Introduction

Exercices pour le Module 1
INF8601 Systèmes informatiques parallèles

Michel Dagenais

École Polytechnique de Montréal
Département de génie informatique et génie logiciel

Accélération parallèle

Une application parallèle prend 4% de son temps pour diviser le travail et 3% pour réconcilier les réponses. Quelle est l'accélération maximale obtenue sur 10, 50, 100 ou 1000 processeurs?

Accélération parallèle

Une application parallèle prend 4% de son temps pour diviser le travail et 3% pour réconcilier les réponses. Quelle est l'accélération maximale obtenue sur 10, 50, 100 ou 1000 processeurs?

Pour $n = 10, 50, 100, 1000$, le temps relatif (ou son inverse l'accélération a), en fonction de la fraction parallélisable f et du nombre de processeurs parallèles p, devient selon la loi d'Amdahl, $a = \frac{1}{(1-f)+\frac{f}{p}}$: 0.163 (6.1), 0.0886 (11.3), 0.0793 (12.6), 0.07093 (14.1).

Hiérarchie de mémoire

L'architecture Intel 64 décompose son adresse 64 bits en 16 bits inutilisés, 4 * 9 bits pour 4 niveaux de table de pages et 12 bits d'adresse dans la page. La cache L1 fait 32KiO, a une associativité de 4 et contient des blocs de 64 octets. Montrez l'organisation de la hiérarchie de mémoire et décrivez comment se fait l'accès à une donnée dont l'adresse virtuelle est fournie.

Hiérarchie de mémoire

L'adresse de 64 bits se décompose en: 16 inutilisés, 9, 9, 9, 9 (index 0, 1, 2, 3), 12 pour octet dans la page. Pour un accès, la cache de prétraduction d'adresse contient 128 entrées de 36 bits d'étiquette vs adresse physique. Dans une bonne proportion des cas, on y trouve l'adresse physique correspondant à l'adresse logique cherchée. Autrement, la table de page de niveau 0 est indexée avec le premier 9 bits, menant vers une page de la table de niveau 1, indexée avec le second champ de 9 bits... jusqu'au niveau 3. Ce dernier niveau contient l'adresse physique et des bits de statut (valide, accessible...).

L'adresse physique obtenue, il faut vérifier si elle se trouve en cache L1. La cache de donnée de 32KiO contient 512 blocs de 64 octets, soit 128 ensembles de 4 blocs. L'adresse de 64 bits donne: 16 inutilisés, 35 étiquette, 7 ensemble, 6 octet du bloc. Pour chaque ensemble, on a 4 blocs avec leur étiquette correspondante dans la cache. Le numéro d'ensemble est pris dans l'adresse et l'étiquette vérifiée avec celle des 4 blocs disponibles dans l'ensemble pour voir si le bloc cherché est en cache. Autrement, il faut charger en L1 le bloc contenant l'adresse physique recherchée à partir de la mémoire centrale ou L2.

Le rôle du système d'exploitation

Votre programme effectue un appel à fgets pour lire des données du disque. Décrivez le traitement fait par la librairie et le système d'exploitation?

Le rôle du système d'exploitation

Votre programme effectue un appel à fgets pour lire des données du disque. Décrivez le traitement fait par la librairie et le système d'exploitation?

Premièrement, la librairie vérifie s'il reste des donnée dans le tampon de lecture pour ce pointeur de fichier. Autrement, un appel système read est effectué. Le système d'exploitation vérifie si le bloc désiré est dans ses tampons d'E/S et sinon envoie une requête au contrôleur de disque et met le processus en attente. Le contrôleur lit le bloc voulu, le copie en mémoire par DMA et envoie une interruption au système d'exploitation. Le système d'exploitation change le statut du processus à prêt et pourrait le relancer immédiatement.

Mesurer la taille de la cache

On vous demande de mesurer la taille des blocs de cache L1 ainsi que la taille de la cache L1. Comment feriez-vous?

Mesurer la taille de la cache

On vous demande de mesurer la taille des blocs de cache L1 ainsi que la taille de la cache L1. Comment feriez-vous?

Un vecteur peut être accédé en boucle déroulée pour différentes tailles de vecteur. Lorsque la taille dépasse la grosseur de la cache L1 de donnée, le temps d'accès moyen augmente rapidement. Ensuite, on peut lire dans un vecteur de taille supérieure à L1 des octets consécutifs, 1 sur 2, 1 sur 3... Le temps d'accès moyen augmente jusqu'à ce qu'on atteigne 1 sur taille du bloc.

Efficacité maximale du matériel

Un additionneur 1 bit peut produire une somme sur 1 bit en une unité de temps avec 6 portes logiques. Un additionneur 32 bits peut produire une somme de 32 bits avec 320 portes logiques en 6 unités de temps. Calculez le rendement de chacun (travail / (matériel * temps))?

Efficacité maximale du matériel

Un additionneur 1 bit peut produire une somme sur 1 bit en une unité de temps avec 6 portes logiques. Un additionneur 32 bits peut produire une somme de 32 bits avec 320 portes logiques en 6 unités de temps. Calculez le rendement de chacun (travail / (matériel * temps))?

Somme 32 bits en 32t avec 6 portes: 1 / (32 * 6) = .0052
Somme 32 bits en 6t avec 320 portes: 1 / (6 * 320) = .00052

Il est facile d'avoir plus de puissance de calcul avec des blocs parallèles mais il faut réussir à exploiter efficacement ce matériel.

Chaque processeur dans un circuit multi-coeur peut exécuter 2G Instructions/seconde avec un taux de succès en mémoire cache de 99.9%, pour des blocs de 64 octets. Quelle est la bande passante requise vers la mémoire pour chaque processeur pour les instructions? Pour 100 processeurs? Un système multi-coeur contient 100 de ces processeurs dans une grille 10x10 avec 3 canaux à 2GO/s partagés par colonne. Le contrôleur de mémoire centralisé peut supporter 50GO/s. Est-ce que cette organisation peut soutenir le volume d'accès mémoire requis pour les fautes de cache d'instructions des 100 processeurs?

2G Instructions/seconde avec un taux de succès 99.9%, blocs de 64 octets, 100 processeurs, 3 canaux à 2GO/s partagés par colonne, contrôleur de mémoire 50GB/s. Ceci donne 2G lectures/s * (1-.999) * 64 octets = 128MO/s ou 1.28GO/s par colonne et 12.8GO/s pour 100 processeurs.

Différents axes de parallélisation

Donnez un exemple de décomposition pour a) une même opération sur des données différentes, b) un même programme sur des données différentes, c) différents programmes sur les mêmes données ou des données différentes?

Différents axes de parallélisation

Donnez un exemple de décomposition pour a) une même opération sur des données différentes, b) un même programme sur des données différentes, c) différents programmes sur les mêmes données ou des données différentes?

Nous pouvons avoir a) un calcul de seuil sur chaque pixel d'une image, b) un calcul de rendu d'image sur différentes scènes d'un film, et c) différents encodages vidéo sur un même film.

Taux de pannes

Votre serveur départemental alimente un laboratoire avec 24 ordinateurs (D=.99) pour 20 équipes. Le serveur est en fait une machine virtuelle qui peut rouler sur l'une ou l'autre de deux machines physiques (D=.999) redondantes, connectées chacune à deux SAN (D=.999) partagés en mirroir. Chaque SAN est composé de 4 disques (D=.99) en RAID 5. Tout ceci est connecté par un réseau (D=.99999) Quelles sont les chances pour au moins une équipe de ne pouvoir travailler? Quel serait l'effet de la fiabilité du logiciel?

Taux de pannes

Serveur: avoir deux défectueux, $.001 \times .001 = .000001$.

Avoir $n/24$ ordinateurs fonctionnels, $\frac{24!}{n! \times (24-n)!} \times 0.99^n \times 0.01^{24-n}$, sommation pour n $= 20$, 21, 22, 23, ou 24: 0.999996373

Avoir n/4 disques fonctionnels, $\frac{4!}{n! \times (4-n)!} \times 0.99^n \times 0.01^{4-n}$,sommation pour n $= 3$ ou 4: 0.9994

SAN: deux défectueux, $(1 - (0.999 \times 0.9994))^2 = 0.000002558$

Réseau défectueux .00001

Probabilités que chaque item fonctionne:

$(1-0.000001) \times 0.999996373 \times (1-0.000002558) \times (1-0.00001) = 0.999982815$

Un problème de grande taille s'exécute en 24h sur un noeud et en 1h sur un ordinateur parallèle de 16 noeuds identiques au premier. Quelle est la fraction parallélisable du programme? Comment expliquez-vous de tels chiffres?

Un problème de grande taille s'exécute en 24h sur un noeud et en 1h sur un ordinateur parallèle de 16 noeuds identiques au premier. Quelle est la fraction parallélisable du programme? Comment expliquez-vous de tels chiffres?

Le problème décomposé entre plus facilement en mémoire centrale ou en mémoire cache et le tout s'exécute plus efficacement, présentant une accélération plus importante que la réduction du problème. Non, la fraction parallélisable n'est pas négative!