POLYTECH SORBONNE

Calcul haute performance : notions de base (4I903)

Examen du 25 mai 2018

Seul le polycopié de cours est autorisé - Durée : 2h00

Le barème est donné à titre indicatif

Le sujet se décompose en 2 exercices indépendants. La qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'évaluation de la copie.

Exercice 1 (Parallélisation d'un algorithme séquentiel (8 points))**.** Considérons l'algorithme séquentiel suivant, où toutes les tâches ont le même temps d'exécution (égal à une unité de temps) :

T_1 : X1=A1+B1
T_2 : X2=A2+B2
T_3 : T=max(abs(1+B1),abs(B2))
T_4 : S=min(abs(1+B1),abs(B2))
T_5 : R=T*T*(1+S*S/T/T)
T_6 : T=X1*(1+B1)+X2*B2 / R
T_7 : S=X2*(1+B1)-X1*B2 / R

1. Établir le système de précédence et le graphe de précédence en détaillant les résultats intermédiaires.
2. Donner les décompositions du graphe de précédence sur une machine parallèle ayant suffisamment de processeurs
   a) dans le cas d'une décomposition par prédécesseurs ;
   b) dans le cas d'une décomposition par successeurs.
   c) Calculer l'accélération et l'efficacité de ces deux décompositions en utilisant le nombre de processeurs optimal pour chaque décomposition. Quelle décomposition est la meilleure ?
3. Quel est le temps d'exécution de l'algorithme parallèle sur une infinité de processeurs ?
4. Donnez une borne maximum du temps d'exécution du même algorithme sur une machine à 2 processeurs.
5. Existe-t-il une décomposition qui réduise la largeur ? Si oui, laquelle ?

Exercice 2 (Solution itérative de grands systèmes linéaires (12 points))**.** De nombreuses applications de calcul scientifique sont basées sur la résolution d'un système linéaire $A\,x = b$. Pour certains types de systèmes linéaires (ceux dont la matrice est dite à diagonale dominante), une résolution est possible avec un algorithme itératif, appelé méthode de Jacobi.
Pour un système linéaire $A\,x = b$ où $A \in \mathbb{R}^{n\times n}$ est la matrice du système (avec les éléments a_{ij}), $b \in \mathbb{R}^n$ le vecteur de droite connu et $x \in \mathbb{R}^n$ le vecteur des inconnues, la méthode de Jacobi consiste à répéter, jusqu'à convergence, la mise à jour suivante d'une solution approchée $x^{(m)}$:

$$x_i^{(m+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij}\, x_j^{(m)} \right), \quad i = 0, \ldots, n-1.$$

Tournez la page S.V.P.

Pour la première itération, tout vecteur $x^{(0)}$ peut être choisi, par exemple le vecteur initialisé comme ceci : $x_i^{(0)} = 1$, $i = 0, \ldots, n-1$.
La convergence, qui conditionne l'arrêt de l'itération, peut être détectée par plusieurs moyens différents. Un procédé simple consiste à comparer les solutions approchées de deux itérations consécutives, $x^{(m)}$ et $x^{(m+1)}$. Si elles ne diffèrent – en valeur absolue – en aucun élément plus qu'un certain seuil ε petit, la mise à jour n'a plus d'effet parce que le procédé a convergé. L'algorithme précis de la méthode de Jacobi est décrit par le pseudo-code donné dans l'algorithme 1.

Algorithme 1 La méthode de Jacobi

Pré-condition : $A \in \mathbb{R}^{n \times n}$ une matrice pour laquelle Jacobi converge,
$b \in \mathbb{R}^n$ un vecteur connu,
$\varepsilon > 0$ une constante pour le test de convergence

Post-condition : Un vecteur $x \in \mathbb{R}^n$, composés des éléments x_i ($i = 0, \ldots, n-1$), résolvant $A\,x = b$ approximativement

1: $x \leftarrow 1$
2: **Répéter**
3: $\quad \delta \leftarrow 0$
4: $\quad$ **Pour** $i = 0, \ldots, n-1$ **faire**
5: $\quad\quad c \leftarrow b_i$ /*c : scalaire temporaire*/
6: $\quad\quad$ **Pour** $j = 0, \ldots, n-1$ **faire**
7: $\quad\quad\quad$ **Si** $i \neq j$ **Alors**
8: $\quad\quad\quad\quad c \leftarrow c - a_{ij} \cdot x_j$
9: $\quad\quad\quad$ **Fin si**
10: $\quad\quad$ **Fin pour**
11: $\quad\quad c \leftarrow \frac{c}{a_{ii}}$
12: $\quad\quad \delta \leftarrow \max(\delta, |x_i - c|)$
13: $\quad\quad x'_i \leftarrow c$ /*$x' \in \mathbb{R}^n$: vecteur temporaire, composé des éléments x'_i ($i = 0, \ldots, n-1$)*/
14: $\quad$ **Fin pour**
15: $\quad x \leftarrow x'$
16: **jusqu'à ce que** $\delta < \varepsilon$

Nous nous intéressons ici à la parallélisation MPI de la méthode de Jacobi, en considérant un ensemble de noeuds avec des processeurs multicœurs comme architecture cible.

1. (1 point) Quel type d'équilibrage de charge est adapté pour cet algorithme parallèle ? Justifier.
2. (3 points) On considère la distribution suivante pour la parallélisation de la méthode de Jacobi : la matrice A est distribuée par blocs de lignes sur tous les processus. De même, le vecteur b est distribué par blocs sur tous les processus. On considère que les vecteurs x et x' peuvent être stockés en entier (si nécessaire) dans chaque processus.

   Décrire l'algorithme d'itération de Jacobi parallèle correspondant (algorithme A).

   Que devient la condition de terminaison de l'itération de Jacobi en parallèle ?

   Est-il possible d'utiliser une ou plusieurs routines MPI de communication collective dans l'algorithme parallèle ?
3. (2 points) Les communications collectives pouvant freiner le passage à l'échelle de l'algorithme parallèle, on vise un algorithme sans routine de communication collective (toujours hors calcul de la condititon de terminaison en parallèle). Pour cela, on considère que les processus forment un anneau sur lequel on va faire circuler les données.

   Décrire ce nouvel algorithme d'itération de Jacobi parallèle (algorithme B).

4. (2 points) Lequel des deux algorithmes (A ou B) vous semble le mieux adapter pour mettre en place un recouvrement des communications par le calcul ? Justifier et décrire le principe de mise en place (en MPI) pour l'algorithme retenu.

5. (2 points) On souhaite mettre en place une programmation hybride MPI-OpenMP pour l'algorithme A. Détaillez les directives OpenMP à utiliser ainsi que leurs emplacements dans l'algorithme.

6. (2 points) Quelle est (asymptotiquement) l'intensité de calcul de chaque itération de cet algorithme ? Que pouvez-vous en conclure sur le facteur limitant de cet algorithme au niveau de chaque cœur de calcul ?