Evaluating Champsim Simulator

Anishka Ratnawat
Computer Science and Automation
IISc, Bangalore, India
anishkar@iisc.ac.in

Snigdha Shekhar
Computer Science and Automation
IISc, Bangalore, India
snigdhas@iisc.ac.in

Tanya Gautam
Computer Science and Automation
IISc, Bangalore, India
tanyagautam@iisc.ac.in

***Abstract*—Branch prediction is a pivotal step and a common architectural method used to avoid delays in pipeline caused due to stalls incurred in fetching the correct instructions following branch execution. In order to be able to fetch instructions continuously, it must know in advance whether the branch will be "taken" (and consequently fetch the target instruction) or "not taken" (and consequently fetch the fall-through instruction). The processor, therefore, should predict the branch outcome since it cannot know the status of the branch instruction until the execution phase is completed. A processor can perform branch prediction while fetching a branch instruction. In this report, we evaluate several existing branch prediction techniques by running several benchmark programs and simulating branch prediction using Champsim simulator.**

I. Introduction

Branch prediction techniques aim to predict the results of execution of branch instructions as accurately as possible, thereby reducing the possibility of flushing of the instructions that are executed following the branch when loaded during the later stages of the branch's execution. [1]

ChampSim is an open-source trace-based simulator maintained at Texas A&M University and through the support of the computer architecture community. In this report, we evaluate various predictors and report the MPKI (Mispredictions per Kilo Instructions) and IPC (Instructions per cycle) for our given 3 SPEC benchmark programs.

A. Gshare

Gshare is a correlated predictor which combines values from global history register with the PC value or the instruction address by XORing the global history register and the PC address and accessing the table of 2-bit predictors with the hash obtained from XORing. The main advantage of Gshare is that it has a relatively small implementation overhead. However, the performance of Gshare suffers when branch resolutions are more accurately correlated with localized patterns.

B. Perceptron

A perceptron is a mathematical model of a neuron that simulates "learning" from a given set of inputs. The model refines itself as the coefficients for a function with binary output get better. This machine learning concept can be applied to branch prediction by maintaining a list of function coefficients for every address whose lower n bits map to a particular row entry in a table of coefficients. The accuracy of perceptron alone can be improved by increasing from a single-layer to multiple layers, however the implementation overhead for a multiple layers is significantly more.

C. TAGE

The TAGE predictor is derived from Michaud's PPM-like tag-based branch predictor and uses geometric history lengths. The TAGE predictor features a base predictor T0 in charge of providing a basic prediction and a set of (partially) tagged T_i. These tagged predictor components T_i, $1 <= i <= M$ are indexed using different history lengths that form a geometric series, i.e, $L(i) = (int)(\alpha^{i-1} * L(1) + 0.5)$ An entry in a tagged component consists in a signed counter ctr which sign provides the prediction, a (partial) tag and an unsigned useful counter u. Throughout this paper, u is a 2-bit counter and ctr is a 3-bit counter.

D. Hybrid

A hybrid predictor, also called combined predictor, implements more than one prediction mechanism. The final prediction is based either on a selection mechanism that stores which of the predictors has made the best predictions in the past, or a majority vote function based on an odd number of different predictors [4].

II. Simulation Methodology

We ran the Champsim simulator for 500 Million instructions in the following manner.

```
bin/champsim --warmup_instructions
200000000 --simulation_instructions
500000000 ~/602.gcc_s-2226B.champsimtrace.xz
```

A. Benchmarks

The benchmarks assigned to us were the following.

- **602.gcc_s**:It generates code for an IA32 processor. All output files are IA32 assembly code files.
- **605.mcf_s**:It is used for single-depot vehicle scheduling in public mass transportation
- **625.x264_s**: Performs Video Compression and compresses portions of Blender Open Movie Project's "Big Buck Bunny"

The storage budget assigned to us was roughly 128KB and we have modified the parameters of various predictors accordingly. We have listed the modified parameters used while running the simulators below.

B. Gshare

```
GLOBAL_HISTORY_LENGTH = 16;
COUNTER_BITS = 2;
GS_HISTORY_TABLE_SIZE = 524288;
```

C. Perceptron

```
PERCEPTRON_HISTORY = 62;
PERCEPTRON_BITS = 9;
NUM_PERCEPTRONS = 1850;
```

D. TAGE

```
α = 1.6
TAGE_INDEX_BITS[TAGE_NUM_COMPONENTS]
= {14,15,14,13};
TAGE_TAG_BITS[TAGE_NUM_COMPONENTS]
= {7,9,9,10};
```

*1) **Varying history lengths and table sizes for TAGE**:* We have explored different history lengths and table sizes for TAGE as follows:

Set 1 parameters (Original):

```
α = 2
TAGE_INDEX_BITS
[TAGE_NUM_COMPONENTS] = {14,15,14,13};
TAGE_TAG_BITS
[TAGE_NUM_COMPONENTS] = {7,9,9,10};
```

Set 2 parameters:

```
α = 1.6
TAGE_INDEX_BITS
[TAGE_NUM_COMPONENTS] = {14,15,13,12};
TAGE_TAG_BITS
[TAGE_NUM_COMPONENTS] = {10,11,13,14};
```

Set 3 parameters:

```
α = 1.2
TAGE_INDEX_BITS
[TAGE_NUM_COMPONENTS] = {14,15,13,12};
TAGE_TAG_BITS
[TAGE_NUM_COMPONENTS] = {12,13,14,15};
```

E. Hybrid

Hybrid branch predictors normally achieve higher prediction accuracy than single-scheme predictors by exploiting the strengths of each of their component predictors [2]. We have built the hybrid predictor using TAGE and Perceptron as the individual component predictors.
We have implemented our Hybrid Predictor using the Bimodal predictor as the meta predictor. The storage budget for the bimodal predictor that we have used is 1 KB. We have included the TAGE predictor and the Perceptron predictor classes in our code so we can directly call their class functions in our code.

1 Predict Branch

$i \leftarrow bimodal\_table[this][hash]$
if $(i == 0$ **or** $i == 1)$ **then**
  $ppred = perceptron\_predict\_branch()$
  return $ppred$
else
  if $(i == 2$ **or** $i == 3)$ **then**
    $tpred = tage\_predict\_branch()$
    return $tpred$
  end if
end if

2 Last branch result

$i \leftarrow bimodal\_table[this][hash]$
if $(taken == tpred$ **and** $taken \neq ppred)$ **then**
  $i = i + 1;$
else
  if $(taken \neq tpred$ **and** $taken == ppred)$ **then**
    $i = i - 1$
  end if
end if

We update our TAGE predictor and perceptron predictor after the prediction is done by calling the function TAGE_update() and perceptron_update(). We also update the Bimodal Predictor appropriately using the function *Last_Branch_Result* described above.

III. Observation

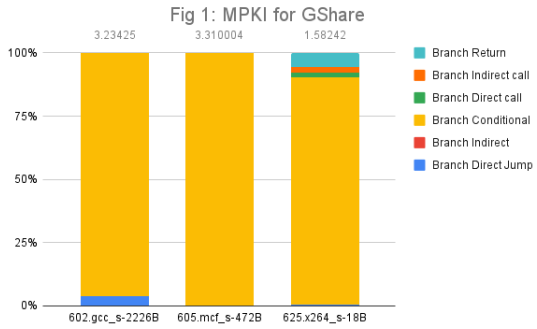
This section includes the results obtained from the simulation and an analysis of the results of the various predictors simulated.

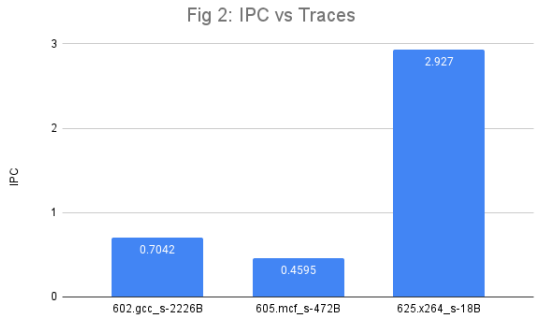
A. GShare

Table 1 lists the MPKI, IPC, and Performance Accuracy of GShare for different traces. Fig 1 shows the MPKI of GShare for different benchmarks. It also illustrates the distribution of MPKI for different types of branches.

Trace	MPKI	IPC	PA
602.gcc_s-2226B	3.23	0.7042	99.09
605.mcf_s-472B	3.33	0.4592	97.59
625.x264_s-18B	1.496	2.939	96.31

TABLE I: GShare Simulations



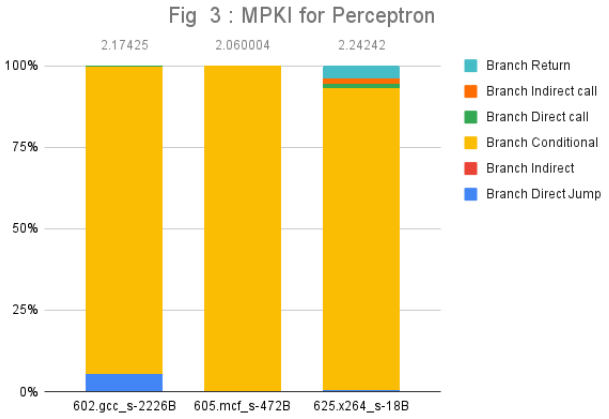


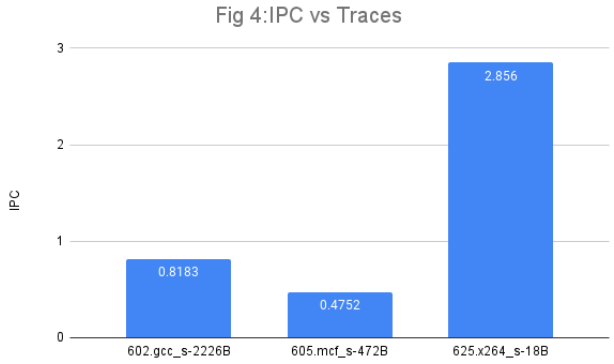
B. Perceptron

Table II lists the MPKI, IPC, and Performance Accuracy of Perceptron for different traces. Fig 3 shows the MPKI of Perceptron for different benchmarks. It also illustrates the distribution of MPKI for different types of branches.

Trace	MPKI	IPC	PA
602.gcc_s-2226B	2.179	0.8183	99.39
605.mcf_s-472B	2.056	0.4752	98.5
625.x264_s-18B	2.238	2.856	94.48

TABLE II: Perceptron Simulations





C. TAGE

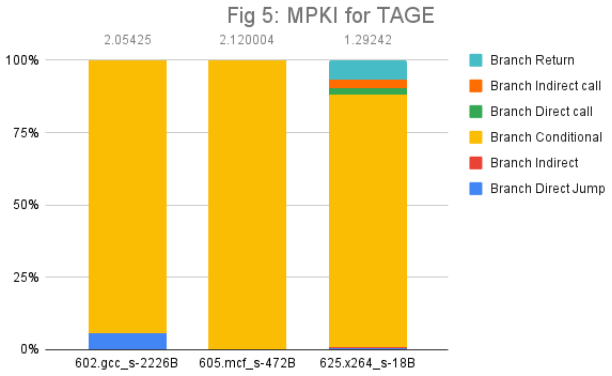
Table III lists the MPKI , IPC and Performance Accuracy of TAGE for different traces. Fig 5 shows the MPKI of TAGE for different benchmarks. It also illustrates the distribution of MPKI for different types of branches.

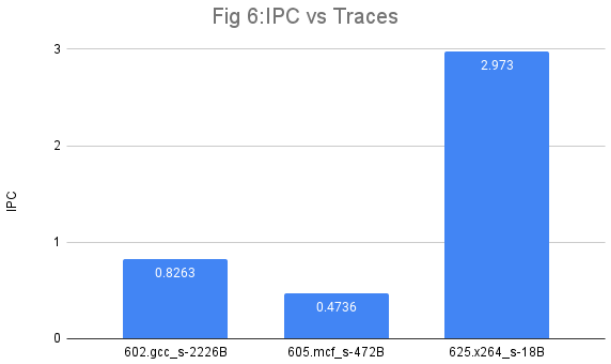
Trace	MPKI	IPC	PA
602.gcc_s-2226B	2.057	0.8263	99.42
605.mcf_s-472B	2.123	0.4736	98.46
625.x264_s-18B	1.295	2.973	96.81

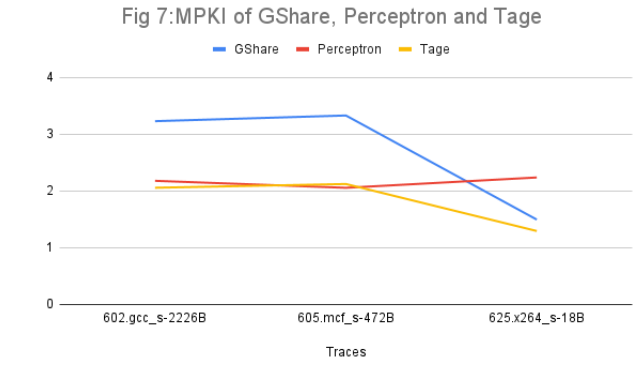
TABLE III: TAGE Simulations

	50:50			30:70			70:30		
Trace	**MPKI**	**IPC**	**PA**	**MPKI**	**IPC**	**PA**	**MPKI**	**IPC**	**PA**
602.gcc_s-2226B	1.843	0.8389	99.48	1.94	0.836	99.45	1.864	0.8358	99.47
605.mcf_s-472B	2.117	0.4737	98.46	2.117	0.4735	98.46	2.115	0.4737	98.46
625.x264_s-18B	1.367	2.959	96.63	1.297	2.969	96.8	1.305	2.963	96.78

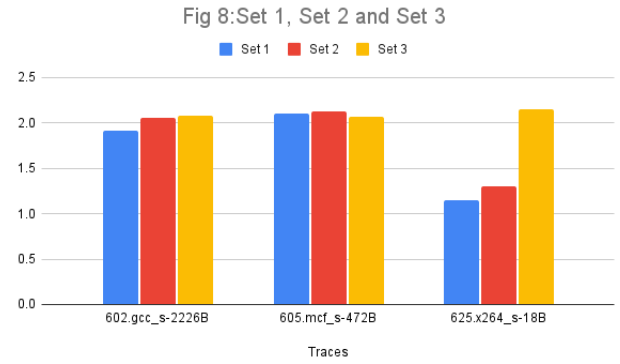
TABLE IV: Perceptron and TAGE Simulations







1) Varying history lengths: FIG 8 shows the misprediction rates for different sets of history lengths and different table sizes. We have 3 sets of table sizes and history lengths. For Set 1, we have taken $\alpha = 2$, for Set 2 we have taken $\alpha = 1.6$ and for Set 3 we have taken $\alpha = 1.2$. We obtained the best results with $\alpha = 2$. This is in line with our understanding that increasing the history lengths in the TAGE components increases the prediction accuracy. Hence, we have used $\alpha = 2$ while simulating the hybrid predictors.



D. Hybrid

We have designed 3 hybrid predictors involving TAGE, Perceptron, and Gshare. The total storage budget allocated to us is 128KB. We have used the Bimodal predictor as the meta-predictor with the storage budget fixed at 1KB.

1) TAGE/Perceptron: Table IV shows the MPKI, IPC, and Performance Accuracy of the Hybrid TAGE/Perceptron predictor. The relative storage allocated to the individual predictors(perceptron and TAGE) is varied as 30:70, 50:50, or 70:30.

2) Perceptron/GShare: Table V shows the MPKI, IPC and Performance Accuracy of Hybrid Perceptron/GShare predictor.The relative storage allocated to each predictor is in the ratio 70:30. We have given 70% budget to Perceptron and 30% budget to GShare as we have observed from Fig 7 that Ghare is the worst performing predictor among the three predictors.

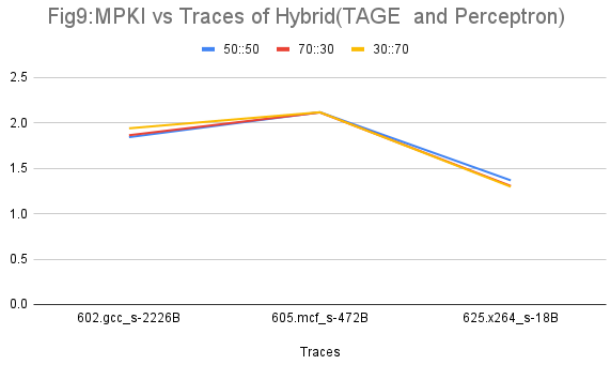
Trace	**MPKI**	**IPC**	**PA**
602.gcc_s-2226B	1.836	0.8393	99.48
605.mcf_s-472B	1.884	0.4789	98.63
625.x264_s-18B	1.582	2.917	96.07

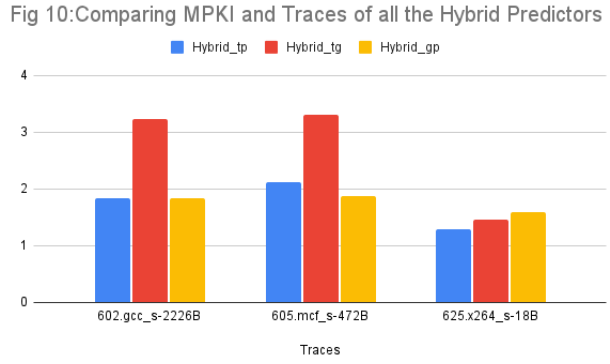
TABLE V: Perceptron/GShare Simulations

3) TAGE/GShare: Table VI shows the MPKI, IPC and Performance Accuracy of the Hybrid TAGE/GShare predictor. The relative storage allocated to each predictor is in the ratio of 70:30.We have given 70% budget to TAGE and 30% budget to GShare as we have observed from Fig 7 that Ghare is the worst performing predictor among the three predictors.

Trace	MPKI	IPC	PA
602.gcc_s-2226B	3.232	0.7037	99.09
605.mcf_s-472B	3.313	0.4594	97.59
625.x264_s-18B	1.468	2.942	96.38

TABLE VI: TAGE/GShare Simulations





IV. Results

This section includes an analysis of the result of the predictors simulated.

1) Fig 7 shows that the MPKI of TAGE is lowest for all three benchmarks. Hence we can say that TAGE works best amongst the three predictors for these benchmarks. For benchmark 625.x264_s-18B, Gshare performs better than Perceptron as GShare has lower MPKI than Perceptron. This may indicate the fact that the benchmark 625.x264_s-18B has more *linearly inseparable* branches, hence GShare performs better. A Boolean function is *linearly separable* if and only if there exist values for the weights such that all of the true instances can be separated from all of the false instances by a hyperplane. [3]

We can also observe from fig 2, fig 4 and fig 6 that the 602.gcc_s-2226B has the lowest IPC and 625.x264_s-18B has the highest IPC in all three predictors. The performance can be measured in terms of the Instructions per Cycle(IPC). Hence, if we use IPC as a performance metric, we can say that the performance of 625.x264_s-18B is higher as compared to the other two benchmarks.

2) Varying History Lengths: From Fig 8, we can observe that as we decrease the value of α, the MPKI increases in benchmark 602.gcc_s-2226B and 625.x264_s-18B. This shows that smaller history lengths lead to larger MPKI. But surprisingly, we made an interesting observation that there is a slight decrease in MPKI for $\alpha = 1.2$ in 605.mcf_s-472B, from $\alpha = 1.6$.

3)The hybrid TAGE/Perceptron performs better than its worst-performing individual predictors. From Fig 9, we can also observe that for the Hybrid TAGE/Perceptron predictor, there is only a minute difference in the MPKI of the traces even on changing the relative storage allocated.

4)All the Hybrid Predictors perform better than the worst-performing individual predictors. For example, in Hybrid TAGE/Perceptron predictor, the MPKI for the 602.gcc_s-2226B is 1.843(50:50), 1.94(30:70), and 1.864(70:30) which is lower than MPKI of TAGE which is 2.057 and also the MPKI of Perceptron which is 2.179. Also, we can observe that the Hybrid Perceptron/GShare Predictor works better than Perceptron for all benchmarks. The hybrid Perceptron/GShare predictor performs better in the presence of context switching, hence the benchmarks may have a lot of context switching hence the hybrid is performing better. [3]

5)On comparing the three benchmarks for different Hybrid predictors, we observed that the Hybrid TAGE/Perceptron Predictor gives the lowest MPKI for 625.x264_s-18B. But Hybrid_gp Perceptron/GShare gives the lowest MPKI for 602.gcc_s-2226B and 605.mcf_s-472B. The best and worst performing for each benchmark is reasoned below:

A. 602.gcc_s-2226B

Hybrid Gshare/TAGE performs worst in this benchmark. As GShare has high MPKI values as observed in TABLE I, this indicates that it is a very simple predictor and does not exploit geometric history length like TAGE. So in this Hybrid predictor, GShare may be limiting the performance and dominating the effects of TAGE which results in high mispredic-

tions. Also for this benchmark Hybrid TAGE/Perceptron has the best performance as this benchmark is linearly separable and we have taken relative storage as 70% Perceptron and 30% GShare.

B. 605.mcf_s-472B

The results of this benchmark are similar to that of 602.gcc_s-2226B.

C. 625.x264_s-18B

As stated before in (1), this benchmark may have linearly inseperable branches, hence, we can see that Hybrid Perceptron/GShare performs the worst as perceptron has 70% storage budget, while Hybrid TAGE/GShare and Hybrid Perceptron/TAGE are comparable as TAGE has a 70% storage budget with Hybrid TAGE/Perceptron being only slightly better than Hybrid TAGE/GShare.

Trace	Best Predictor	Worst Predictor
602.gcc_s-2226B	Perceptron/GShare	TAGE/GShare
605.mcf_s-472B	Perceptron/GShare	TAGE/GShare
625.x264_s-18B	TAGE/Perceptron	Perceptron/GShare

TABLE VII: Best and Worst Performing Hybrid Predictors

V. Conclusion

In this report, we have analysed several branch prediction techniques and evaluated the performance of several benchmark programs using these techniques. We have observed that the TAGE Predictor gives the minimum MPKI for these benchmarks amongst TAGE, Perceptron/GShare since TAGE uses geometric history lengths. We have also observed that by increasing α, we can vary geomtric lengths used by the TAGE predictor which is inversely proportional to the MPKI metric, i.e, on increasing α, we observe a drop in mispredictions for the same storage budget.

We have also evaluated several hybrid predictors and found that some hybrid predictors are better for some benchmarks while not being the optimal choice for others. These hybrid predictors also are limited by their worst performing predictor hence, they might not always perform better than their individual predictor components on the same storage budget.

The evaluation of all of these predictors gives MPKI values ranging between 1 to 3 for all the traces provided. The prediction accuracy is well above 95% for most predictors, with the exception of the perceptron predictor on the trace 625.x264_s-18B (due to the presence of linearly inseparable branches).

References

[1] Hennessy, J. L., Patterson, D. A. (2019). Computer Architecture: A Quantitative Approach. India: Elsevier Science.

[2] McFarling, Scott (June 1993). "Combining Branch Predictors" (PDF). Digital Western Research Lab (WRL) Technical Report, TN-36.

[3] 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. 197-206, doi: 10.1109/HPCA.2001.903263.

[4] P. Trivedi and S. Shah, "Reduced-hardware Hybrid Branch Predictor Design, Simulation & Analysis," 2019 7th International Conference on Smart Computing & Communications (ICSCC), Sarawak, Malaysia, 2019, pp. 1-6, doi: 10.1109/ICSCC.2019.8843638.