1 Introduction

The article [1] introduces a novel dynamic branch predictor based on one-layer perceptrons. In this paper, the writers introduce how it works and give corresponding explanations. Besides, they provide proper evaluation of the prediction and compare to other state-of-the-art predictors. They also give the details of implementation. The main contributions include

- The article introduces the perceptrons predictor, first neural-networks based predictor. The article reaches SOTA performances on two benchmark. Especially, for a 4K byte hardware budget, the predictor improves misprediction rates on the SPEC 2000 integer benchmarks by 10.1%.
- The article explore the design space and illustrates the circumstances where the perception predictor works outstanding (Linearly separable branches and longer history length).
- The article gives out details of implementation.

2 Analysis

2.1 Strength and weakness

The article have such strengths.

- The idea of using perception is innovative. It perform well when meeting very long history length and context switching.
- When history length grows, percptron predictor only need linearly growing resources.
- The implementation and application are feasible.

However, the article also have such weakness.

- The predictor may work worse than traditional predictors if hardware budget is small, the history length is short and the branches are linearly inseparable.
- In modern computer, the circumstances where percetron predictor performs well are less frequent. It have longer delay comparing to traditional predictors.

2.2 Combination with other ideas

The technique claims longer delay, which means several instructions between fetching instructions and predict will be wasted. Thus, this method is not consistent with multi-way processor.
Nevertheless, the memory-access frequent equipment and multi-thread processor are consistent with it. Because, if memory-access miss happens, processor may need to wait. During this latency, it gives enough time for predictor to work. Besides, multi-thread processor makes use of idle clock cycle and increase the history length, where can this technique be applied.
To make full use of the technique, we need enough SRAM and large hardware budget. Also, linearly separable

branches and longer history length (frequent loop nesting). It is rare to meet these situation in daily PC usage but in machine learning and neural network computing. Long history length is frequent and most branches are linearly separable. This predictor is fit for GPU situation.
However, the implementation of precetron predictor need large hardware resources and is complex. Too much chip area used on controllers is not GPU design friendly.

3 Further work

The idea is very innovative. In [2], Two primary issues in predictor is systematically illustrated (1) hard-to-predict (H2P) branches and (2) rare branches with low dynamic execution counts over tens of millions of instructions. Using more training data, better online algorithm and machine learning models are three directions. In [3], many ML based predictor are proposed and they are feasible solution to both previous issues. The idea of using local and global history is introduced into them as well. [4] provides tutorial developing convolutional neural network (CNN) helper to reduce mispredictions by an average of 36.6% on 47% of H2Ps in SPECint 2017. Runtime data are trained offline. A two-layer CNN global history model after training can give out predicted branches, reaching 100% accuracy for H2P-1.
The idea of combining traditional predictors eg.TAGE-SC-L with ML based helpers is promising. We can further develop ML based helpers to solve online problems and H2P branches. Reinforcement learning is a vital solution. We can view the prediction problem as a Markov process.We have a state space E. Each instruction is a state and we will predict next state (next branch). In further direction, LSTM may be introduced as a helpers, which can obtain longer memory (long history length) and also short memory (very close branch), having advantages of both types of predictors. Besides, RNN is less time-consuming and data demanding than CNN. We can design the space of numbers of recurrent neurons, since predictions problems do not have to refer to deep neural networks. We can transfer the idea in [1] to LSTM based helpers since they are both training and ML based techniques.

References

[1] D.A. Jimenez and Calvin Lin. Dynamic branch prediction with perceptrons. 2001.

[2] Chit-Kwan Lin and Stephen J. Tarsa. Branch Prediction Is Not a Solved Problem: Measurements, Opportunities, and Future Directions. *International Symposium on Workload Characterization*, pages 228–238, June 2019.

[3] Drew D. Penney and Lizhong Chen. A Survey of Machine Learning Applied to Computer Architecture Design. *arXiv e-prints*, page arXiv:1909.12373, September 2019.

[4] Stephen J Tarsa, Chit-Kwan Lin, Gokce Keskin, Gautham Chinya, and Hong Wang. Improving Branch Prediction By Modeling Global History with Convolutional Neural Networks. *arXiv e-prints*, page arXiv:1906.09889, June 2019.