ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF8601: Systèmes informatiques parallèles (Automne 2013)
3 crédits (3-1.5-4.5)

CONTRÔLE PÉRIODIQUE

DATE: Mardi le 22 octobre 2013

HEURE: 14h45 à 16h35

DUREE: 1H50

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

a) Une application utilise énormément de mémoire et le système de mémoire virtuelle doit ainsi régulièrement transférer certaines pages sur le disque. Une optimisation suggérée est d'avoir la possibilité de compresser des pages en mémoire. Lorsqu'une page est très utilisée elle est conservée telle quelle, lorsqu'une page est moins utilisée elle est compressée mais conservée en mémoire et peut être décompressée au prochain accès, et lorsqu'une page est très peu utilisée elle est transférée sur disque pour être relue au prochain accès. Cette technique permet de conserver jusqu'à deux fois plus de pages en mémoire lorsqu'elles sont compressées mais demande un certain effort de calcul pour la compression / décompression. En raison de ce compromis, cette technique est présentement marginalement utile. Pour chacun des paramètres suivants, dites s'il rend plus avantageux ou moins avantageux cette technique: i) augmentation de la vitesse du disque en passant à un disque à semi-conducteur (SSD), ii) augmentation de la vitesse du CPU, iii) augmentation du nombre de CPU, iv) augmentation de la taille de la mémoire? Justifiez? **(2 points)**

b) Un centre de données contient 10 000 noeuds. Chaque noeud est connecté à une unité externe de disques en RAID contenant 5 disques dont au moins 4 doivent être fonctionnels pour opérer correctement. Chaque noeud en lui-même est en panne en moyenne 8 heures par année. Par ailleurs, chaque unité RAID est en panne lorsque 2 disques ou plus sont défectueux, avec chaque disque qui a une probabilité de panne de 0.001. Quelle est la probabilité de panne pour un noeud (noeud lui-même ou unité de disque en panne) et donc quel est le nombre moyen de noeuds en panne à un instant donné dans le centre de données. **(2 points)**

c) En une seconde, une application sur un processeur exécute 2 000 000 000 instructions et génère autant d'accès mémoire pour les instructions, en plus de générer 500 000 000 accès pour les données, dans le cas idéal où il n'y a aucune attente après la mémoire (pas de faute de cache). Le taux de succès en cache est de 99.9% pour les instructions et de 99.5% pour les données. La pénalité d'échec est de 80ns. Quelle est la vitesse de cette application sur ce processeur, en nombre d'instructions par seconde, en tenant compte des attentes après la mémoire? **(1 point)**

Question 2 (5 points)

a) Dans le premier travail pratique, vous avez utilisé l'outil Valgrind / Callgrind. Quelle information en avez-vous tiré? Comment avez-vous pu avec cette information prédire le facteur d'accélération de votre application en multi-processeur? Est-ce que cette prédiction est plutôt une borne inférieure ou supérieure de l'accélération qui sera réellement obtenue? **(1.5 point)**

b) Toujours pour le premier travail pratique, vous avez utilisé LTTng afin de mieux comprendre le comportement de TBB en comparaison avec la librairie PThread. Combien de fils d'exécution sont utilisés dans chaque cas? Comment se fait la répartition du calcul entre les fils d'exécution? Quelles sont les données écrites dans la trace qui permettent ainsi d'afficher une vue temporelle de l'exécution? **(1.5 point)**

c) Un serveur est responsable de fournir des images en différents formats et résolutions. Chaque requête pour une image demande en moyenne 150ms de temps de processeur, 50ms de disque et 300ms d'attente après un client lors de la communication réseau. Chaque fil d'exécution traite une requête à la fois, séquentiellement. L'unité centrale de traitement contient 8 processeurs (physiques sans hyperthread). En supposant que les disques et le réseau sont toujours disponibles, combien de fils d'exécution devrait-on avoir au minimum pour toujours occuper les 8 processeurs? Combien de disques devrait-on avoir afin de pouvoir fournir au même débit que les 8 processeurs. **(2 points)**

Question 3 (5 points)

a) Votre ordinateur contient 8 processeurs. Cependant, vous ne savez pas si ce sont en fait 4 processeurs physiques avec hyperthread ou 8 processeurs physiques. De plus, vous ne savez pas si certains processeurs partagent on non leur cache de niveau L1. Cependant, on vous a bien mis en garde contre le fait de mettre deux tâches temps réel sur deux processeurs qui partagent la même cache L1. Sachant que vous pouvez facilement spécifier au système d'exploitation quel fil exécuter sur quel processeur, proposez une méthode (description d'un programme d'essai à exécuter) pour déterminer les processeurs physiques et logiques ainsi que les processeurs physiques qui partagent leur cache L1. **(2 points)**

b) Un jeu multi-fil s'exécute sur deux processeurs à mémoire partagée. C'est un jeu classique à plusieurs joueurs, chacun ayant le choix entre plusieurs outils, le but du jeu étant de construire quelque chose. Une section de code montrée plus bas effectue la sélection d'outil sur le processeur 0 alors que les informations sur cet outil peuvent être accédées sur le processeur 1. Pour que l'exécution soit cohérente, il faut que les informations sur le nouvel outil choisi soient disponibles avant d'être accédées. Quelles sont les barrières mémoire minimales à insérer (et où) afin d'assurer un comportement correct du programme dans tous les cas? **(2 points)**

```
(initialement outil = &marteau; marteau.debut = 0)
Processeur 0                    Processeur 1

tournevis.debut = 1;            outil_courant = outil;
outil = &tournevis;             debut = outil_courant->debut;
```

c) Un système de cache utilise le protocole MOESI et un maintien de la cohérence par répertoire. Expliquez quelle information doit être conservée avec cette organisation pour chaque bloc en mémoire cache? **(1 point)**

Question 4 (5 points)

a) La fonction suivante décale la valeur de chaque pixel d'une image. La variable KEY est une constante et l'image reçue en argument n'est pas accédée par d'autres fonctions en parallèle. Proposez plusieurs améliorations qui devraient permettre d'accélérer significativement cette fonction tout en obtenant le même résultat à la fin. Justifiez. **(2 points)**

```
int encode(struct image *img)
{
    int i, j, index;
    int checksum = 0;
```

```
    #pragma omp parallel for private(i,j,index)
    for (i = 0; i < img->width; i++) {
        for (j = 0; j < img->height; j++) {
            index = i + j * width;
            img->data[index] = img->data[index] + KEY;
            #pragma omp atomic
            checksum += img->data[index];
        }
    }
    return checksum;
}
```

b) Quelle est la différence entre les stratégies *static*, *dynamic*, *guided* et *auto* pour partitionner les itérations des boucles et les ordonnancer sur les fils d'exécution en OpenMP? Expliquez dans quelle situation chacune peut être particulièrement intéressante. **(2 points)**

c) Quel est l'effet de la clause *nowait* sur le déroulement d'une boucle *for* en OpenMP? Est-ce que ceci peut faire une différence appréciable? Expliquez? **(1 point)**

Le professeur: Michel Dagenais