Compte-rendu TP n°4, MULTI, 2019-2020

Aymeric Agon-Rambosson

Dimanche 23 février 2020

Cette semaine, on se propose de faire varier les caractéristiques des caches, pour en déduire par analyse numérique les caractéristiques optimales.

Système mémoire presque parfait

Question C1

On se propose, pour mesurer le temps d'exécution, de regarder le contenu du pseudo-registre c_total_cycles qui est gentiment affiché dans le pseudo-tty à la fin du calcul.

Avec 256 sets, 16 mots par ligne, un degré d'associativité de 4, et un tampon d'écritures postées d'une profondeur de 8 mots, le programme se termine à la fin du cycle 74502.

Question C2

Maintenant qu'on connaît le nombre de cycles nécessaires pour terminer le calcul, on peut demander 75000 cycles, et une période d'échantillonnage des statistiques de 1000, de manière à regarder dans un premier temps les taux de miss cumulatifs et le CPI sur la totalité de la période.

Sur le calcul considéré dans sa totalité, on a un taux de miss sur le cache d'instructions de 0,06 %, un taux de miss sur le cache de données de 0,1 %, et un CPI de 1,32.

Si on regarde l'évolution des taux de miss cumulatifs sur la totalité du calcul (des cycles 1 à 75000), on voit qu'ils sont décroissants pour les instructions et les données. La décroissance est très rapide au début, puis ralentit à partir du 10000ème cycle. Le CPI suit la même tendance.

Si on regarde en particulier les 1000 premiers cycles, on a une décroissance en tendance de même, avec une légère remontée autour des cycles 200 et 400 pour le taux de miss de données (et donc pour le CPI). Les trois grandeurs se stabilisent à partir des cycles 700-800.

La décroissance rapide des taux de miss est dû au miss compulsifs qui sont par définition concentrés au début de l'exécution des programmes : dès lors que les instructions sont chargées dans le cache, on fait très peu de miss, surtout si le cache est presque parfait : peu de miss de conflit, et peu de miss de capacité.

La remontée du taux de miss sur les caches de données est probablement dûe à un changement de contexte (passage du système à l'applicatif, par exemple).

On trouvera en annexe deux graphiques.

Question C3

Voilà la section du code qui décrit la définition des statistiques, dans le fichier pibus_mips32_xcache.cpp.

```
void PibusMips32Xcache::printStatistics()
{
  std::cout << "*** " << name() << " at cycle " << std::dec <<
      c_total_cycles << std::endl;
  std::cout << "- INSTRUCTIONS       = " << c_total_inst << std::endl ;
  std::cout << "- CPI                = " <<
      (float)c_total_cycles/c_total_inst << std::endl ;
  std::cout << "- CACHED READ RATE   = " <<
      (float)(c_dread_count-c_dunc_count)/c_total_inst << std::endl;
  std::cout << "- UNCACHED READ RATE = " <<
      (float)c_dunc_count/c_total_inst << std::endl ;
  std::cout << "- WRITE RATE         = " <<
      (float)c_write_count/c_total_inst << std::endl;
  std::cout << "- IMISS RATE         = " <<
      (float)c_imiss_count/c_total_inst << std::endl;
  std::cout << "- DMISS RATE         = " <<
      (float)c_dmiss_count/(c_dread_count-c_dunc_count) << std::endl;
  std::cout << "- IMISS COST         = " <<
      (float)c_imiss_frz/c_imiss_count << std::endl;
  std::cout << "- DMISS COST         = " <<
      (float)c_dmiss_frz/c_dmiss_count << std::endl;
  std::cout << "- UNC COST           = " <<
      (float)c_dunc_frz/c_dunc_count << std::endl;
  std::cout << "- WRITE COST         = " <<
      (float)c_write_frz/c_write_count << std::endl;
}
```

À chaque cycle, est gardé dans un pseudo-registre (c_total_instructions) le nombre total d'instructions exécutées depuis le démarrage de la machine, simplement incrémenté à chaque fois que :

- une requête d'instructions est valide
- la réponse est valide
- l'adresse demandée n'est pas la même qu'à l'instruction précédente

Voyons plutôt :

```
if (m_ireq.valid && m_irsp.valid && (m_ireq.addr != r_icache_save_addr.read()))
  c_total_inst++;
```

Dans un autre pseudo-registre, c_total_cycles, on garde le nombre de cycles passés depuis le démarrage de la machine, simplement incrémenté à chaque cycle.

Dans un autre pseudo-registre, c_imiss_count, on garde le nombre de miss d'instructions depuis le démarrage de la machine. Ce compteur est incrémenté à chaque miss d'instructions.

Dans un autre pseudo-registre, c_imiss_frz, on garde le nombre de cycles passés à attendre en raison d'un miss d'instructions. Ce compteur est incrémenté à chaque cycle passé à attendre une inscription.

Dans un autre pseudo-registre, c_dmiss_count, on garde le nombre de miss de données depuis le démarrage de la machine. Ce compteur est incrémenté à chaque miss de données.

Dans un autre pseudo-registre, c_dread_count, on garde le nombre d'instructions qui sont des lectures. Ce compteur est incrémenté à chaque fois que l'opcode de l'instruction assembleur correspond à lw, lb, ou lh.

Dans un autre pseudo-registre, c_dunc_count, on garde le nombre d'instructions de lecture vers des adresses non cachables.

Avec tous ces registres, on a tout ce qu'il faut pour définir correctement toutes les statistiques qui nous intéressent :

Le CPI cumulatif est le quotient du nombre total de cycles par le nombre total d'instructions.
Le taux de miss d'instructions est le quotient du nombre total de miss d'instructions par le nombre total

d'instructions.
Le coût d'un miss d'instructions est le quotient du nombre total de cycles de gel passés à attendre une instruction par le nombre total de cycles.
Le taux de miss de données est le quotient du nombre total de miss de données par le nombre d'instructions de lecture.
Le coût d'un miss de données est le quotient du nombre total de cycles de gel passés à attendre une donnée pour lecture par le nombre total d'instructions de lecture vers des adresses non cachables.

Les statistiques affichées par le programme simul.x correspondent bien à ces grandeurs, cf. le code source plus haut.

Influence de la capacité du cache d'instructions

Question D1

Faisons le résumé des statistiques demandées :

Nombre de sets	Durée	Taux de miss	Coût du miss	CPI
256	85956	1.48 %	15.3	1.54
64	106270	3.88 %	15.7	1.95
16	152943	10.01 %	15.6	2.97
4	166455	12.04 %	15.5	3.29
1	234079	23.50 %	15.2	5.11

Il est tout à fait logique que le taux de miss et le CPI se dégradent, conduisant à des temps d'exécution plus longs, avec la diminution du nombre de cases. Tout simplement parce qu'en diminuant le nombre de cases, et en laissant les autres paramètres inchangés, on diminue la taille du cache : les miss de conflits et de capacité seront beaucoup plus fréquents : les lignes se battront pour un plus petit nombre de cases, et le cache ne pourra tenir qu'un plus petit nombre d'instructions à la fois : les localités spatiale et temporelle seront moins bien exploitées.

Le coût du miss reste constant en la taille du cache, parce qu'il dépend d'autre chose : il dépend des caractéristiques des matériels qui passent avant lui sur le bus, en particulier de la profondeur du tampon d'écritures postées.

Question D2

Comme on vient de le mentionner, le coût du miss d'instructions dépend des caractéristiques des matériels qui passent avant lui sur le bus, au premier rang desquels le tampon d'écriture postées, et aussi de la fréquence à laquelle le programme fait accéder à ce tampon d'écritures postées.

Le tampon d'écritures postées a les mêmes caractéristiques que la semaine dernière (profondeur de 8), donc ce n'est pas la cause ici. En revanche, on n'exécute pas du tout le même programme que la semaine dernière, celui de cette semaine fait beaucoup plus d'entrées-sorties (des écritures dans le pseudo-terminal, en particulier). Le tampon d'écritures postées sera beaucoup plus souvent non-vide, donc va occuper le bus de manière prioritaire plus souvent, augmentant le coût des miss pour ceux qui passent après lui.

Le coût du miss peut être représenté par une variable aléatoire : il dépend de l'état de remplissage du tampon d'écritures postées au moment où le miss se produit. On ne peut pas donner une valeur a priori du coût du miss, celui-ci sera variable.

Le coût ici donné est une moyenne des coûts des miss constatés **dans ce programme bien précis**. Une moyenne de valeurs entières peut parfaitement être une valeur non entière.

Influence de la largeur de la ligne de cache

Nombre de sets	Taille du set	Durée
256	1	193355
128	2	148381
64	4	130274
32	8	124855
16	16	129468
8	32	149237

La configuration la plus efficace, c'est 32 sets de 8 mots, soit 32 sets de 32 octets chacun.

Si on augmente la taille de la ligne, on espère faire diminuer le taux de miss, mais on augmente le coût d'un miss : on doit transmettre plus de choses sur le bus.
Si on diminue la taille de la ligne, on espère faire diminuer le coût d'un miss, mais on augmente le taux de miss.

Le coût effectif du miss en nombre de cycles étant le produit du taux et du coût par miss, il est logique, par raison mathématique (on parle d'une fonction quadratique), que la configuration optimale sera sur les valeurs médianes de l'un et de l'autre.

On trouvera le graphique qui montre les durées en cycle en fonction de la largeur en annexe.

Influence de la capacité du cache de données

Nombre de sets	Durée	Taux miss données	Coût miss données	CPI
256	74553	0.2 %	17.7	1.32
64	74638	0.2 %	17.8	1.32
16	94444	7.1 %	18.4	1.68
4	144880	22.3 %	16.77	2.56
1	218474	38.0 %	15.8	3.86

Encore une fois, des résultats très logiques : diminuer le nombre de cases de cache de données augmente mécaniquement le taux des miss de conflit et de capacité : les lignes se battent pour un moindre nombre de cases, les localités spatiales et temporelles des données sont moins bien exploitées.

Influence de la profondeur du tampon d'écritures postées

Question G1

Le tampon d'écritures postées est un canal de communication FIFO à N cases implémenté en matériel. Il n'y a que deux opérations possibles :

- poser une donnée à une extrémité, opération accessible uniquement au composant "WBUF_FSM" (qu'on a pas encore vu, mais qu'on sait exister).
- prendre une donnée à l'autre extrémité, opération accessible uniquement au composant PIBUS_FSM.

Il n'est en particulier, dans cette configuration du cache, pas possible de consulter une donnée en plein milieu de la FIFO.

Dans le tampon, on doit stocker :

- un bit de validité
- 30 bits d'adresse
- 4 bits de byte enable
- 32 bits de données (un mot)

Si le processeur fait une requête d'écriture alors que le tampon d'écritures postées est plein, le processeur se gèle.

Si le processeur fait une requête de lecture qui fait miss alors que le tampon d'écritures postées est non-vide, le processeur se gèle jusqu'à ce que son cache lui donne la donnée, ce qui suppose que le cache (d'instructions ou de données) obtienne le bus, ce qui suppose que le tampon d'écritures postées se soit vidé au préalable.

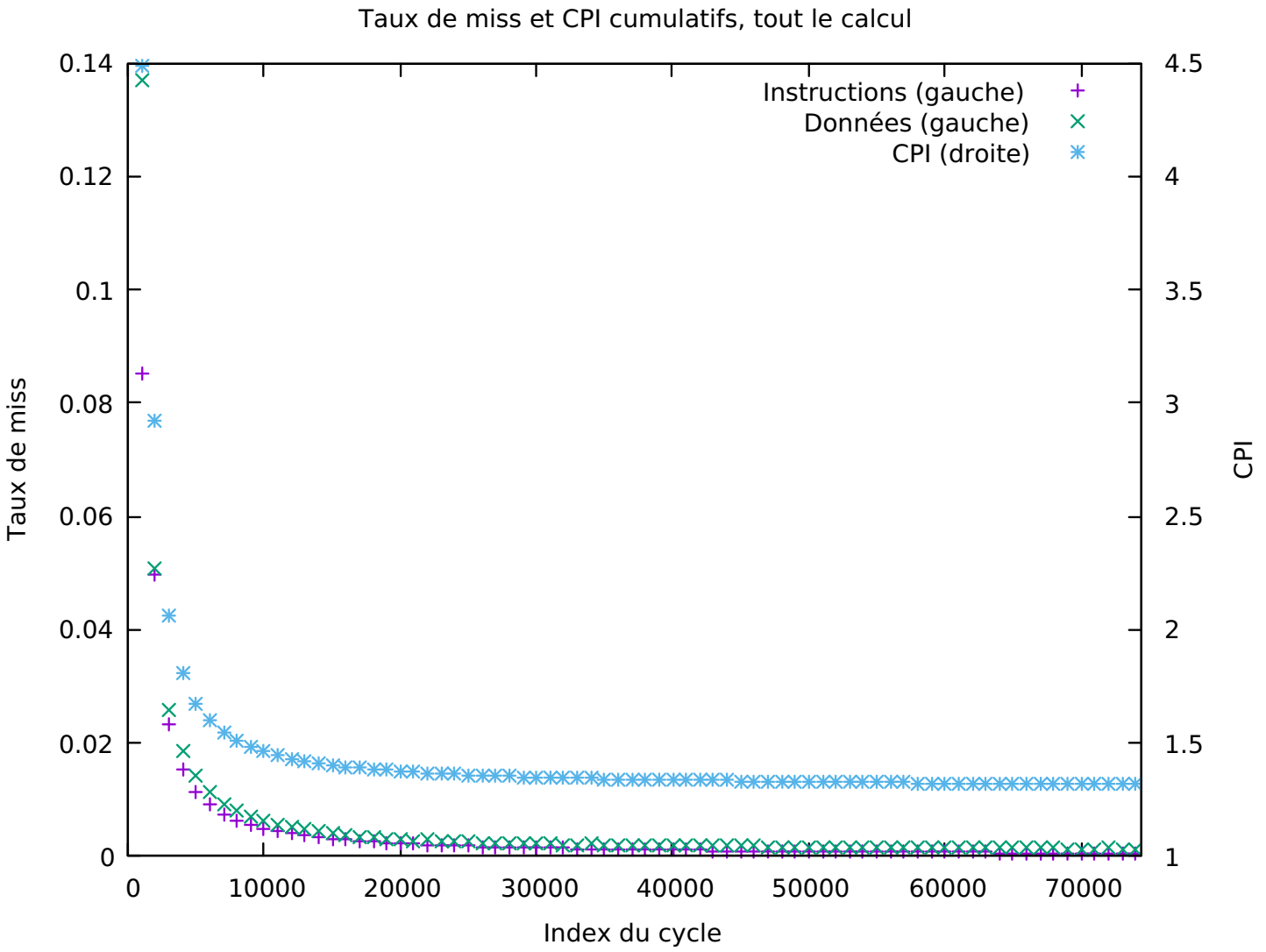
On a ce comportement parce que les écritures sont prioritaires, et les écritures sont prioritaires pour assurer la consistance mémoire.

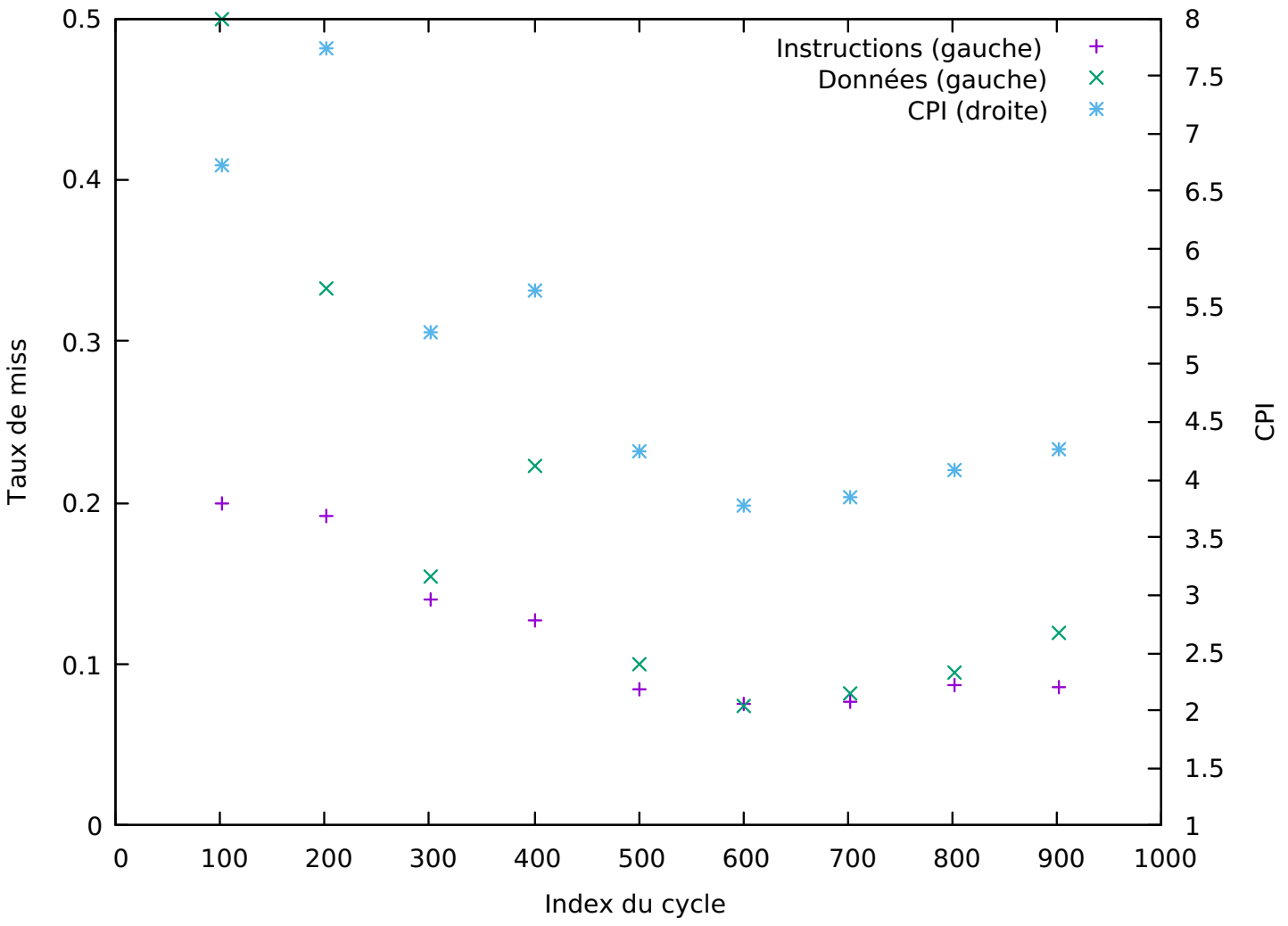
Question G2

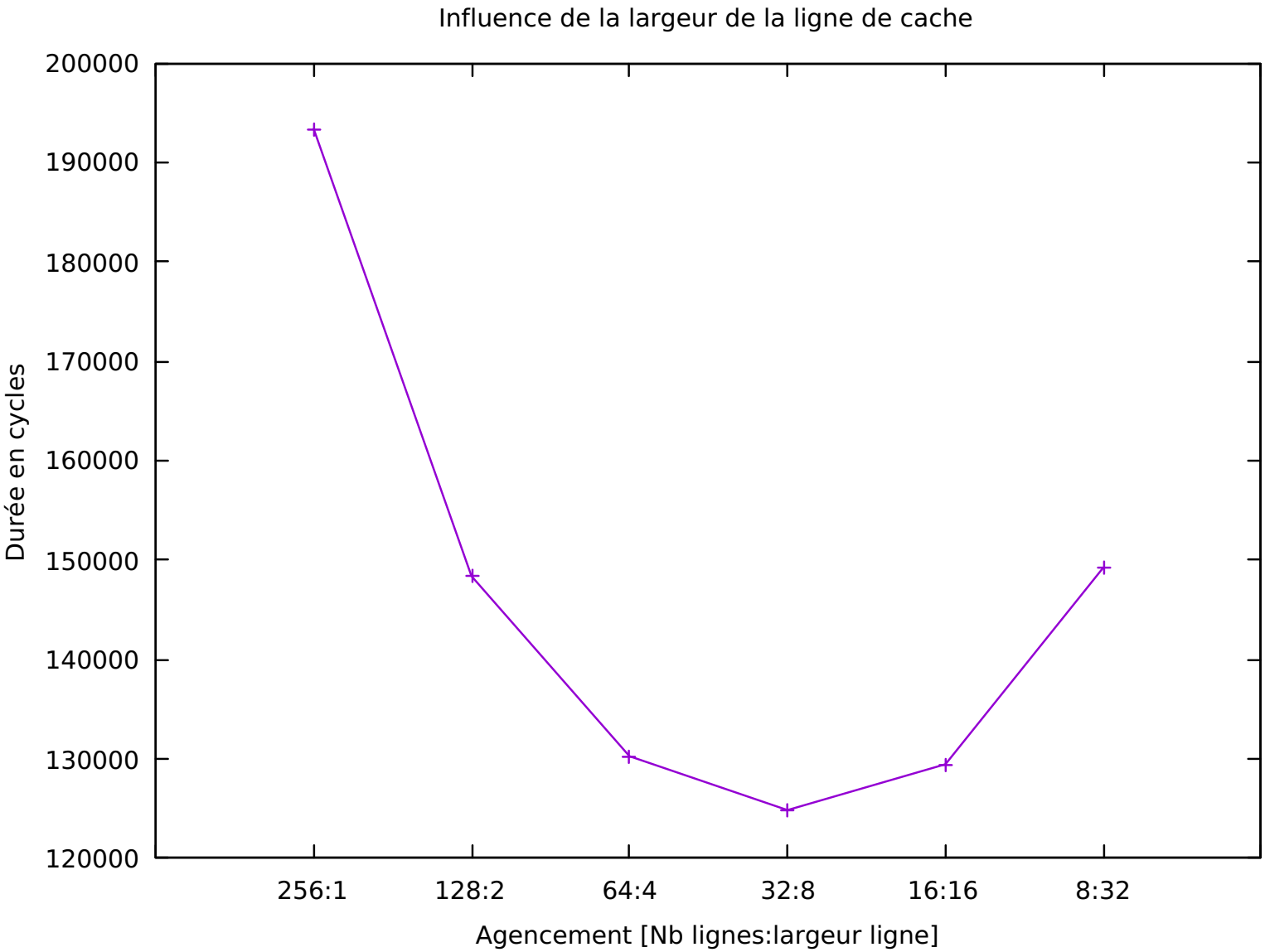
WBUF	Durée	CPI	Coût écriture	Fréquence écritures
1	77624	1.37	0.51	12.5 %
2	75396	1.33	0.13	12.5 %
4	74502	1.32	0	12.5 %
8	74502	1.32	0	12.5 %

Le coût des écritures, c'est le nombre de cycles de gel moyen occasionnés par une écriture.

Ce coût est très faible, parce que pour occasionner un cycle de gel, il faut remplir le tampon. Puisque la vidange du tampon est prioritaire par rapport aux lectures, le tampon d'écritures postés n'occasionne de cycle de gel que si on a un peu plus de n écritures consécutives avec n la taille du tampon. La probabilité d'avoir ces n écritures consécutives est en fait très faible (rappelons que en moyenne 10 % des instructions sont des instructions d'écriture), et diminue encore avec n augmentant.


Taux de miss et CPI cumulatifs, tout le calcul
Instructions (gauche)
Données (gauche)
CPI (droite)
Taux de miss
CPI
Index du cycle
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
1
1.5
2
2.5
3
3.5
4
4.5
0
10000
20000
30000
40000
50000
60000
70000


Taux de miss et CPI cumulatifs, 1000 premiers cycles
Instructions (gauche)
Données (gauche)
CPI (droite)
Taux de miss
CPI
Index du cycle
0
0.1
0.2
0.3
0.4
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
0
100
200
300
400
500
600
700
800
900
1000


Influence de la largeur de la ligne de cache
Durée en cycles
200000
190000
180000
170000
160000
150000
140000
130000
120000
256:1
128:2
64:4
32:8
16:16
8:32
Agencement [Nb lignes:largeur ligne]