Université Gaston Berger de Saint-Louis UFR DES SCIENCES APPLIQUEES ET DE TECHNOLOGIE Section Informatique	Master Développement et de Systèmes d'Information	Année Universitaire 2017-2018 B. DIOP

Correction de la fiche TD[1] N°1 - Traitement de problèmes parallèles

Exercice 1

Traitement de tableau à deux dimensions

Réponses aux questions

1. Puisque le calcul des éléments est indépendant l'un de l'autre, le problème peut être transformé en solution parallèle.
2. Les données du tableau sont réparties de manière équitable de sorte que chaque processus possède une partie du tableau (sous-tableau).
3. Le schéma de distribution est choisi pour un accès mémoire efficace ; par exemple. Le flot unitaire (flot de 1) à travers les sous-matrices. Le flot de l'unité maximise l'utilisation du cache et de la mémoire.
4. Comme il est souhaitable d'avoir un flot unitaire à travers les sous-matrices, le choix d'un schéma de distribution dépend du langage de programmation.
5. Le calcul indépendant des éléments du tableau garantit qu'il n'y a pas besoin de communication ou de synchronisation entre les tâches.
6. Étant donné que la quantité de travail est uniformément répartie entre les processus, il ne devrait pas y avoir de problèmes d'équilibre de charge.
7. Après la distribution du tableau, chaque tâche exécute la partie de la boucle correspondant aux données qu'elle possède. Par exemple, les distributions de blocs sont représentées sur la figure 1:

1

Introduction to Parallel Computing
Blaise Barney, Lawrence Livermore National Laboratory

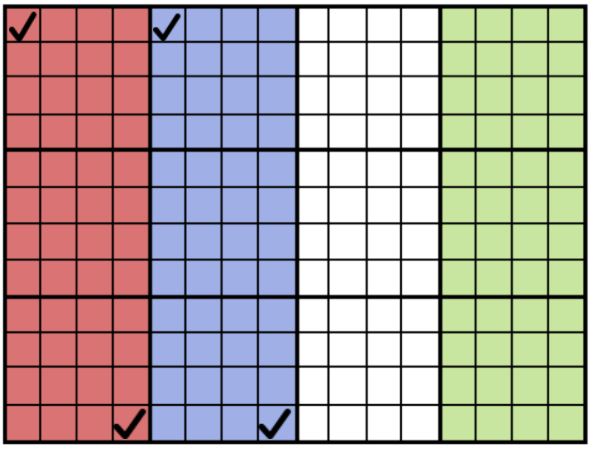

Figure 1. Division des taches

Solution 1

Une solution faisable c'est de mettre en œuvre en tant que modèle SIMD (Single Instruction Multiple Data) : chaque tâche exécute le même programme sur une ensemble de données différent.

Il existe un processus maître lequel initialise le tableau, envoie des informations aux autres processus esclaves et reçoit les résultats.

Chaque processus esclave reçoit des informations, effectue sa part de calcul et envoie les résultats au maître.

Solution de pseudo-code :

Le gras met en évidence les changements de parallélisme.

```
Vérifier si je suis MAITRE ou ESCLAVE

Si MAITRE

   Initialiser le tableau
   Envoyer à chaque esclave les infos sur une partie du tableau
   Envoyer à chaque ESCLAVE sa partie du tableau initial

   Recevoir de chaque ESCLAVE les résultats

Sinon
      Si ESCLAVE
      Recevoir de MAITRE les infos sur une partie du tableau
      Recevoir de MAITRE ma partie du tableau initial

      # calcul de ma portion de tableau
      Faire j = première colonne, dernière colonne
            Faire i = 1, n
                  a (i, j) = fcn (i, j)
```

```
        Fin do
    Fin do

    Envoyer des résultats au MAITRE

Fin si
```

Solution 2

- La solution précédente a démontré l'équilibrage de charge statique :
    - Chaque tâche a une quantité fixe de travail à faire
    - Peut y avoir un temps d'inactivité significatif pour des processus plus rapides ou plus légers - les tâches les plus lentes déterminent les performances globales.

- L'équilibrage statique de la charge n'est généralement pas une préoccupation majeure si toutes les tâches effectuent la même quantité de travail sur des machines identiques.
- Si vous avez un problème d'équilibrage de la charge (certaines tâches fonctionnent plus rapidement que d'autres), vous pouvez bénéficier d'un graphe de "pool de tâches".

Graphe de "pool de tâches

- Deux processus sont employés

  Processus MAITRE :
    - Contient un pool de tâches pour les processus de travail à faire
    - Envoie une tâche à un travailleur à la demande
    - Recueille les résultats des travailleurs

  Processus ESCLAVE :

  De manière répétitive faire
    - Obtenir une tâche à partir du processus MAITRE
    - Effectuer le calcul obtenu du MAITRE
    - Envoie les résultats au MAITRE

- Les processus de travail ne savent pas avant l'exécution quelle partie du tableau ils vont gérer ou combien de tâches ils vont effectuer.
- L'équilibrage de charge dynamique se produit au moment de l'exécution : les tâches les plus rapides auront plus de travail à faire.

Solution de pseudo-code :

Le gras met en évidence les changements pour le parallélisme.

```
Vérifier si je suis MAITRE ou ESCLAVE
Si MAITRE
   Répéter jusqu'à ce qu'il n'y ait plus de tache
      Si demande
            Envoyer à ESCLAVE prochain travail
      Sinon
            Recevoir les résultats de ESCLAVE
   Fin Répéter

Sinon Si ESCLAVE
      Répéter jusqu'à ce qu'il n'y ait plus de tache
            Demande d'emploi à MAITRE
            Recevoir du MAITRE la tache suivante

            Calculer l'élément de tableau : a (i, j) = fcn (i,j)

            Envoyer les résultats à MASTER
      Fin Répéter

fin si


------------------------------------------------------------
```

Exercice 2

Estimation de PI

Solution parallèle

- Le problème est facile à paralléliser
- Tous les calculs de points sont indépendants. Pas de dépendances de données
- Le travail peut être divisé de manière équitable. Aucun problème d'équilibre de charge
- Pas besoin de communication ou de synchronisation entre les tâches

Stratégie parallèle

- Diviser la boucle en portions d'itérations égales pouvant être exécutées par le pool de tâches
- Chaque tâche effectue indépendamment son travail
- Un modèle SPMD est utilisé
- Une tâche agit comme le MAITRE pour recueillir des résultats et calculer la valeur de PI

Solution de pseudo-code :

Le gras met en évidence les changements pour le parallélisme.

```
nbPoints = 10000
cercle_count = 0

p = nombre de tâches
num = nbPoints / p

Vérifier si je suis MAITRE ou ESCLAVE

Faire j = 1, num
   Générer 2 nombres aléatoires entre 0 et 1
   xcoordonnée = random1
   ycoordonnée = random2
   Si (xcoordonnée, ycoordonnée) se trouve dans le cercle alors

      cercle_count = cercle_count + 1

fin Faire

Si MAITRE
   Recevoir de ESCLAVE leur cercle_count
   Calculer PI (utiliser les calculs MAITRE et ESCLAVE)

Sinon si ESCLAVE
   Envoyer à MAITRE cercle_count

fin si
```

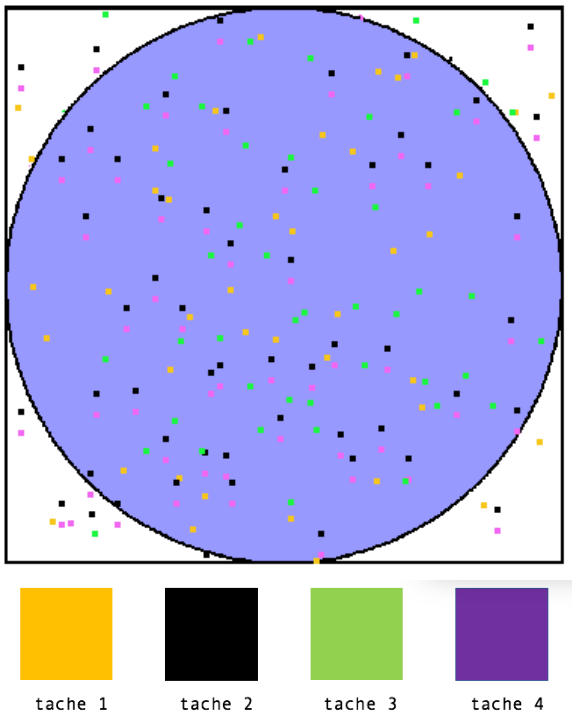

Figure 2. Division des taches

Exercice 3

Équation de chaleur simple

Solution parallèle

- Ce problème est plus difficile, car il existe des dépendances de données, qui nécessitent des communications et la synchronisation.
- Le tableau entier est partitionné et distribué comme sous-programmes à toutes les tâches. Chaque tâche possède une partie égale du tableau total.
- Parce que la quantité de travail est égale, l'équilibrage de charge ne devrait pas être une préoccupation
- Déterminer les dépendances de données :
    - Les éléments intérieurs appartenant à une tâche sont indépendants des autres tâches
    - Les éléments de bordure dépendent des données d'une tâche voisine, ce qui nécessite une communication.

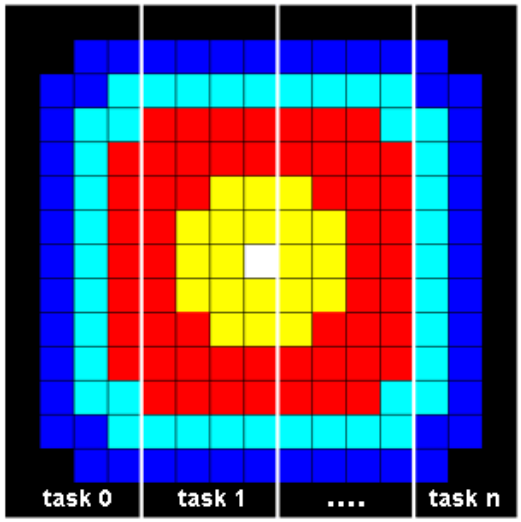

Figure 3. Division des taches

- Mettre en œuvre en tant que modèle SPMD :
    - Le processus ESCLAVE envoie les informations initiales aux travailleurs, puis attend de recueillir les résultats de tous les travailleurs
    - Les processus MAITRE calculent la solution dans le nombre d'étapes de temps spécifié, en communiquant si nécessaire avec les processus voisins

Solution de pseudo-code :

Le gras met en évidence les changements de parallélisme.

```
Vérifier si je suis MAITRE ou ESCLAVE

Si je suis MAITRE
   Initialiser le tableau
   Envoyer à chaque ouvrier des informations de départ et un sous-tableau
   Recevoir les résultats de chaque ESCLAVE

Sinon ESCLAVE
   Recevoir de MAITRE informations de départ et sous-tableau
```

```
   # Effectuer des pas de temps
   Faire t = 1, nsteps
     Mettre à jour le temps
     Envoyer aux voisins mes informations de frontière
     Recevoir des voisins leurs informations frontalières
     Mettre à jour ma partie du tableau de solutions

   Fin faire

   Envoyer des résultats MAITRE

Fin si
```

Exercice 4

Équation d'onde 1-D

Solution parallèle

- Ceci est un autre exemple d'un problème impliquant des dépendances de données. Une solution parallèle impliquera des communications et une synchronisation.
- Le tableau d'amplitude est partitionné et distribué comme sous-réseaux à toutes les tâches. Chaque tâche possède une partie égale du tableau total.
- Équilibrage de charge : tous les points nécessitent un travail égal, donc les points doivent être divisés de manière égale
- Une décomposition par blocs ferait partitionner le travail en nombre de tâches, ce qui permettrait à chaque tâche de posséder principalement des points de données contigus.
- La communication doit uniquement avoir lieu sur les bordures de données. Plus la taille du bloc est grande, moins la communication est importante.

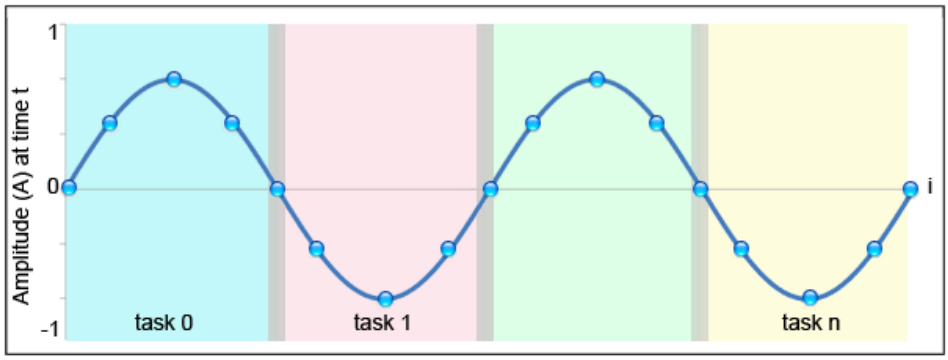

Figure 4. Division des taches

- Mettre en œuvre en tant que modèle SPMD :
    - Le processus maître envoie les informations initiales aux travailleurs, puis attend de recueillir les résultats de tous les travailleurs
    - Les processus ESCLAVE calculent la solution dans le nombre d'étapes de temps spécifié, en communiquant si nécessaire avec les processus voisins

Solution de pseudo-code :

Le gras met en évidence les changements de parallélisme.

```
Déterminer le nombre de tâches et l'identité des tâches

#Identifier les voisins de gauche et de droite
left_neighbor = mytaskid - 1
right_neighbor = mytaskid +1
Si mytaskid = first alors left_neigbor = last
Si mytaskid = last alors right_neighbor = premier

Vérifier si je suis MAITRE ou ESCLAVE

Si je suis MAITRE
  Initialiser le tableau
  Envoyer à chaque ESCLAVE des informations de départ et un sous-tableau

Sinon si ESCLAVE
  Recevoir les informations de départ et le sous-tableau de MAITRE
fin Si

#Perform time steps
#Dans cet exemple, le maître participe aux calculs
Faire t = 1, nsteps
  Envoyer le point de terminaison gauche au voisin de gauche
  Recevoir l'extrémité gauche du voisin droit
  Envoyer le bon point final au voisin droit
  Recevoir l'extrémité droite du voisin de gauche

  # Mettre à jour les points le long de la ligne
        Faire i = 1, npoints
          newval (i) = (valeurs de 2.0 * (i)) - oldval (i)
        + (sqtau * (valeurs (i-1) - (valeurs 2.0 * (i)) + valeurs (i + 1)))
        fin Faire

fin Faire

# Collecter les résultats et écrire dans le fichier
Si je suis MAITRE
  Recevoir les résultats de chaque ESCLAVE
  Écrire les résultats dans un fichier
Sinon si ESCLAVE
  Envoyer les résultats à MAITRE
fin Si
```