The Effect of Process Switches on Branch Prediction Accuracy

T.Kisuki[†] H.Corporaal[‡] P.M.W.Knijnenburg[†]

[†]Department of Computer Science, Leiden University
Niels Bohrweg 1
2333 CA Leiden, the Netherlands
TEL:+31-71-5277037 FAX:+31-71-5276985
[‡]Department of Electrical Engineering,
Delft University of Technology
Mekelweg 4
2628 CA Delft, the Netherlands
{kisuki,peterk}@cs.leidenuniv.nl
heco@nidhog.et.tudelft.nl

Keywords:
Branch Prediction, Process Switch, Operating System,
Multi-threading Architecture, Simulator

Abstract

Recent superscalar processors that use deep pipelines and wide issue rates, highly depend on efficient branch prediction to obtain a large amount of instruction level parallelism. Much work has been carried out in this area and the outcome of branches is very predictable. Mostly, superscalar processors are used in a multiprogramming environment to obtain higher throughput. In this paper, we analyze the impact of process switches on prediction accuracy and show that the accuracy of branch prediction is still high in a multiprogramming environment where process switches occur in the order of milliseconds. However, in multi-threading architectures, the prediction accuracy will be severely degraded due to the high frequency of process switches.

1 Introduction

In order to fully exploit the performance of recent superscalar processors, effective branch prediction schemes are essential. To improve branch prediction accuracy, many hardware and software schemes have been proposed.

In [11] [12], two-level adaptive branch prediction is introduced. This scheme adjusts its prediction dynamically according to the behavior of the branch instructions and achieves substantially higher accuracy than other schemes.

In [5], a compiler synthesized dynamic branch prediction is proposed using profile feedback information. At compile time, explicit instructions per branch are added into compiled code to predict the direction of branches. The experiments show that the performance of this algorithm is significantly better than other branch prediction strategies.

In [2], the prediction accuracy of indirect jumps is considered. It is shown that a prediction schemes based on a branch target buffer are not effective for indirect branches. To improve the accuracy of indirect jumps, a target cache is introduced. The target cache uses branch history to distinguish different dynamic occurrences of each indirect branch. This mechanism reduce the indirect jump misprediction rate by 63.3 to 93.4 percent in SPECint benchmarks.

Using these techniques, the outcome of branches is very predictable and the accuracy of branch prediction reaches more than 97 percent. With an accurate branch predictor, a processor can keep its execution units busy and maintains a high issue rate.

In addition, to increase the total throughput, these high performance processors are mostly used in a multiprogramming environment. In this environment several processes are running concurrently and these processes are scheduled by the operating system. In the UNIX operating system, a process switch occurs every small interval of time [1]. If there are inter-process communications, this interval becomes much shorter based on the frequency of communications [7].

In such an environment, the branch predictor may be influenced since after a process switch, the contents of branch predictor tables will be flushed or most likely the information of the previous process will be used.

In [3], the impact of process switches on the accuracy of branch prediction is examined. A new hybrid branch predictor is introduced and it is shown that this hybrid mechanism is more accurate especially in the presence of process switches. Multiple branch predictors including a static predictor whose prediction accuracy is independent of process switches are used to improve the prediction accuracy. To model process switches, all history information is reinitialized. However, in a real architecture the information of the previous process will likely be used after a process switch. In our experiments, we find that the prediction accuracy is highly depend on the way of initialization of pattern history tables.

In this paper we present a detailed analysis of the two-level adaptive branch predictor in a multiprogramming environment and extend previous work in two ways. First, we examine the impact of initialization of pattern history tables in different ways. Second, we also consider the multi-threading architectures, in which even a second level cache miss may cause a process switch.

This paper is organized as follows. In Section 2, the structure of the contemporary two-level adaptive branch predictor is explained. Section 3 considers the interval of process switches in general purpose processors and multi-threading architectures. Section 4 describes the simulation method and benchmarks. The simulation results are presented in Section 5. Finally, Section 6 provides concluding remarks.

2 Branch Prediction Model

In this section, the concept of two-level adaptive branch prediction is described. A two-level adaptive branch predictor consists of two levels of branch history information [11] [12]. At the first level, the outcome of branches is kept in a register. This is called the *branch history register* (BHR). A k-bit BHR keeps the outcome of the last k branches dynamically.

The second level of the predictor keeps branch behavior for each pattern of the BHR. This is called *pattern history table* (PHT). To obtain a branch behavior an n-bit counter is used. This counter is incremented if a branch is taken and decremented if not taken. A branch is predicted as 'taken' if the value of this counter is greater than one half of its maximum value, otherwise predicted as 'not taken'. The structure of two-level adaptive branch predictor is shown in figure 1.

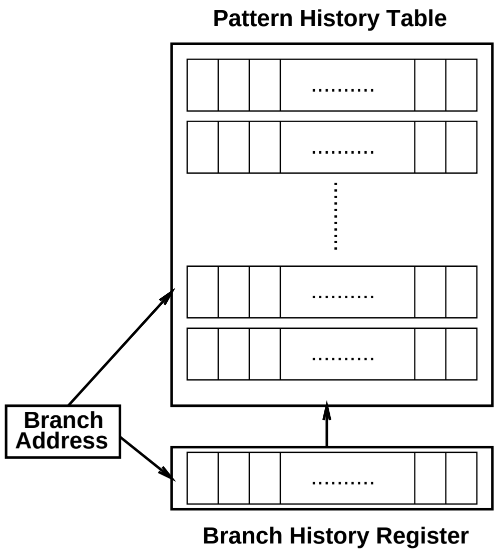

Figure 1: Structure of Two-Level Branch Predictor

There are nine variations of two-level adaptive branch predictor (see table 1). At the first level, there are three variations to keep branch history information, Global(G), per-Set(S) and Per-address(P). In the Global history scheme, only one BHR is used for every branch. The BHR is updated with the results from all the branches. In the per-Set history scheme, one BHR is associated with a set of branches. The outcome of branches from the same set is kept in each BHR. In the Per-address history scheme, the outcome of each branch is kept in its own BHR. Therefore, there is no interference among different branches. At the second level, pattern history tables also have three variations, global(g), per-set(s) and per-address(p). In the global history table, only one PHT is used by all branches. In the per-set history table, one PHT is associated with a set of branches. In the per-address history table, each entry is indexed by the branch address and each branch has own PHT.

In [12], the cost effectiveness of these branch predictor has been intensively examined. The results show that the effective implementation cost ranges from 2K to 128K bits.

In the following, GAg, GAs, SAg and SAs are

Variations	Description
GAg	Global Adaptive Branch Prediction using one global pattern history table
GAs	Global Adaptive Branch Prediction using per-set pattern history tables
GAp	Global Adaptive Branch Prediction using per-address pattern history tables
PAg	Per-address Adaptive Branch Prediction using one global pattern history table
PAs	Per-address Adaptive Branch Prediction using per-set pattern history tables
PAp	Per-address Adaptive Branch Prediction using per-address pattern history tables
SAg	Per-Set Adaptive Branch Prediction using one global pattern history table
SAs	Per-Set Adaptive Branch Prediction using per-set pattern history tables
SAp	Per-Set Adaptive Branch Prediction using per-address pattern history tables

Table 1: Variations of Two-level Adaptive Branch Predictor

considered as a suitable candidate for a branch predictor which has a 2K to 128K-bit PHT.

3 Process Switch Interval

In the UNIX operating system, a process switch occurs at fixed intervals of time, from 150 ms to 200 ms [1]. However, this value becomes more shorter if there are inter-process communications. In [7], a detailed decomposition of execution time for kernel services is presented. The characteristics of a modern UNIX operating system (Silicon Graphics IRIX 5.3) have been evaluated using three types of workloads (program development workload, database workload and engineering workload). The database workload requires many kernel services to handle inter-process communication. This results in both a high system call and a high process switching rate. Using a MIPS R4400-like processor (200MHz), a process switch occurs every 1.2 ms on average. In other workloads, the average process switch intervals are 10.9 ms and 29.4 ms.

In multi-threading architectures [6], many threads and lightweight processes are running concurrently and process switches occur very frequently. In such architectures, even second level cache misses may cause process switches. Table 2 shows the data cache hitrate, the number of process switches and the average interval of process switches where a data cache miss causes a process switch. In compress, the cache hitrate is more than 98 percent. However, the average interval of a process switch that would be caused by a cache miss is very frequent, every 143 clock cycles.

We assume that in general purpose processors, a process switch occurs between 1 ms and 200 ms. In multi-threading architectures, it happens more frequently. These time intervals in general purpose processors correspond to a number of clock cycles in a 200 MHz processor between 2×10^5 and 4×10^7. For multi-threading architectures, we set the minimum time interval 100 clock cycles considering cache misses. In such a short time interval, only a few branches occur.

	Cache Size 64K Byte	
	Hit Rate (%)	Number of Switches
LU	97.8	11261 (216 clocks)
OCEAN	93.7	45395 (74 clocks)
WATER	99.9	4186 (6348 clocks)
compress	98.8	14642 (143 clocks)

Table 2: Cache Hit Rate and Number of Process Switches

4 Simulation Methodology

In this section we discuss our methodology of measuring the impact of a multiprogramming environment on branch prediction accuracy. Since the frequency of process switches depends on a fixed time interval, precise clock cycles including pipeline stalls and cache misses are required to evaluate the impact of process switches.

4.1 Simulation Environment

In this paper, an instruction level simulator, called ISIS [9], is used. An instruction level simulator simulates the behavior of a processor on the assembler level. ISIS actually simulates a MIPS R3000 processor pipeline, instruction and data cache. Therefore, exact behavior of each pipeline state and pipeline stall can be simulated. This feature is particularly important for the study of the branch prediction considering the effect of multiprogramming environment where a process switch occurs after fixed number of clock cycles. For this experiment, branch prediction units are implemented and integrated into ISIS.

4.2 Applications

Three applications from the SPLASH benchmark suites [10] [8] and one application from SPECint95 have been selected for the evaluation. SPLASH benchmark programs are designed for multiprocessors. However, as these benchmarks contain sufficient number of branches and demand high cache performance, they are suitable for the study of branch prediction in a multiprogramming environment. Table 3 lists characteristics of these applications.

- SPLASH
  - LU
    This program factors a dense matrix into the product of a lower triangular and an upper triangular matrix. We use the non-contiguous block allocation version.
  - OCEAN
    This program studies the role of eddy and boundary currents in influencing large-scale ocean movements.
  - WATER
    This application evaluates forces and potentials that occur over time in a system of water molecules. In this study, the spatial version of WATER is used.
- SPECint95
  - compress
    Compress reduces the size of the named files using adaptive Lempel-Ziv coding. Compressed files can be restored to their original form using Uncompress.

4.3 Parameters of Simulation

In this paper, a 2-bit counter is used in the PHT. We initialize the PHT in different ways after a process switch to examine how these values affect prediction accuracy. (Note that if the value of PHT indicates 00 or 01, the branch is predicted as 'not taken', otherwise predicted as 'taken'.)

Benchmarks	Dynamic Conditional Branches (Taken:%)	Execution Cycles
LU	177756 (86.1 %)	2422913
OCEAN	122102 (96.0 %)	3354252
WATER	4249633 (54.6 %)	26570457
compress	183393 (64.7 %)	2090057

Table 3: Characteristics of Benchmark Programs

1. PHT is reset to 00, 01, 10 or 11.
2. The contents of the PHT before the process switch is used.

In a real architecture, to reset the PHT is expensive and the PHT is most likely left unchanged after a process switch. Therefore, we have included the second case also in our study where the PHT left by the previous process is used. To model second case, the PHT is set randomly after a process switch occurs. Note that in many applications, more than half of the outcomes of branches are 'taken'. Therefore, the contents of the PHT may be biased towards 10 or 11. Hence, the effect of a process switch may be less severe and this model may be pessimistic.

In the following experiment, we use these PHT sizes 2K, 8K, 32K and 128K bits. The detailed parameters of branch predictors are shown in table 4.

	Bits of BHR (PHT size: 2K, 8K, 32K, 128K)	Sets of BHR	Sets of PHT
GAg	10, 12, 14, 16	1	1
GAs	8, 10, 12, 14	1	4
SAg	10, 12, 14, 16	4	1
SAs	8, 10, 12, 14	4	4

Table 4: Parameters of Branch Predictor

5 Simulation Results

In this section, we present the results of our experiments into prediction accuracy in a multiprogramming environment. In the following experiments, the contents of BHR are flushed after a process switch. Figures 2 to 5 show the branch accuracy of our four benchmarks where the PHT is

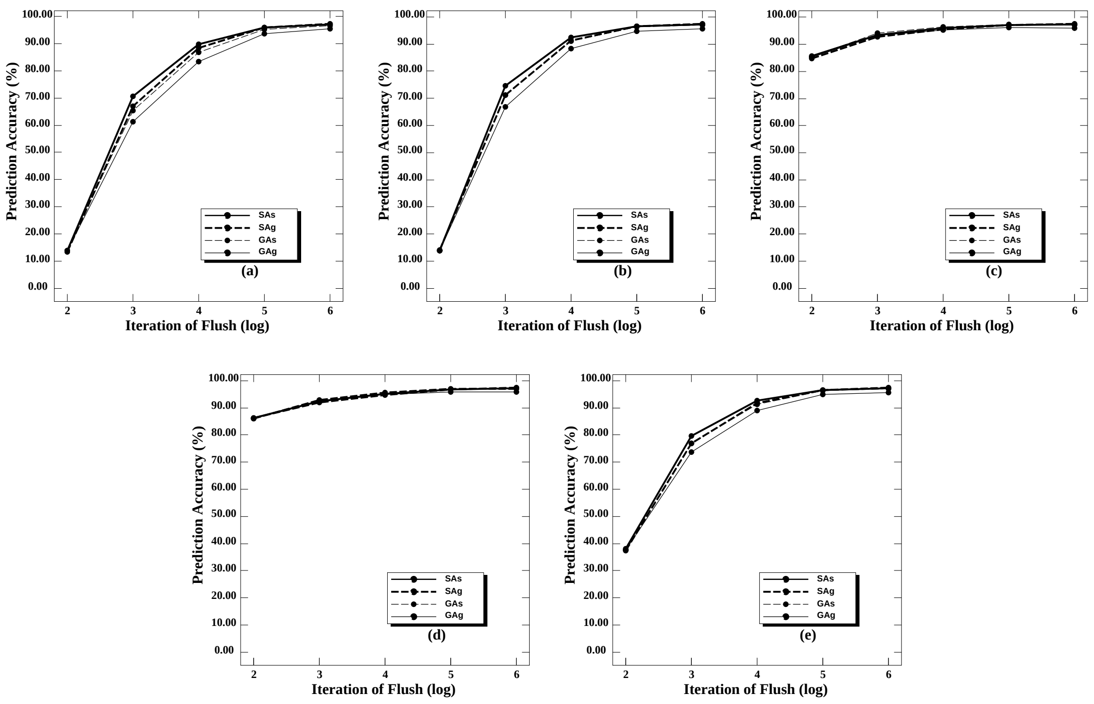

Figure 2: LU: PHT Size 8K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

initialized in different ways. The interval between process switches is set from 100 to 1000K (10000K in WATER) clock cycles. Figures 6 to 9 show the average branch prediction accuracy of four benchmarks. Finally, the results of the sensitivity on different PHT sizes are shown.

5.1 Impact of Process Switches

First, case (e) in all figures where the contents of the PHT are initialized randomly after a process switch, none of branch predictor maintains high prediction accuracy if a process switch occurs within 100K clock cycles in every benchmark. Even with larger size of the PHT, the results show the same tendency (shown in [4]).

Assume we have a 200MHz RISC processor (same clock rate as R4400), there is a no significant impact on the branch predication if the interval of process switches is longer than 0.5 ms.

In general purpose processors, a process switch occur from 150 ms to 200 ms. In this case, the results are very optimistic. Even in a database workload, where a process switch occurs every 1.2 ms, there is little impact of process switches.

However, in multi-threading architectures, the frequency of process switches is much higher. If the process switch occurs every 100 cycles due to second level cache misses, the accuracy of branch prediction is severely degraded. In this case, a branch predictor needs some scheme to reduce the impact of a process switch.

5.2 Different Initialization

The results of case (a) to (d), sets the PHT after a process switch, highly depend on the characteristic of applications. The results of LU and OCEAN are the most intriguing ones. If the value of the PHT is set to be 01 or 11, almost the entire influence of process switches is eliminated. LU and OCEAN are scientific programs and most branches are taken. Therefore, these applications are particularly predictable. However, they are also sensitive to the exact value to which the PHT is reset. If the PHT is set to 00, 01 or if the value of previous process is used, the prediction accuracy is severely degraded. In this case, the PHT should be set to 10 or 11 after a process switch.

WATER and compress behave badly and the

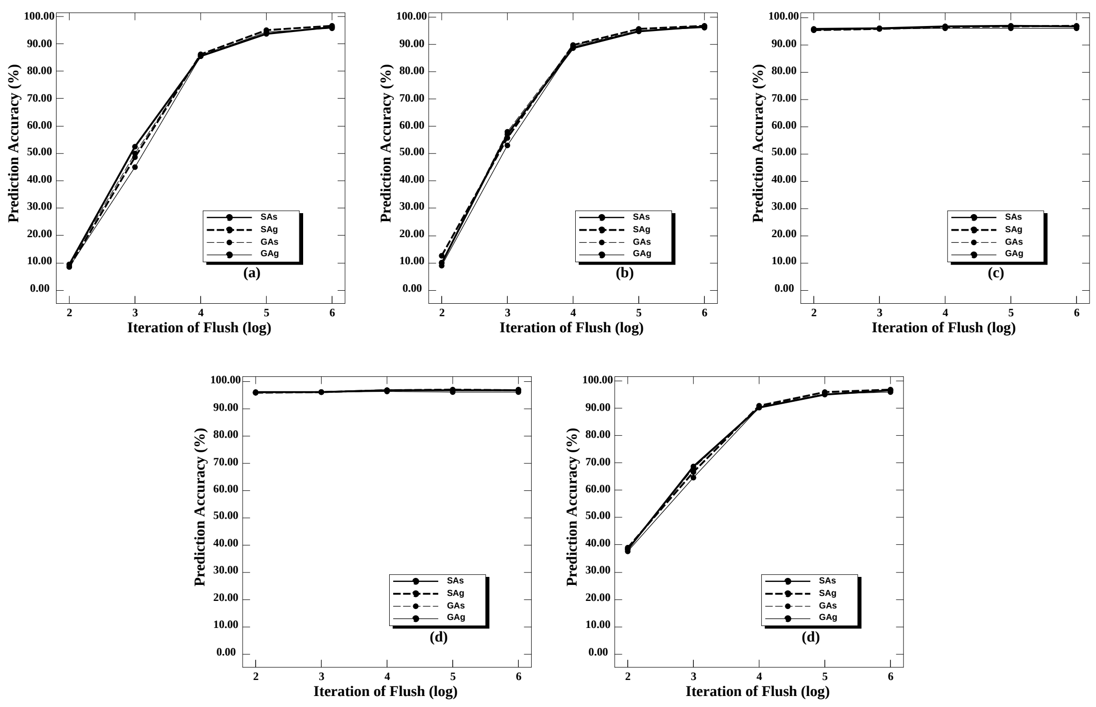

Figure 3: OCEAN: PHT Size 8K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

value of the PHT does not affect the accuracy of the branch prediction as much as in LU and OCEAN. However, in WATER the result of setting the PHT to 10 shows the best performance. In general, many branches are 'taken'. One solution to reduce the impact of process switches is setting the PHT to 10 after a process switch.

5.3 Sensitivity on Different PHT Size

Figure 10 shows the sensitivity of branch prediction accuracy. In this figure, PHT is set randomly after a process switch. Each line shows the prediction accuracy where a process switch occurs between 100 and 10K clock cycles. Remarkably, with larger sizes of the PHT, the accuracy of branch prediction drops significantly. In all schemes, branch predictors with a larger size of PHT are more sensitive to process switches, especially where a process switch occurs every 10K to 100K clock cycles. Since larger size of branch predictors need more branches to fill its table, they require more clock cycles to predict the branch outcome correctly.

6 Concluding Remarks

In this paper, we have examined the impact of a process switch on branch prediction accuracy. The performance of four variations of two-level adaptive branch predictors, GAg, GAs, SAg and SAs were evaluated.

Choosing a reasonable hardware size for a branch predictor, our results show that the influence of a process switch can be ignored if the process switch occurs less than once at every 100K clock cycles.

This results suggest that in general purpose processors, there is no significant influence of process switches on branch prediction accuracy. However, in multi-threading architectures where cache misses cause process switches, the frequency of process switches is much higher and the prediction accuracy will be severely degraded. In such cases, some scheme to reduce the impact of a process switch is necessary to maintain high prediction accuracy. According to our results, it is promising to set the PHT to 10 after a process switch. Adding hardware which sets the PHT after a process switch, eliminates almost the entire

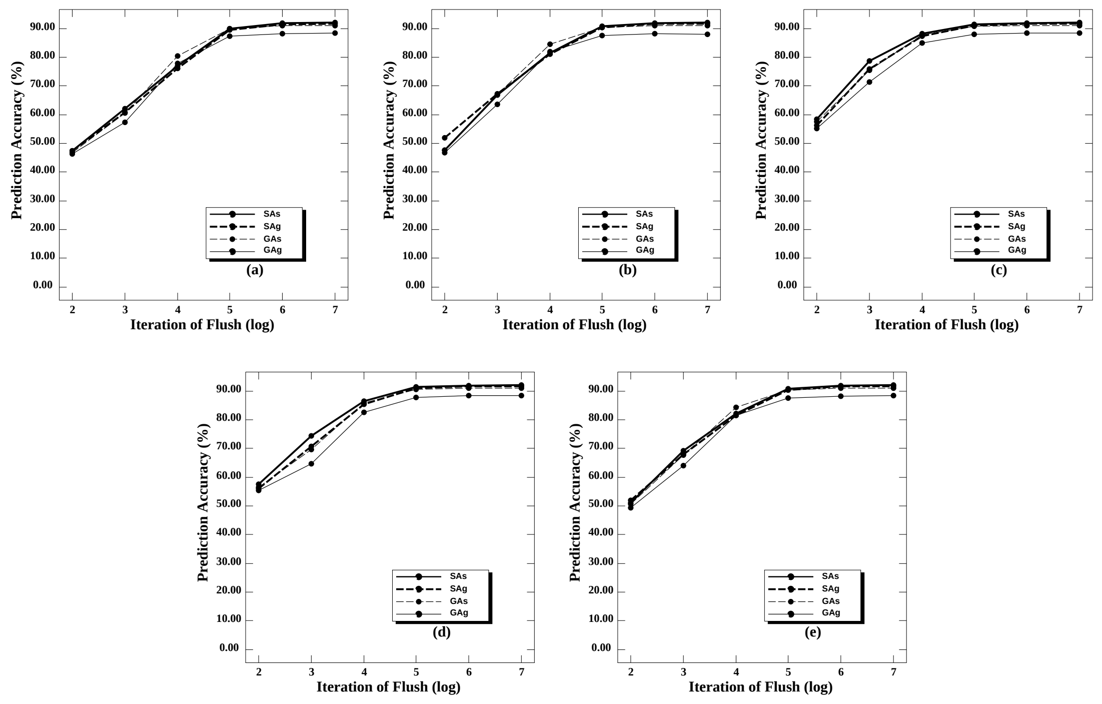

Figure 4: WATER: PHT Size 8K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

influence of a process switch in applications like LU and OCEAN. The results of sensitivity also suggest that it is effective having multiple small PHTs for a set of processes instead of having one large PHT.

Future work will address this problem and propose a simple scheme to enhance the accuracy in this situation.

Acknowledgment

I want to thank all people who took the ILP seminar given in Leiden University and Delft University of Technology. In addition, I wish to gratefully acknowledge my previous colleagues, Masaki Wakabayashi and Junji Yamamoto, who implemented the instruction simulator ISIS. This instruction simulator is or will be distributed by the Japanese Government.

References

[1] *The Compatible Time-Sharing System : A Programmer's Guide.* The M.I.T. Press.

[2] Po-Yung Chang, Eric Hao, and Yale N. Patt. Target Prediction for Indirect Jumps. *ISCA24*, 1997.

[3] Marius Evers, Po-Yung Chang, and Yale N. Patt. Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches. *ISCA23*, pages 3–11, 1996.

[4] T. Kisuki, H. Corporaal, and P.M.W. Knijnenburg. The Effect of Process Switches on Branch Prediction Accuracy. Technical Report 99–02, Department of Computer Science, Leiden University, 1998.

[5] Scott Mahlke and Balas Natarajan. Compiler Synthesized Dynamic Branch Prediction . *Proceedings of the 29th Annual International Symposium on Microarchitecture*, pages 153–164, 1996.

[6] M.L. Powell, S.R. Kleiman, S. Barton, D. Shah, D. Stein, and M. Weeks. SunOS Multi-thread Architecture. *USENIX*, 1991.

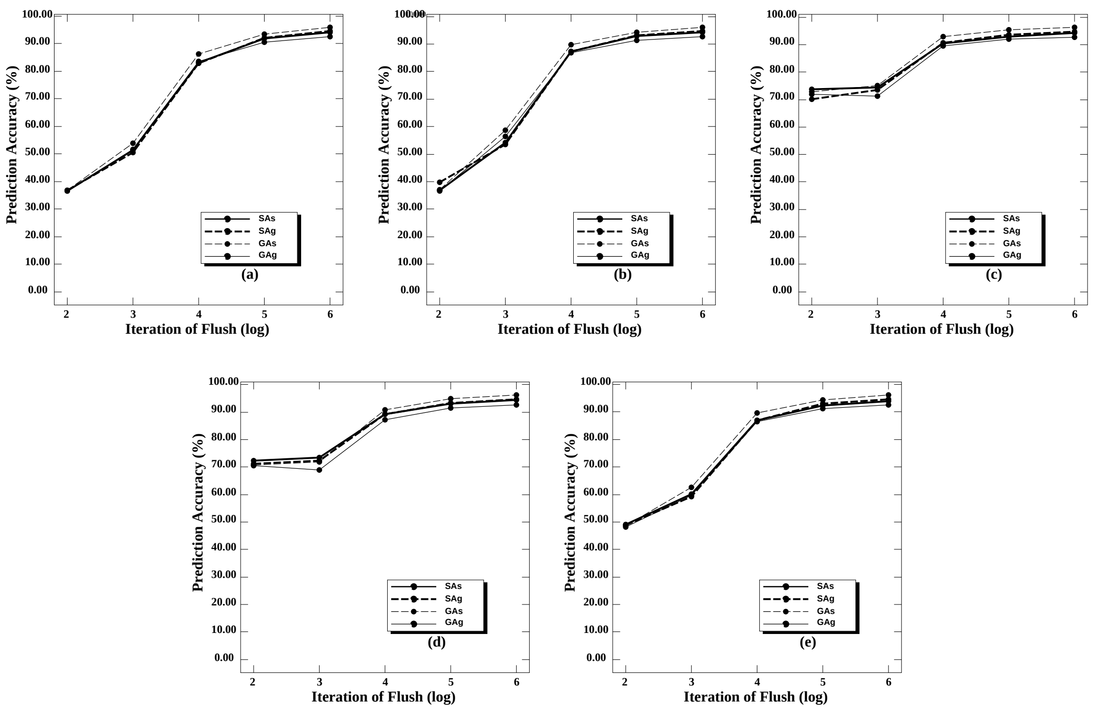

Figure 5: compress: PHT Size 8K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

[7] Mendel Rosenblum, Edouard Bugnion, Stephen Alan Herrod, Emmett Witchel, and Anoop Gupta. The Impact of Architectual Trends on Operating System Performance. *SIGOPS*, December 1995.

[8] J. P. Singh, W. Weber, and A. Gupta. SPLASH: Stanford parallel applications for shared-memory. Technical report, Computer System Laboratory, Stanford University, 1992.

[9] M. Wakabayashi. What is ISIS. *Available through* `http://www-amano.aa.cs.keio.ac.jp/isis/`.

[10] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. *ISCA*, pages 24–36, June 1995.

[11] Tse-Yu Yeh and Yale N. Patt. Two-Level Adaptive Branch Prediction. *Proceedings of the 24th Annual ACM/IEEE International Symposium and Workshop on Microarchitecture*, pages 51–61, November 1991.

[12] Tse-Yu Yeh and Yale N. Patt. A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History. *The 20th Annual International Symposium on Computer Architecture*, pages 257–266, May 1993.

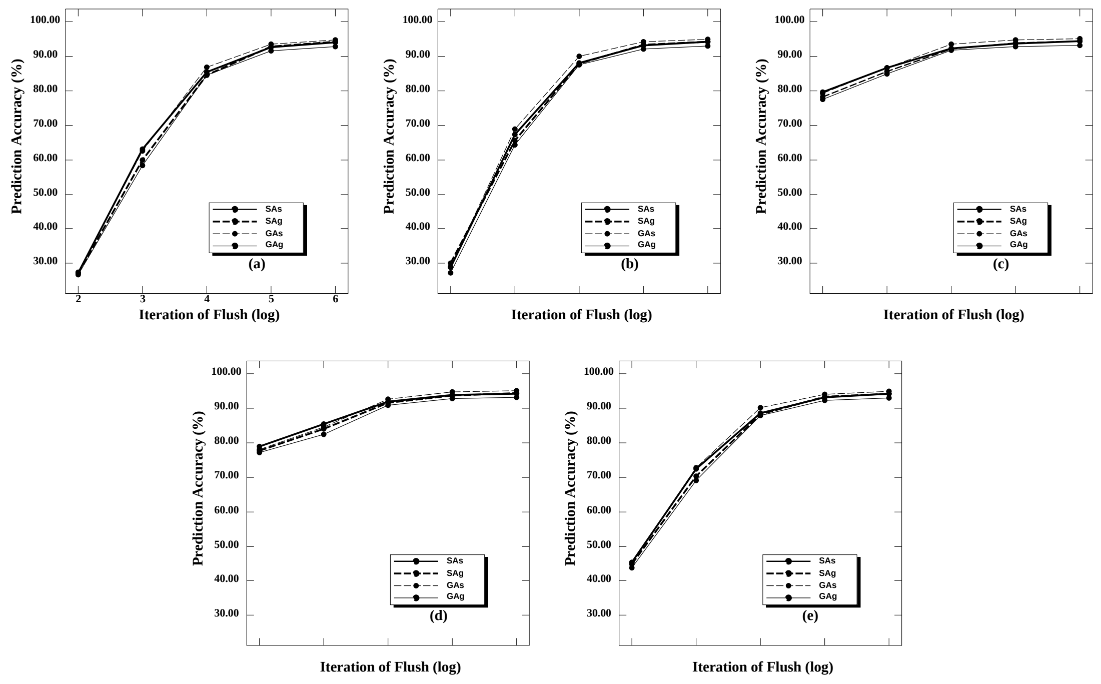

Figure 6: Average Accuracy of Four Appricaitons: PHT Size 2K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

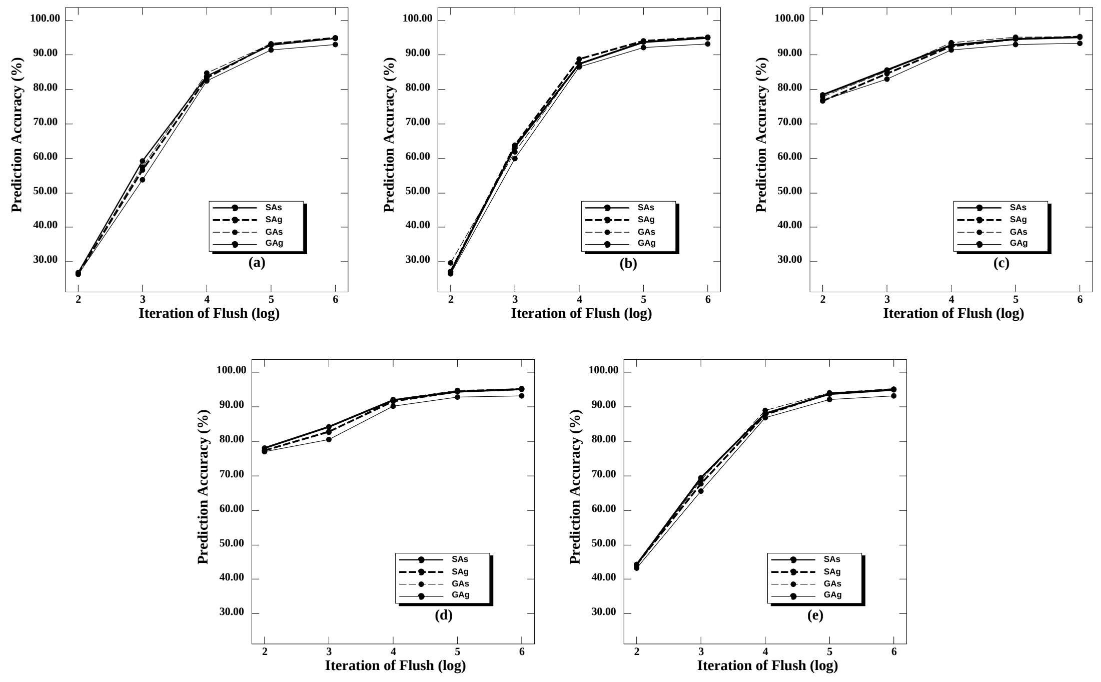

Figure 7: Average Accuracy of Four Appricaitons: PHT Size 8K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

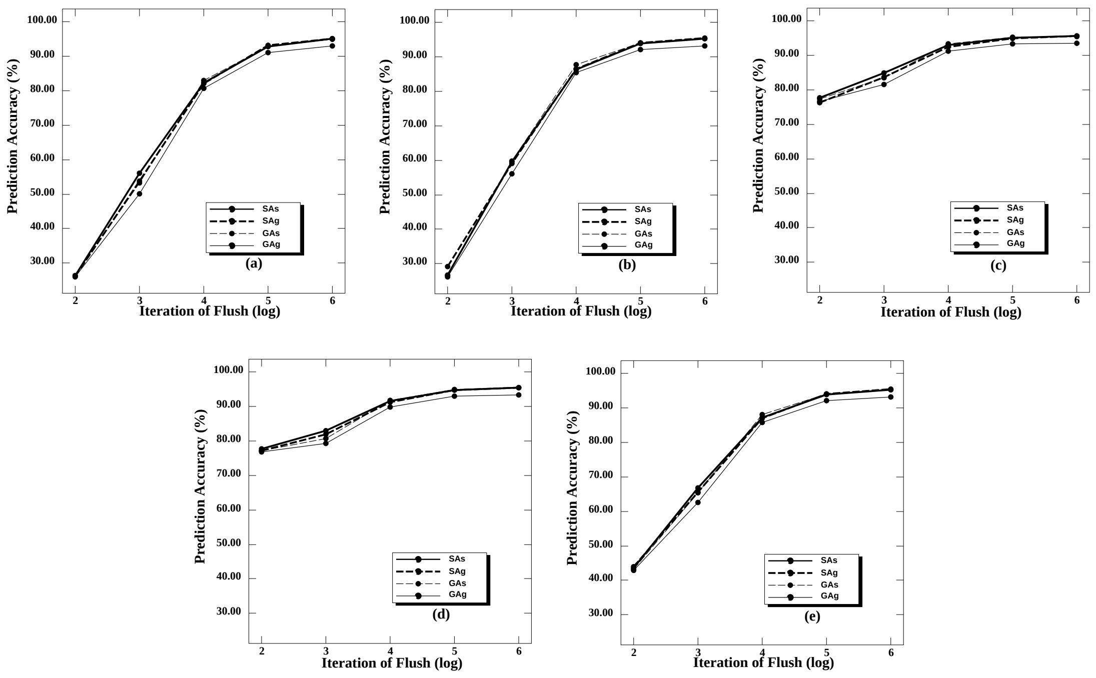

Figure 8: Average Accuracy of Four Appricationsː PHT Size 32K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

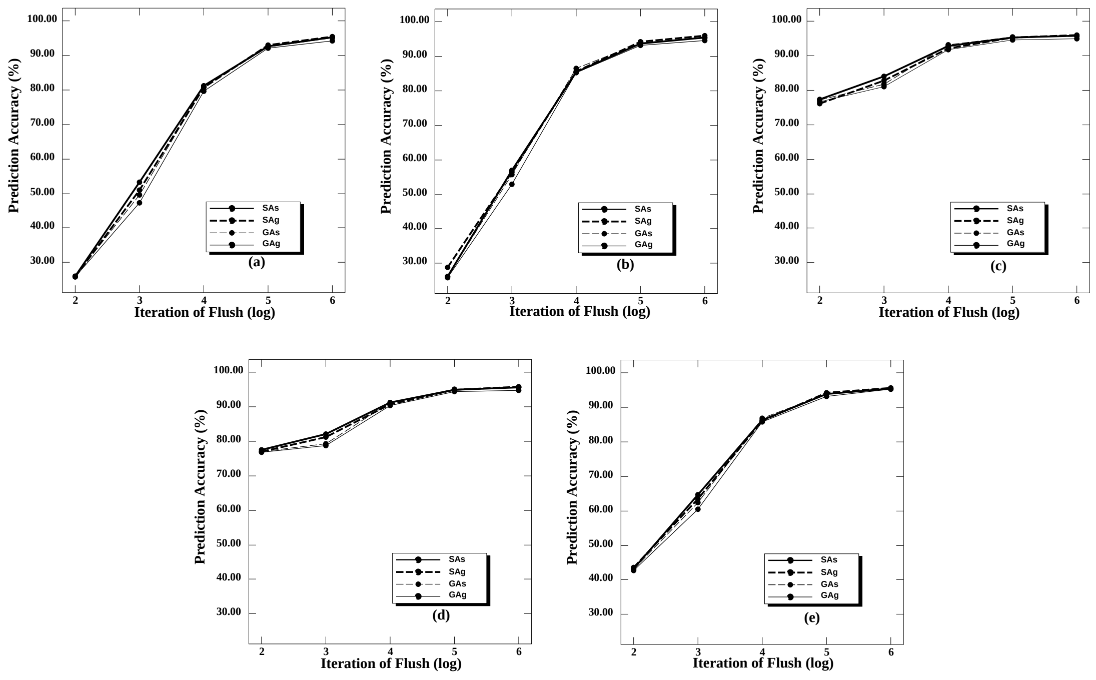

Figure 9: Average Accuracy of Four Appricationsː PHT Size 128K
(a) 00 (b) 01 (c) 10 (d) 11 (e) random

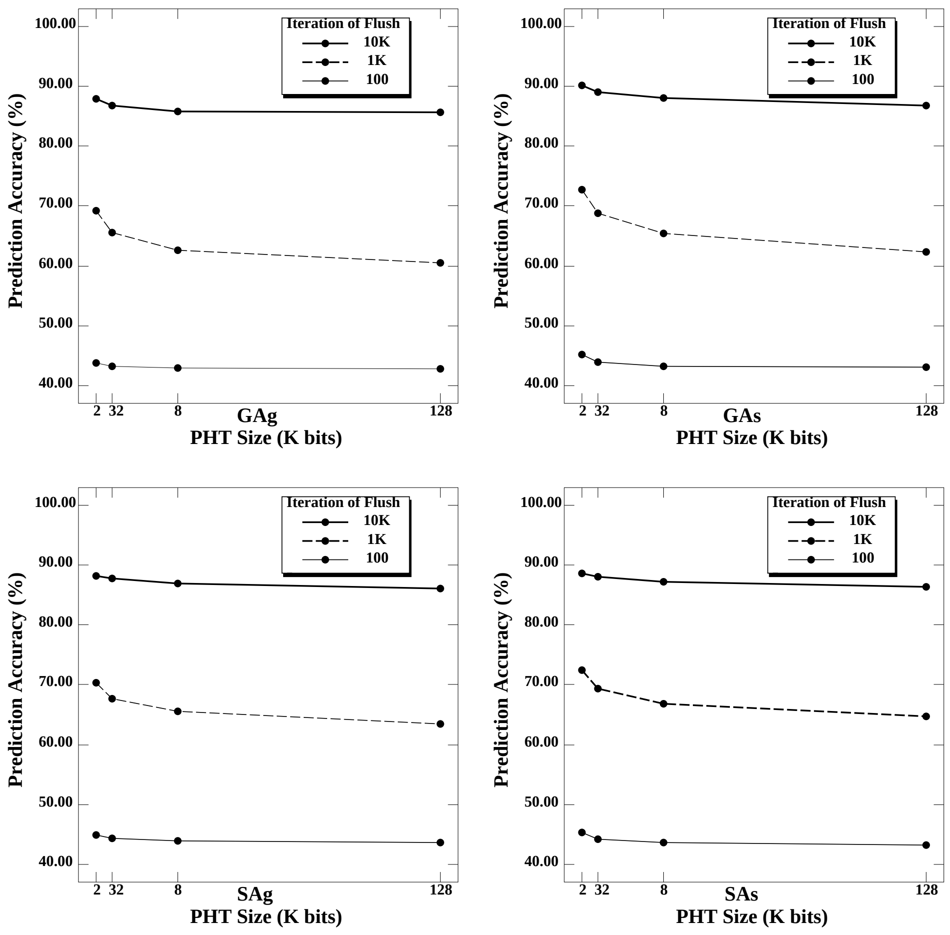

Figure 10: Sensitivity of Four Branch Predictors
(a) GAg (b) GAs (c) SAg (d) SAs