Temporal Stream Branch Predictor

Yongming Shen Michael Ferdman
Stony Brook University

Abstract

Branch predictors play an important role in high performance processors. Designing a branch predictor that balances simplicity, resource usage, and accuracy remains a challenge today.

In this paper, we present the *temporal stream (TS) branch predictor*, a branch predictor that adds temporal streaming on top of a base predictor to deliver improved accuracy. Experiments show that when using gshare as the base predictor, the TS predictor can achieve far better accuracy than gshare alone by correcting gshare's mistakes. While a 512KB gshare predictor achieves 4.675 MPKI on the championship traces, TS with 32KB gshare as base predictor can achieve 3.695 MPKI.

Introduction

Today's deeply pipelined super-scalar out-of-order processors require high accuracy branch predictors to achieve peak performance. Designing a branch predictor requires striking a balance between simplicity, resource usage, and accuracy.

In this paper, we present the temporal stream (TS) branch predictor, a branch predictor that adds temporal streaming on top of a base predictor to deliver improved accuracy. The TS predictor is based on the observation that a simple predictor often repeats its mistakes. To take advantage of this fact, the TS predictor first records the base predictor's mistakes and then uses the recording to prevent the same mistakes from happening again, thus achieving higher overall branch-prediction accuracy.

In the implementation of the TS predictor, we record the correctness of the base predictor in a circular buffer of bits ("1" for correct, "0" for incorrect). Whenever the base predictor makes a mistake, TS looks up a table to check if there is an appropriate point to start replay. TS also records the current buffer tail pointer in the table for future use. When replaying, a "1" in the buffer means to pass on the base prediction, and a "0" in the buffer means to flip the base prediction. If the TS predictor makes a mistake, replay is stopped until the base predictor makes a mistake again.

For this unlimited size submission, the TS predictor uses a 512KB gshare predictor as the base predictor. While gshare alone achieves 4.675 MPKI on the championship traces, the TS predictor improves accuracy to 3.487 MPKI. This demonstrates that temporal streaming is effective in achieving better branch prediction accuracy.

Motivation and Solution

To better explain the TS branch predictor, we will first discuss a TS predictor without a base predictor (one can also think of this as a TS predictor with an always-taken base predictor).

The fundamental idea behind temporal stream branch prediction is to record a sequence of conditional branch outcomes and then to replay the outcomes as predictions at an appropriate time. Because a sequence of branch outcomes is highly repetitive, the recording of past outcomes serves as an accurate predictor of future outcomes. The main challenge for the TS predictor is to figure out a past location within the recorded sequence from which to begin replaying the branch outcomes.

To visualize how the predictor works, imagine an application whose body is made up of a large loop containing complex program logic. At the beginning of the first loop iteration, the TS predictor begins sequentially recording the outcomes of the conditional branches, writing a "1" to indicate that a conditional branch was taken and a "0" to indicate that a conditional branch was not taken. When the first loop iteration completes, there is an opportunity to replay previously observed outcomes and make correct predictions, given the assumption that the branch outcomes of the second loop iteration are the same as the outcomes of the first. The TS pre-

dictor can take advantage of this opportunity if it starts replay at the appropriate time and place.

In practice, the power of the TS predictor lies in leveraging extensive historical context to determine the outcome of hard-to-predict branches. Once replay of a temporal stream starts, the further the stream gets without making a mistake, the more likely the stream is to correctly predict future outcomes. However, for branches that are easy to predict, long history is not needed or is harmful because of increased training time.

To mitigate the long training time problem, we couple the TS predictor with a simple base predictor, allowing the base predictor to handle most common cases but leverage the temporal stream for cases where the base predictor makes mistakes. When coupled with a base predictor, rather than recording the outcomes of each conditional branch, the TS predictor records whether or not the base predictor made a mistake. Recording a "1" in the sequence indicates that the base predictor was correct; recording a "0" indicates that the base predictor made a mistake. When the TS predictor is not replaying a temporal stream, the base predictor's output is used directly. When the TS predictor is replaying the temporal stream, a "1" indicates that the base prediction should be used, while a "0" indicates that the base predictor's prediction is likely incorrect and should be reversed.

Design and Implementation

Our implementation of the TS predictor comprises three structures: (1) a base branch predictor, (2) a circular buffer for recording base predictor mistakes, and (3) a head table containing pointers to locations in the circular buffer.

The TS predictor maintains a tail pointer, pointing to the location where the next base correctness bit is to be recorded; a head pointer, pointing to the location from where the next base correctness bit is to be read; and a replaying flag to indicate whether or not the predictor is in replay mode. The overall predictor's operation is described in the following algorithms:

```
predict():
  base_outcome = base_predict();
  if (replaying && buffer[head++]==0)
    return !base_outcome;
  else
    return base_outcome;


update():
  update_base_predictor();
  update_features();//e.g., global history
  buffer.append(
    base_outcome == actual_outcome);
  if (TS_outcome != actual_outcome)
    replaying = false;
  if (base_outcome != actual_outcome) {
    key = key_from_features();
    if (!replaying &&
          !empty(head_table[key]) {
      head = head_table[key];
      replaying = true;
    }
    head_table[key] = tail;
  }
```

For the purpose of the competition submission, we use a gshare predictor as the base predictor, an infinitely large buffer for recording base predictor correctness, and an infinitely large head table for mapping keys to replay starting points. The *key_from_features* function simply concatenates the current PC and global history bits.

Results

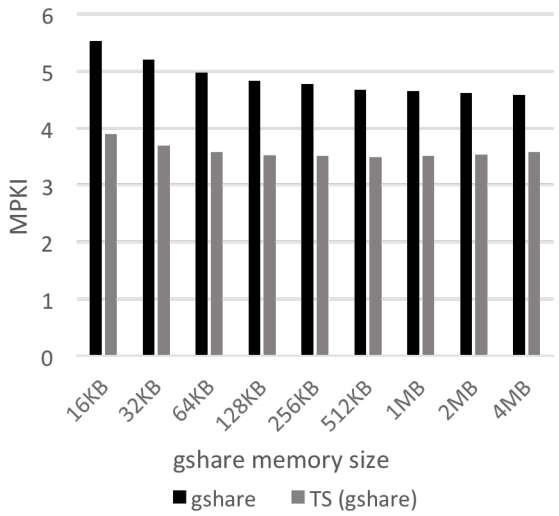
In our experiments, gshare predictors with memory sizes from 16KB to 4MB are used. For the head table, concatenations of 140-bit global histories and PC values are used as keys. Table 1 and Figure 1 compare the MPKI scores of gshare predictors with gshare-based TS predictors. Our results indicate that

Table 1: Predictor Accuracies (MPKI)

gshare memory size	16KB	32KB	64KB	128KB	256KB	512KB	1MB	2MB	4MB
gshare	5.530	5.196	4.971	4.823	4.768	4.675	4.643	4.612	4.586
TS (gshare)	3.897	3.695	3.575	3.517	3.505	3.487	3.507	3.537	3.577

TS can consistently provide better accuracy than gshare alone, with the best accuracy being at 3.487 MPKI, when the base gshare predictor uses 512KB of memory.

Figure 1: Predictor Accuracies (MPKI)



Related Work

Prior work on temporal streaming was done in the context of data prefetching [2][3] and instruction prefetching [1]. In this work, we find that temporal streaming is also a promising technique for branch prediction. In many ways, our work is also similar to stream compression using branch prediction [4]. Overriding branch prediction [5] also uses the idea of correcting a simple predictor's mistakes, but correction is made after the simple predictor's output is used, whereas the TS predictor corrects mistakes immediately.

Conclusions

We find that using temporal streaming for branch prediction is promising. Experiments show that gshare-based TS predictors consistently provide better accuracy than gshare predictors. In this submission, we do not place restrictions on the size of the head table and the circular buffer. However, there are numerous techniques that can be used to convert the current implementation to one that uses a reasonable amount of memory. Additionally, our submission simply uses global history and PC bits to compute keys for the head table, whereas further accuracy improvements should be achievable with more complex keys.

References

[1] Ferdman, Michael, et al. "Temporal instruction fetch streaming." Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on. IEEE, 2008.

[2] Wenisch, Thomas F., et al. "Practical off-chip meta-data for temporal memory streaming." High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on. IEEE, 2009.

[3] Wenisch, Thomas F., et al. "Temporal streaming of shared memory." Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on. IEEE, 2005.

[4] Burtscher, Martin, et al. "The VPC trace-compression algorithms." Computers, IEEE Transactions on 54.11 (2005): 1329-1344.

[5] Seznec, André, et al. "Design tradeoffs for the Alpha EV8 conditional branch predictor." Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on. IEEE, 2002.