Predictor Analysis

The Key Components of a Successful Predictor.

Robert Herriot

Tristan Islam

Daniel Okazaki

Stephen Pacwa

Table of Contents

1 List of Figures

2 List of Tables

3 Report Information

Date	Version Number	Author	Comment
11/24/2019	0.0	Robert, Tristan, Daniel, Stephen	Initial report version.
11/30/2019	0.1	Robert, Tristan, Daniel, Stephen	Minor changes to graphs and experiments.
12/7/2019	1.0	Robert, Tristan, Daniel, Stephen	Final report version.
12/10/2019	1.1	Robert, Tristan, Daniel, Stephen	Minor changes to sentences.

Table 1: Report Version Information

4 Introduction

The purpose of this report is to run simulations of branch predictors with different parameters, present the data that was collected, and analyze the results that were attained. For the purpose of analysis, the simulations can be grouped into several overarching categories, each dependent on a different critical aspect of the predictor. The first group is related to the number of bits that are stored in each entry. The second group is related to the number of entries that each predictor contains. The third group presents the advantages of correlating predictors. The fourth group presents the advantages of global predictors. A final group presents the usefulness of a simple tournament predictor.

The report will be organized in the following manner. Section 5 will go into the background of branch prediction as well as the types of predictors that will be tested. Section 6 will describe how the experiments were completed. Section 7 will describe the results of the experiments. Section 8 will analyze the results. Finally, Section 9 provides a conclusion to the paper and restates the overall trends in predictor design.

5 Background

Branch predictors have become an important aspect of computer performance with the introduction of pipelined CPUs. The processor attempts to predict the outcome of a branch based on a simple heuristic, allowing it to prefetch instructions that would proceed after the branch statement. This optimization leads to an overall decrease in the latency of the pipeline. However, if the prediction is later discovered to have been incorrect, the pipeline must be flushed as the instructions that were prefetched are no longer going to be completed. This flushing of the pipeline and fetch of the new instructions introduces a much greater latency penalty than if the CPU had simply stalled at the branch. To amortize the cost of this penalty, the branch predictor must almost never miss. This paper analyzes the different branch predictors in order to determine what aspects of the predictor will result in ones with the fewest misses and therefore decrease the overall latency of the pipeline the most.

As presented in the introduction, the tested predictors will be split into five groups. These groups will consist of several different implementations. The first group will consist of six predictors with 32 entries. Each one will have either one, two, three, four, five, or six bits of history stored in each entry. The second group will consist of four predictors with two bits of history stored in each entry. They will range in size, consisting of 32, 64, 128, and 256 entries. The third group will consist of predictors with two bits of history stored in each entry and thirty-two entries per group. They have either zero, one, two, or four bits of shift history. The next group will contain an assortment of global predictors with two bits per entry and either two, four, or eight bits of global history. Finally, a tournament predictor will constructed out of a large correlating predictor and a large global predictor. The total size of the tournament predictor was chosen to be the same as the size of the gshare predictor originally bundled with the software, distributed equally between the two predictors. This group will also contain the same correlating and global predictor on their own to facilitate analysis.

6 Setup

The original testing environment was changed to collect a number of important statistics including the number of hits, the number of misses, and the number of branches total, in addition to the original misses per thousand instructions (MPKI). For the tournament predictor tests, the choice of predictor used in each branch is also collected. From this information the percent miss, percent hit, percent predictor one is chosen, and percent predictor two is chosen can be extrapolated. To summarize our findings and present a high overview of the results of each predictor, a geometric mean was used on each of these categories. This was primarily done on the MKPI, hit and miss statistics, which we collected on a trace by trace basis. Because of the similarity between MPKI and percent miss, only the MPKI was retained as a metric used to compare predictors.

7 Results

In effect, we determined that the advantage of using tournament predictors was marginal at best. Once we narrowed down the size which we were operating in, the competition field became a narrow race between the gshare predictor and the best predictors we'd settled on via trial and error. In general, the gshare predictor was marginally but still notably more effective than the best general and global predictors of its size, and the best tournament predictors at that size (two-predictor tournament predictors with the sum of the sub-predictors' sizes equaling 32kb) lagged behind the gshare predictor by marginal values, sometimes less than a tenth of a percentage point more hits. In the best tournament predictors, the two sub-predictors both were used, with complete domination of all test cases by one predictor being eliminated. This follows intuitively from the fact that the utility of a tournament predictor is having sub-predictors tailored for specific situations, making up for each other's faults. Additionally, tournament predictors also require additional storage and logic on top of the two 16kb sub-predictors, so the fact that they can't seem to outperform the simpler gshare predictor makes their use wasteful.

Another conclusion one can easily draw from our comparison of the gshare predictor and the best of our tournament predictors is that there is a sort of “maximum effectiveness curve” determined by the size of the predictor, where a predictor of a certain size cannot exceed a certain maximum average hit fraction across a varied set of test traces simply due to the maximum amount of information it is capable of storing being insufficient to do so. The gshare predictor and the best of our tournament predictors all clustering very close to the same hit rate, with all others tested being inferior, would imply that they are very close to the maximum performance obtainable from this volume of memory.

Getting tournament predictors to perform effectively was a long process. One of the main optimizations was giving the predictors enough history that they could each behave effectively. A more important optimization however was making the counter length larger. One of the setups for tournament predictors that worked was combining general and global predictors.

8 Analysis

This section will present several plots of the data as well as analysis on why the data is acting a certain way. It is broken down into six sections based on the five groups presented above as well as a section that goes into further detail on the relationship between the types of several of the predictors.

8.1 Entry Size

The first experiment that was completed focused on the importance of the size of each entry. The goals were to explore the effectiveness of entry size and to determine if there was a point where a larger entry no longer grants an increase in performance. These predictors utilize a simple entry field that increments or decrements a certain number of bits with no overflow. If the number is above the halfway point, it is predicted as taken, otherwise it is not taken.

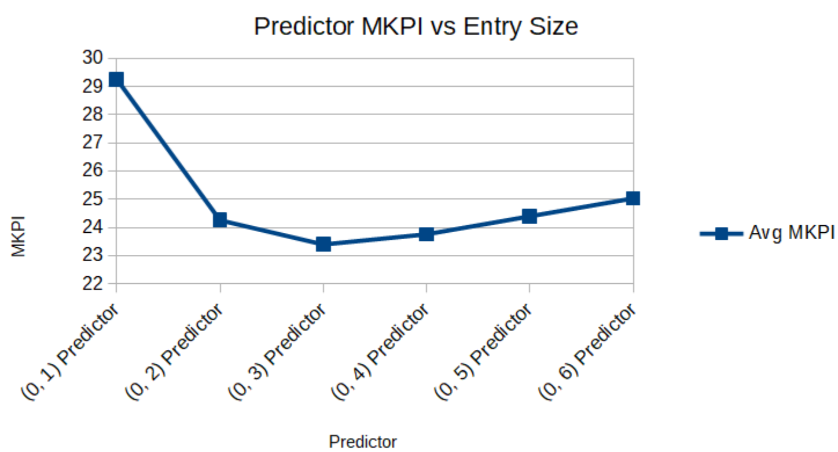

Figure 1: Average MPKI versus Number of Bits in Each Entry

This figure plots the average MPKI against the number of bits in each entry. It consists of six predictors which only differ in how many bits are stored in each entry. Each predictor contains a total of 32 entries.

Figure 1 displays the results of this experiment. It shows that after three bits of entry, adding more bits becomes counterproductive. It is believed that this is due to the "inertia" that is experienced when there is not enough of a trend to move out of the half that the numbers are currently in. It is believed that the introduction of a more complex state machine for each entry could change these results. It is also important to note that the gain achieved between two and three bits of state is marginal. Because adding a third bit makes the predictor 50% larger for very little gain, two bit predictors were considered optimal, and were used for all later tests.

8.2 Number of Entries

The next experiment aimed to explore the benefit of increasing the number of entries inside each predictor. By adding more entries, the amount of competition between two branches can be reduced and performance can increase.

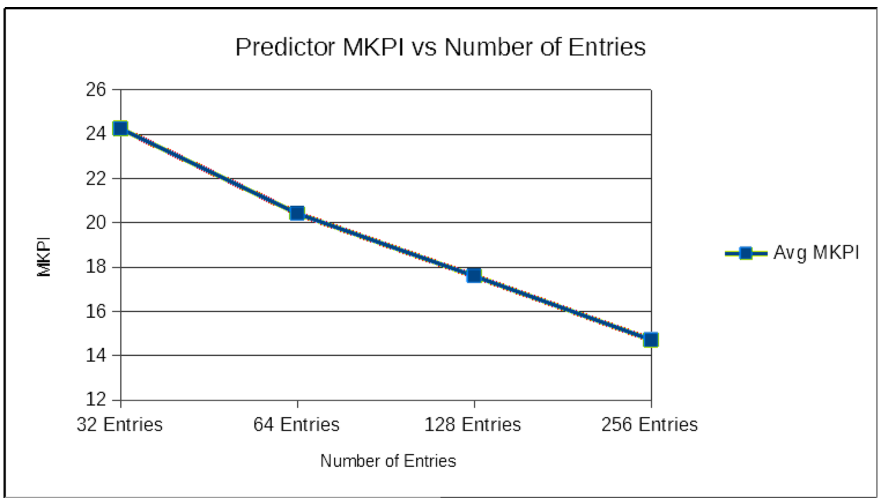

Figure 2: Average MPKI versus the Number of Total Entries

This figure plots the average MPKI against the number of entries in each predictor. It consists of four predictors which only differ in how many entries there are. Each predictor is based on a (0, 2) design otherwise.

Figure 2 shows a linear result, which suggests, that at least at sizes relative to the ones tested, that the number of entries is one of the few areas that will almost always benefit from being increased. This is primarily due to the fact that there is less conflict between any two branches that would originally be sharing the same entry and that there is more room for state to be stored.

8.3 Correlating Predictors

The next experiment tested the benefit of correlating predictors. This was done by testing four different predictors that each introduce more bits of correlation, ranging from none, as the baseline, to four bits.

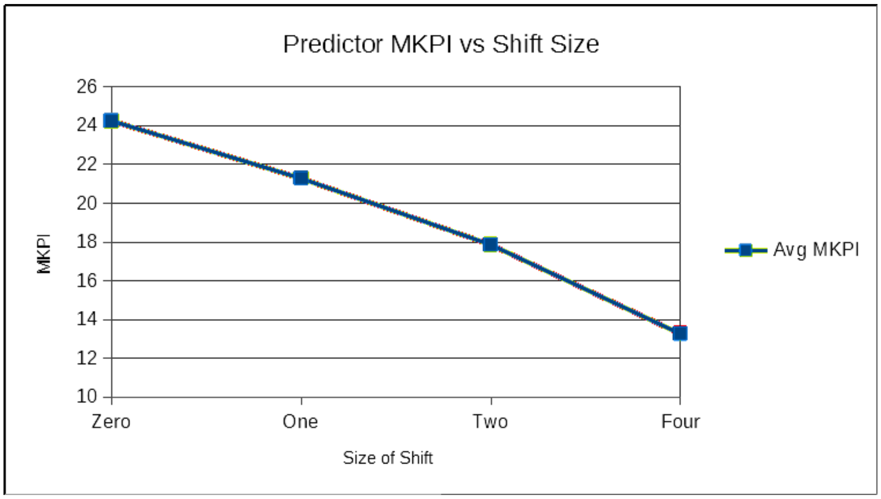

Figure 3: Average MPKI versus the Number of Bits of Shift History

This figure plots the average MPKI against the number of bits of correlation that are stored in the shift register. It consists of four predictors which only differ in the number of these bits. Each predictor contains two bits of information per entry with 32 entries per table that is referenced by the shift register.

Figure 3 displays the difference in average MKPI that is the result of the addition of and expansion of a shift register. This plot is also linear in nature. Because increasing the size of the shift register will, in effect, increase the size of the predictor, It is hard to determine if this linear result is due to the correlation factor or the size of the predictor. These results are explored later in this paper alongside the global predictor.

8.4 Global Predictors

The next experiment attempted to determine the effectiveness of a global predictor. The global predictor uses a global history of taken and not taken to index a list of entries directly. Therefore, the size of the global register is related to the overall size of the predictor.

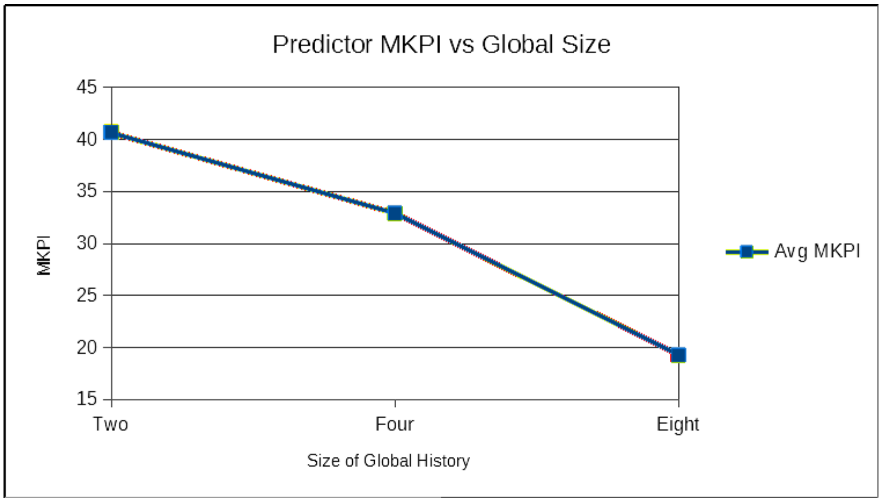

Figure 4: Average MPKI versus the Number of Bits of Global History

This figure plots the average MPKI against the number of bits of global history that is stored in a register. It consists of three predictors which only differ in the number of these bits. Each predictor contains two bits of information per entry.

Figure 4 shows a similar correlation to the correlating predictor tested above. The growth of the global history results in a strong increase in the predictor performance. This is also difficult to analyze on its own as it could still be correlated to the predictor size. Because of this, the next section will compare the results of these various predictors.

8.5 Predictor Type Comparison

This section presents the data from the simple, correlating, and global predictors placed side by side to further explore the relationship between the improvements of those predictors and their size. These experiments were done with nine predictors presented on a sliding scale. To ensure the tests are completed fairly, the total number of entries of each predictor, 256, is kept stable. Note that a (0,2) predictor is a direct-map predictor (no branch-history correlation) and an (8,2) predictor is a global predictor in this case (only branch-history correlation).

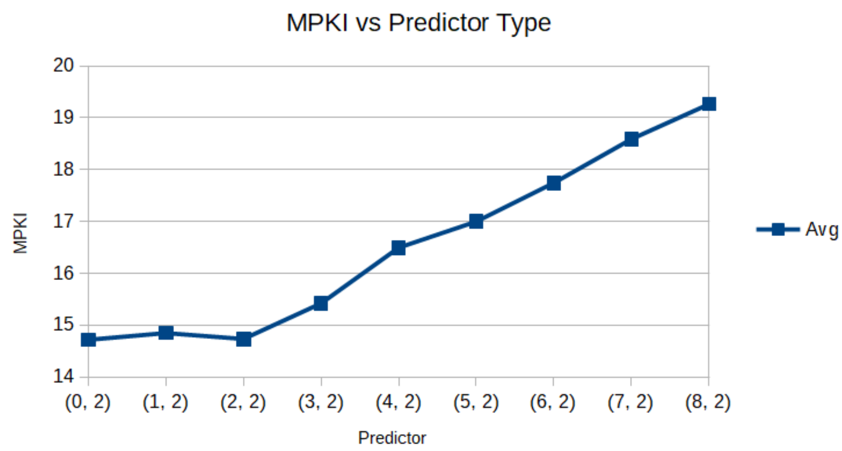

Figure 5: Average MPKI versus Type of Predictor

This figure plots the average MPKI of a range of different predictors. These predictors all have the same fixed number of entries. The (0, 2) predictor is considered identical to a simple predictor and the (8, 2) predictor is considered identical to a global predictor. All predictors in between are a range of correlating predictors.

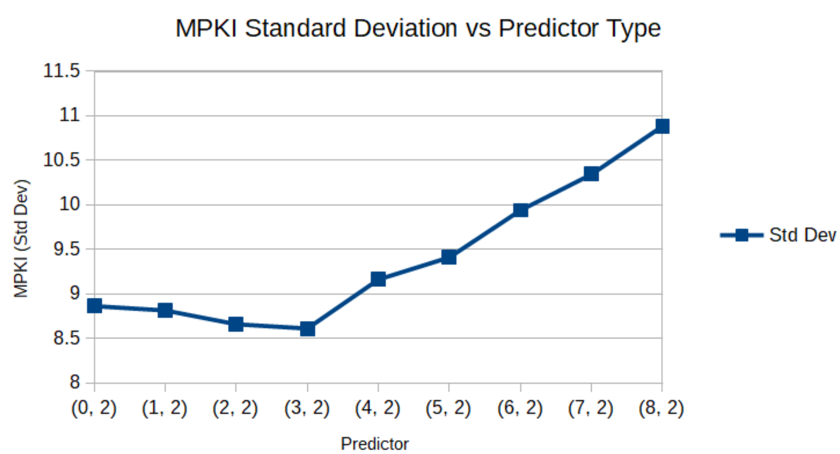

Figure 6: MPKI Standard Deviation versus Type of Predictor

This figure plots the standard deviation of the MPKI of a range of different predictors. These predictors all have the same fixed number of entries. The (0, 2) predictor is considered identical to a simple predictor and the (8, 2) predictor is considered identical to a global predictor. All predictors in between are a range of correlating predictors.

The standard deviation presented in Figure 6 increases along with MPKI as the complexity (amount of history bits) increases, indicating a greater variance in test-case results. This shows that for more complex (more history bits) predictors there were test cases for which they performed well, but there was also a lot of variance by test case. As can be seen in Figure 5, the high-correlation predictors are not very good on average. This is because correlating predictors store a large amount of state information and this will take a while to adjust if the pattern of branches in the benchmark shifts. This variance is what makes a mix of high-correlation and low-correlation predictors good for creating a tournament predictor: the low-correlation predictor is going to be more effective in most cases, but the few where a high-correlation predictor is better are improved.

8.6 Tournament Predictors

The final experiment attempts to create a tournament predictor that is able to perform better than its constituent parts. The tournament predictor uses a simple global predictor history to choose which one should be used in each prediction. When comparing to the constituent predictors, care was taken to ensure that the global and correlating predictors were twice the size they were inside the tournament predictor to ensure that this did not muddy the results.

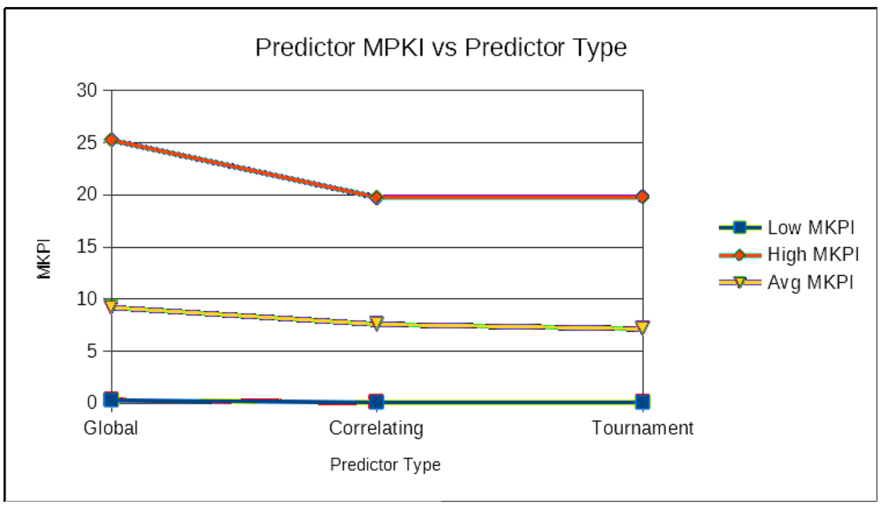

Figure 7: Predictor MPKI versus Type of Predictor with regards to Low, Average, and High

This figure plots the low, average, and high MPKI of a tournament predictor and the two predictor types that it is made of. These predictors all contain the same number of total bits in size to aid in comparison.

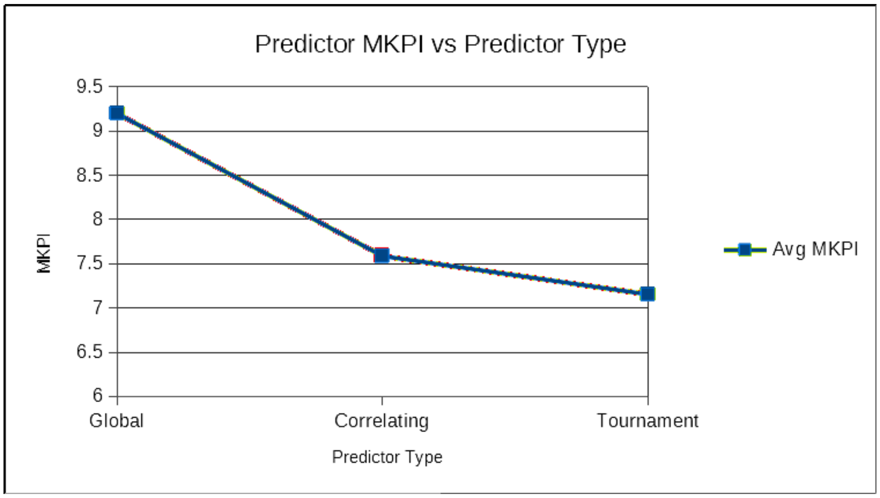

Figure 8: Predictor MPKI versus Type of Predictor

This figure plots the MPKI of a tournament predictor and the two predictor types that it is made of. These predictors all contain the same number of total bits in size to aid in comparison.

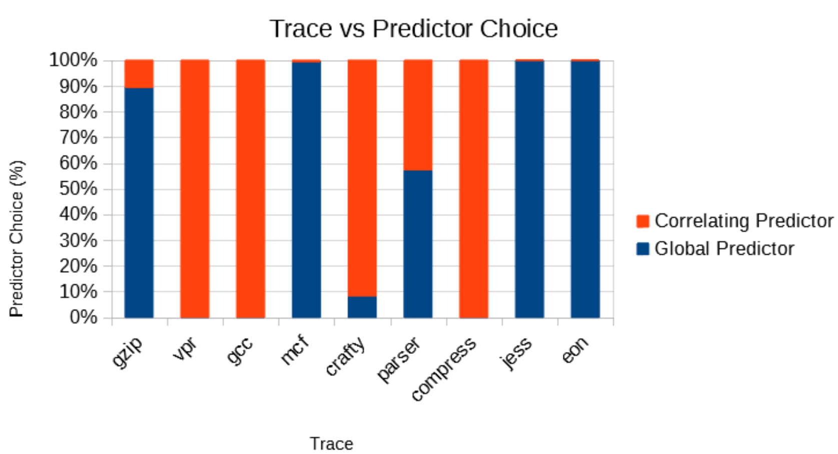

Figure 9: Trace versus Percent Choice of Predictor Type

This figure plots the predictor choice of a variety of different trances that were run in a tournament predictor. The majority of traces display dominance of one predictor type over the other.

These plots offer a strong defense of the capability of the tournament predictor. Figures 7 and 8 display the ability of the tournament predictor to improve the performance of the predictor past any single predictors capability. Although these gains are relatively small, they should be put into the context of predictors that are already at a highly optimal state. In the plots that display a high, average, and low value, we can see that the tournament predictor takes on the strong high and low of the correlating predictor, which incorporates the strengths of the global predictor in some places to reduce the overall MPKI. Figure 9 presents the breakdown of predictor choice across many of the chosen traces. The dominance of one predictor over the other shows that each type of predictor has a workflow that has a trend that allows for one predictor to perform better than the other. The traces that seems to flip between the two predictors more than the others may suggest that there is a third “type” of workflow that would benefit from a third, different predictor implementation.

9 Conclusion

In conclusion, we have analyzed the properties of various predictors to determine which properties are most important to consider and tried to understand why certain properties changed the predictors effectiveness. We also applied these principles to the design of a tournament predictor that performs well. It was important to choose two predictors that do not perform similarly to each other and still perform well enough on their own to work well when they dominate the other predictor. By taking this approach, a gshare sized tournament predictor which is able to match the performance of gshare.