Evaluating the Performance of Dynamic Branch Prediction Schemes with BPSim

Norman Lam, Si-En Chang, and Mark L. Manwaring
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99163
(Email: {nlam, chang, manwarin}@eecs.wsu.edu)

Extended Abstract

1. Introduction

BPSim (**B**ranch **P**rediction **S**imulator) is a trace-driven simulator for predicting branch outcomes in high performance processors. It incorporates various dynamic prediction schemes widely reported in the literature and practiced in existing processors such as the MIPS R10000, Pentium Pro and Alpha 21164 [1, , 10, 13]. Users can benefit from the completeness and flexibility of this simulator in gaining an understanding of the trade-off between prediction accuracy and hardware cost especially in an education environment.

The problem of branch prediction has become one of the key issues in computer architecture; therefore, most recent computer architecture textbooks discuss this issue extensively [2, 3, 4, 5]. Branch instructions will cause pipeline stalls during execution and result in substantially poor performance. In single-issue pipelined processors, the pipeline flow is disrupted by branch instructions if the branch outcome and target address cannot be resolved prior to the next instruction fetch. This performance degradation is even more significant in wide-issue superscalar processors as more instructions can be fetched, issued, and executed in one cycle. To remedy this problem, researchers have proposed various dynamic schemes to predict the branch outcome and its target address at program run time.

Dynamic prediction schemes can achieve very high accuracy but in the expense of excessive amount of hardware for implementation. The hardware costs include target address buffers, pattern history tables, and other related data bits. Since poor prediction accuracy can result in significant penalty in flushing the pipeline and reverse the execution of the wrongly fetched instructions, the trade-off between prediction accuracy and hardware cost must be carefully studied to determine the most cost-effective scheme and hardware configuration. A simulator is, therefore, a valuable tool for such a purpose.

Some computer architecture textbooks provide instruction set simulators and cache simulators such as SPIM, DLX and Dinero [4]. However, none of these textbooks provides any branch prediction simulator. Unfortunately, this kind of simulator is unavailable in the public domain. Our objective is to develop a branch prediction simulator BPSim with the following goals:

- Trace-driven simulator: Since the traces are generated from the actual execution of real programs, accurate simulation results can be obtained from trace-driven simulation.
- Including all the popular dynamic branch prediction schemes: branch-target-buffer (BTB) with variable number of counter bits, all variations of 2-level adaptive branch prediction scheme with variable counter bits, and combining branch predictors such as gshare and gselect.
- Fast simulation speed: Since a long simulation time is usually required for the benchmarking programs, a fast simulation is extremely important for long traces such as SPEC92 and SPEC95 program. Steps that would slow down the simulation time are avoided or simplified. For example, since not all basic blocks contain a branch instruction, those blocks are first removed from the trace buffer before feeding into the simulator.
- extensive command line options: Users can specify the hardware configurations in each predictor scheme to identify the most cost-effective approach for their predictor schemes.

At Washington State University, we have developed the BPSim simulator to aid the students in understanding various dynamic branch prediction schemes. The simulator will be used in a one-semester graduate level computer architecture course (EE 524/CS 561) in the school of Electrical Engineering and Computer Science. The current platform is targeted toward the MIPS instruction set which is implemented in the DEC 3000 and DEC 5000 workstations. There are 15 workstations available, and they are easily accessible to the students in this school. We expect the students to experiment on this simulator using the SPEC92 and SPEC95 benchmarking programs.

2. Simulation Method and Features

A trace-driven simulation approach is used in BPSim as illustrated in Fig. 1. An annotated code of the target benchmarking program is first generated by *pixie*, which is available on the DEC workstations[11]. Because of the enormous amount of traces, it is necessary to generate the traces on-the-fly rather than saving them in a disk file and retrieve from it later. When running, the simulator creates a child process to execute the annotated target program to generate the required input traces. Each trace includes only the starting address and the number of instructions of each basic block. BPSim then extracts the branch instructions in each block from the original object code file, and uses the input trace to determine the target

address and the actual branch outcome of each branch instruction. The branch predictor scheme can then use this information to predict and determine its accuracy.

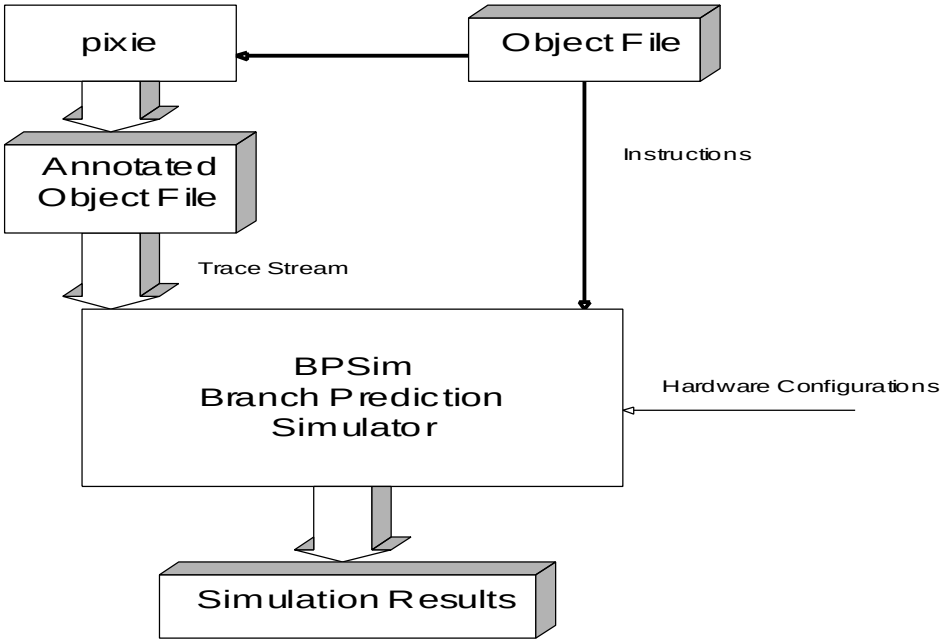

Figure 1. Block Diagram for Branch Prediction Simulation Using BPSim.

BPSim includes the following dynamic branch prediction schemes:

?? N-bit saturating up-down counters [6, 8]: User can select the number of bits in each counter, which is used to keep track of the history of each branch instruction. The prediction direction is determined by the counter value.

?? Two-level adaptive branch predictors with N-bit counters [15, 16]: This class of predictors achieves high accuracy by exploiting the fact that branch outcomes are correlated. There is a total of nine variations from this scheme based on the number of history registers and pattern history tables: GAg, GAs, GAp, PAg, PAs, PAp, SAg, SAs, and SAp.

?? gselect [9]: In this scheme, the branch address is concatenated with the global history before indexing the branch history table. The purpose is to identify the current branch more efficiently and, therefore, provides a more accurate prediction.

?? gshare [7]: Unlike gselect, the branch address is exclusive-OR with the global history before indexing the branch history table. The purpose is to isolate the history of one branch to prevent it from being corrupted by the history of another branch.

3. Sample Simulation Results

Four integer programs from the SPEC92 benchmark suite are utilized. Our intention here is to correlate the simulation results of BPSim with those reported in the literature. These four programs were first annotated by *pixie* with the trace generation codes and then simulated by BPSim. Table 1 lists the prediction accuracy for various predictors with the best performing hardware configurations. The size of the branch target buffer (or branch history table in 2-level adaptive schemes) is set to 4096 and is a direct-mapped cache. Figures 2 - 4 illustrate the effects of various physical parameters on the prediction accuracy.

	Prediction Accuracy (%)				
Predictor	**compress**	**espresso**	**eqntott**	**xlisp**	**average**
2-bit BTB	89.94	95.01	82.99	83.90	87.96
GAs (11, 32)	91.44	97.07	94.74	88.62	92.97
GAs (7, 32)	90.84	96.79	92.25	86.60	91.62
PAg (12, 1)	90.79	96.32	95.37	89.19	92.92
PAs (6, 16)	91.01	96.15	94.93	88.08	92.54
PAs (8, 256)	91.41	96.36	95.91	89.34	93.26
SAs (6, 4x16)	90.09	96.15	92.06	84.64	90.74
SAs (9, 4x32)	91.40	96.70	93.90	88.08	92.52
gselect	90.84	96.95	92.34	88.59	92.18
gshare	91.61	96.83	96.15	89.20	93.45

Table 1. Detailed Prediction Accuracy of Various Dynamic Branch Prediction Schemes. Note: SAs (6, 4x16) indicates 4 sets of history registers, each with a length of 6 bits; 16 sets of pattern history tables to store the branch history information.

4. Conclusions

Branch prediction is an important issue in increasing the instruction fetch rate and boosting the performance of high performance processors. A good design and implementation of the prediction algorithm must be preceded by a careful investigation of the trade-off between hardware cost and prediction accuracy. We have introduced a versatile and complete simulator for evaluating the performance of dynamic branch prediction schemes. We also have run an extensive set of experiments to demonstrate the accuracy of the simulator. The results correlate closely with the data reported in the literature. BPSim will

be beneficial as an educational tool in computer architecture as well as an performance evaluation tool for practicing engineers.

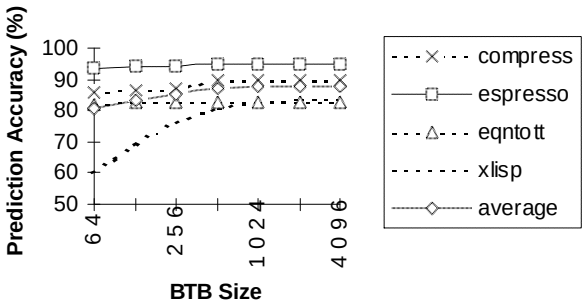

Figure 2. The Effect of Branch Target Buffer Size on Prediction Accuracy.

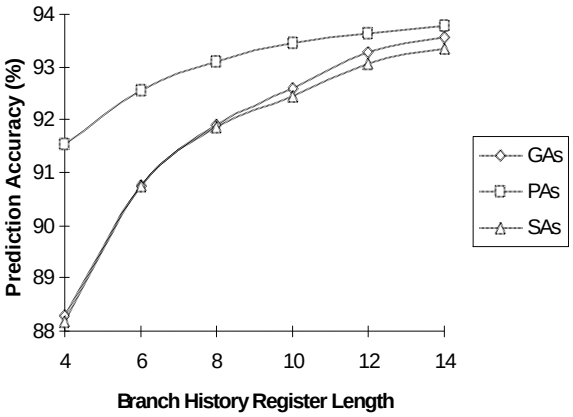

Figure 3. The Effect of Branch History Length on Prediction Accuracy in 2-Level Adaptive Schemes.

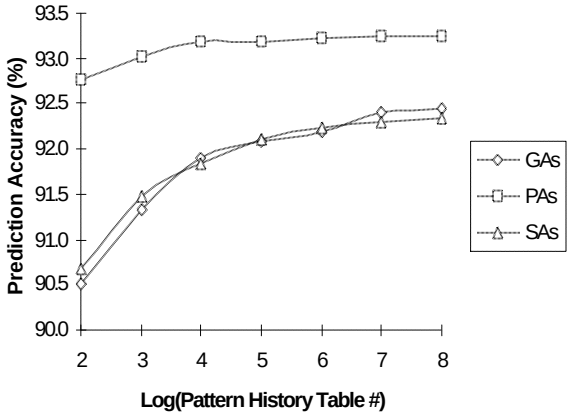

Figure 4. The Effect of the Number of Pattern History Tables on Prediction Accuracy in 2-Level Adaptive Schemes.

References

[1] J. H. Edmondson, P. Rubinfeld, R. Preston, and V. Rajagopalan, "Superscalar Instruction Execution in the 21164 Alpha Microprocessor," IEEE Micro, Vol. 15, No. 2, April 1995, pp. 33-43.

[2] M. Flynn, *Computer Architecture: Pipelined and Parallel Processor Design*, Jones and Bartlett, Boston, 1995.

[3] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, New York, 1993.

[4] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd Ed., Morgan Kaufmann, San Francisco, 1996.

[5] M. Johnson, *Superscalar Microprocessor Design*, Prentice Hall, Englewood Cliffs, 1991.

[6] J. Lee and A. Smith, "Branch Prediction Strategies and Branch Target Buffer Design," Computer, 17(1), Jan. 1984, pp. 6-22.

[7] S. McFarling, "Combining Branch Predictors," WRL Technical Note TN-36, Digital Equipment Corp., June 1993.

[8] R. Nair, "Optimal 2-Bit Branch Predictors," IEEE Transactions on Computers, Vol. 44, No. 5, May 1995, pp. 698-702.

[9] S. Pan, K. So, and J. Rahmeh, "Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation," Proc. 5th Annual Intl. Conf. On Architectural Support for Prog. Lang. And O. S., Oct. 1992, pp. 76-84.

[10] D. B. Papworth, "Tuning the Pentium Pro Microarchitecture," IEEE Micro, Vol. 16, No. 2, April 1996, pp. 8-15.

[11] M. Smith, *Tracing with Pixie*, Technical Report CSL-TR-91-497, Stanford University, CA 94305-4055, Nov. 1991.

[12] Perleberg, C. and Smith, A.J., "Branch Target Buffer Design and Optimization." IEEE Trans. On Computers, Vol. 42, No. 4, April 1993, pp. 396-412.

[13] J. Smith, "A Study of Branch Prediction Strategies," Proc. 8th Annual Intl. Symp. On Computer Architecture, June 1981, pp. 135-148.

[14] K C. Yeager, "The MIPS R10000 Superscalar Microprocessor," IEEE Micro, Vol. 16, No. 2, April 1996, pp. 28-40.

[15] T. Yeh and Y. Patt, "Two-Level Adaptive Training Branch Prediction," Proc. 24th Annual ACM/IEEE Symp. and Workshop on Microarchitecture, Nov 1991, pp. 51-61.

[16] T. Yeh and Y. Patt, "A Comparison of Dynamic Branch Predictors That Use Two Levels of Branch History," Proc. 20th Annual Intl. Symp. On Computer Architecture, May 1993, pp. 257-266.