ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF8601: Systèmes informatiques parallèles (Automne 2016)
3 crédits (3-1.5-4.5)

CONTRÔLE PÉRIODIQUE

DATE: Mercredi le 26 octobre 2016

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) Dans une grappe de calcul, la probabilité qu'un ordinateur donné fonctionne sans panne pendant une tâche de durée t (en heures) est donnée par la fonction $p(t) = 1/(.01t + 1)$. Par exemple, la probabilité qu'une tâche puisse se terminer correctement, sans panne du noeud qui l'exécute, est de 1 pour $t = 0$, 0 pour $t = \infty$ et 0.80645 pour $t = 24h$. Si une grappe contient 100 noeuds, quelle est la probabilité que tous les noeuds soient opérationnels pendant une tâche parallèle de 1h? De 2 h? **(2 points)**

b) Lors du premier TP, vous avez utilisé l'outil Callgrind de l'environnement Valgrind. Quelle information est-ce que cet outil fournit? Comment avez-vous utilisé cette information, quelle était son utilité? **(1 point)**

c) Dans le cadre du premier TP, vous avez effectué le même calcul à l'aide des POSIX Threads et à l'aide de TBB. Quelle version était la plus efficace? Est-ce que la différence était grande? Comment expliquez-vous cette différence? **(1 point)**

d) Un programme est composé de sections parallélisables et non parallélisables. La fraction parallélisable est de 88%. Ce programme est parallélisé et exécuté sur un ordinateur de 4 coeurs, quel sera le facteur d'accélération? Sur un ordinateur à 64 coeurs? **(1 point)**

Question 2 (5 points)

a) Un ordinateur 64 bits, dont la mémoire est adressable à l'octet, utilise des tables de pages à 4 niveaux et des pages de 8Kio. Chaque noeud de la table de pages entre dans une page. Montrez comment l'adresse se décompose en décalage dans la page, index pour chacun des 4 niveaux dans la table, et s'il y a lieu les bits restants. A quoi servent les bits restants s'il y en a? Décomposez l'adresse virtuelle 0x0000ABCDEF012345 en ces différentes composantes (décalage, index3, index2...). **(2 points)**

b) Un ordinateur possède une cache L1 MESI de 64Kio, avec des ensembles de 8 blocs, pour chacun de ses 4 coeurs. Chaque bloc en cache contient 128 octets. Cette cache est initialement vide. Les adresses suivantes sont accédées en séquence, en lecture (R) ou en écriture (W), sur les processeurs spécifiés (P0 à P3). Donnez l'état des cases non vides (M, E, S ou I) après chaque accès. **(2 points)**

```
P0 R 0x12A; P0 R 0x02A; P1 W 0x12A 2;
P2 W 0x12A 4; P2 R 0x12A; P1 W 0x02A 8
```

c) La cache de pré-traduction d'adresse (TLB) permet de convertir les adresses virtuelles d'un processus en ses adresses physiques, en mémorisant un sous-ensemble (cache) du contenu de la table de pages d'un processus. Sur certaines variantes plus avancées de TLB, une étiquette est ajoutée aux adresses virtuelles dans le TLB pour identifier le processus associé. Doit-on mettre à jour le contenu du TLB lorsque l'espace d'adressage du processus est modifié (e.g., une zone de mémoire partagée est ajoutée par un appel à mmap())? Expliquez. Que doit-on faire avec le contenu du TLB, selon que ce soit un TLB régulier ou une "variante avancée", lorsque l'ordonnanceur change de thread mais tout en restant dans le même processus? Lorsque l'ordonnanceur change de processus? **(1 point)**

Question 3 (5 points)

a) Ecrivez un programme, utilisant la librairie TBB, qui offre la fonction *int CheckSorted(float v[], int size)* permettant d'effectuer le calcul suivant. L'argument v contient un vecteur de nombres à virgule flottante alors que l'argument size contient la taille de v. La fonction CheckSorted doit calculer et retourner le nombre de fois que des items dans le vecteur ne sont pas en ordre croissant (i.e., l'item à la position i est plus petit que l'item en position i - 1, pour i entre 1 et size - 1 inclusivement). **(3 points)**

b) Un programme multi-thread s'exécute sur plusieurs processeurs en parallèle. Afin d'aller plus vite, il n'utilise pas de verrou. Un thread veut mettre à jour des valeurs data_a_1 et data_a_2 puis data_b_1 et data_b_2 et indiquer lorsqu'elles sont valides en mettant à 1 valid_a et valid_b respectivement. Le programme proposé suit. Doit-on ajouter des barrières mémoire pour avoir un comportement correct (i.e., ne pas lire les données avant qu'elles aient été mises à jour)? Si oui, modifiez le programme en ajoutant les barrières mémoire minimales requises? **(2 points)**

```
(initialement toutes les variables sont à 0)
Processeur 0                    Processeur 1

data_a_1 = 20;                   if(valid_a) {
data_a_2 = 18;                     new_a = data_a_2;
data_b_2 = 8;                    }
data_b_1 = 10;                   if(valid_b) {
valid_b = 1;                       new_b = data_b_1;
valid_a = 1;                     }
```

Question 4 (5 points)

a) On vous fournit la section de code suivante en OpenMP. Vous devez proposer plusieurs améliorations qui permettront d'obtenir le même résultat mais plus efficacement. Fournissez une nouvelle version de cette section de programme et expliquez en une ou deux lignes de texte chaque amélioration apportée. **(2 points)**

```
#pragma openmp parallel for
for(int i = 0; i < nb_cpu; i++) {
  for(int j = 0; j < nb_task; j++) {
    for(int k = 0; k < nb_resource; k++) {
      by_cpu[i] += time[k][i][j];
      #pragma openmp atomic
      sum += time[k][i][j];
} } }

#pragma openmp parallel for
for(int i = 0; i < nb_cpu; i++) {
  for(int j = 0; j < nb_task; j++) {
    for(int k = 0; k < nb_resource; k++) {
      by_task[j] += time[k][i][j];
} } }

#pragma openmp parallel for
for(int i = 0; i < nb_cpu; i++) {
  for(int j = 0; j < nb_task; j++) {
    for(int k = 0; k < nb_resource; k++) {
      by_resource[k] += time[k][i][j];
} } }
```

b) Le programme OpenMP suivant s'exécute et produit une sortie sur stdout. Donnez une sortie possible produite par l'exécution de ce programme. Est-ce que la sortie peut changer d'une exécution à l'autre? Expliquez. **(2 points)**

```
int main(int argc, char **argv)
{ int nb_thread, thread;
  printf("a)\n");
  omp_set_num_threads(4);
  #pragma omp parallel private(nb_thread, thread)
  { printf("b)\n");
    #pragma omp for schedule(static)
    for(int i = 0; i < 8; i++) {
      nb_thread = omp_get_num_threads();
      thread = omp_get_thread_num();
      printf("c) thread %d / %d, i = %d\n", thread, nb_thread, i);
    }
    #pragma omp master
    printf("d) thread %d / %d\n", thread, nb_thread);
    #pragma omp critical
    printf("e) thread %d / %d\n", thread, nb_thread);
    #pragma omp single
    printf("f) thread %d / %d\n", thread, nb_thread);
  }
  printf("g) thread %d / %d\n", thread, nb_thread);
}
```

c) Peut-on changer la taille de la pile en OpenMP? Comment? Comment sait-on si on doit changer la taille de la pile? **(1 point)**

Le professeur: Michel Dagenais