ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF8601: Systèmes informatiques parallèles (Automne 2014)
3 crédits (3-1.5-4.5)

CONTRÔLE PÉRIODIQUE

DATE: Lundi le 3 novembre 2014

HEURE: 9h30 à 11h20

DUREE: 1H50

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

a) Une entreprise offre un service d'espace disque à distance aux usagers, AieNuage. L'entreprise veut offrir un service fiable mais ne veut pas assumer les coûts de prendre des copies de sécurité sur ruban. Elle utilise donc un système avec un grand nombre de noeuds avec redondance. Chaque noeud est constitué d'une unité de traitement et d'un disque de 1TB, pouvant accueillir les fichiers de 100 usagers. Les fichiers de chaque usager sont stockés en permanence sur 3 noeuds différents, n'importe lequel des 3 pouvant alors servir les requêtes pour cet usager. Si une unité de traitement est en panne, le noeud est indisponible temporairement. Si un disque est en panne, les données de cette copie sont irrémédiablement perdues. La probabilité qu'une unité de traitement soit en panne est de 0.001. La probabilité qu'un disque soit en panne est de .002. Quelle est la probabilité que les fichiers d'un usager à un moment donné soient indisponibles (perdus ou non)? Irrémédiablement perdus? Si le service compte 2 milliards d'usagers, combien ont perdu leurs fichiers irrémédiablement à un instant donné? **(2 points)**

b) Sur les processeurs Intel, il est possible de varier la taille des pages en mémoire virtuelle, et d'utiliser soit des petites pages de 4Kio ou des grandes pages de 4Mio. Quel est l'impact de ce choix sur le taux de succès pour la cache de pré-traduction d'adresse (TLB)? Le taux de succès en mémoire cache? L'utilisation efficace de la mémoire physique? **(1 point)**

c) Expliquez ce qui peut causer un faux partage de données en cache. Quel en est l'impact? Comment peut-on se débarrasser d'un tel problème? Donnez un exemple d'allocateur en TBB qui permet d'allouer une donnée qui ne présentera pas de tel problème. **(1 point)**

d) Vous venez de paralléliser une fonction F d'un programme qui était sériel. La nouvelle version de F peut ainsi utiliser efficacement les 64 processeurs de votre ordinateur. Les autres fonctions sont inchangées. Le programme s'exécute maintenant 10 fois plus vite au total. Quelle fraction du temps prenait initialement la fonction F? **(1 point)**

Question 2 (5 points)

a) Un très grand vecteur v contient des entiers signés qui représentent les entrées et les sorties de participants pendant *la fête de la plage* à Polytechnique. Par exemple, +1 indique une entrée et -2 la sortie de deux personnes. Sachant que la salle était initialement vide, il faut calculer le vecteur t qui contient le nombre de participants dans la salle après avoir traité la donnée du vecteur v à la position correspondante. On veut effectuer ce calcul avec la librairie TBB. Quelle structure (parallel for, parallel reduce, parallel scan ou parallel do) suggérez-vous d'utiliser? Ecrivez le coeur de ce programme, comme dans les exemples utilisés pour les dispositives du cours expliquant TBB. **(2 points)**

b) Dans le premier travail pratique, vous avez examiné et comparé le travail de PThread et TBB. Quels sont les appels système impliqués dans la synchronisation des fils d'exécution de PThread et TBB? **(1 point)**

c) Un ordinateur 64 bits fonctionne avec une table de pages à 4 niveaux. Seul un bit de son espace adressable n'est pas utilisé, laissant 63 bits pour ses adresses virtuelles. Toutes les pages ont la même taille et sont utilisées autant pour peupler l'espace virtuel des processus que pour contenir les différents niveaux de la table de page. Quelle est la taille d'une page? **(1 point)**

d) Un processeur 64 bits possède une cache L1 de 256Kio avec des blocs de 256 octets. La fonction de correspondance est directe-associative avec un degré d'associativité de 4, ce qui donne des ensembles de 4 blocs chacun. Si on vous donne l'adresse suivante en hexadécimal `0x000000000A0B0C0D`. Donnez le numéro d'ensemble en cache, le décalage dans le bloc et l'étiquette associés à cette adresse. **(1 point)**

Question 3 (5 points)

a) Un des modèles étudiés de cohérence en mémoire partagée est l'ordre partiel des écritures (PSO). Expliquez lesquels réordonnancements des accès mémoire (lecture ou écriture versus lecture ou écriture) sont possibles? Donnez un exemple de structure matérielle qui pourrait expliquer de tels réordonnancements. **(2 points)**

b) La section de code suivante s'exécute sur un ordinateur avec ordonnancement faible. Quelles sont les sorties possibles produites par printf? Quelles barrières devrait-on insérer, et où, pour avoir comme seules sorties possibles `1 2 3` ou `0 0 0`? **(2 points)**

```
(initialement valid, a, b et c sont 0)
Processeur 0                    Processeur 1

a=1;                            if(valid) printf("%d %d %d\n", a, b, c);
b=2;                            else printf("0 0 0\n");
c=3;
valid=1;
```

c) Vous devez programmer une application avec plusieurs fils d'exécution de manière à maximiser l'utilisation de toutes les ressources disponibles sur le serveur (processeurs, disques...). Chaque fil traite une requête à la fois qui peut nécessiter du calcul, des accès réseau et des accès disque. Puisque ce logiciel sera installé sur une grande variété de serveurs, vous ne pouvez savoir à l'avance quel sera le nombre de processeurs ou de disques, ni la fraction du temps passée à attendre après les requêtes réseau. Par ailleurs, il faut éviter d'avoir beaucoup trop de fils d'exécution, puisque cela encombre les queues du système d'exploitation, et il ne faut pas non plus continuellement créer et détruire des fils d'exécution, puisqu'il y a un coût non négligeable associé à cela. Proposez une organisation efficace pour décider combien de fils d'exécution créer et pour gérer l'utilisation et l'évolution de ces fils d'exécution. **(1 point)**

Question 4 (5 points)

a) Le programme suivant calcule la distribution des couleurs dans une grande image. Convertissez ce programme sériel en programme OpenMP efficace. **(2 points)**

```
void ColorDistribution(unsigned char image[2048][2048],
    unsigned int d[256])
{
  for(int i = 0; i < 2048; i++)
    for(int j = 0; j < 2048; j++) d[image[i][j]]++;
}
```

b) Lorsqu'une boucle parallel for est rencontrée en OpenMP, la librairie de support à l'exécution doit répartir les itérations entre les différents fils d'exécution. En supposant que, dans le court programme montré ici, la valeur de `img->width` est de 1024, que le nombre de processeurs disponibles est 8, et que la durée de la fonction f1 est assez variable, expliquez comment le travail pourrait être réparti efficacement sur plusieurs fils d'exécution par OpenMP, en spécifiant bien quel fil traitera quelles itérations à quel moment. **(2 points)**

```
int encode(struct image *img)
{
    int i, j;

    #pragma omp parallel for private(i,j)
    for (i = 0; i < img->width; i++) {
        for (j = 0; j < img->height; j++) {
            img->data[i][j] = f1(img->data[i][j]);
        }
    }
}
```

c) Dans une boucle OpenMP, vous avez besoin de calculer une somme globale. Vous pourriez prendre une variable globale protégée par un verrou (section critique ou mutex), déclarer une variable comme une réduction pour la boucle, ou utiliser une opération d'incrément atomique sur une variable globale. Qu'est-ce qui sera le plus performant? Pourquoi? **(1 point)**

Le professeur: Michel Dagenais