Travail pratique #2

IFT-1215

March 23, 2015

¡¡ Dû le 16 avril !!

1 Survol

Ce TP a pour but de vous familiariser avec les effets concrets de la hiérarchie mémoire.

Les étapes sont les suivantes:

1. Lire et comprendre cette donnée.
2. Compiler le code fourni, et l'utiliser pour faire diverses mesures.
3. Écrire un rapport expliquant votre expérience. Il doit contenir une explication de quelles mesures vous avez faites, pourquoi, et comment expliquez les résultats ainsi que décrire **votre** expérience pendant les points précédents: problèmes rencontrés, surprises, choix que vous avez dû faire, options que vous avez sciemment rejetées, etc... Le rapport ne doit pas excéder 8 pages.

Ce travail est à faire en groupes de 2 étudiants. Le rapport doit être écrit en LaTeX (compilable sur `frontal.iro`). Le rapport est à remettre par remise électronique avant la date indiquée. Aucun retard ne sera accepté. Indiquez clairement vos noms au début de chaque fichier.

Si un étudiant préfère travailler seul, libre à lui, mais l'évaluation de son travail n'en tiendra pas compte. Si un étudiant ne trouve pas de partenaire, même après avoir annoncé sa disponibilité dans le forum de discussion, il doit me contacter au plus vite. Des groupes de 3 ou plus sont **exclus**.

2 Localité et temps d'accès a la mémoire

Le travail consiste à mesurer et expliquer le temps d'exécution d'un programme qui parcourt un tableau en mémoire. Le programme prend 3 paramètres

- L'ordre dans lequel le tableau est traversé: il peut être soit séquentiel, soit aléatoire.
- La taille du tableau, en nombre d'éléments.
- Le nombre de fois que le tableau est traversé.

Le nombre de répétitions est utile pour 2 raisons:

- Mesurer le temps nécessaire pour initialiser le tableau (en demandant de traverser le tableau 0 fois).
- Augmenter le temps d'exécution de manière à être mesurable.

La taille du tableau détermine si tout ou seulement une partie du tableau peut être gardé dans un cache. Donc selon la taille du tableau, la performance reflètera plutôt la performance du cache de niveau 1, ou du cache de niveau 2, ou de la mémoire centrale (ou encore d'autres niveau de la hiérarchie mémoire bien sûr).

L'ordre de traversée va déterminer ce que l'on mesure:

- lorsque l'accès est séquentiel, la localité dans l'espace est maximale, l'accès suivant est très prévisible (et même parfois connu avant que l'accès antérieur soit terminé), et les transferts peuvent se faire par grands blocs sans peur de gaspiller du temps à aller chercher des infos qui ne seront pas utilisées. Ce mode permet donc de mesurer la *bande passante* maximale, qui se mesure habituellement en MB/s ou GB/s.
- À l'inverse, dans le mode aléatoire, la localité dans l'espace est très mauvaise, l'accès suivant est complètement imprévisible avant que l'antérieur soit terminé, et toute info transférée en sus du strict minimum est gaspillée. Ce mode permet donc de mesurer plutôt la *latence* qui se mesure habituellement en nanosecondes par accès mémoire.

2.1 TLB

Chaque accès à la mémoire est en fait un double accès:

- un accès à une table qui traduit l'adresse logique de l'objet en une adresse physique.
- l'accès en tant que tel à l'objet en mémoire.

La table de traduction d'adresses travaille par *pages* (de typiquement $4KB$), donc elle est plus petite que la mémoire qu'elle traduit, mais elle peut être quand même assez grande. Elle a donc elle-aussi un cache, qui s'appelle *TLB*. Un TLB de 32 éléments avec des pages de $4KB$ signifie que si l'on utilise moins que $128KB$ de mémoire en même temps, le premier accès se fera rapidement mais que si l'on utilise plus que $128KB$ le premier accès sera parfois ralenti par le besoin d'aller chercher une autre partie de la table en mémoire. Tout comme le cache de la mémoire, le TLB peut avoir plusieurs niveaux.

Pour un accès séquentiel, le TLB n'a pas d'effet significatif sur la performance (vu qu'il n'ajoute au pire qu'un accès mémoire supplémentaire pour chaque page transférée), par contre dans le mode d'accès aléatoire il peut avoir un impact important. Par exemple, dans un système avec un L1-TLB très rapide de 12 éléments et un L2-TLB plus lent de 1024 éléments, la latence tendrait à augmenter lorsque le tableau traversé dépasse $48KB$, puis une nouvelle fois lorsque que l'on passe au-delà de $4MB$.

3 Exemples de session

Voici un exemple où l'on compile le code fourni puis on l'exécute de manière à obtenir son temps d'exécution (les lignes qui commencent par `%` montrent les commandes envoyées, les autres montrent les messages reçus en réponse):

```
% make
cc -Wall -O2    cache-test.c   -o cache-test
% time ./cache-test --rand 100000 1000
Looped randomly 1000 times through an array of 100000 elements of 4 bytes
./cache-test --rand 100000 1000  1.74s user 0.00s system 99% cpu 1.746 total
%
```

Et voici un autre exemple qui montre comment trouver le type de processeur utilisé (ainsi que diverses autres infos) sur une machine de type GNU/Linux:

```
% cat /proc/cpuinfo
...
model name      : AMD E-350 Processor
...
cache size      : 512 KB
...
```

```
TLB size        : 1024 4K pages
...
%
```

4 Travail

Le travail consiste à utiliser le programme fourni pour mesurer la performance de la hiérarchie mémoire d'une machine dans différentes circonstances, et ensuite expliquer les résultats en terme de performance des différents niveaux de la hiérarchie mémoire: taille, brande passante, latence de chaque niveau mesuré (i.e. chaque niveau de cache ainsi que la mémoire principale), ainsi que si ce niveau est partagé entre les différents core ou pas. Si vous voulez, vous pouvez aussi essayer de pousser ce test jusqu'au niveau supérieur (la mémoire virtuelle, sur le disque), mais soyez averti que cela peut rendre la machine tellement lente qu'elle devient inutilisable pendant un long moment.

Faites ce travail sur 2 machines différentes et qui ont des CPUs différents (idéalement de constructeurs différents).

Ces machines doivent avoir plusieurs processeurs (ou plusieurs "core"s), et il vous faudra comparer aussi le temps d'exécution selon le nombre de copies de notre programme de test exécutées en même temps, ce qui devrait vous permettre de déterminer quels niveaux de cache sont exclusifs à chaque "core", et lesquels sont au contraire partagés.

Comparez vos résultats aux informations officielles du fabricant: certaines de ces informations peuvent être relativement difficiles à trouver, mais trouvez au moins la taille de chaque niveau de cache.

4.1 Mesures

Le temps d'exécution d'un programme dépend de beaucoup de facteurs parfois difficiles à contrôler. Essayez de vous assurer (e.g. avec la commande `top`) que la machine n'est pas déjà occupée à faire autre chose. Faites vos mesures plusieurs fois et prenez-en le minimum, la médiane, ou la moyenne des résultats, selon votre préférence, sans oublier de l'indiquer dans votre rapport.

5 Remise

Il faudra remettre électroniquement, via la page Moodle (aussi connu sous le nom de StudiUM) du cours, un fichier `rapport.tex`, qui doit pouvoir se compiler sans soucis avec LaTeX sur `frontal`. Indiquez clairement vos noms au début de chaque fichier.

6 Barème

Les 22 points en jeux seront divisés comme suit:

- 4 points pour les mesures sur chaque machine.
- 4 points pour les explications des résultats sur chaque machine.
- 6 points pour le reste du rapport.