onera.fr`, `augustin.parret-freaud@safrangroup.com`.

1 Compter les lettres et les nombres

Le programme "StatistiqueTexte.cpp" consiste à parcourir un fichier texte pour compter le nombre de lettres (minuscules et majuscules), le nombre de chiffres et le nombre de caractères autres.

1. Paralléliser à l'aide de directives OpenMP le programme
2. Tester sur 1,2,4, 8 et 16 threads la performance du programme. Qu'en conclure ?

2 Orthonormalisation d'une base libre

Le programme "Orthonormalisation.cpp" génère une famille libre de vecteur puis l'orthonormalise à l'aide d'un algorithme de Gram-Schmidt modifié :

où (u, v) représente un produit scalaire.

On veut paralléliser le code à l'aide de MPI.

1. Expliquer (en commentaire dans votre source si vous voulez) pourquoi il est difficile de paralléliser le code en distribuant la famille de vecteurs.
2. On distribue du coup chaque vecteur de la famille sur les processeurs afin de pouvoir paralléliser l'orthonormalisation ainsi que la génération des vecteurs de la famille de vecteurs. Mettez en œuvre cette stratégie de parallélisation. On supposera que la dimension du vecteur est divisible par le nombre de processus.

Algorithm 1 Calcul la base orthonormée générant la famille libre de n vecteurs $\mathcal{F} = \{u_1, u_2, \ldots, u_n\}$ de dimension d donnée en entrée

Require: $d \geq n$
Ensure: $\mathcal{F}$ contient en sortie la base orthonormée
  for $i \leftarrow 1$ to n **do**
    for $j \leftarrow 1$ to i-1 **do**
      $u_i \leftarrow u_i - (u_i, u_j).u_j$
    end for
    $u_i \leftarrow \frac{u_i}{\|u_i\|}$
  end for

3. Mesurez les temps pris pour la génération et l'orthonormalisation en fonction du nombre de processus pris.

3 Calcul d'enveloppe convexe

On veut calculer l'enveloppe convexe d'un nuage de point sur un plan. Pour cela, on utilise l'algorithme de Graham décrit sur le lien suivant :

`https://fr.wikipedia.org/wiki/Parcours_de_Graham`

On obtient en sortie une sous-liste de points du nuage qui définissent l'enveloppe convexe. Ces points sont rangés de manière à parcourir le polygone de l'enveloppe dans le sens direct.

Le code séquentiel peut être trouvé dans le fichier "enveloppe_convexe.cpp". En sortie, le code écrit un fichier de type svg (graphisme vectoriel) directement lisible de votre browser internet (Edge, Firefox, Safari, etc.) qui vous permettra de valider votre code.

Afin de paralléliser le code en distribué avec MPI, on veut distribuer les sommets sur plusieurs processus puis utiliser l'algorithme suivant :

- Calculer l'enveloppe convexe des sommets locaux de chaque processus
- Puis en échangeant deux à deux entre les processus les enveloppes convexes locales, calcul sur chacun la fusion des deux enveloppes convexes en remarquant que l'enveloppe convexe de deux enveloppe convexe est l'enveloppe convexe de la réunion des sommets définissant les deux enveloppes convexes.

1. Dans un premier temps, mettre en œuvre l'algorithme sur deux processus;
2. Dans un deuxième temps, en utilisant un algorithme de type hypercube, de sorte qu'un processus fusionne son enveloppe convexe avec le processus se trouvant dans la direction d, mettre en œuvre l'algorithme sur 2^n processus.
   Exemple sur 8 processus
3. Calculer le speed-up de votre algorithme à volume constant de points par processus et au total.

Algorithm 2 Calcul d'une enveloppe convexe 2D sur un nuage réparti de points sur huit processus

Require: Huit processus

Ensure: En sortie, tous les processus contiennent l'enveloppe convexe

Les processus 0 à 7 calculent l'enveloppe convexe de leur nuage de points local.

Le processus 0 et le processus 1 fusionnent leurs enveloppes, idem pour 2 avec 3, 4 avec 5 et 6 avec 7.

Le processus 0 et le processus 2 fusionnent leurs enveloppes, idem pour 1 avec 3, 4 avec 6 et 5 avec 7.

Le processus 0 et le processus 4 fusionnent leurs enveloppes, idem pour 1 avec 5, 2 avec 6 et 3 avec 7.