Bimode Cascading: Adaptive Rehashing for ITTAGE Indirect Branch Predictor

Yasuo Ishii
The University of Tokyo, NEC
yishii@is.s.u-tokyo.ac.jp

Takeo Sawada
The University of Tokyo
tsawada@is.s.u-tokyo.ac.jp

Keisuke Kuroyanagi
The University of Tokyo
ksk9687@is.s.u-tokyo.ac.jp

Mary Inaba
The University of Tokyo
mary@is.s.u-tokyo.ac.jp

Kei Hiraki
The University of Tokyo
hiraki@is.s.u-tokyo.ac.jp

ABSTRACT

As the success rate of branch prediction improves more and more, indirect branch prediction is becoming increasingly important. To find out what feature is important in modern indirect branch predictors, we analyzed all the workloads provided in CBP3. Our analysis shows that 10 out of 40 workloads include more than 1000 monomorphic indirect branches, which take only one destination. It also shows that polymorphic branches, which take two or more destinations, covers by more than 80% of executed branches in 8 out of 40 workloads. However, conventional predictors cannot handle both types of workloads efficiently because their configurations are fixed and non-adaptive.

In order to handle those two different types of workloads in limited hardware resources, adaptive reconfiguration is essential. We propose Bimode Cascading ITTAGE predictor (BCTAGE). The BCTAGE predictor combines adaptive rehashing with the ITTAGE predictor. The BCTAGE predictor tracks workload characteristics and adapts its configuration to the workload dynamically. In our experiments, the BCTAGE predictor improves prediction accuracy significantly.

1. INTRODUCTION

Branch target prediction is becoming more and more important as a conditional branch prediction has been becoming more and more accurate. Especially, branch target predictions for indirect branches are essential for several workloads. Generally, the indirect branches are categorized as the monomorphic branch, which takes only one target, and the polymorphic branch, which takes two or more targets. The branch target buffer (BTB) [1] is useful to predict monomorphic branch because it correlates a branch target address with an address of the indirect branch. However, the BTB cannot predict polymorphic branch correctly. The tagged target cache (TTC) [2] supports polymorphic branches by using the branch history. However, it wastes the hardware resource for the prediction of the monomorphic branch because it provides a dedicated cache entry for each branch history pattern even for the monomorphic branch.

To improve the prediction accuracy, two hybrid approaches have been proposed. One is the cascading hybrid predictor and the other approach is the adaptive rehashing. The cascaded predictor [3] employs both the TTC-like tagged component and the BTB-like base component. The BTB-like base component is used to predict monomorphic branches and only polymorphic branches are predicted by the TTC-like tagged component. This approach is also used in the ITTAGE branch predictor [4]. The ITTAGE employs multiple TTC-like tagged components to improve the prediction accuracy. As the adaptive rehashing, Rehashable BTB (R-BTB) [5] uses the BTB for polymorphic branches by employing the additional hash function that uses the branch history. When the R-BTB detects the hard-to-predict polymorphic indirect branch, the R-BTB uses the additional hash function to predict polymorphic branches correctly. Otherwise, the R-BTB worked as the normal BTB.

In this paper, we propose the Bimode Cascading ITTAGE branch predictor (BCTAGE). The BCTAGE combines adaptive rehashing to the ITTAGE. The BCTAGE tracks characteristics of the current workload and adapts its configuration dynamically. It improves the cost-efficiency and the accuracy compare to the ITTAGE.

We analyze the workload used in the third championship branch prediction (CBP3) and discuss the requirement of the indirect branch predictor in Section 2. We propose the BCTAGE branch predictor and describe its detailed design in Section 3. The implementation parameters and the budget counting are described in Section 4, and we conclude this paper in Section 5.

2. WORKLOAD CHARACTERIZATION OF INDIRECT BRANCHES

As a preliminary study, we examined the number of indirect branch instructions and the coverage ratio of polymorphic branches in the 40 workloads provided in CBP3. The coverage ratio is a proportion of the number of executed polymorphic branches to the total number of executed indirect branches.

Table 1 and Figure 1 show our analysis result. 10 out of 40 workloads, such as INT01 and WS03, are excluded from Figure 1 because the number of indirect branches in these workloads is too small (less than 200000). We categorized rest of the workloads into three categories.

Workloads in the first category have relatively large number of indirect branch instructions, and most of the branch executions are issued by monomorphic branches. It shows that the coverage ratio of polymorphic branch becomes less than 50% when the workload has more than 1000 indirect branch. For example, CLIENT10 has 10524 branches, but only 636 of them are polymorphic.

Workloads in the second category have relatively fewer indirect branch instructions, and most of the branch

Table 1: Characteristics of distributed traces for CBP3 (representative examples)

Workload name	CLIENT05	CLIENT10	INT01	INT05	SERVER01	WS03
# of indirect branches (static)	10524	3852	10	416	28	0
# of polymorphic branches (static)	636	236	2	19	11	0
# of indirect branches (dynamic)	725053	1002783	452	215588	886150	0
Coverage of polymorphic branches	42.1%	35.3%	3.3%	95.7%	84.5%	N/A

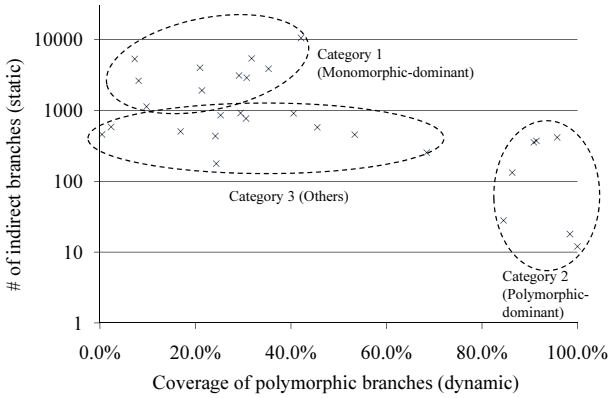

Figure 1: Correlation between the number of indirect branches and coverage ratio of polymorphic branch (workloads with >200000 indirect branches).

executions are issued by polymorphic branches. For example, SERVER01 includes only 28 indirect branch instructions and more than 80% of branch execution is polymorphic branches.

Workloads in the third category have both monomorphic and polymorphic branches.

From these observations, we concluded that adaptive reconfiguration feature is indispensable to achieve best accuracy in all three categories.

3. THE BIMODE CASCADING ITTAGE BRANCH PREDICTOR (BCTAGE)

As we investigated in Section 2, the dynamic reconfiguration such as adaptive rehashing is a cost-effective approach. We propose Bimode Cascading ITTAGE (BCTAGE) which combines adaptive rehashing to the ITTAGE. By using the adaptive rehashing, the BCTAGE adapts to both many easy-to-predict monomorphic branches and a small number of hard-to-predict polymorphic branches dynamically.

3.1 Overview

In the ITTAGE, the tagged components are optimized for polymorphic branches. The BCTAGE modifies these tagged components to support BTB-like prediction by the adaptive rehashing. The modified tagged components are called the bimode tagged components and the original tagged components are called the normal tagged components in this paper. Figure 2 shows the overview of the 6-component BCTAGE. The BCTAGE contains a BTB-like base component (labeled as base), multiple TTC-like normal tagged components (labeled as T0, T2, and T4), and multiple bimode tagged components (labeled as T1 and T3) that employ adaptive rehashing. The cascading multiplexers

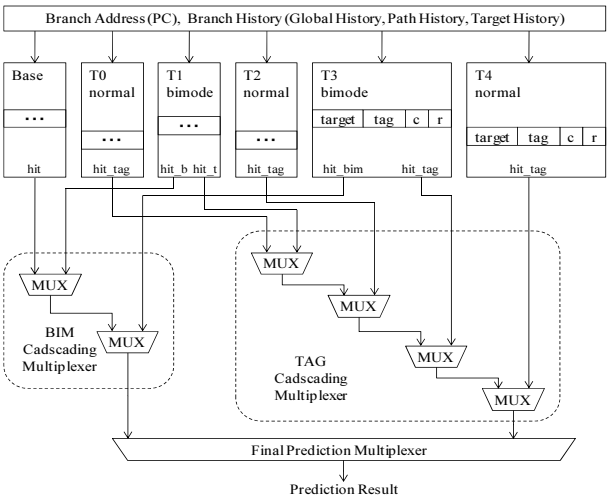

Figure 2: The BCTAGE Branch Predictor.

are used to select the prediction using the longest branch history length, which is called the longest matching.

3.2 Bimode Tagged Components

Figure 3 shows the block diagram of the bimode component for the BCTAGE. Originally, the normal tagged component has a hash pair using the branch history (labeled as TAG) to calculate the tag and the index. The bimode component employs an additional hash pair (labeled as BIM) that uses only branch address. One hash function specified by the mode register is used for the prediction. The value of the mode register is updated when the BCTAGE detects that the characteristics of the current workload is changed. When the workload contains many indirect branches, the bimode component uses the pair using only branch address to predict monomorphic branches. We call it BIM mode or BIM component. When the workload contains small number of indirect branches, the bimode component uses the pair using both branch address and branch history to predict polymorphic branches. We call it TAG mode or TAG component.

Entry of the bimode component contains four fields, which are the branch target address, the confidence counter, the partially tag, and the replacement counter. The replacement counter is called useful bit in the ITTAGE, but we change the name because it is used for the replacement information in the BIM mode.

3.3 Partially Tagged Base Components

The BCTAGE employs a set-associative partially tagged branch target buffer as the base component. Each entry holds an 8-bit partially tag. The partially tag is used to select a hit way and to track characteristics of the current

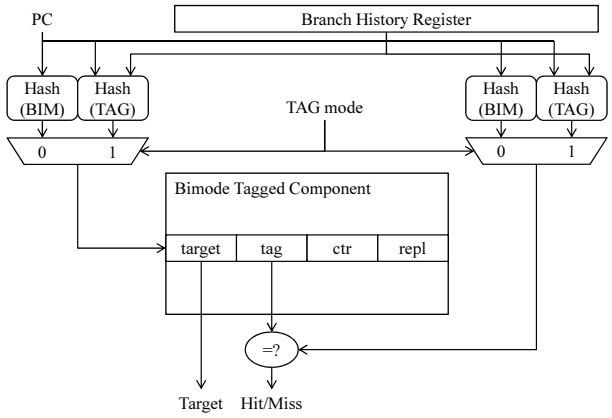

Figure 3: The Bimode Tagged Component.

workload. When no entries hit in the tag, the BCTAGE assumes that many indirect branches are recently appeared in the current workload. This feature helps to track the characteristic of the current workload.

3.4 Prediction Algorithm

All predictions are performed in the fetch stage. The base component and the tagged components are accessed in parallel. The base component is read by the branch address. The tagged components are read by the indexes that are generated by the hash functions specified by the mode register. Each tagged components provide its prediction on a tag match. The BIM components provide their predictions for BIM cascading multiplexer and the TAG components provides their predictions for TAG cascading multiplexer.

The TAG cascading multiplexer selects the longest matching prediction and the second longest matching prediction. It is known that the longest matching prediction often becomes less accurate than that of the second longest matching prediction when the longest matching prediction is generated by a newly allocated entry. In such case, the predictor uses the second longest matching prediction as the final prediction like the ITTAGE. Otherwise, the longest matching prediction is used as the prediction result. The entry whose confidence counter and replacement counter show the lowest level is used as newly allocated entry. To track the accuracy of the newly allocated entries, an 8-bit policy selection counter is employed by the BCTAGE. The update of the policy counter is described in Section 3.5.

The BIM cascading multiplexer selects a prediction using the longest tag length from predictions provided by the base component and the BIM components. Unlike TAG cascading multiplexer, the prediction using the longest tag length is always selected as the output because all corresponding components use only branch address.

The final prediction multiplexer chooses a prediction from outputs of two cascading multiplexers. The prediction from BIM cascading multiplexer is used when all TAG components miss to their tag matches.

3.5 Updating Existing Entry

An update of prediction components are performed in the retire stage. On updating the predictor, the BCTAGE searches the prediction component that makes the longest matching. Only the component providing the prediction is updated.

The confidence counter of the updating entry is incremented when the target address is correct. Otherwise, the confidence counter is decremented. The target address is updated when the confidence counter shows the lowest level. On updating the target address, the branch outcome is simply written to the corresponding entry. The update of the replacement counter is different between the TAG mode and the BIM mode. The replacement counter is updated when the second longest matching prediction is different from the prediction of the longest matching component. When the prediction of the longest matching component is correct, the replacement counter is incremented. When the second longest matching prediction is correct, the replacement counter is decremented. When the longest matching entry is newly allocated entry and the replacement counter is updated, the policy selection counter is updated. The policy selection counter is incremented when the longest matching prediction is correct. The policy selection counter is decremented when the longest matching prediction is wrong. The update of the replacement counter of the BIM component is different from that of the ITTAGE. The replacement counter of the BIM component is always incremented because the counter shows only a priority of the victim selection in the BIM components.

The base predictor is updated when all tagged components misses in the tag matching. When the base predictor is updated, the target address and the confidence counter are updated as performed in the tagged component. The replacement counter is incremented as performed in the BIM component.

3.6 Allocating New Entry

On a misprediction, the BCTAGE allocates new tagged component for future prediction. The allocation policy is derived from ITTAGE. The predictor scans entries of TAG components that use longer history length than that of the longest matching component. The predictor selects one out of those candidates. Selected entry is then initialized. The partially tag and the target address are initialized to the branch outcome. The other fields are initialized to the lowest level.

When there are no BIM components that hit in the tag matching, the predictor also allocates BIM components. The predictor selects a victim entry from multiple BIM components. The prediction entry whose replacement counter is the lowest in the candidates is replaced by the new entry. The new entry is initialized as performed in the TAG components.

The base predictor also allocates new entry on a misprediction. When no entry hit in partially tags, an entry whose replacement counter is the smallest is replaced by the new entry. The new entry is initialized as performed in the TAG and BIM components.

3.7 Workload Detection

As discussed in the previous section, the number of the indirect branches correlates with the coverage ratio of the polymorphic branches. Therefore, the BCTAGE tracks the number of the indirect branches to detect the workload characteristic. Two performance counters are used for tracking the characteristics of the current workload. The number of mispredictions due to capacity shortage of the

Table 2: Configuration of Tagged Components (history length = 0 indicates BIM mode).

	T0,T1	T2,T3	T4,T5	T6,T7	T8,T9	T10,T11	T12,T13	T14,T15	T16,T17	T18
# of Entry	256	512	1024	512	512	512	512	256	256	256
Tag Width	13	14	15	16	17	18	19	21	22	23
Hist. Length (easy)	5,6	8,10	12,15	19,24	30,37	47,59	73,92	115,144	180,225	281
Hist. Length (normal)	5,0	8,10	12,0	19,24	30,0	47,59	73,0	115,144	180,0	281
Hist. Length (hard)	5,0	8,0	12,0	19,0	30,0	47,0	73,0	115,0	180,0	281
Table Cost (Kbit)	12.25	25.00	51.00	26.00	26.50	27.00	27.50	14.25	14.50	14.75

base and BIM components is counted as the metric. When the performance counters show a large value, large amount of BIM components are required.

The performance counters are update on a misprediction. The performance counters are increased when all base and BIM components fail to tag matching. In this case, the BCTAGE assumes that the capacity shortage of the BIM components causes the misprediction. On the other hand, the performance counters are decreased when the base component hit to the partially tag. In this case, the BCTAGE assumes that more TAG components are required because the base prediction is already provided by the base component. When the counter values are overflow, the BCTAGE allocates more bimode tagged components to the BIM components because the current workload contains many indirect branches that should be predicted by the BTB-like predictor. The BCTAGE allocates more bimode tagged components to the TAG components when the counter values are underflow.

The BCTAGE uses three modes in this work. When there are many indirect branches, the predictor assigns all bimode tagged components to the BIM mode, which is called a hard mode. When there is small number of indirect branches, the predictor assigns all bimode tagged components to TAG mode, which is called an easy mode. Otherwise, the predictor uses a half of the bimode tagged components as BIM components, which is called a normal mode. The mode state is written to the mode register. The mode register is used to switch the hash function for the prediction as shown in Figure 3.

4. IMPLEMENTATION

We implement 20-components BCTAGE (one base component, ten normal tagged components, and nine bimode tagged components) for the competition. Table 2 shows the configuration of the tagged components.

We use 281-bit global history length for the BCTAGE. The global history contains branch direction, path history, and target history. All information for branch history is duplicated for speculative updating. One is updated in the fetch stage and the other is updated in the retire stage. The branch history updated in the fetch stage is used for the prediction. The branch history updated in the retire stage is used for the updating the predictor. The detailed budget count is shown in Table 3.

5. CONCLUSION

The amount of monomorphic and polymorphic branch executions varies from workload to workload. To support both monomorphic-dominant and polymorphic-dominant workloads in limited hardware resources, indirect predictors should adapt their configuration to running workload.

Table 3: Configuration and Budget Count.

Resource	Configuration	Budget
Global History	281-bit branch direction × 2	562
	281-bit path history × 2	562
	281-bit target history × 2	562
	Path reg. (32-bit) × 2	64
Base Predict.	1280-entry, 5-way BTB, 8b tag 32b addr, 2b ctr, 2b repl info.	56320 (55.00K)
Tagged Comp.	19-components (see Table 2) 32b addr, 2b ctr, 2b repl info.	473856 (462.75K)
Others	Policy selection counter (8-bit)	8
	Prediction mode reg. (2-bit)	2
	Perf. counter (16-bit) × 2	32
	Random seed (16-bit)	16
Total		531984

In this paper, we propose the Bimode Cascading ITTAGE branch prediction (BCTAGE). The BCTAGE predictor has base and BIM components for monomorphic branch prediction, and TAG components for polymorphic branch prediction. BIM components and TAG components can be dynamically reconfigured according to current workload. It effectively supports both monomorphic-dominant workloads and polymorphic-dominant workloads in limited budget resources.

6. REFERENCES

[1] J. K. F. Lee and A. J. Smith, "Analysis of branch prediction strategies and branch target buffer design," Tech. Rep. UCB/CSD-83-121, EECS Department, University of California, Berkeley, Aug 1983.

[2] P.-Y. Chang, E. Hao, and Y. N. Patt, "Target prediction for indirect jumps," in *Proceedings of the 24th annual international symposium on Computer architecture*, ISCA '97, pp. 274–283, 1997.

[3] K. Driesen and U. Hölzle, "The cascaded predictor: economical and adaptive branch target prediction," in *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*, MICRO 31, pp. 249–258, 1998.

[4] A. Seznec and P. Michaud, "A case for (partially) tagged geometric history length branch prediction," *The Journal of Instruction Level Parallelism*, vol. 8, Febrary 2006.

[5] T. Li, R. Bhargava, and L. K. John, "Rehashable btb: An adaptive branch target buffer to improve the target predictability of java code," in *Proceedings of the 9th International Conference on High Performance Computing*, HiPC '02, pp. 597–608, 2002.