Evaluating Different Branch Predictors Using ChampSim

Anshika Modi
M.Tech CSA
Indian Institute of Science
Bangalore,India
SR: 24-1-24095
anshikamodi@iisc.ac.in

Archie Gaur
M.Tech CSA
Indian Institute of Science
Bangalore,India
SR: 24-1-24826
archiegaur@iisc.ac.in

***Abstract*—Hardware Branch Predictors are considered critical for maintaining high performance in Modern Architectures as they ensure that instruction buffers are filled by speculating correct execution path. Significant performance degradation can be cause by mispredictions, especially in deep pipelines. This study is based on the evaluation of four branch predictors: Bimodal, Gshare, Perceptron, and TaGe for a hardware budget of 64KB. All of them follows different methodology to predict if the branch should be taken or not. Moreover, a hybrid predictor is implemented by combining Gshare and TAGE with Bimodal as meta predictor to improve accuracy.**

***Index Terms*—branch predictor, bimodal, gshare, perceptron, tage, hybrid predictor**

I. Introduction

Single-core Performance can be significantly improved by using deeper pipelines, but some branches in the pipeline can cause "stalls". Speculative Execution is required to reduce these stalls. To predict the correct execution path, hardware branch predictors are used, but when a branch predictor causes a misprediction, all instructions from the incorrect execution path must be squashed. This misprediction causes great latency leading to performance degradation in longer pipelines. For example, experiments on real processors showed that reducing the branch mispredictions by half improved the processor performance by 13 percent.[1]
Branch predictor tells whether a branch will be taken or not and determine branch target address. The Branch Target Buffer(BTB) contains the previously computed Branch Target addresses. The major focus is to find out how efficiently branch predictors can predict branch directions. To evaluate this, we simulate traces from 3 cloud suite benchmarks [2] and 1 SPEC benchmark [3]and assess the accuracy of four branch predictors: Bimodal, Gshare, Perceptron and TAGE. All of these predictors different methods to predict the direction of the branches.
Bimodal predictor takes advantage of bimodal distribution of the branch behavior and attempts to distinguish usually taken from usually not taken branches. Gshare uses global history hashed with instruction pointer the index into the table of saturating counters. Perceptron predictor uses simple linear neuron to find correlation between branches by training weights for each bit in global history. Tage predictor uses tag matching and different lengths of global history to index into different tables. All these predictors are dynamic predictor. i.e. they can dynamically learn behavior of branches and adapt for the given input.

II. Simulation Methodology

We are using ChampSim [4] in this study. ChampSim is an open-source fast trace-based simulator. ChampSim uses four kinds of modules – Branch Direction Predictors, Memory Prefetchers, Cache Replacement Policies and Branch Target Predictors. The simulator doesn't execute actual programs but replays recorded traces to model the CPU's behavior. In this study, the Traces for three Cloudsuites benchmarks used are:

A. $cloud9\_phase0\_core1.trace.xz$

B. $nutch\_phase1\_core0.trace.xz$

C. $streaming\_phase1\_core0.trace.xz$

D. $600.perlbench\_s-210B.champsimtrace.xz$

These traces are fed into Champsim to decode instructions, predict branches and calculate branch prediction accuracy. We know that if misprediction occurs, then all instructions on incorrect speculative path should be squashed, but not in the case of Champsim because Champsim doesn't include incorrect path instructions Champsim also provides data about the hits and misses for load and stores at each level in cache Hierarchy and also stimulates TLB behaviour. Champsim used a Branch Target Buffer (BTB) with 2048 entries and a 2-way set associative structure. It predicts non-return branch targets and has a Return Address Stack(RAS) of 64 entries for return address prediction. an The size of the Branch Target Buffer is kept constant across different predictors to ensure fair testing. Champsim cannot fast forward, so in the warmup phase, 2 Billion Instructions are run to stabilize the behaviour of the program and then 500 million instructions are simulated to evaluate branch prediction. Champsim reports various performance metrics such as IPC(Instructions per

cycle), MPKI(mispredictions per kilo instructions), Branch prediction accuracy, cache performance ,TLB performance etc. Our main area of interest lies in calculating IPC, MPKI, and Branch prediction accuracy.

III. Observation

In this section we report the results of simulations for different configurations of bimodal, gshare, perceptron, tage and hybrid predictor for above mentioned traces.

A. Bimodal Predictor

Bimodal branch prediction takes advantage of bimodal distribution of branch behavior and attempts to distinguish usually taken from usually not taken branches. Each counter is two bits long. For each taken branch, the appropriate counter is incremented. Likewise for each not-taken branch, the appropriate counter is decremented. In addition, the counter is saturating. In other words, the counter is not decremented past zero, nor is it incremented past three [5]. ChampSim already provide bimodal implementation. So we are using it with modification in sizes of table size and counter bits. Table I reports instructions per cycle(IPC) and mispredictions per kilo instructions(MPKI) for bimodal predictor with hardware budget of 64KB and Figure 1 shows their comparison.

TABLE I
IPC and MPKI for Bimodal with 64KB Budget

	cloud	nutch	streaming	perl
IPC	0.8672	0.629	0.511	2.22
MPKI	0.01522	0.3421	0.3903	6.521

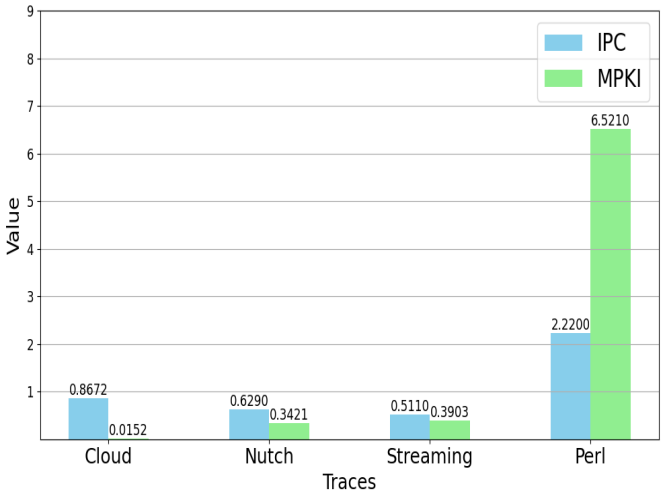

Fig. 1. IPC and MPKI for Bimodal with 64KB Budget

B. GShare Predictor

Gshare predictor was first introduced by McFarling [5]. It uses XOR result of branch history register(BHR) and PC value to index into pattern history table(PHT). PHT is table of two bit saturating counters. ChampSim already provide gshare implementation. So we are using it with modification in sizes of BHR and PHT. Table II reports instructions per cycle(IPC) and mis-prediction per kilo instructions(MPKI) for gshare predictor with hardware budget of 64KB and Fig 2 shows their visual comaprison.

TABLE II
IPC and MPKI for GShare with 64KB Budget

	cloud	nutch	streaming	perl
IPC	0.8672	0.629	0.511	2.344
MPKI	0.01522	0.3421	0.3903	5.315

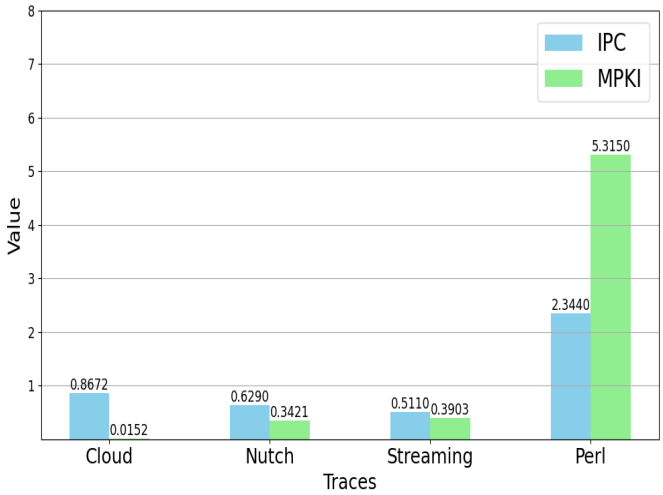

Fig. 2. IPC and MPKI for GShare with 64KB Budget

C. Perceptron Predictor

The perceptron predictor was introduced by Jimenez and Lin[6] . It uses a simple linear neuron known as a perceptron to perform branch direction prediction. It exploits correlations between branches. Perceptron predictor stores weight for each bit of global history. During prediction it performs multiply accumulate operation with weights and global branch history. If the result is negative than branch won't be taken, otherwise branch will be taken. Biggest down side of perceptron predictor is that it can only learn linearly separable branches. It can't predict branches which have XOR like correlation. ChampSim also provide implementation of perceptron branch predictor. We are using this implementation with modification size of global history, number of perceptron and number of perceptron bits to evaluate accuracy of perceptron predictor for different hardware budgets. Table III reports instructions per cycle(IPC) and misprediction per kilo instructions(MPKI) for perceptron predictor with hardware budget of 64KB and Fig 3 shows their visual comparison.

TABLE III
IPC and MPKI for Perceptron with 64KB Budget

	cloud	nutch	streaming	perl
IPC	0.8671	0.629	0.5111	2.116
MPKI	0.01522	0.3421	0.3926	7.405

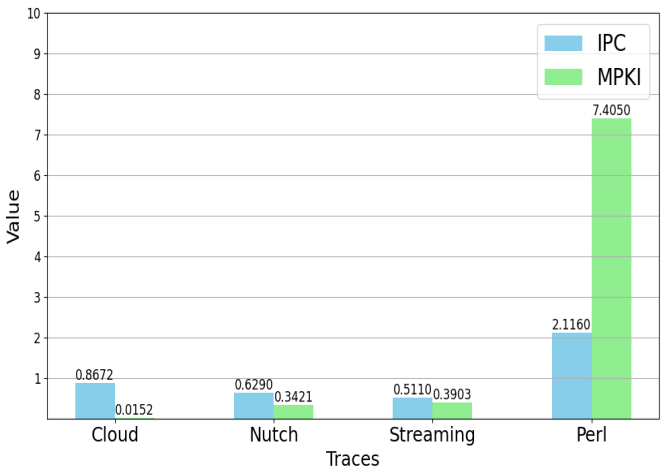

Fig. 3. IPC and MPKI for Perceptron with 64KB Budget

D. TaGe Predictor

Tage predictor was introduced by Seznec, Andre and Michaud [7]. It uses different history lengths to index into predictor tables. Different history lengths are in geometric progression. If predictor has M distinct tables $T_i, 0 <= i < M$. Distinct history length are used to index distinct tables. $L(i)$ will be used to compute the index of the table T_i. $L(i)$ can be calculated using the formula:

$$L(i) = \alpha^{(i-1)} \cdot L(1) + 0.5 \quad (1)$$

Table T_0 will be directly indexed using branch address. We are using open source tage implementation from [13]. We are taking 4 prediction tables with one bimodal table as T_0. We are keeping the hardware budget fixed at 64KB.

Table IV reports IPC and MPKI for Tage predictor and Fig 4 shows their visual comparison.

TABLE IV
IPC AND MPKI FOR TAGE WITH 64KB BUDGET

	cloud	nutch	streaming	perl
IPC	0.8671	0.629	0.5111	2.444
MPKI	0.01522	0.3421	0.3903	4.541

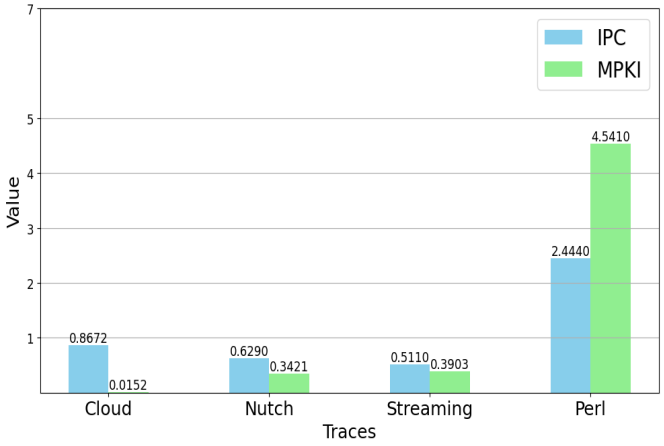

Fig. 4. IPC and MPKI for TaGe with 64KB Budget

We have also evaluated the TaGe predictor by changing the parameters such as history lengths and table size but keeping the hardware cost same as 64KB. The MPKI and IPC value for various configurations of TaGe predictor for all the four traces is shown in Table V, VI, VII, VIII respectively.

TABLE V
MPKI & IPC FOR VARIOUS TAGE CONFIGURATIONS FOR TRACE1 (CLOUD)

Banks	BiModal_Size	Len_Global	Len_Tag	MPKI	IPC
2	65536	13	10	0.01522	0.8672
4	40960	9	10	0.01522	0.8672
4	65536	12	9	0.01522	0.8672
3	512	7	8	0.01522	0.8671

TABLE VI
MPKI & IPC FOR VARIOUS TAGE CONFIGURATIONS FOR TRACE2 (NUTCH)

Banks	BiModal_Size	Len_Global	Len_Tag	MPKI	IPC
2	65536	13	10	0.3421	0.629
4	40960	9	10	0.3421	0.629
4	65536	12	9	0.3421	0.629
3	512	7	8	0.3421	0.629

TABLE VII
MPKI & IPC FOR VARIOUS TAGE CONFIGURATIONS FOR TRACE3 (STREAMING)

Banks	BiModal_Size	Len_Global	Len_Tag	MPKI	IPC
2	65536	13	10	0.3903	0.5111
4	40960	9	10	0.3903	0.5111
4	65536	12	9	0.3903	0.5111
3	512	7	8	0.3903	0.5111

TABLE VIII
MPKI & IPC FOR VARIOUS TAGE CONFIGURATIONS FOR TRACE4 (PERL)

Banks	BiModal_Size	Len_Global	Len_Tag	MPKI	IPC
2	65536	13	10	4.102	2.528
4	40960	9	10	4.541	2.444
4	65536	12	9	3.348	2.648
3	512	7	8	8.278	2.037

As seen from Table VIII, the MPKI value is improved when the table size is 64KB and global history length is 9 bit. Thus, the MPKI value is reduced for this configuration.

E. Hybrid Predictor

Hybrid predictors combine multiple predictors to increase the prediction accuracy. We are using method described in [5] implement hybrid predictor. Our hybrid predictor uses gshare predictor, tage predictor and one bimodal like meta predictor. Meta predictor is a table of 2 bits saturating counter with hardware budget of 1KB. This table is indexed using branch address. If MSB of counter is zero, we choose Tage predictor to predict the branch. If MSB of counter is one, we choose gshare predictor. When branch is evaluated, if gshare is correct and tage is wrong we increase the counter, if tage is correct and gshare is wrong we decrease the counter. If both are correct or both are wrong we don't change the counter.

Total hardware budget of hybrid predictor is 64KB distributed between gshare and tage predictor and 1KB for meta predictor.

Table IX reports IPC and MPKI for hybrid predictor with 50:50 budget distribution, Since it seems to be most accurate predictor and Fig 5 shows their visual comparison.

TABLE IX
IPC AND MPKI FOR HYBRID WITH 64KB BUDGET

	cloud	nutch	streaming	perl
IPC	0.8681	0.627	0.515	2.659
MPKI	0.0152	0.3423	0.3901	3.298

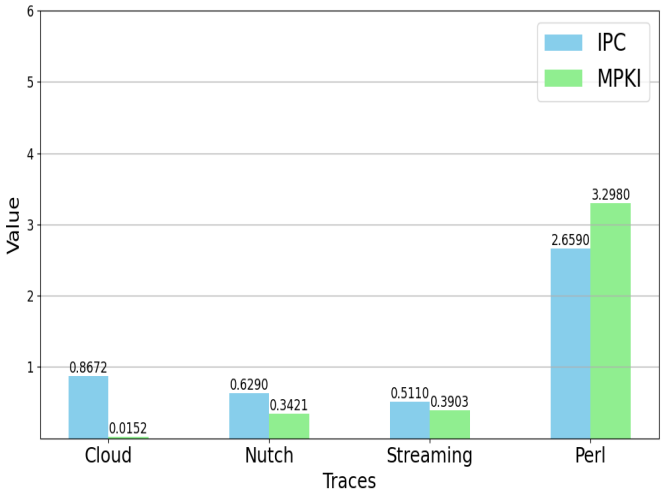

Fig. 5. IPC and MPKI for Hybrid with 64KB Budget

IV. CONCLUSION

In this study we evaluated four existing branch predictors on four cloudsuite benchmark programs using the Champsim simulator: A) BiModal B) Gshare C) Perceptron D) TaGe We also have compared the MPKI and IPC for various values of history lengths and table size in TaGe predictor. After evaluating these four predictors we proposed our own hybrid predictor which uses gshare predictor and tage predictor. Meta predictor is used for choosing between these two predictors. Table X reports prediction accuracy for different branch predictors.

From the table we can say that every predictor has more or less the same prediction accuracy for the cloudsuite traces, but for the SPEC(perl) trace, Hybrid is giving the best performance. Although the difference in hybrid and TaGe Predictor's accuracy is quite low, but the mixture of TaGe with Gshare in 50:50 tends to give better prediction accuracy. Fig 6 shows the comparison of various Predictors prediction accuracy for 64KB hardware budget.

TABLE X
PREDICTION ACCURACY FOR DIFFERENT PREDICTORS (IN PERCENTAGE)

	cloud	nutch	streaming	perl
BiModal	99.81	96.67	97.76	95.88
GShare	99.81	96.67	97.76	96.64
Perceptron	99.81	96.67	97.75	95.32
TaGe	99.81	96.67	97.76	97.13
Hybrid	99.81	96.67	97.76	97.91

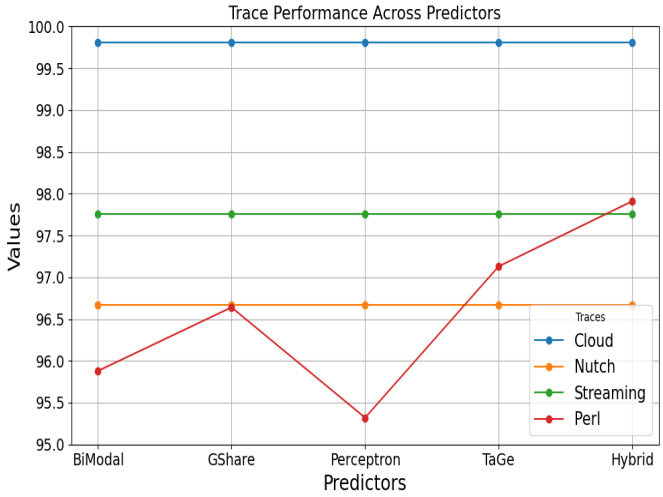

Fig. 6. Prediction Accuracy Comparison of all predictors for 64KB hardware budget

REFERENCES

[1] JD. A. Jimenez. "An optimized scaled neural branch predictor". In: (2011). DOI: 10.1109/ICCD.2011.6081385.

[2] CloudSuite Traces. URL: https://www.dropbox.com/sh/hh09tt8myuz0jbp/AACAS5zMWHL7coVuS-.

[3] SPEC Traces. URL: https://dpc3.compas.cs.stonybrook.edu/champsim-traces/speccpu/.

[4] Champsim. URL: https://github.com/ChampSim/ChampSim.

[5] Scott McFarling. "Combining branch predictors. Vol. 49. Technical Report TN-36, Digital Western Research Laboratory". In: (1993). URL: https://www.ece.ucdavis.edu/~akella/270W05/mcfarling93combining.pdf.

[6] D. A. Jimenez and C. Lin. "Dynamic branch prediction with perceptrons, Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture, Monterrey, Mexico, 2001, pp. 197206," in: (). DOI: 10.1109/HPCA.2001.903263.

[7] Andre Seznec and Pierre Michaud. "A case for (partially) TAgged GEometric history length branch prediction". In: (2006).