IUT, dÃĺpartement Informatique
2ÃÍme annÃľe
AnnÃľe Universitaire 2012-2013

Programmation système et réseaux

TP n° 4
Calcul parallèle multicoeur, mutex et sémaphores

Durée 2 semaines

Objectifs :
Le but de ce TP est de créer des programmes multithreads pour faire du calcul parallèle. On comparera différentes techniques de protection des accès concurrents aux données partagées.

Note. Chaque semaine, le listing commenté du TP est à rendre en début de la scéance de la semaine suivante.

Exercice 1 Tri par bulles randomisé parallèle Le but de cet exercice est de comparer différentes métthodes de protection des accès concurrents à un tableau lors de l'application d'un algorithme parallèle de tri.

Le principe du tri par bulles randomisé pour trier un tableau de taille n est très simple :

```
tant que le tableau n'est pas trié
faire
   tirer deux indices i et j du tableau au hasard
      si tableau[i] et tableau[j] ne sont pas dans le bon ordre
         échanger tableau[i] et tableau[j]
fin faire
```

a) Implémenter des routines de base sur les tableaux : allocation, génération aléatoire, recopie, affichage d'un tableau. Implémenter aussi le tri par bulle du tableau pour avoir une méthode de tri fonctionnelle.

b) Implémenter en C le tri par bulles randomisé. Le programme affichera la durée de l'algorithme de tri (en secondes).

c) Sauvegardez votre programme du b). Réaliser une implémentation multithrédée de l'algorithme du tri par bulles randomisé. On créera N threads, où N est une constante définie par une directive de précompilation. Dans un premier temps, on ne gérera pas les conflits d'accès au tableau. On créera aussi un thread qui, toutes les secondes, vérifiera si le tableau est trié et le cas échéant terminera le programme en affichant la durée de l'algorithme de tri. On verifiera aussi qu'il n'y a pas de bug, c'est à dire que le tableau obtenu est bien le même que le tableau obtenu par le tri par bulles standard du a). Observez le taux de charge des différents cores

de votre machine pour le programme du a) et du b) à l'aide d'un *system monitor* tel que `gnome-system-monitor`.

d) Sauvegardez votre programme du c) Faire une nouvelle version du programme dans laquelles les accès au tableau sont protégés par un mutex.

e) Sauvegardez votre programme du d) Faire une nouvelle version du programme dans laquelles les accès au tableau sont protégés par un modèle lecteur-rédacteur basé sur des sémaphores POSIX.

f) Comparez les performances en temps de calcul des méthodes du b), c), d) et e) sur une dixaine (ou plus) de tableaux aléatoires. Calculez la moyenne, la médiane, le minimum et le maximum des runtimes pour chaque méthode. Comment interprétez-vous les résultats ?

Exercice 2 Tri fusion parallèle Le tri fusion est une méthode de tri de complexité $O(n \log n)$. Elle reprose sur le pricipe suivant :

1. on partage le tableau en deux moités de taille plus ou moins égales ;
2. On trie les deux moitiés de tableau par récursivité ;
3. On fusionne les deux moitiés de tableau triées par un interclassement (de complexité linéaire).

a) Réaliser le tri fusion (séquentiel) d'un tableau.

b) Sauvegardez votre programme du a). Créez une version parallèle multithreadée du tri fusion dans laquelle chaque appel récursif est traité par un thread différent. En fait, on créera des nouveaux threads uniquement pour les appels récursifs jusqu'à une certaine profondeur P de la pile d'appel, où P est une constante définie par une directive de précompilation.

c) Comparez les version du tri fusion du a) et du b) pour différentes valeurs de P.