Imperial College London

Discovering Predictive Patterns: Contextual Factors for Next Generation Branch Predictors

Author: Andrei-Florin Sburlan

Supervisor: Prof. Paul Kelly

Abstract

This research is centred around exploring new ways of improving branch prediction, in particular by augmenting previous methods with extra information about the program execution. We begin by analysing the effectiveness of a number of context types derived from the *Return Address Stack*, though we also dive into the field of Machine Learning by introducing the novel perspective of branch prediction as a Machine Translation problem. In light of this, we train a Transformer Neural Network predictor, show its effectiveness in a comparative study with other approaches and finish by illustrating how analysing the internal weights of the model can give insight into subtle branch correlations.

Key Findings

- Integrating the Return Address Stack into branch prediction has the potential to reduce the misprediction rate by up to 12% of its initial value
- For object oriented languages, the Return Address Stack at the time of object allocation is predictive for branch outcomes in virtual methods
- Transformer Neural Networks can be successfully trained to predict branch outcomes. Moreover, their internal weights can be examined to gain insight into subtle branch correlations

Background & Motivation

- State-of-the art branch predictors (e.g. TAGE, Perceptrons) use the Taken/Not Taken branch history as context for their predictors
- There is strong evidence that such approaches are reaching a **fundamental performance ceiling** [1]
- Better branch prediction could **enable wider pipelines** with up to **2x more performance** [1]
- Future predictors need to use more context when making decisions

Simulation Environment

- Using Gem5 to simulate an Out-of-Order core
- Using the **TAGE-SC-L 64KB** Branch Predictor as a **baseline**
- Capturing the micro-architectural state before each branch prediction for correlation experiments
- Evaluating correlations on the SPEC CPU 2006 benchmark suite
- Using **SimPoints** [2] to speed up simulation:

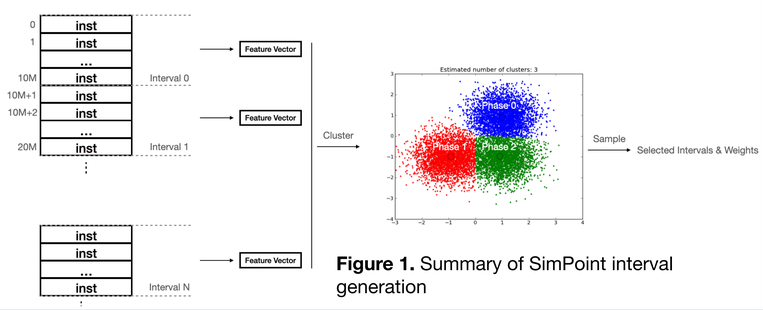

Figure 1. Summary of SimPoint interval generation

Return Address Stack Correlations

- Using the top entry of the Return Address Stack as context for a simple correlating branch predictor
- Experiment with using an *oracle* to pick *optimally* between TAGE-SC-L and the RAS-based predictor to obtain an **upper bound** for the performance of this approach
- On average, from 6.1% misprediction rate to 5.4% misprediction rate

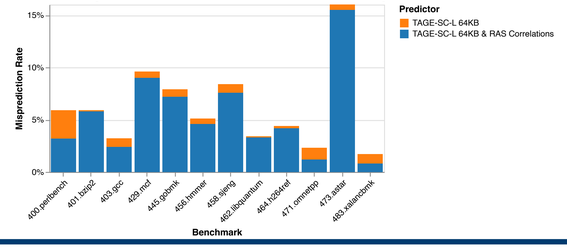

Figure 2. Upper bound for the optimisation opportunity when using RAS correlations and TAGE-SC-L 64KB

Object Allocation Context Correlations

- For object oriented languages
- Using the top entry of the Return Address Stack at the time of object allocation as context for a simple correlating branch predictor
- Experiment with using an *oracle* to pick *optimally* between TAGE-SC-L and the object allocation context-based predictor to obtain an **upper bound** for the performance of this approach
- On average, from 6.6% misprediction rate to 6.1% misprediction rate

Table 1. Upper bound for the optimisation opportunity when using object allocation context correlations and TAGE-SC-L 64KB

Benchmark	TAGE-SC-L 64KB Misprediction Rate	Object Allocation Context Correlation Opportunity (Upper Bound)
471.omnetpp	2.3%	0.9%
473.astar	16.0%	0.0%
483.xalancbmk	1.7%	0.7%
Average	6.6%	0.5%

Training Transformers for Branch Predictions

- Leveraging ML to assess the potential for enhanced predictions using the Path History
- Branch Prediction as a translation problem: Path History → Taken/Not Taken History
- Use a Transformer model [3] to predict branch outcomes from Path, Taken Histories
- **Train on SimPoint interval traces, weight training loss with interval weights**
- Train on SPEC 2006 CPU *Train* input set, evaluate on *Test* input set
- Evidence that Transformers can discover correlations that appear at **varying offsets** in the history
- Hardware implementation cost too high, but could be used in a compiler PGO setting

	TAGE-SC-L 64KB Accuracy	Transformer Model Accuracy		Delta Accuracy	
		Known Branches	All Branches	Known Branches	All Branches
400.perlbench	94.1%	85.6%	83.0%	-8.5%	-11.1%
401.bzip2	94.1%	96.8%	94.9%	**+2.7%**	**+0.8%**
403.gcc	96.8%	94.7%	91.3%	-2.1%	-5.5%
429.mcf	90.4%	94.7%	94.5%	**+4.7%**	**+4.1%**
445.gobmk	92.1%	94.5%	94.1%	**+2.4%**	**+2.0%**
456.hmmer	94.9%	90.3%	90.4%	-4.6%	-4.5%
458.sjeng	91.6%	91.1%	91.0%	-0.5%	-0.6%
462.libquantum	96.6%	97.2%	94.6%	**+0.6%**	-2.0%
464.h264ref	95.6%	97.8%	97.7%	**+2.2%**	**+2.1%**
471.omnetpp	97.7%	95.6%	82.8%	-2.1%	-14.9%
473.astar	84.0%	91.1%	90.9%	**+7.1%**	**+6.9%**
483.xalancbmk	98.3%	90.0%	87.8%	-8.3%	-10.5%
Average	93.8%	93.3%	91.1%	-0.5%	-2.7%

Table 2. Comparison between TAGE-SC-L 64KB and the Transformer model on the *Test* set. The Transformer model has been trained offline on the *Train* input set.

```
if (condition A) { ... }                    // B1

int N = random_int();
for (int i = 0; i < N; i++) {
   if (random_boolean()) { ... }            // B2
}

if (condition A && condition B) { ... } // B3
```

Listing 1. Hard-to-predict branch (B3) for Taken/Not Taken History-based predictors

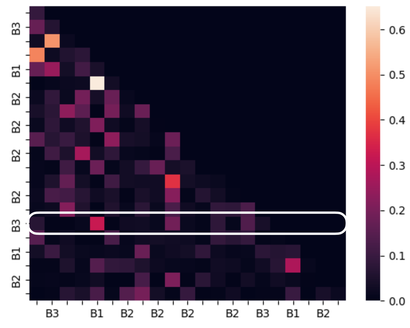

Figure 3. Attention weights generated by the model on an example trace of the program in Listing 1. Note that branch B3 pays high attention to the position in the history corresponding to branch B1.

Future Work

- Explore hardware implementation of Transformers/Attention e.g. using binarised Neural Networks [4]
- Profile Guided Optimisations based on Transformer-learned branch correlations
- Oracle approximations for the RAS/Object Allocation Context predictors e.g. using them in a tournament predictor

References

[1] Lin CK, Tarsa SJ. Branch Prediction Is Not A Solved Problem: Measurements, Opportunities, and Future Directions. In: 2019 IEEE International Symposium on Workload Characterization (IISWC); 2019. p. 228-38.
[2] Hamerly G, Perelman E, Lau J, Calder B. SimPoint 3.0: Faster and More Flexible Program Phase Analysis. J Instr Level Parallelism. 2005;7.
[3] [14] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, et al., editors. Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc.; 2017. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
[4] Courbariaux M, Hubara I, Soudry D, El-Yaniv R, Bengio Y. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1; 2016.