*Schemes of Branch Prediction*

Nicolae-Andrei Vasile

*Faculty of Computer Science and Automatic Control*

*POLITEHNICA University of Bucharest*

Bucharest, Romania

*Abstract – In high-performance computer systems, performance losses due to conditional branch instructions can be minimized by predicting a branch outcome and fetching, decoding, and/or issuing subsequent instructions before the actual outcome is known.*

1. **INTRODUCTION**

As the design trends of modern superscalar microprocessors move toward wider issue and deeper super-pipelines, effective branch prediction becomes essential to exploring the full performance of microprocessors. A good branch prediction scheme can increase the performance of a microprocessor by eliminating the instruction fetch stalls in the pipelines [1]. As a result, numerous branch prediction schemes have been proposed and implemented on new microprocessors.

Branch instructions can break the smooth flow of instruction fetching and execution. This results in delay, because a branch that is taken changes the location of instruction fetches and because the issuing of instructions must often wait until conditional branch decisions are made.

To reduce delay, one can attempt to predict the direction that a branch instruction will take and begin fetching, decoding, or even issuing instructions before the branch decision is made. Unfortunately, a wrong prediction may lead to more delay if, for example, instructions on the correct branch path need to be fetched or partially executed instructions on the wrong path need to be purged. The disparity between the delay for a correctly predicted branch and an incorrectly predicted branch points to the need for accurate branch prediction strategies.

[2] We present a framework that categorizes branch prediction schemes by the way in which they partition dynamic branches and by the kind of predictor that they use. The framework allows us to compare branch prediction schemes, and to analyze why they work. We use the framework to show how a static correlated branch prediction scheme increases branch bias and thus improves overall branch prediction accuracy. We also use the framework to identify the fundamental differences between static and dynamic correlated branch prediction schemes. This study shows that there is room to improve the prediction accuracy of existing branch prediction schemes.

[3] Our study begins in the next section, with two branch prediction strategies that are often suggested. These strategies indicate the success that can reasonably be expected. They also introduce concepts and terminology used in this paper. Strategies are divided into two basic categories, depending on whether history was used for making a prediction or not. In subsequent sections, strategies belonging to each of the categories are discussed, and further refinements intended to reduce cost and increase accuracy are presented. Levels of confidence are attached to branch predictions to minimize delay when there are varying degrees to which branch outcomes can be anticipated (for example, prefetching instructions is one degree, pre-issuing them is another).

[1] In this paper, we apply techniques from data compression to establish a theoretical basis for branch prediction, and to illustrate alternatives for further improvement. To establish a theoretical basis, we first introduce a conceptual model to characterize each component in a branch prediction process. Then we show that current "two-level" or correlation-based predictors are, in fact, simplifications of an optimal predictor in data compression, Prediction by Partial Matching (PPM).

[4] In this paper, we explore various types of artificial neurons and propose a two-level scheme that uses perceptrons instead of two-bit counters. A key advantage of our approach is its ability to utilize long branch history lengths. In our predictor, each static branch is ideally allocated its own perceptron to predict the branch outcome, and the space required by our scheme scales linearly with the history length.

1. **BASIC DEFINITIONS**

Given a conditional branch in a program, the goal of a branch prediction scheme is to predict accurately the outcome of that conditional branch (i.e. that the branch will take or that the branch will fall through).1 The most accurate branch prediction schemes predict the next action of a branch based on some function of the past actions of that branch and possibly other branches in the program.

To understand the capabilities of these branch prediction schemes and to compare competing schemes in a meaningful manner, we must be able to identify and quantify the important properties of branch prediction schemes. To achieve this goal, this section defines a set of mathematical tools that allow us to analyze program and branch behavior in an abstract manner.

Let a branch execution be a pair consisting of an identifier and a direction variable. Intuitively, the identifier uniquely specifies a static branch in a program, and the direction variable indicates the direction that the branch went. We define an execution stream or just stream as a sequence of branch executions. Intuitively, this corresponds to a branch trace of one invocation of a program, identifying in trace order the conditional branches executed and the directions that they went. A stream can also be formed by concatenating the streams of multiple invocations of a program (possibly with different inputs). We refer to the original stream of all executions in a run of the program as the program execution stream. A substream of a stream is a subsequence of. A predictor is a simple mechanism that predicts the next direction of a stream. A predictor may consider program characteristics (e.g. the opcode of the next branch to predict) in addition to any part of the past program execution stream.2 The accuracy of a predictor is the number of correct predictions divided by the total number of predictions; accuracy measures how closely the predicted stream matches the actual stream. A prediction scheme is a comprehensive mechanism that takes a program execution stream, divides it into substreams, and directs each substream to a unique predictor. Figure 1 illustrates this concept. The objective in dividing the execution stream into substreams is that each substream should be more accurately predictable by its predictor. The accuracy of the prediction scheme is the total number of correct predictions divided by the total number of predictions.

# **REFERENCES**

|  |  |
| --- | --- |
| [1] | I.-C. K. Chen, J. T. Coffey and T. N. Mudge, "Analysis of branch prediction via data compression," *ACM SIGPLAN Notices,* pp. 128-137, 1996. |
| [2] | C. Young, N. Gloy and M. D. Smith, "A comparative analysis of schemes for correlated branch prediction," *Proceedings 22nd Annual International Symposium on Computer Architecture,* pp. 276-286, 1995. |
| [3] | J. E. Smith, "A study of branch prediction strategies," *Proceedings of the 8th annual symposium on Computer Architecture,* pp. 135-148, 1981. |
| [4] | D. A. Jimenez and C. Lin, "Neural methods for dynamic branch prediction," *ACM Transactions on Computer Systems,* pp. 369-397, 2002. |