整合靜態與動態方法之新數值預測結構

New Value Prediction Architecture By Combining Static and Dynamic Method

黃樹林

Shu-Lin Hwang

摘要

數值預測機制嘗試去消除真實資料相依，其藉由預測指令執行後的結果數值。使用這預測的結果，隨後的相依指令可在相同的週期被執行而不用被停下來等待。在[4][5]中的作者提出一種包含三種預測器(最後數值、步距及有限內容預測器)的混合式預測器可以對所有指令得到較高的預測率。但是仍有一些更複雜的資料順序無法被分類而無法對他們做任何預測。為了能實現混合式數值預測的機制，大量的數值預測表通常被使用來儲存相關預測特性的資訊。在本篇論文中我們使用程式執行紀錄技術及程式特性來探討數值預測表的使用效率。除外，我們基於程式執行紀錄的分析技術整合靜態與動態的配置方法於新的混合式的預測架構中。實驗結果顯示新提出的方法能降低硬體成本並且相較於[4][5]中提出的混合式預測架構能達到較高的預測效能。

關鍵詞：真實資料相依、數值預測、程式執行紀錄

ABSTRACT

Value prediction attempts to eliminate true data dependencies by predicting the outcome values of instructions at run-time. Using the predicted outcome, the dependent instructions can be executed in the same cycle rather than stalled. In [4][5], they propose a hybrid mechanism with three predictors (last value, stride, and finite context based predictor) to get high prediction rate for all instructions. But there are still a lot of more complex data sequences that are hard to classify, so we can not make any prediction on them. In order to carry out mechanism of the hybrid value prediction, huge value prediction table usually was used to keep information about the prediction characteristics. In this paper, we explore the utilization of prediction table by program profiling technique and program's characteristic. Additionally, we combine static and dynamic allocating methods in our new hybrid predictor based on the program profiling technique. Simulation results show the proposed method can reduce the hardware cost and achieve higher performance compared with the hybrid predictor proposed in [4] [5].

Keyword: True data dependency, Value Prediction, Program Profiling

1. Introduction

Control flow dependence and data flow dependence are the two fundamental restrictions that limit the instruction level parallelism (ILP) that can be extracted from programs. In order to achieve higher ILP, speculation execution technique has been adopted. There is an uprising demand for speculative execution, as the superscalar architectures have become increasingly popular in the current processor designs. Now people use various methods to remove

黃樹林　明志技術學院電機工程系副教授

the constraints of sequential instruction execution and exploit instruction parallelism.

Control dependences occur due to conditional branch instructions that can cause a potential change in control flow based on the outcome of the branch. In order to eliminate the control dependences, branch prediction and control speculative execution are used. Value prediction is a technique used to break the true data dependence by predicting the outcome of data value. That is, when an instruction is fetched, its result can be predicted so that the subsequence instructions that depend on the result of the instruction can be executed in parallel with the instruction rather than stalled. Many predictors have been proposed, for example, last value prediction [1], stride prediction [2], context predictors [3], and hybrid predicting approaches [4]. Though these methods can achieve high ILP, their hardware cost is high. In [4][5], they suggest us to use a hybrid mechanism with three predictors (last value, stride, and context based predictor) to get high prediction rate for all instructions. In this paper, we propose a new technique combining hardware method and profiling technique to improve value prediction performance and the utilization of prediction table.

2. Related Work

2.1. Value Predictor

Value Predictor is necessary for making correct value prediction. There are many kinds of value predictor. In the following paragraph, those predictors will be introduced. We should use different value predictors to make prediction based on the behavior of instructions.

By observing the data sequences generated from the instruction, some interesting patterns give us some ideas to predict data values. Some instructions always make the same value, for example: 3, 3, 3, 3, 3, ⋯. Some experiment results have shown that such sequence occurs very often. Some instructions make regular values, we call those values: stride data sequence --- the next data value is always a stride difference with the previous data value. For example: 2, 4, 6, 8, 10, ⋯, but some data sequences are complex, for example: 3, 3, 3, 4, 5, 3, 3, 3, 4, 5, 3, 3, ⋯ , it needs to use context predictor to make prediction for those complex data sequences. There are still a lot of more complex data sequences that are hard to classify, so we think those data sequences are unpredictable. When those instructions are encountered, no prediction is made.

2.1.1 Last Value Predictor

Last Value Predictor is the simplest and typical predictor. When we encounter data sequences that are constants, then last value predictor should be used to make prediction [4]. Last value predictor simply predicts the last value as its target value. The main part of the predictor is a Value History Table (VHT) that stores the last result produced by the previous instructions that are currently mapped to the entry. The VHT of last value predictor has two fields --- Tag, Value. Tag field stores the identity of the instruction that is currently mapped to that entry, and the Value field stores the last result for that instruction.

2.1.2 Stride Predictor

Stride value predictor is used to capture the characteristic of stride data sequences. Customarily, we call the difference of the two most recent values as stride [4,7]. For example, if the stride data sequences are 2, 4, 6, 8, ⋯, then the stride of such sequences is 2. Stride value predictor adds the stride to the most recent value to produce the next value.

Counter field stores a value, and we can determine whether a prediction should be made by the value. This value will be adjusted based on the historical performance of last value predictor. For example, the counter field increases by 1, when predictor makes correct prediction; the counter field decreases by 1, when predictor makes incorrect

prediction. When the counter is below certain threshold, we don't make prediction. Initially the counter was set to 1; the threshold was set to 2. So we don't make any prediction when instructions are encountered for the first time.

2.1.3 FCM

There are still a lot of instructions that produce complex data sequences. For example, a, a, a, b, c, a, a, a, b, c, a, … . Those data sequence can be predicted by their history. Context based predictor is designed for such data sequences. By the instruction's history, Context based predictors predicts the next value [4,6].

Finite Context Method Predictor (*fcm*) is the typical context based predictor [4]. " *fcm* predictors rely on mechanisms that predict next value based on a finite number of preceding values. An order *k fcm* predictor uses *k* proceeding history values." *fcm* predictors use counters to count the number of repetitions of values that occurs immediately following a certain context pattern. Hence for each context pattern there must be as many counters as the values following the context. The value with the maximum counter is the predicted next value.

Another Scheme to capture the recurrence of a behavior pattern among instruction results is to use an elaborate two-level prediction scheme [5]. This method incorporates with the level prediction and correlated base mechanism, and has shown that such a method can carry out highly accurate branch prediction.

However, incorporating the 2-level prediction concept into data value prediction is not as straightforward as incorporating it into branch prediction. The primary difficulty is that the result of an instruction can take any one of 2^W value, for reasonable values of W such as 32 or 64. There are so numerous possibilities in value prediction not just two target values for the branch prediction.

Nevertheless, because of program characteristic and value locality, the value outcome of the instruction cannot be such random. Experiment has showed that most parts of instructions generate less than 4 values. Figure 1 is an example of the two level predictors. The prediction unit contains two parts: VHT (value history table) and PHT (pattern history table). The value history table has four fields: Tag, LRU Info, Data Values, and Value History Pattern.

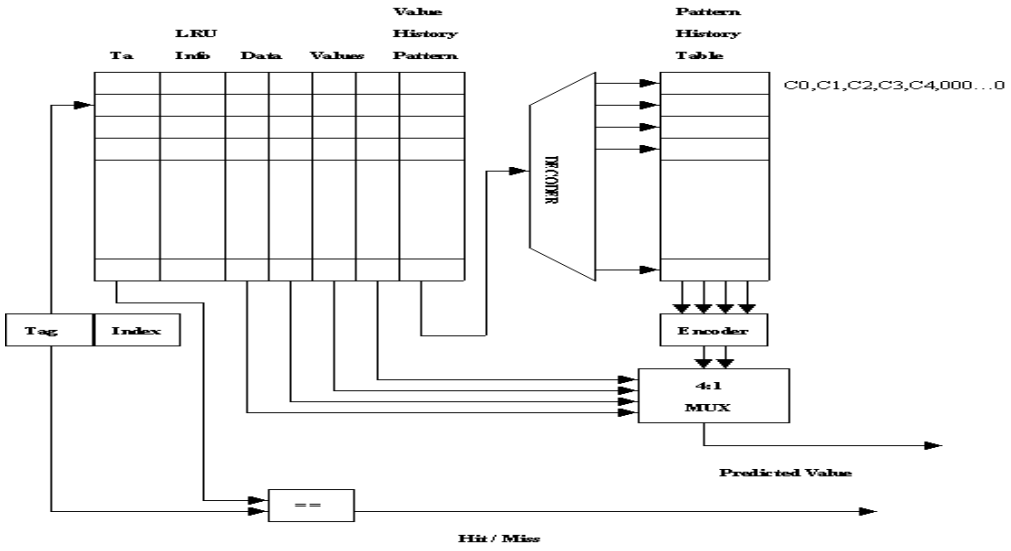

Figure 1. Two level predictor

The Data Values field stores up to 4 most recent unique values. The 4 values binary encoding by {00, 01, 10, 11}. By selecting one of the 4 outcomes from {00, 01, 10, 11}, and taking the value which currently associated with that outcome to be predicted value. The LRU Info field records the order in which the 4 data values were last seen. When a fifth unique value is produced, it replaces the Data Values field the least recently seen value out. Value History Pattern fields stores the pattern of values which produced by instruction, and Value History Pattern is the index to Pattern History Table (PHT) which is second level of the predictor. Each entry of the pattern history table holds four count values {C0, C1, C2, C3} that associate to the four data values in value history table and are 4 bit-size.

2.2. Hybrid Predictor

Last value predictor simply predicts the last previous value as its target value. Stride value predictor is used to capture the characteristic of stride data sequences. Still a lot of instructions that produce data sequences that can't be predicted by the predictors described above. However, such data sequences could be predicted by their history. Context based predictors predicts the next value by the instruction's previous history. It predicts one of the history values when the same context repeats.

But no single prediction scheme can make high prediction rate for all types of instructions. Even in the same program, different instruction blocks have different value localities and different characteristics. Each predictor is just suitable for some instructions. It suggests us to use hybrid predictor to get high prediction rate for all instructions [5]. There are some defects in hybrid predictor. For example, if we use a hybrid predictor with three predictors (last value predictor, stride predictor, and context based predictor) then we must keep the prediction information in the three predictors for each instruction. That is, for each instruction, we must use three times of prediction table in hybrid predictor than in single predictor. But we only use one predictor's result by some mechanism (for example, using confidence counter). Besides, experiments have shown that the predictability of instructions is not uniformly among the program. Some instructions are highly predictable, and some are highly unpredictable.

2.3. Program Profiling

We can use hybrid predictor to cover all different instruction characteristics. However, by such pure hardware methods, the hardware cost is too high to be accepted, and the efficiency is low. Due to conflict miss problem, the unpredictable instructions could uselessly occupy the prediction table and evacuated the predictable instructions. As a result, the utilization of prediction table is low.

Feddy Drabby [8] proposed a compiler-aided scheme to help value prediction by program profiling. It uses program-profiling method to collect information about the predictability of instructions in a program. The compiler that acts as a mediator passes this profiling information to the value prediction hardware mechanism. The collected information is used by the hardware in order to reduce misprediction and to achieve better utilization of prediction table. In general, the idea of profiling techniques is to study the behavior of the program based on its previous run. In each run, the program can be executed based on different sets of input parameters and input files. Experiments [5] have shown that different input files do not dramatically affect the prediction accuracy of several examined benchmarks. And the correlation between the predictability of instructions under different runs of a program with different input files and programs is high.

3. Our modified method and new architecture

Our method is a modification of the profiling

technique which uses the collected information to determinate the allocation of our new hybrid predictor. It consists of two phases: profiling phase and run-time phase. The profiling phase is divided into three steps. In the first step, the program is compiled as usual and the code is generated. In the second step, the profile image of the program is collected.

The profile image describes the prediction tendency of each instruction. In order to get this information, the programs is run on the Shade simulator that is designed to emulate the operation of the value predictor and can measure the profiling image for each instruction. Such profiling image includes execution count, the prediction rate of the instruction and the suitable predictor for the instruction.

The output of the profile image is organized as a table. Each entry is associated with one instruction and consists of four fields: the instruction address, the executing count, the prediction accuracy and suitable predictor. Note that the table has been sorted by the executing count and the instructions with the lower prediction accuracy (e.g. <0.7) are filtered out. In the final step, the compiler generates the function codes that can initialize the allocation of our new hybrid predictor with the output table. Figure 2 shows the profiling phase process.

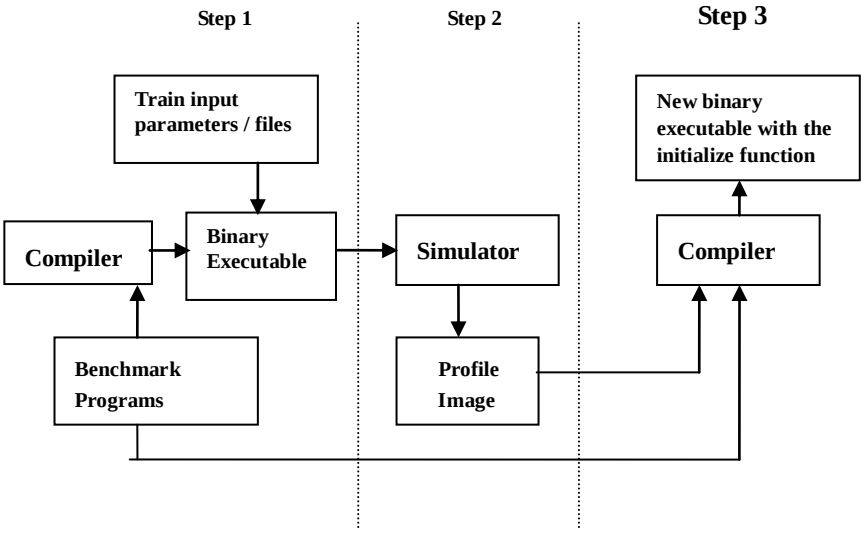

Figure 2. The process of the profiling phase

The run time phase is divided into two steps. The first step makes initial allocation by using the profiling output. We only do the static allocation for the last-value predictor and stride predictor. Because the FCM predictor needs to learn repeating sequences to predict arbitrary repeating patterns. So, it is unnecessary to allocate the FCM predictor statically. The next step makes the dynamic value prediction at run time.

Our new hybrid predictor includes three parts: last-value predictor, stride predictor and FCM predictor. In order to design the architecture of our new predictor, we make the experiment of the hybrid predictor with infinite size in each predictor.

The hybrid predictor of the experiment is composed of one FCM predictor and one stride predictor. The stride predictor is based on the two-delta predictor. The two predictors make predictions simultaneously for each instruction. If one of the two predictions is correct, the correct

prediction count will be increased by one. The experiment results are shown in Fig. 3.

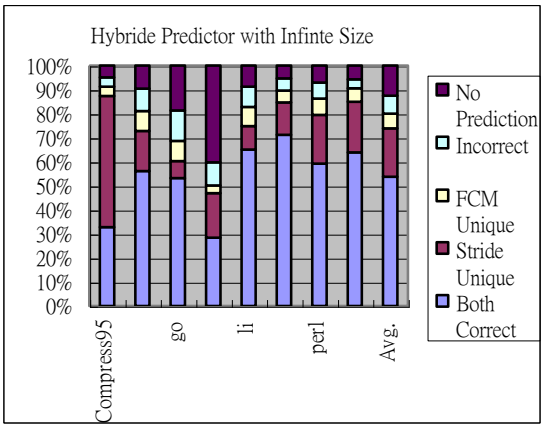

Figure 3.The Prediction Rate of Hybrid Predictor with Infinite Size

From the simulation results, we know the Both Correct has the highest rate of 53.7%, the prediction rate of the Stride Unique and FCM Unique are 20.2% and 6% respectively. Because the last-value predictor is of the low cost and the FCM predictor is too expensive. So, we prefer to use the last-value predictor or stride predictor as the major component. In order to achieve high performance, different types of the correct prediction must be allocated in the suitable predictor.

We use the profile image output to help the allocation of the last-value predictor and stride predictor. Firstly, the last-value predictor will be allocated for the instructions with high prediction accuracy. We use the static allocation method for the last-value predictor. This means it could not be replaced any time. Then the stride predictor is allocated initially for the rest instructions. But the stride predictor uses dynamic allocation and can be replaced at run time.

The last-value predictor in our architecture is simple and very accurate as shown in Figure 4. If it is hit, the other predictors don't need to make prediction. Thus the three predictors are unnecessary to make predictions simultaneously for each instruction. If it is missed, the prediction is chosen from the other predictors (stride and FCM predictors) the same as the operation of general hybrid predictor.

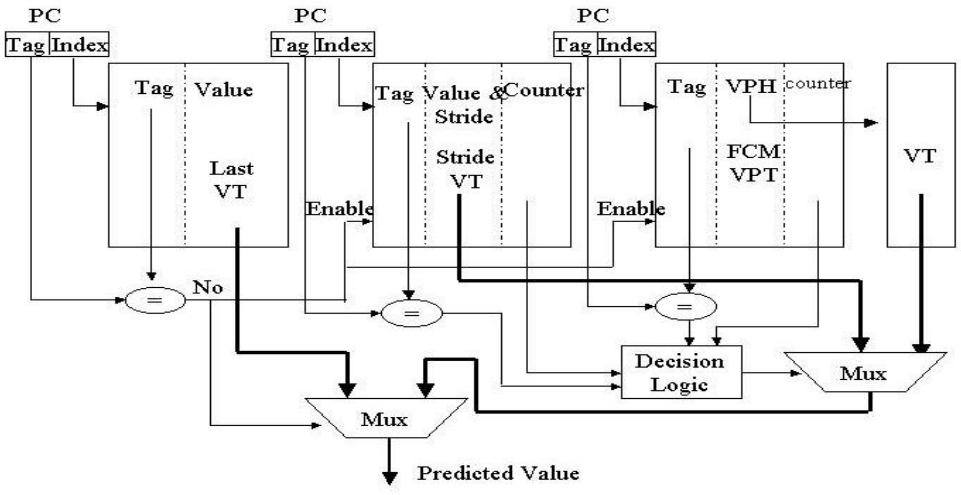

Figure 4. The Architecture of New Value Predictor

4. Numerical Results

We use the SPECint95 as the benchmarks for the simulation. All simulations were run on Sun Ultra-sparc processor with Solaris O.S. The simulation tools we use are Shade and SpixTools released by SUN. The simulation results of the general hybrid predictor and our new predictor with various entries (from 1024 to 4096) are shown in Figure 5 and Table 1, respectively. The general hybrid predictor includes one stride predictor and one FCM predictor with the same entries. But in our new predictor the last-value predictor and the FCM predictor are of fixed size (512 entries and 256 entries) for all simulated architectures. Only the size of the stride predictor can be changed in our new predictor. We use Table 1 to compare the cost and the prediction rate for the different architectures. The *H_n* represents the general hybrid predictor with *n* entries and the *New_n* represents our new predictor with *n* entries stride predictor. As we discussed in the previous paragraph, the performance of the new predictor is better than the general hybrid predictor and the hardware cost is reduced obviously.

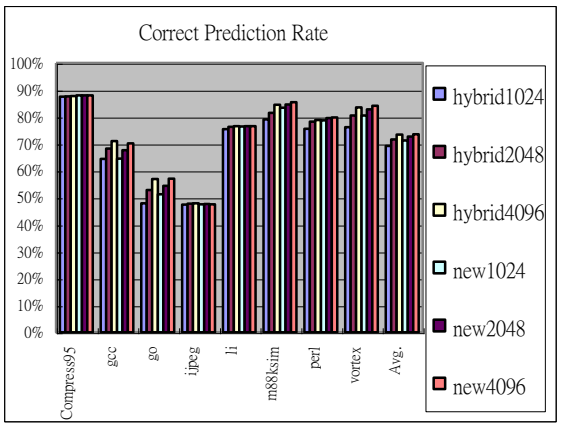

Figure 5. Correct Prediction Rate of Our New Hybrid Predictor

Type	H_1024	New_1024	H_2048	New_2048	H_4096	New_4096
Cost (Bits)	194K	140.75K	384K	225.75K	760K	393.75K
Correct Rate	0.6918	0.7127	0.7160	0.7264	0.7336	0.7356
Incorrect Rate	0.0637	0.0457	0.0642	0.0472	0.0641	0.0449
No Prediction	0.2445	0.2416	0.2198	0.2264	0.2023	0.2195
Prediction Accuracy	0.9094	0.9397	0.9122	0.9390	0.9146	0.9425

Table 1. The Comparison between Hybrid and Our New Predictors

5. Conclusions

From the simulations shown above, the 512-entry last-value predictor has a very high prediction rate (>95%) for all the case. This will be a good choice for the high confidence prediction at run time. In this paper, we have proposed a new hybrid predictor using an appropriate predictor for each instruction.

References

[1] Lee, J.K.F., and A.J. Smith, "Branch Prediction Strategies and Branch Target Buffer Design", In *Proceeding of 8th Annual International Symposium on Computer Architecture*, May 1981, pp. 135-148.

[2] M. H. Lipasti and J. P. Shen, "Exceeding the data flow limit via value prediction," in Proceedings of the 29th Annual ACM/IEEE International Symposium and WorkShop on Microarchitecture,

pp.226-237, Dec. 1996.

[3] Feddy Gabbay, Avi Mendelson, " Speculative execution based on value prediction. EE Department TR 1080, Technion – Israel Institute of Technology, Nov. 1996.

[4] Yiannakis Sazeides, James E. Smith, "The Predictability of Data Values," in Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 248 –258,1997.

[5] Kai Wang and Manoj Franklin, "Highly Accurate Data Value Prediction using Hybrid Predictors," in Proceedings of the fourth International Conference on High Performance Computing, pp. 358 –363,1997.

[6] Yiannakis Sazeides and James E. Smith, "Modeling Program Predictability," in Proceedings of the 25th Annual International Symposium on Computer Architecture, pp.73 –84,1998.

[7] Brad Calder, Glenn Reinman and Brad Calder, "Selective Value Prediction," in Proceedings of the 26th annual international symposium on Computer architecture, pp.64 – 74,1999.

[8] Feddy Gabbay and Avi Mendelson, " Can Program Profiling Support Value Prediction?", in Proceedings of the Thirtieth Annual IEEE/ACM International Symposium on Microarchitecture, pp.270 –280, 1997