Travaux Pratiques Archi SEOC-3A - 2021/2022
Sujet de Frédéric Pétrot et Arthur Perais

Implantations et évaluations de quelques prédicteurs de branchements

Durée : 4 heures

1 Organisation

Ce TP se fait soit seul, soit en binôme, à votre convenance. Le travail demandé TP fait l'objet d'un rendu contenant un court rapport qui inclut les *résultats* d'expérimentations et un *tarball* (ou un git) contenant vos sources à fournir en fin de séance. Un gabarit en LaTeX qui reprend le predicteur BHT avec un compteur à saturation est donné en exemple. Vous avez également un script fourni pour vous permettre de systématiser les simulations et l'obtention des graphes des résultats. Ce script marche avec le prédicteur BHT qui a deux arguments, il est donc nécessaire de le modifier pour utiliser un prédicteur avec un nombre d'arguments différents. Le sous-répetoire `sle3a/rendu` contient un exemple de .tex permettant de faire un petit rapport sur le travail.

2 Introduction

On cherche à étudier le comportement en situation réelle de différents prédicteurs de branchement, du plus simple au plus complexe. Pour cela, on va développer un modèle en C++ de chacun de ces prédicteurs sur lequel on fera passer un ensemble de traces d'exécutions, nous permettant d'obtenir en sortie le ratio de mauvaises prédictions (en fait le nombre de *Miss Prediction per Kilo Instructions*, MPKI). On fera varier les différents paramètres du prédicteur, en particulier la taille des tables, afin d'en extraire un comportement asymptotique (par benchmark), dont on tirera un graphe.

L'infrastructure dans laquelle insérer le modèle de prédicteur est celle qui a été utilisée par le *The 5th JILP Championship Branch Prediction Competition (CBP-5)*[1]. Cette infrastructure effectue la lecture des traces et l'appel à des fonctions qui implantent le prédicteur, et donne en sortie diverses statistiques dont les MPKI.

Le prédicteur par défaut dans l'archive fournie est un prédicteur bimodal à saturation « n-bits » vu en cours et rappelé en Figure 1 pour le cas 2-bit. L'automate régissant chaque compteur est rappelé en Figure 3.

Dans une case donnée, un branchement « prédit pris » aura une valeur $v \geq 2^{n-1}$ et « prédit non pris » aura une valeur $v < 2^{n-1}$ (on note qu'il s'agit d'une convention, on pourrait très bien faire l'inverse). Nous ferons en séance une analyse rapide du code afin de comprendre comment il a été implanté dans l'infrastructure.

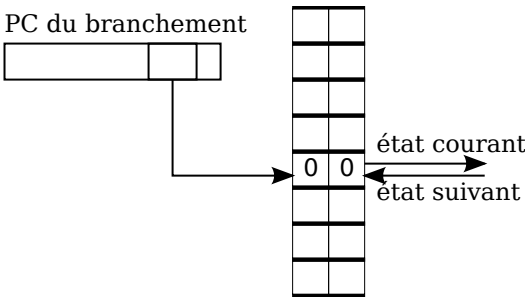


Figure 1 – Prédicteur bimodal

3 Travail demandé – Prédicteurs simples

On implantera différents prédicteurs en prenant soin de paramétrer les tailles afin de pouvoir lancer facilement plusieurs exécutions (*cf.* l'exemple fourni) et ainsi pouvoir tracer des courbes et voir les asymptotes. **Important :** Les figures montrent souvent des prédicteurs de petites tailles, mais vous êtes invités à considérer des prédicteurs commençant au moins à 128 entrées (log2(128) = 7).

Les prédicteurs que vous devrez implanter sont les suivants :

1. Dont on trouvera la version originale sur `https://www.jilp.org/cbp2016/framework.html`

1. un prédicteur global simple, montré en Figure 2 avec un historique global de branchement de longueur (nombre de bits) H permettant d'atteindre une entrée bimodale contenant un compteur 2-bit utilisant le graphe d'état donné en Figure 3. Dans le cas du prédicteur global simple, la structure contenant les prédictions est appelée *Pattern History Table* (comme le premier niveau du 2-level adaptive).

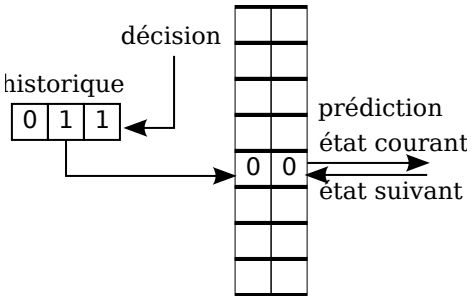


Figure 2 – Prédicteur global simple

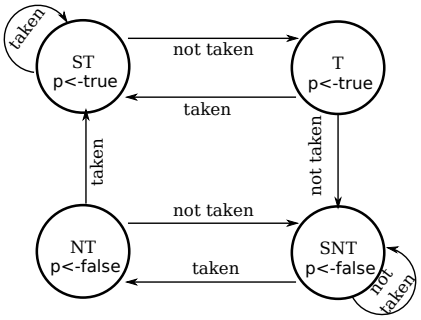


Figure 3 – Automate d'un compteur 2-bit bimodal

L'historique global des branchements est un registre à décalage à gauche dans lequel on injecte sur le poids faible la décision prise ($H = 3$ dans la figure 2).

2. un prédicteur *gshare* qui utilise un historique global sur H bits qui est combiné à l'adresse du branchement à prédire via un XOR, comme illustré sur la Figure 4. On pourra tenter de trouver une fonction de hachage plus performante qu'un simple XOR ;

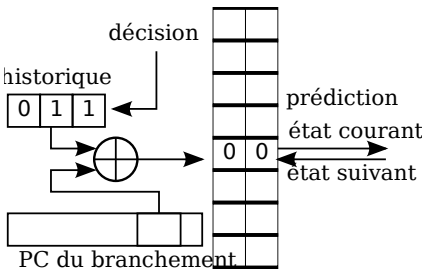


Figure 4 – Prédicteur gshare

3. un prédicteur local qui utilise une table d'historiques sur H bits (la *Pattern History Table*) indexée par les poids faibles de l'adresse du branchement. En utilisant l'entrée d'historique, on indice une BHT bimodale avec compteurs 2-bit de taille 2^H, comme illustré sur la figure 5 ;
4. une combinaison de deux prédicteurs et le métaprédicteur associé (celui de l'alpha 21264) qui utilise un prédicteur global simple (pas gshare !) avec un historique de $H_g = 12$ (soit 4096 entrées), et un prédicteur local comme celui de la précédente question, avec $H_l = 10$ par entrée de la PHT (soit 1024 entrées). La BHT indexée est un simple prédicteur bimodal, mais qui utilise des compteurs **3 bits**.

   Le choix du prédicteur à utiliser se fait grâce au métaprédicteur qui comme les BHT des prédicteurs global et local contient 4K entrées de compteurs 2 bits. On accède au compteur en utilisant l'adresse du branchement à prédire,

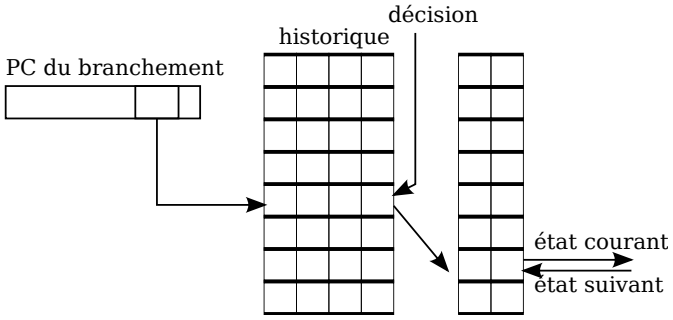


FIGURE 5 – Prédicteur local

et ce compteur nous dit simplement quel prédicteur du prédicteur global ou du prédicteur local va effectuer la prédiction. Le compteur est incrémenté lorsque le prédicteur *prédit* est correcte et l'autre prédicteur a fait le mauvais choix, et est décrémenté dans le cas opposé (*cf.* la machine d'état de la figure 6).

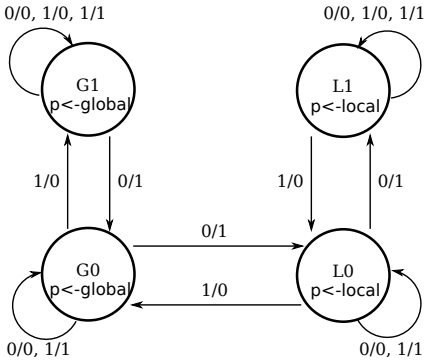


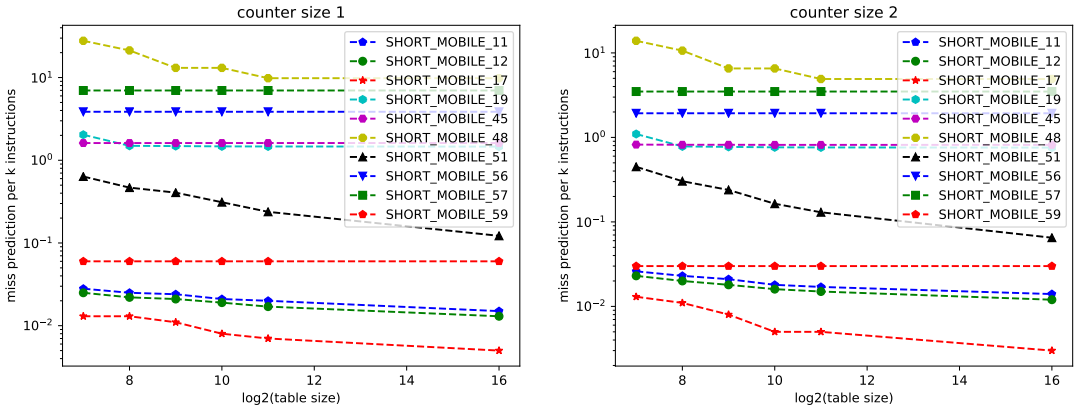
FIGURE 6 – Machine à états du choix du prédicteur. Légende : $global/local$, 0 décision incorrecte, 1 décision correcte

On ne tentera pas (du moins dans le temps imparti) de faire une version paramétrable de ce dernier prédicteur.

Il y a un script `sle3a-doit.sh` dans le répertoire `script` qui lance automatiquement l'exécution sur un sous-ensemble des traces pour un prédicteur donné, qu'il a fallut préalablement compiler dans le répertoire `sim`. Il y a dans le script pour le prédicteur n-modal 2 boucles imbriquées : la boucle externe, indice i, fait varier le nombre de bits du compteur, et la boucle interne, d'indice j, fait varier le nombre de bits de PC à utiliser pour indexer les tables. Vous serez amené à modifier les valeurs des bornes en fonction des prédicteurs, voir à changer un peu cette partie si vous faites des prédicteurs exotiques.

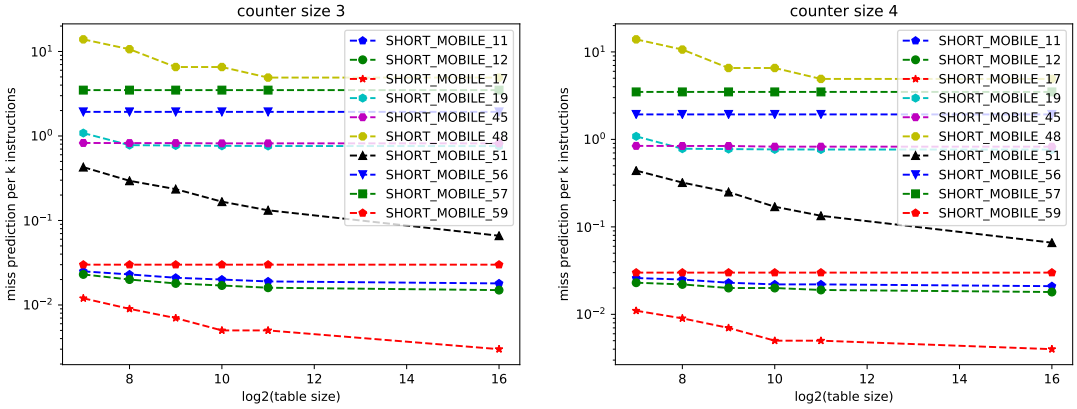
Le script `sle3a-doit.sh` produit ses résultats dans `../results/xxx`, ou `xxx` est un répertoire dont le nom peut-être choisi à l'envie. Il les analyse ensuite et génère des programmes `python` qui servent à tracer des courbes (avec `matplotlib`). Ces courbes sont crées dans le répertoire où on appelle `sle3a-doit.sh` et ce sont ces courbes que l'on vous demande d'analyser.

On donne sur la figure 7 le type de résultat attendu pour le prédicteur n-bit.



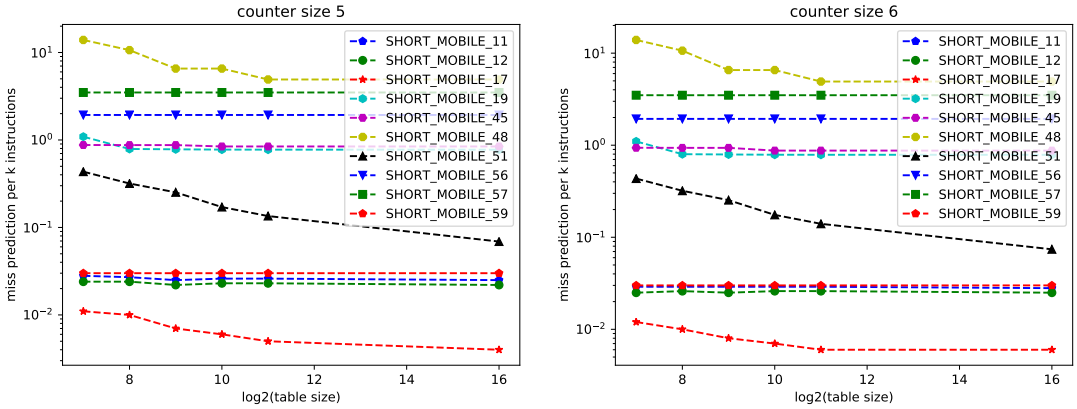
(a) 1 bit par compteur

(b) 2 bits par compteur



(c) 3 bits par compteur

(d) 4 bits par compteur



(e) 5 bits par compteur

(f) 6 bits par compteur

FIGURE 7 – Les courbes de MPKI pour le prédicteur bimodal simple, en fonction du nombre de bits par compteur et du nombre d'entrées.

4 Travail demandé – Aller plus loin

La deuxième partie de ce TP est plus propective, car il s'agit d'améliorer la performance des prédicteurs simples implémentés jusqu'ici. On se propose de suivre deux chemins possibles.

1. Choisir un des prédicteurs de la première partie et chercher à minimiser les MPKI par tous les moyens possibles : taille des tables, fonction de hachage, associativité, combinaisons de prédicteurs, type d'historique (dans la mesure de ce que permet le simulateur), tout est permis. Ici, on vous encourage à analyser les traces, par exemple en construisant des statistiques sur chaque branchement dans la fonction *UpdatePredictor()*, afin de tenter de déterminer pourquoi un branchement est mal prédit alors qu'il a l'air régulier ou corrélé à un voisin. Dans votre rapport, il conviendra d'expliquer les techniques utilisées et vos analyses quand à ce qui fonctionne le mieux.
2. Implémenter un prédicteur moderne tel que le prédicteur neural à base de perceptrons de Jiménez et Lin, à partir du papier de recherche éponyme (Lien, et en comparer la performance avec les prédicteurs de la première partie. A noter que pour implémenter le prédicteur, il n'est pas nécessaire de lire tout le papier de recherche. La Figure 3 ainsi que les Sections 3.2, 3.3, 3.5 et 4 sont suffisantes. Dans votre rapport, il conviendra de décrire votre implémentation afin que l'on puisse s'assurer que vous avez bien compris le papier.