Travaux Pratiques Archi SLE-3A
Cours de Frédéric Pétrot

Implantations et évaluations de quelques prédicteurs de branchements

Durée : 3 heures

1 Organisation

Attention, le travail demandé dans ce TP doit faire l'objet d'un court rapport qui inclut les *sources* des prédicteurs et les *résultats* d'expérimentations à fournir en fin de séance. Un gabarit en LaTeX qui reprend le predicteur 1-bit donné en exemples et quelques scripts sont fournis pour vous permettre de le faire vite, il est néanmoins nécessaire de réserver au moins 20 minutes pour consolider vos écrits. Vous avez intérêt à ajouter vos résultats au fur et à mesure.

2 Introduction

On cherche à étudier le comportement en situation réelle de différents prédicteurs. 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 et qui fournira en sortie le ratio de mauvaises prédictions (en fait le nombre de *Miss Prediction per Kilo (Branch) 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 2nd JILP Championship Branch Prediction Competition (CBP-2)* [1]. Cette infrastructure effectue la lecture des traces et l'appel à des fonctions qui implantent le prédicteur, et donne en sortie uniquement la valeur du MPKI.

Le prédicteur par défaut dans l'archive est le prédicteur « 1-bit » vu en cours et rappelé figure 1. Nous ferons une analyse rapide en séance afin de comprendre comment implanter un prédicteur dans l'infrastructure. Un

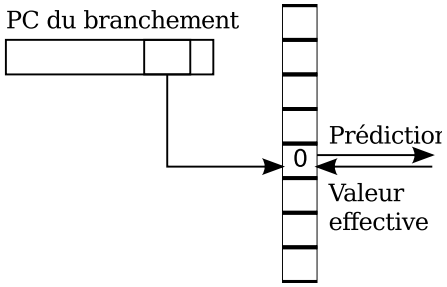


FIGURE 1 – Prédicteur 1-bit

branchement pris (ou prédit pris) sera à `true` (ou encore 1 si considéré comme un bit) et non pris (ou prédit non pris) sera à `false` (ou encore 0 si considéré comme un bit).

3 Travail demandé

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.

Les prédicteurs que l'on pourra d'implanter sont les suivants :

1. un prédicteur bimodal à 4 états dont le principe est illustré sur la figure 2.

   On fera l'expérience avec le graphe de transitions vu en cours et rappelé ci-après.

1. `http://cava.cs.utsa.edu/camino/cbp2/`

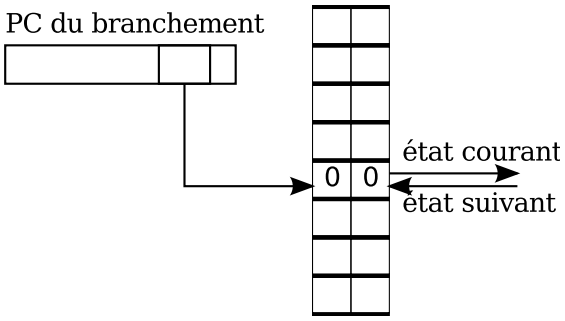


FIGURE 2 – Prédicteur bimodal

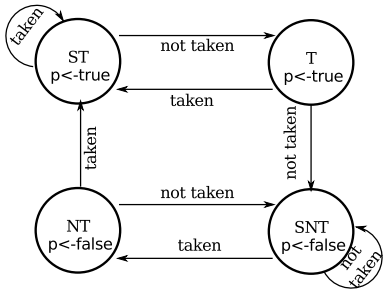


FIGURE 3 – Graphe de transitions du prédicteur bimodal

2. un prédicteur global simple avec un historique des branchement de longueur (nombre de bits) H permettant d'atteindre une entrée bimodale utilisant le graphe d'état n°1 (la *Pattern History Table*). Pour mémoire, l'historique 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 4).

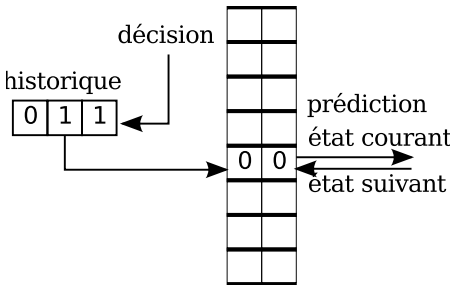


FIGURE 4 – Prédicteur global simple

3. un prédicteur *gshare* qui est un prédicteur global pour lequel l'index définissant l'entrée dans la table de taille 2^n est calculé comme le ou-exclusif d'un registre d'historique avec le pc du branchement, comme illustré sur la figure 5. Si $H > n$, alors on prendra les n bits de poids fort de H pour faire le ou-exclusif.
4. un prédicteur corrélé qui utilise un historique sur H bits permettant de sélectionner une PHT parmi 2^H et utilisant l'adresse du branchement pour indicer cette table, comme illustré sur la figure 6.
5. un prédicteur local qui utilise une table d'historiques sur H bits indexée par les poids faibles de l'adresse du branchement. En utilisant l'entrée d'historique, on indice une PHT bimodale de taille 2^H, comme illustré sur la figure 7.
6. et pour finir un prédicteur mixte (celui de l'alpha 21264) qui utilise un prédicteur global simple 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 table d'historique (soit 1024 entrées). La PHT indexée est un simple prédicteur de type bimodale, mais sur 3 bits, implantée avec un compteur à saturation (comme celle du graphe n°3 de la figure 3).

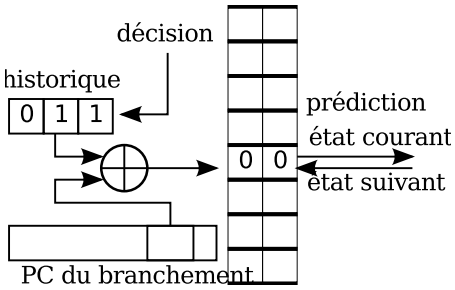


FIGURE 5 – Prédicteur *gshare*

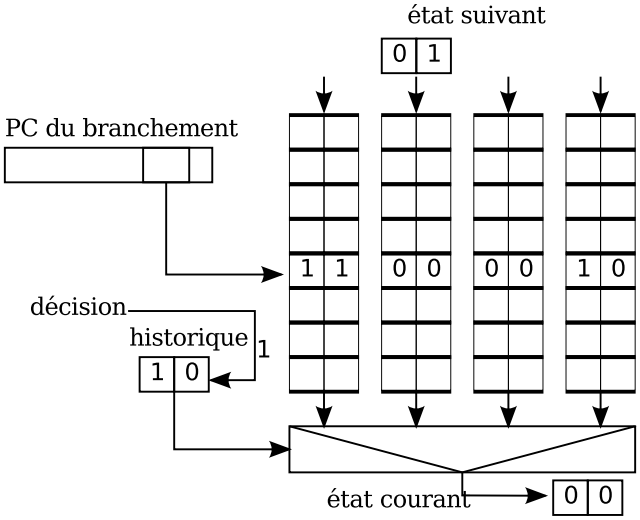


FIGURE 6 – Prédicteur corrélé

Le choix du prédicteur à utiliser se fait grâce à une BHT de 4K entrées de 2 bits. 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 8).

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

Il y a un script `run` à la racine qui lance automatiquement l'exécution sur la totalité des traces pour un prédicteur donné, dont le nom de guerre est passé en premier argument sur la ligne de commande. Il y a dans le script 3 boucles imbriquées : la boucle externe, indice k fait varier la longueur de l'historique (k ne prend qu'une valeur quelconque pour les prédicteurs n'ayant pas d'historique), la boucle du milieu, d'indice j, fait varier le nombre de bits de PC à utiliser pour indexer les tables, et la boucle interne, d'indice i, elle, parcours l'ensemble des traces. Vous serez amené à modifier les valeurs des bornes en fonction des prédicteurs, voir à changer un

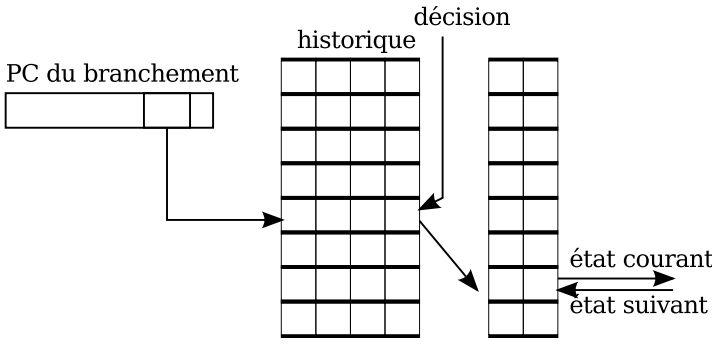


FIGURE 7 – Prédicteur local

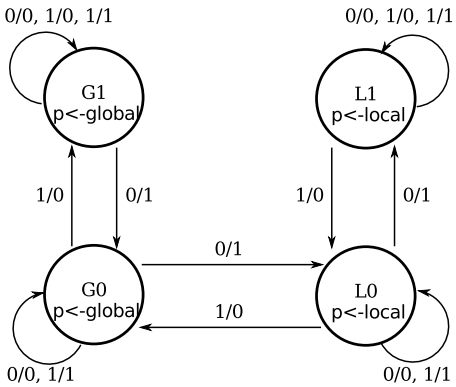


FIGURE 8 – Machine à états du choix du prédicteur. Légende : $global/local$, 0 décision incorrecte, 1 décision correcte

peu cette partie si vous faite des prédicteurs exotiques.

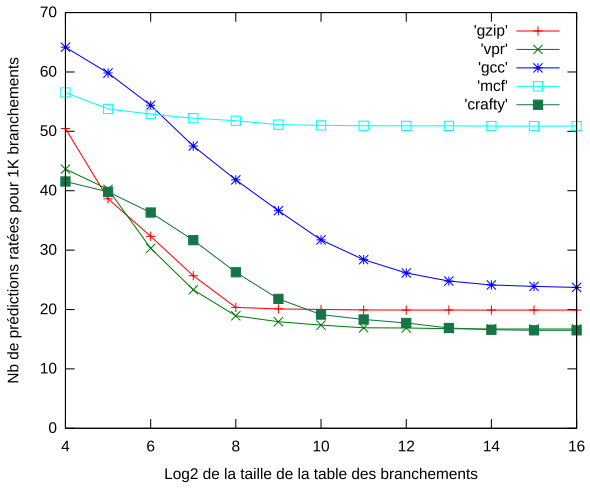
Le script `run` crée deux répertoires par prédicteur, l'un contenant les résultats l'autre les figures.

Afin de paramétrer simplement le prédicteur (sans avoir à toucher à l'infrastructure), on inclura (`#include`) le prédicteur comme unique fichier dans `my_predictor.h`), les temps de compilation étant très faibles [2].

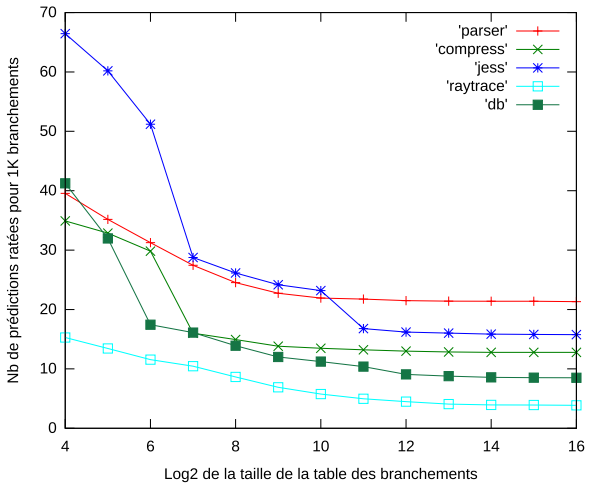
On utilisera `gnuplot` avec le script `plot` pour obtenir les courbes. On donne sur la figure 9 le type de résultat attendu pour le prédicteur 1 bit.

Le sous-répetoire `rendu` contient un exemple de .tex permettant de faire un petit rapport sur le travail.

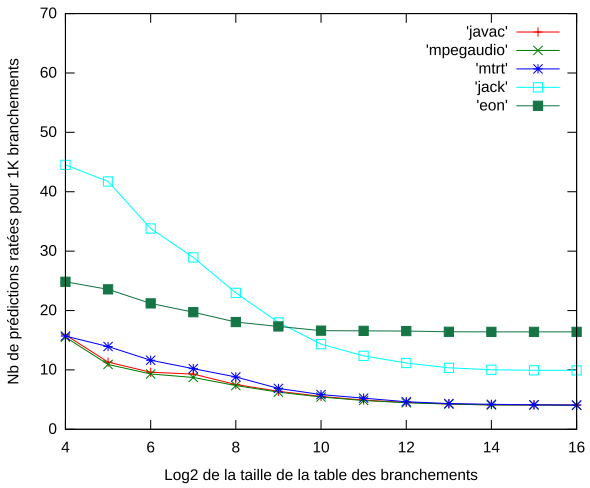
2. `time ./run` donne `267,91s user 9,73s system 117% cpu 3:55,91 total`



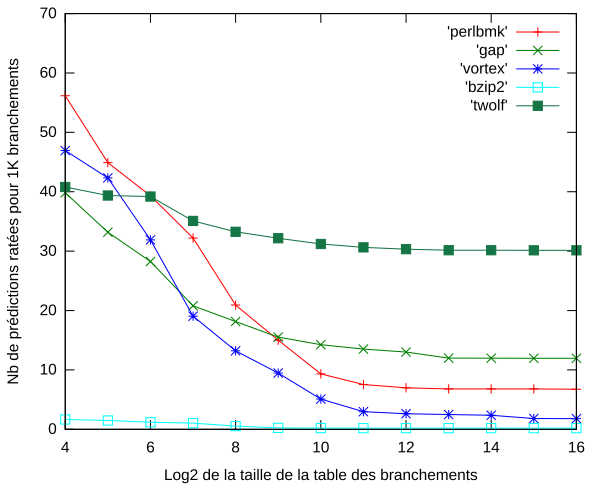
(a) Graphes 0 à 4



(b) Graphes 5 à 9



(c) Graphe 10 à 14



(d) Graphe 15 à 19

FIGURE 9 – Les courbes de MPKI