**TD 7**

# *Caches*

***2 :1 Cache Rule :*** *The miss rate of a direct-mapped cache of size $N$ is about the same as that of a two-way set-associative cache of size $N/2$.*
(Hennessy-Patterson, CAQA 1ère édition, page 0)

## 1 Questions de cours

Rappeler le(s) principe(s) de localité.

Rappeler le principe d'un cache, d'un cache completement associatif *(fully associative)*, d'un cache à correspondance directe *(fully mapped)*, d'un cache associatif par ensemble à $n$ voies *(n-way set-associative)*.

Expliquer la citation poétique ci-dessus.

Quels sont les avantages et inconvénients des caches séparés programmes/données par rapport aux caches unifiés ?

Pouvez-vous estimer le pourcentage d'opérations mémoires qui sont des écritures ? Expliquez les stratégies dans ce cas : écriture directe *(write through)*, écriture différée *(write back)*, tampon d'écriture ?

Dessiner les blocs fonctionnels pour un cache à correspondance directe de 2K mots pour un processeur 64 bits, avec des lignes de cache de 8 mots.

Modifiez le dessin précedent pour en faire un cache associatif par ensembles à deux voies.

Discuter son insertion dans un pipeline.

Discuter son insertion dans le processeur superscalaire du TD précédent.

## 2 Optimisation

Dans la suite on se rappellera que

$$T_{\text{moyen d'accès mémoire}} = T_{\text{hit}} + R_{\text{miss rate}} \times T_{\text{miss}}$$

et on évaluera l'impact des différentes optimisations possibles sur les trois termes $T_{\text{hit}}$, $R_{\text{miss rate}}$ et $T_{\text{miss}}$.

## 2.1 Introduction

Les trois types possibles de caches représentent trois compromis entre ces trois termes. Lesquels ?

Quels sont les avantages et inconvénients des caches séparés programmes/données par rapport aux caches unifiés (je l'ai déjà demandé mais les réponses n'étaient pas très précises) ?

## 2.2 Comment diminuer le pourcentage de défauts de caches

Il y a trois catégories de *miss* :
– Les obligatoires (de «démarrage à froid»).
– Ceux dûs à la capacité du cache.
– Ceux dûs à des conflits, si le cache n'est pas pleinement associatif.

Discuter les techniques suivantes d'optimisation du $R_{\text{miss rate}}$. On regardera quel type de défaut de cache ils minimisent, mais aussi leur coût en matériel et leur impact sur les deux autres paramètres $T_{\text{hit}}$ et $T_{\text{miss}}$.

### 2.2.1 Augmentation de la taille de la ligne de cache

(à taille de cache constante, évidemment).

### 2.2.2 Augmentation de l'associativité

Le débat est réglé par deux résultats empiriques importants : celui qui est présenté en en-tête, et un autre, qui dit qu'un cache complètement associatif est à peine meilleur qu'un cache associatif par ensemble à 8 voies.

### 2.2.3 Cache des victimes

L'idée est d'ajouter un petit cache de quelques lignes, qui récupère une copie des dernières lignes virées du cache. Quel intérêt ?

### 2.2.4 Caches pseudo-associatifs

Comment obtenir le $R_{\text{miss rate}}$ d'un cache associatif par ensembles et le $T_{\text{hit}}$ d'un cache à correspondance directe ?

### 2.2.5 Préchargement (prefetch) matériel

Tiens, si la logique de remplissage de cache n'a rien de mieux à faire, elle peut charger les lignes de cache qui suivent la dernière ligne chargée...

### 2.2.6 Instructions de préchargement

... à disposition du compilateur ou du hackeur ...

### 2.2.7 Optimisations purement logicielles (compilateur ou programmeur)

Pour le cache instruction : déplacement de procédures, fusion de boucles, éclatement de boucles, déroulement de boucle en tenant compte de la taille du cache...

Pour le cache données : fusion (alignement) de tableaux, permutation de boucles, découpage de matrices en blocs (c'est de l'algèbre, pas seulement de la réecriture de code), ...

Inconvénient : ce type de manipulation est peu portable d'une version du processeur à l'autre.

## 2.3 Réduction de la pénalité en cas de défaut de cache

### 2.3.1 Donner la priorité aux défauts de lecture sur les défauts d'écriture

Attention au maintien de la cohérence.

### 2.3.2 Caches non bloquants dans les superscalaires

### 2.3.3 Critical word first

Le processeur a demandé une adresse, on peut essayer de remplir la ligne de cache en commençant par celle-là.

### 2.3.4 Caches de second niveau

Il paraît qu'il existe même des hiérarchies mémoire à trois niveau.

## 2.4 Réduction du temps d'accès en cas de requête réussie

### 2.4.1 Faire simple et petit

Lorsqu'on fait grossir le cache ou son associativité, il faut marchander l'amélioration de $R_{\text{miss rate}}$ et la dégradation de $T_{\text{hit}}$.

Dès lors qu'on a deux niveaux de caches, le marchandage peut devenir plus facile.

### 2.4.2 Positionnement de la TLB

La TLB fait la traduction des adresses virtuelles (disons, sur 64 bits) en adresses physiques (30-40 bits). S'il te plaît, dessine moi une TLB.

On peut placer la TLB entre cache et processeur (cache en adresses réelles) ou bien entre le cache et la mémoire principale (cache en adresses virtuelles). Avantages et inconvénients comparés ?

### Et pour conclure, une anecdote

Lorsqu'ARM a voulu placer ses processeurs sur le marché de Windows CE, ils ont du doubler la taille de tous les caches, qui jusque là offraient des performances suffisantes dans les applications embarquées (téléphones mobiles) et pour d'autres systèmes d'exploitation (EPOC32, RISC OS).

Quelle est la triste morale de cette histoire ?

S'en inspirer pour discuter l'impact sur la hiérarchie mémoire de la tendance actuelle des OS vers les processus légers *(threads)* et les machines virtuelles.