Systèmes et supports d'exécution pour le calcul parallèle et distribué

Université de Bordeaux 1 — Master 2 SDRP, 2006 – 2007

Partiel

Alexandre DENIS

29 novembre 2006

Durée : 2 h. **Documents interdits.**
Les exercices sont indépendants les uns des autres. Le barème est donné à titre indicatif. Les parties de code demandées doivent être rédigées en langage C.

1 Questions de cours : microprocesseurs et parallélisme (7 points)

1. Rappelez brièvement la définition de :
   (a) parallélisme
   (b) processeur superscalaire
   (c) processeur multithread
   (d) machine SMP, machine NUMA
2. Pourquoi a-t-on introduit du parallélisme à l'intérieur des processeurs ? Est-ce une tendance durable ? Justifiez ?
3. Énumérez et décrivez les formes de parallélisme qui existent à l'intérieur des microprocesseurs.
4. Comment utilise-t-on, au niveau logiciel, ces différentes formes de parallélisme offertes par les microprocesseurs ?

2 Portabilité des *threads* (6 points)

1. Qu'appelle-t-on une construction d'intergiciel par abstraction ?
2. Qu'appelle-t-on la *portabilité* ?
3. Quels sont les aspects internes des bibliothèques de *threads* qui sont non-portables (ou difficilement portables) ? Par rapport à quels types de ressources la question de la portabilité se pose-t-elle pour les *threads* ?
4. Comment situez-vous les bibliothèques de *threads* dans les différents modèles d'abstraction ? Pourquoi a-t-on procédé de cette façon ? Que représente alors l'API `pthread` ?

3 Problème : multiplication de matrices parallèle (7 points)

Nous étudions dans ce problème la parallélisation à l'aide de *threads* d'une multiplication de matrices, telle qu'elle peut être mise en oeuvre à l'intérieur d'une bibliothèque d'algèbre linéaire comme LAPACK. La version séquentielle est donnée ci-dessous, qui calcule le produit de la matrice **`A`** par la matrice **`B`**, place le résultat dans la matrice **`C`**, toutes les matrices étant carrées de taille **`N`** fixe :

```
double A[N][N], B[N][N], C[N][N];
int i, j, k;
for(i = 0; i < N; i++)
{
  for(j = 0; j < N; j++)
  {
    double sum = 0.0;
    for(k = 0; k < N; k++)
      sum += A[i][k] + B[k][j];
    C[i][j] = sum;
  }
}
```

3.1 Parallélisation directe

1. Nous allons paralléliser la boucle la plus externe (sur **`i`**). Pourquoi ?
2. Donnez l'idée générale d'une parallélisation de ce code en OpenMP.
3. Soit **`P`** le nombre de processeurs. On supposera que **`N`** est multiple de **`P`**. Donnez une version de ce code parallélisée à l'aide de `pthread`.
4. Quelle est l'accélération attendue ? Justifiez.

3.2 Équilibrage de charge

Nous allons ajouter à notre code une gestion de l'équilibrage de charge.

1. Qu'est-ce que l'équilibrage de charge ? Dans quel cas est-il important ?
2. Est-il crucial de gérer l'équilibrage de charge dans le cas présent ?
3. Proposez une version parallèle `pthread` qui gère l'équilibrage de charge. On pourra considérer comme granularité de tâche une ligne de la matrice **`C`**.