Vol.2016-SE-192 No.11
Vol.2016-EMB-41 No.11
2016/6/3

A Comprehensive Analysis of Branch Prediction in Multi-core Processors

Lin Meng[1] WenWen Wang[2] Shigeru Oyanagi[3]

Abstract: Branch prediction is an effective method for improving performance of the processor. This paper focus on the comprehensive analysis of branch prediction in multi-core processors for considering the effective of branch prediction to the performance of multi-core processors. The experimental items of branch prediction include the kinds, multi-thread numbers of the applications. These experiments use the Zsim simulator and using the benchmark of Parsec.

1. Introduction

Currently multi-core processors which have 2 or more cores, is wildly used. The current general core of multi-core processors is a processor, which uses pipelining and superscalar processor technique. These cores have the function of fetch, decode, execution, memory access and writeback. Hence, the the branch prediction is an effective method for reducing the control hazard and improving performance of the processor .

This paper focus on the comprehensive analysis of branch prediction in multi-core processors for considering the effective of branch prediction to the performance of multi-core processors. The experimental items of branch prediction include the kinds, multi-thread numbers of the applications. These experiments use the Zsim simulator and using the benchmark of Parsec.

2. Related Work

2.1 Multi-core Processors

Multi-core processors are a single computing component with two or more independent actual processing units. The unit is called core too. The current general core is a processor, which using pipelining and superscalar processing technique. the core has the function of fetch, decode, execution, memory access and writeback. Between the every cores, the multi-core have a shared cache for sharing the data.

The typical multi-core processes include the Intel Xeon, Sun UltraSPARC, AMD Opteron and IBM Cell.

Intel Xeon (Clovertown)[1]: The Clovetown is the Quad-Core of Xeon Processor 5300 Series. The processor have 4 cores, 32KB-instruction and 32KB-data cache as L1 cache in per core. Figure 1 shows the configuration of Xeon 5300 series.

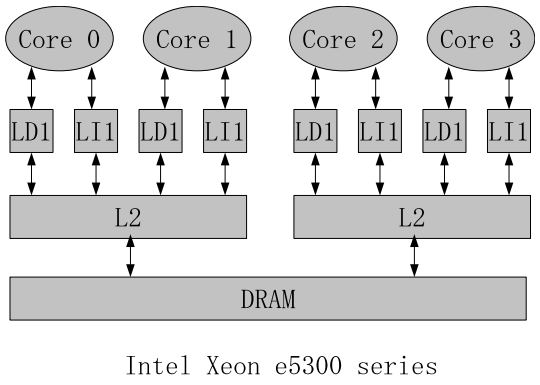


Fig. 1 Intel Xeon e5300 series.

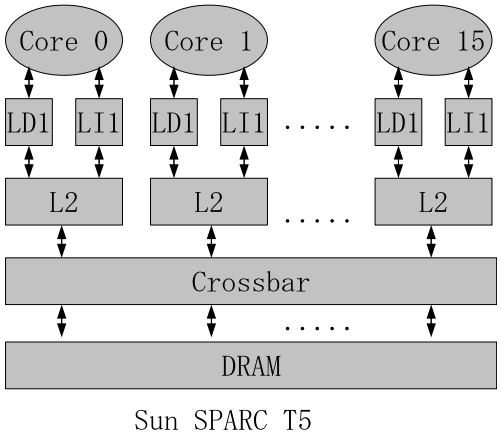


Fig. 2 Oracle's SPARC T5 series.

Sun SPARC (T5)[2]: The Sun SPARC T5 processor is a single chip multi-core processor and contains 16 cores processor. Each core processor has full hardware support for eight strands, two integer execution pipelines, one floating-point execution pipeline, and one memory pipeline. Figure 2 shows the configuration of Oracle's SPARC T5.

AMD Opteron 6200[3]: The ADM Opteron 6200 processor up to 16 Bulldozer cores. L1 instruction cache and L2 cahche are

[1] Department of Electronic and Computer Engineering, Ritsumeikan University,Kusatsu, Shiga, Japan
[2] Department of Computer Science and Engineering, University of Minnesota at Twin Cities, Minneapolis, Minnesota, USA
[3] College of Information Science and Engineering, Ritsumeikan University,Kusatsu, Shiga, Japan

shared between the cores. Every core has a private data cache L1. L3 cache is shared at the chip level.

IBM Cell[4]: Cell processor is a kind of heterogeneous processor which has a PPE（PowerPC Processor Element）as general-purpose processor for control the Operating system and several Cell Broadband Engines（CBE） for the execution of Fast real-time processing of multimedia data.

2.2 Branch Prediction

Control hazard is a serous problem to improve the performance, which is cased by branch instructions in current processors. Branch prediction is an effective method to reduce control hazard for improving the performance of the processors. Current branch predictor uses caches to keep the history of TakennotTaken for predict the jump direction of branch, and lots of these caches use global history to utilize the correlation among recently executed branches.

Despite the steady improvements that have been made, it is difficult to completely avoid conflict aliasing, hence many branches are still miss-predicted.

With increasing the caches size and complexity, the predict accuracy can be improved, however the power consume problem become more serous.

This paper aim to search for the optimal branch predictor for Multi-core processors by analyzing the branch predictor Here we survey and list some current predictors. The predictors who keeps one caches are call single predictor, the others are call hybrid predictor [5].

2.2.1 Single predictor

Bimodal[6]: Bimodal predictor is the first branch predictor which was proposed in 1981. Bimodal predictor uses a 2 bits saturating counter to keep the behavior of branch. These 2 bits saturating counters configure the PHT (Pattern History Table) which is indexed by the lower branch instruction address bits. Because Bimodal predictor uses 2 bits saturating counter and lower branch instructions address, the prediction just utilize the correlation among the last 4 branch histories of the predicted branch. Bimodal branch is very simple, predict the branch quickly. However if there are branch with the same lower branch instruction address bits is executed, the entry of the predictor will be polluted, and miss-prediction will happen. In a words, Bimodal prediction is polluted very easily. Figure 3 configuration of Bimodal predictor.

Gshare[7]: Gshare predictor is widely used in current processors. Gshare predictor uses the exclusive OR of GBH (global branch history) and branch address to make the PHT index. One of the main reasons of miss-predictions is conflict aliasing, which is caused by the different branches accessing the same PHT entry. Because it uses the GBH, warming up time for GBH is necessary. Hence,when the processor change the application or change the context frequently, the prediction accuracy will reduced. Figure 3 configuration of Gshare predictor.

2.2.2 Hybrid predictor

The widely used base predictors are Bimodal and Gshare predictors. Several predictors using this approach as hybrid predictors are shown as follows.

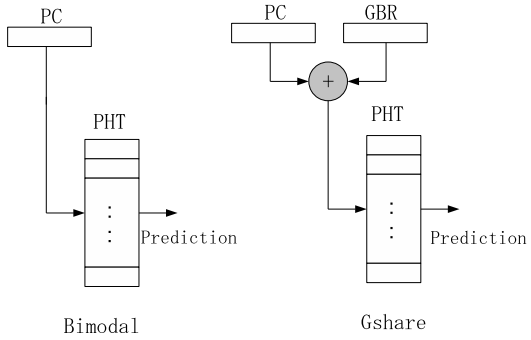


Fig. 3 Single Branch Predictors.

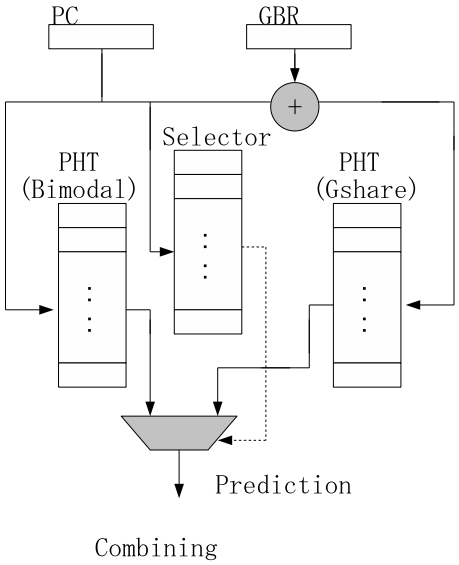


Fig. 4 Combining Branch Predictors.

Combining[7]: Combining predictor consists of a Bimodal predictor which works well for local history and a Gshare predictor which works well for global history. A selector is constructed by 2 bits saturating counter to select the result from either Bimodal predictor or Gshare predictor. Figure 4 configuration of Combining predictor.

Bimode[8]: Bimode predictor is consists of two Gshare predictors and a selector predictor. one of the Gshare predictors for the branches biased toward Taken (Taken Gshare predictor), and the other for the branches biased toward NotTaken (NotTaken Gsahre predictor). The selector predictor uses ChoicePHT to choose the result from two Gshare predictors. Bimode predictor can reduce conflict aliasing by dividing biased branches into different PHTs.

Bimode-Plus[11]: Bimode-Plus uses ones of branch behaviors which is that some branches are strongly biased toward one direction (Taken or NotTaken) until the program finishes. Bimode-Plus predictor provides a Bias Table which keeps Taken bit and NotTaken bit to detect the branches which are strongly biased toward Taken or NotTaken. When the strongly biased branches are detected, the predictor uses Bias Table without using the result of Bimode predictor nor updating into Bimode Predictor. By this way, Bimode-Plus predictor reduces the conflict aliasing between the strongly biased branches and normal branches.

Agree[10]: Agree predictor keeps the Taken biased bit and

Table 1 Processor configuration

Issue	6-issue
Cores	x86 Out-of-Order cores
Line size	64
L1I	32KB, 4-way-set associate, 3 latency
L1D	32KB, 8-way-set associate, 3 latency
L2I	2MB, 8-way-set associate, 7 latency
L2D	2MB, 8-way-set associate, 7 latency
L3	12MB, 6-way, 27 latency
mem	DDR3-1066-CL8, hash-h3, 4 bank

NotTaken biased bit in BTB. PHT keeps the result whether the prediction is same to the biased bit. When the branch result is same to the biased bit, the entry of PHT is incremented, otherwise it is decremented. Exclusive OR of the biased bit and the PHT result is used for prediction.

Hybrid[9]: Hybrid predictor keeps several predictors (2bc, Gshare, GAS, AVG) to predict the branch. BTB keeps the result of each predictor, and is used at prediction to select the most accurate one among the several predictors.

TAGE,L-TAGE[12], [13]: These predictors use PPM (prediction by partial matching) to search the patterns in the branch direction history[14]. This method has 4 PHTs which are accessed by the exclusive OR of different parts of GBHR and branch address. Every PHT entry has a tag (a part of branch address) to protect the conflict. Hence, this method uses the pattern of GBHR to improve the prediction accuracy.

3. Analyzing the Branch Prediction in Multi-core Processors

3.1 Experimental condition

These experiments use the Zsim simulator and the benchmark is Parsec. Zsim is an fast and accurate microarchitectural simulation of thousand-Core systems. Zsim introduces several simulation techniques that significantly speed up both single and multi-thread performance, enabling fast and accurate simulation of large-scale systems [15].

The PARSEC (Princeton Application Repository for Shared-Memory Computers) is a benchmark suite composed of multi-threaded programs. The PARSEC benchmarks satisfy the Multi-threaded Application, Emerging Workloads, Diversity, State-of-Art Algorithms and Research Support for chip-multiprocessors [16].

3.2 Analyzing by the thread numbers

We change the thread numbers of the benchmark to get the experimental results about the IPC(Instruction Per Cycle) and MPKI (Miss-prediction in 1K instruction). Figure 5 shows the default branch predictor in Zsim, the first Level BHSR (branch shift register) is consisted as Bimodal and the second level PHT is consisted as Gshare predictor. We use the default size of the branch predictor.

Figure 6 shows the IPC and MPKI in the case of 4-thread, Figure 7 shows the IPC and MPKI in the case of 8-thread, We found that when the MPKI is larger, the IPC will become lower. In the case of 4-thread, the MPKI of dedup, ferret, fluidanimate and x264, MPKI are different, especially in the fluidanimage the difference are very large.

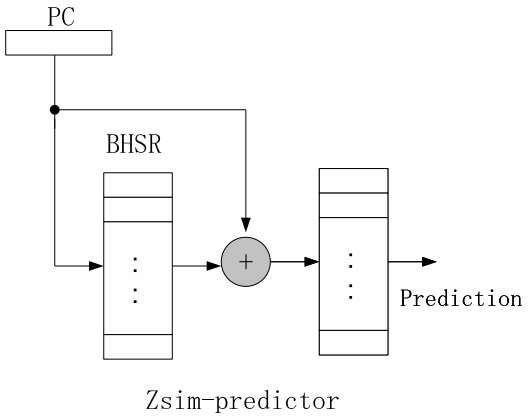


Fig. 5 Default predictor in Zsim.

In the case of 8-thread, the differences of MPKI also disappear in the benchmark of facesim, ferret. It means that when the thread number is increased, the executed path are changed, causing the miss-prediction increases.

Because benchmark swaptions just uses one core, hence the MPKI and IPC are 0 in the core1, core2 and core3.

3.3 Analyzing on the several branch predictors

We equip the Gshare predictor and Bimodal predictor into Zsim for analyzing the relationship between the performance and branch predictor in Multi-core processor.

Figure 8 shows the experimental results about IPC and MPKI by using Bimodal predictor, Figure 9 shows the experimental results about IPC and MPKI by using Gshare predictor.

By comparing the MPKI between Figure **??**, Figure 8 and Figure 9, we found default predictor is better than Gshare and Bimodal predictor. However, the default predictor is not the best predictor, because that Bimodal predictor is better than default predictor in benchmark freqmine, Gshare predictor is better than default predictor in benchmark facesim.

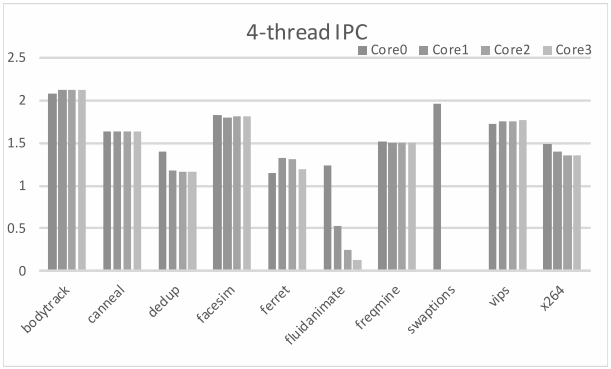
From the experimental results, we find that in the same application

- the MPKI of every cores are different
- when increasing the thread number, the MPKI will increase.
- the MPKI may different by using the different branch predictor, and the best predictor are not exist.

Hence, by these conclusions we aim to equip heterogeneous branch predictors in the multi-core processors for improve the performance of the processor. The heterogeneous branch predictor means, in the multi-core processor, the branch predictors are different in the cores.

4. Conclusion

Branch prediction is an effective method for improving performance of the processor. This paper focus on the comprehensive analysis of branch prediction in multi-core processors for considering the effective of branch prediction to the performance of multi-core processors. The experimental items of branch prediction include the kinds, multi-thread numbers of the applications. These experiments use the Zsim simulator and using the benchmark of Parsec. From the experimental results, we find that in the



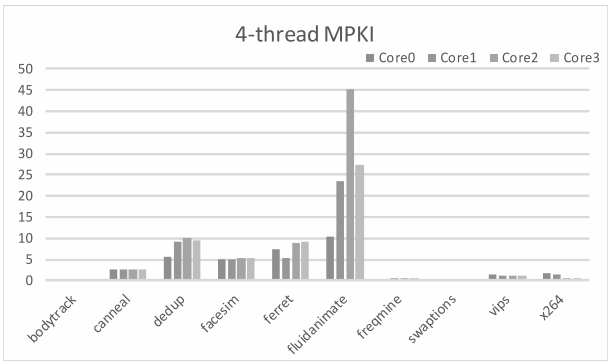
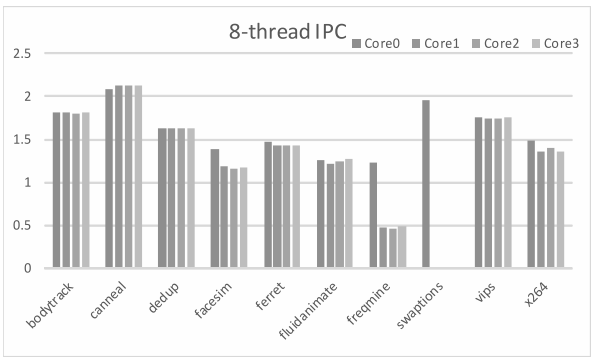


Fig. 6 Experimental results on 4-thread.



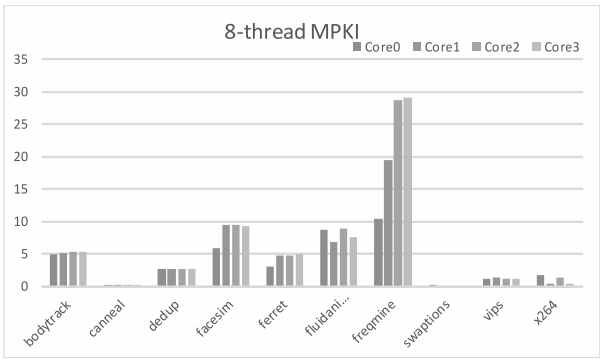
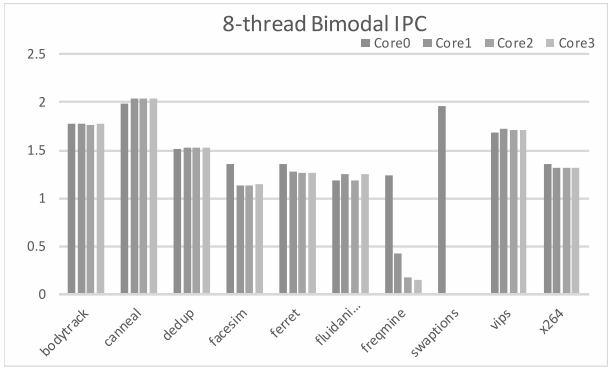


Fig. 7 Experimental results on 8-thread.



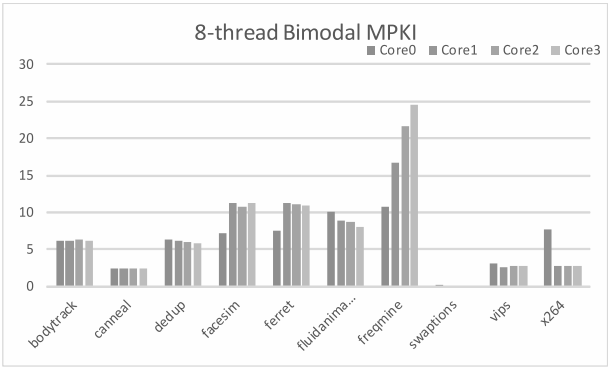
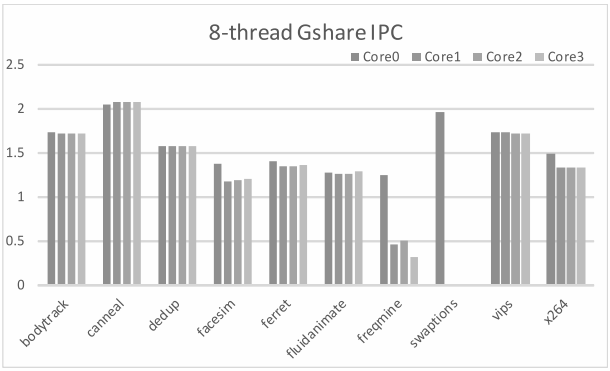


Fig. 8 Experimental results by using Bimodal Predictor.



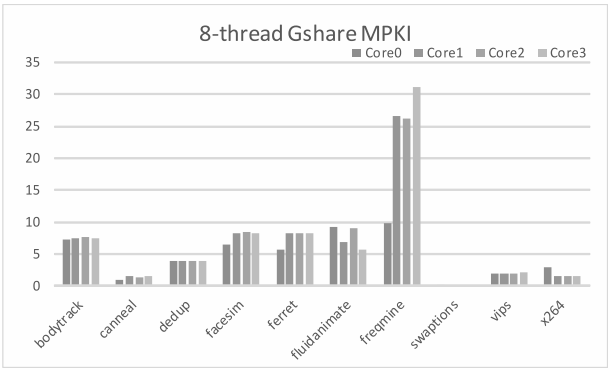


Fig. 9 Experimental results by using Gshare Predictor.

same application, the MPKI of every cores are different, when increasing the thread number the MPKI will increase. the MPKI may different by using the different branch predictor, and the best predictor are not exist. Future work is equipping a heterogeneous branch predictors in the multi-core processors for improve the performance of the processor.

Acknowledgments This work was supported by JSPS KAKENHI Grant Numbers 15K00163 (Grant-in-Aid for Scientific Research C) and 26870713 (Grant-in-Aid for Young Scientists B).

References

[1] *Quad-Core IntelR XeonR Processor 5300 Series datasheet*, 1993.
[2] *An Oracle White Paper: Oracle's SPARC T5-2, SPARC T5-4, SPARC T5-8, and SPARC T5-1B Server Architecture*, 2014.
[3] *AMD Opteron 6200 Series Processor*.
[4] Y. KUROSAWA, Y. Watanabe, and H. Tago : *Cell Broadband Engine Next-Generation Processor*, Toshiba Review, Vol.61, No.6 (2006).

[5] L.Meng, K. Yamazaki, S. Oyanagi: *A Novel Branch Predictor Using Local History for Miss-Prediction*. Proceedings of the International Conference on Computer Design (CDES), 2012.
[6] J.E.Smith: *A Study of Branch Prediction Strategies*, Proc. of 8th International Symposium on Computer Architecture (ISCA), pp.135-148, 1981.
[7] S.McFarling: *Combining Branch Predictors*, Technical report TN-36, Digital Western Research Laboratory,1993.
[8] Chih-Chieh Lee, I-Cheng K. Chen and Trevor N. Mudge: *The bi-mode branch predictor*, Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture, pp.4-13, 1997.
[9] M.Evers, P-Y.Chang and Y.N.Patt, : *Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches*, Proc. of 23th International Symposium on Computer Architecture (ISCA), pp.3-11, 1996.
[10] E.Sprangle, Robert S. Chappell, Mitch Alsup and Yale N. Patt, : *The Agree Predictor: A Mechanism for Reducing Negative Branch History*, Proc. of 24th International Symposium on Computer Architecture (ISCA), pp.284-291, June 1997.
[11] K.Kise, T.Katagiri, H.Honda and T.Yuba,: *The Bimode-Plus Branch Predictor*, IPSJ Trans.ACS-10, pp.85-102,2005.
[12] A.Seznec,: *The L-TAGE branch predictor*, The 2nd JILP Championship Branch Prediction Competition (CBP-2), vol.9, 2007.
[13] M.pierre, : *A PPM-Like, tag-based predictor*, The 1st JILP Championship Branch Prediction Competition (CBP-1), vol.7, April 2005.
[14] I.-C.K.Chen, J.T.Coffey and T.N.Mudge, : *Analysis of branch prediction via data compression*, ACM International Conference on Architectural Support for Programming Languages and Operating Systems-VII, pp.128-137, 1996.
[15] S. Daniel and K.Christos : *ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems*, Proc. of 40th International Symposium on Computer Architecture (ISCA), pp.475-486, 2013.
[16] C. Bienia : *Benchmarking Modern Multiprocessors*, Ph.D. Thesis. Princeton University, 2011.