Computer Architecture

Assignment 4

Part A

Performance of Two-level Branch Predictors

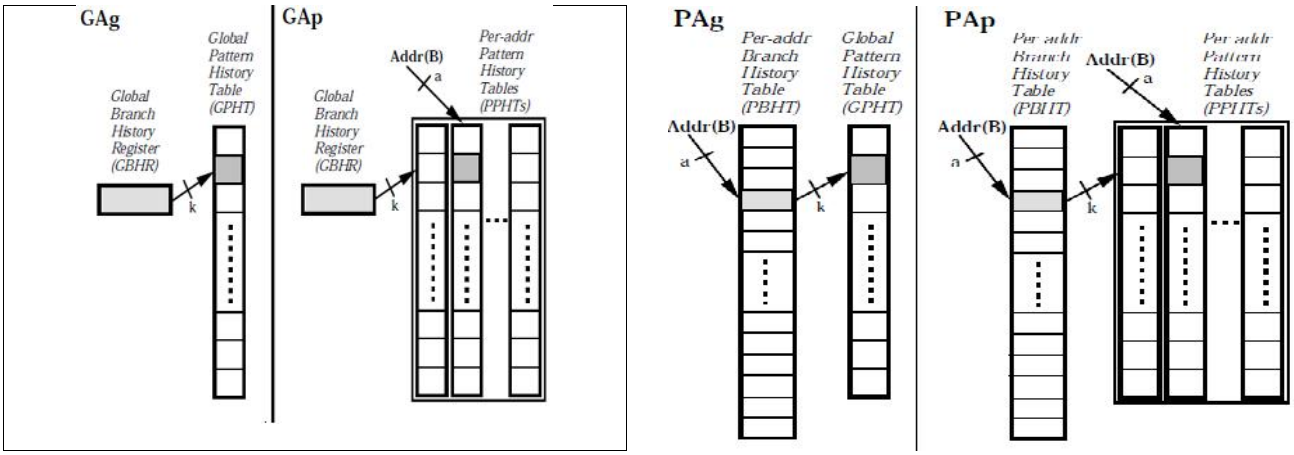
The two-bit predictor schemes use only the recent behavior of a branch to predict the future behavior of that branch. It may be possible to improve the prediction accuracy if we also look at the recent behavior of other branches rather than just the branch we are trying to predict. These predictors are called correlating predictors or two-level predictors.

The first-level branch execution history is the history of the last k branches encountered. The second-level pattern history is the branch behavior for the last j occurrences of the specific pattern of these k branches. Prediction is based on the branch behavior for the last k occurrences of the current branch history pattern. Since the history register has k-bits, at most 2^k different patterns appear in the history register. Each of these 2^k patterns has a corresponding entry in what we called the Pattern History Table (PHT). The first-level and second-level pattern history are collected at run-time, eliminating the disadvantages inherent in Lee and Smith's method, that is, the differences in the branch behavior of the profiling data set and the run-time data sets.

In the part A of this report we analyze the performance of the following two level branch prediction schemes according to the way the first-level branch history is collected and

1. GAg: 1 global history register and 1 global prediction table
2. GAp: 1 global history register and 8 per-address prediction tables
3. PAg: 8 per-address history registers and 1 global prediction table
4. PAp: 8 per-address history registers and 8 per-address prediction tables

The four schemes are depicted in figure below:

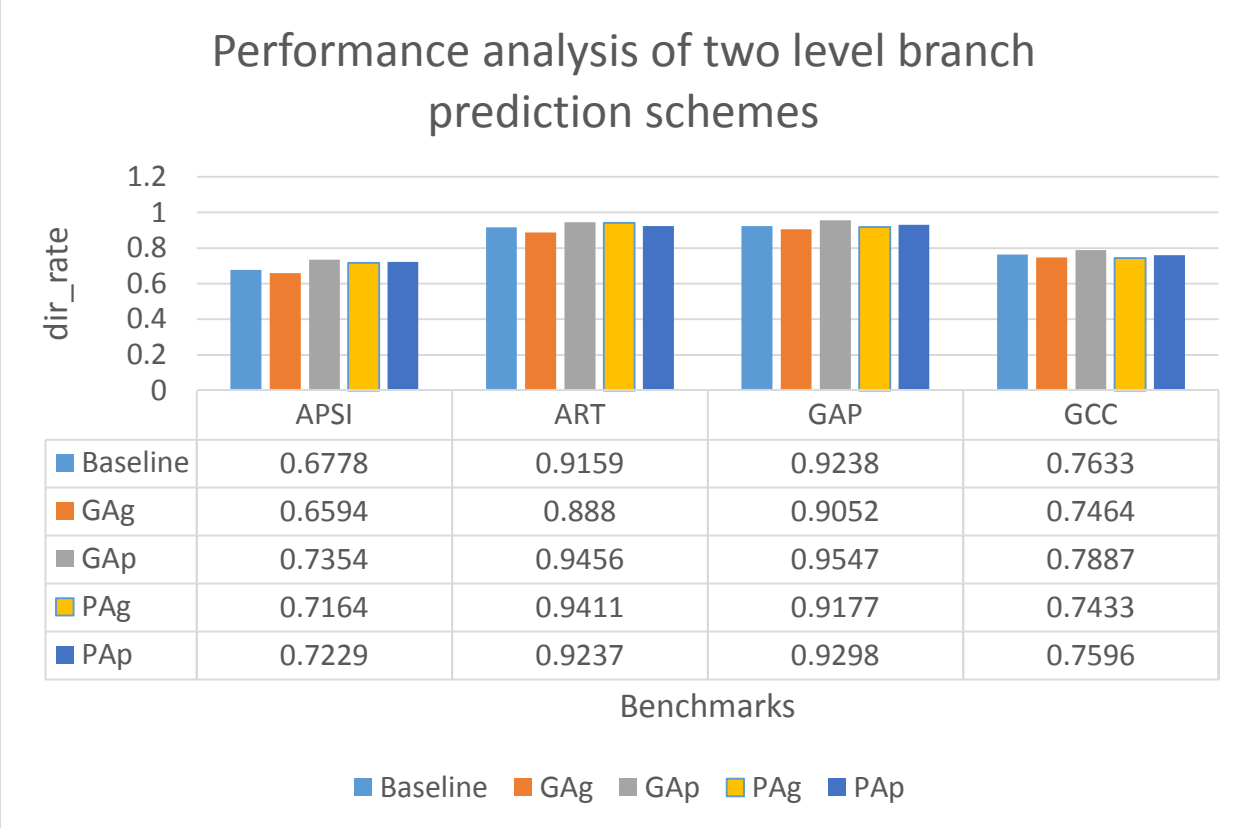


Computer Architecture

The following is the corrected configuration examples for 2-level predictors.
-bpred: 2lev <l1size> <l2size> <hist_size> <xor>
Configurations: N, M, W, X
N # entries in first level (# of shift register(s))
M # entries in 2nd level (# of counters, or other FSM)
W width of shift register(s) (# of bits in each shift register)
X (yes-1/no-0) xor history and address for 2nd level index (We use 0 for this homework).

The following configuration is used for the four branch prediction schemes

GAg: 1 512 9 0 (1, M, W, 0 where M = 2^W)
GAp: 1 512 6 0 (1, M, W, 0 where M = C * 2^W, C is the number of per-address prediction tables)
PAg: 8 512 9 0 (N, M, W, 0 where M = 2^W)
PAp: 8 512 6 0 (N, M, W, 0 where M = N * 2^W)



	APSI	ART	GAP	GCC
Baseline	0.6778	0.9159	0.9238	0.7633
GAg	0.6594	0.888	0.9052	0.7464
GAp	0.7354	0.9456	0.9547	0.7887
PAg	0.7164	0.9411	0.9177	0.7433
PAp	0.7229	0.9237	0.9298	0.7596

Observation1:
GAp predictor scheme performs well compared to all other two level prediction schemes.

Observation 2:
For integer programs (gap, gcc), global history schemes perform better than other schemes.

This is because integer programs usually have more if-else loops whose behavior depends on the outcome of the previous branches and hence has a higher correlation to other branches. Using one single history register exploits the correlation between different branches better.

Observation 3:
For floating programs (apsi, grt), a per-address history schemes perform better than other schemes.

This is because floating programs usually have more for loops whose outcome is independent of the outcome of the previous branches. This requires a per-address history register in first level.

Observation 4:
Per-address history schemes is preferred for both integer and floating point programs.
By providing multiple PHTs in the second level, Interference can be reduced and hence per-address history schemes is preferred for both integer and floating point programs.

Observation 4 (Anolamy):
For art, PAg (0.9411) performs better than PAp(0.9237)

PAg uses a separate local history register for each branch, or a per-address history register, and a single shared global PHT. Each branch prediction is therefore based entirely on the history of the branch being predicted. Since all branches share a single PHT, PAg is characterized by the interference between different branches

Replacing single global history register with multiple registers improves the prediction accuracy of FP programs but worsens that of integer programs.
Using 1k more bits for the global history shift registers, on average, the local scheme does not offer any performance win over the cheaper schemes such as gselect which uses only one global history register. Different from FP programs, which have many long looping structures, branches in integer programs are more correlated..

3.4.4 Results

We have characterized the global, per-address, and per-set history schemes and compared them with respect to their branch prediction performance and cost effectiveness. Global history schemes perform better than other schemes on integer programs but require higher implementation costs to be effective overall. Per-address history schemes perform better than other schemes on floating point programs and require lower implementation costs to be effective overall. Per-set history schemes have performance similar to global history schemes on integer programs; they also have performance similar to per-address history schemes on floating point programs. To be effective, however, per-set history schemes require even higher implementation costs than global history schemes due to the separate pattern history tables

of each set. As a rule of thumb, with respect to the cost-effectiveness of different variations, PAs is the most cost effective among low-cost schemes

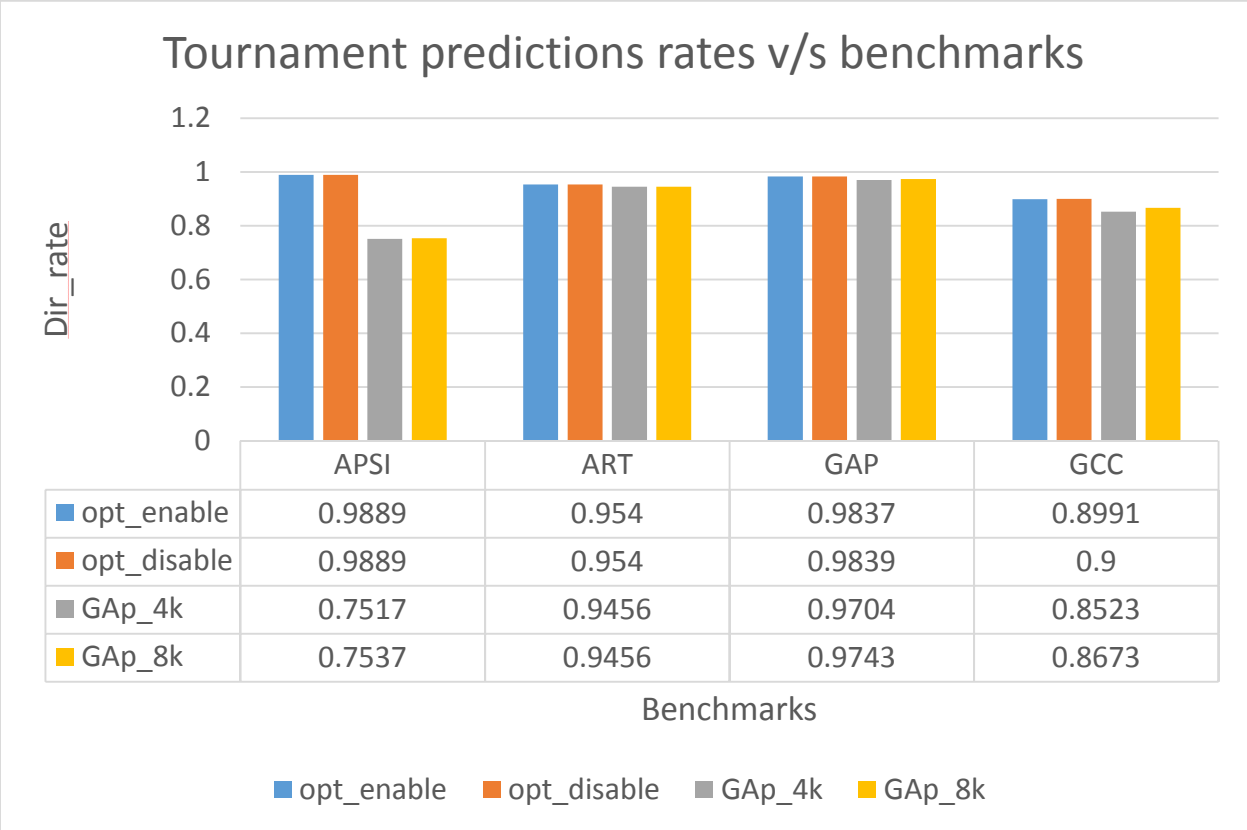
Part B

The following table lists the prediction rate recorded for tournament predictor

Benchmark	T_opt_enable	T_opt_disable	GAp_4k	GAp_8k
APSI	0.9889	0.9889	0.7517	0.7537
ART	0.954	0.954	0.9456	0.9456
GAP	0.9837	0.9839	0.9704	0.9743
GCC	0.8991	0.9	0.8523	0.8673

Since Gap was the best predictor in part A, comparison is made between tournament predictor with that of GAp. The following configuration is used for GAp

-bpred: 2lev 1 4096 9 0
-bpred: 2lev 1 8192 10 0



It can be clearly seen from the above table that tournament predictor outperforms the best two level predictor GAp. However no such significant difference is found when optional case is implemented i.e without local prediction update when global predictor is selected and it provided the correct prediction.