ADVANCED TECHNIQUES FOR IMPROVING INDIRECT BRANCH PREDICTION ACCURACY

Adrian Florea and Lucian N. Vintan
Computer Science Department
"Lucian Blaga" University of Sibiu,
E. Cioran Street, No. 4, Sibiu-550025, Romania
Tel./Fax: +40-269-212716, E-mails: adrian.florea@ulbsibiu.ro, lucian.vintan@ulbsibiu.ro

KEYWORDS

Indirect branches / function calls, dynamic branch and value prediction, execution-driven simulation, SPEC benchmarks, prediction accuracy, correlation information, branch arity.

ABSTRACT

Deep pipelines and fast clock rates are necessitating the development of high accuracy branch predictors. From microarchitectural viewpoint, in the last decade the importance of indirect branch prediction increased even though, in the computing programs the indirect jumps remain less frequent than the more predictable conditional branches. One reason refers to predicative execution that implies decreasing of conditional branches number. The dimension took by the desktop, visual or object-oriented applications development (C++, Java – characterized by a large amount of indirect calls comparative to procedural programs), represents another reason which illustrates that indirect branch prediction misses start to dominate the overall misprediction cost. Since the maximum prediction accuracy obtained by a feasible PPM predictor and reported in literature is around 90% implies the necessity of implementing new efficient indirect branch prediction schemes. Thus, we developed a hybrid predictor with arity-based selection that improves indirect branch prediction accuracy reaching in average to 93.77%, comparable with a multi-stage cascaded predictor. We also showed that a modified Target Cache structure based on confidence mechanism and indexed with extended global correlation information represents a more simple and feasible solution that could replace the more complex PPM predictor.

I. INTRODUCTION

Deep pipelines and fast clock rates are necessitating the development of high accuracy branch predictors. The most branch prediction research is based on two closely related correlation mechanisms (local and global). The global method exploits correlation between the outcome of a branch and the outcome of neighboring branches that are executed immediately prior to the branch. In contrast, the local method depends on the observation that the outcome of a specific instance of a branch is determined not simply by the past history of the branch, but also by the previous outcomes of the branch when a particular branch history was observed. Yet branch prediction is a specific example of a far more general time series prediction problem that occurs in many diverse fields of science. It is therefore surprising that there has not been more cross-fertilization of ideas between different application areas. A notable exception is a paper by Mudge (Mudge et al. 1996) that demonstrates that all two-level adaptive predictors implement special cases of the PPM (markovian) algorithm that is widely used in data compression. Mudge uses the PPM algorithm to compute a theoretical upper bound on the accuracy of branch prediction, while Steven (Steven et al 1999) demonstrates how a two-level predictor can be extended to implement the PPM algorithm with a resultant reduction in the misprediction rate. Other researchers developed some more sophisticated predictors based on neural networks algorithms (Jimenez 2002, Egan et al 2003, Seznec 2004).

A particularly difficult challenge consists in target prediction for indirect jumps and calls. Because the target of an indirect jump (call) can change with every dynamic instance of that jump, predicting the target of such an instruction is really difficult. From microarchitectural point of view object-oriented programming techniques exercise different aspects of computer architecture to support the object-oriented programming style (Calder and Grunwald 1994, Florea et al. 2004). In the last decade the importance of indirect branch prediction increased even though, in the computing programs the indirect jumps and calls remain less frequent than the more predictable conditional branches. One of the reasons refers to predicative execution that implies decreasing of conditional branches number. The dimension took by the desktop, visual or object-oriented applications development (C++, Java – characterized by a large amount of indirect branches comparative to procedural programs), and respectively, the portability trend of many of them, as well as the usage of Dynamically-Linked Libraries, represent other reasons which illustrate that indirect branch prediction misses start to dominate the overall misprediction cost. Knowing that the Pentium4 equivalent processor performance degrades by 0.45% per additional branch missprediction cycle and, additional, a very small number of static indirect

Proceedings 19th European Conference on Modelling and Simulation
Yuri Merkuryev, Richard Zobel, Eugène Kerckhoffs © ECMS, 2005
ISBN 1-84233-112-4 (Set) / ISBN 1-84233-113-2 (CD)

branches is responsible for more than 90% of dynamic indirect jumps (Calder and Grunwald 1994, Florea et al. 2004), results that the overall performance of architectures are very sensitive to indirect branch prediction. Since the prediction accuracy generated by classical schemes Branch Target Buffer or Last Value Predictor is less than 75% (Chang et al 1997) and, the maximum value obtained by a feasible PPM (Partial Prefix Matching) predictor, reported in literature is around 90%, implies the necessity of implementing new performing indirect branch prediction schemes (hybrid or cascaded predictors, or even adapted from value prediction – contextual, PPM predictors).

One goal of this paper is to show that a modified Target Cache structure based on confidence mechanism and indexed with extended global correlation information represents a more simpler and feasible solution that could replace the more complex PPM predictor. We developed a path based predictor derived from the native Target Cache (Chang et al. 1997), attaching a confidence counter (degree of reliability) at every Target Cache location, together with a replacement mechanism based on LRU (least recently used), confidence and both of them. The confidence mechanism performs speculation control by limiting the prediction to those that are likely to be correct. We also extend the global correlation information to identify the right context of indirect jump occurrences. The prediction accuracy generated by the new scheme proposed is only with 0.4% under the accuracy provided by a PPM predictor, but our scheme is much simpler and more feasible to be implemented in hardware.

However, the best result reported in this work consists in developing of a hybrid predictor with arity-based selection. Our new scheme improves indirect branch prediction accuracy reaching in average to 93.77%, comparable with a more complex multi-stage cascaded (Driesen and Hoelzle 1999). We classified branches according to a dynamic measure, the number of different targets encountered in a program run, or branch *arity*. The arity of a branch we determined in a profiling run. The profile information table helps to select which component predictor will predict at every moment. The component predictors are: a *LastValue* predictor (Lipasti et al. 1996, Florea et al. 2002) for monomorphic (1 target) or duomorphic (2 targets) indirect jumps, and respectively, the best contextual (markovian) predictor (Sazeides 1999, Florea et al. 2004) for polymorphic branches (more than 2 targets). Therefore, we determine first, based on laborious simulations what is the optimum search pattern when different contexts are used. Thus, we obtained that the markovian predictor has the best behavior in two different contexts: using a *short history* (the last 32 targets and a search pattern of 3) and a *rich history* (last 256 targets and a search pattern of 6).

The organization of the rest of this paper is as follows. In section II we review related work in the field of indirect branch prediction. Section III describes the implemented predictors. Section IV includes simulation methodology and experimental results obtained using the simulator that we developed. Finally, section V suggests directions for future works and concludes the paper.

II. RELATED WORK

Several forms of BTBs, applied to indirect branch prediction, have been studied by Lee (Lee and Smith 1984). The simplest BTB keeps the most recent target for each branch. In the case of a BTB target mispredict, the predicted target address is replaced. An improvement upon that basic BTB configuration was proposed by Calder (Calder and Grunwald 1994), where a two-bit counter is used to limit the update of the target address only after two consecutive mispredictions have occurred (BTB2b strategy). The BTB2b can produce a better branch prediction ratio for C++ applications by taking advantage of the locality exhibited by targets of virtual function calls (C++ polymorphic calls). Similar results we obtained in a previous work (Florea et al. 2004). We showed that target localities for some object oriented test programs and SPEC benchmarks ('95 and 2000) is over 90%, considering different context orders, varying from 1 to 32.

In (Kaeli and Emma 1991) and (Kaeli and Emma 1997) the authors described two mechanisms that accurately predict the targets of two special classes of indirect branches: (i) subroutine returns, and (ii) indirect jumps generated by switch statements. A Call/Return Stack (RAS) was described which uses the inherent correlation between procedure calls and returns to pair up the correct target address with the current return invocation.

In (Driesen and Holzle 1998a) it is examined a modified structure of the two-level adaptive branch predictor to predict the targets of indirect branches. Their design allows path-based correlation to be exploited by recording partial previous targets in the history register (instead of branch directions since the indirect jumps have always the same direction). Driesen performed an exhaustive search on a large number of two-level predictor configurations. The best prediction accuracy was generated using the GAp structure (Global History Register, Per-address Pattern History Table (PHT) configuration). However, the application of their results is limited since they recorded full target addresses in the history register and assumed infinite PHTs. Also, the authors explore realistic designs, varying how and when targets are recorded in the history register, the size and associativity of the PHTs, and the size of the history register. They also proposed using dual-path hybrid predictors with confidence counters to improve prediction accuracy under a fixed hardware budget. Each component was a two-level predictor with a different path length. Their findings suggest that the best dual path predictor had components with the short and a long path length. Some of our results exhibit the same conclusion. If the context would permit it could be seen a correlation between branches

situated at a large distance in the dynamic instruction stream. Whether in the current path-based predictors, the *N* most recent target addresses are hashed together to form an index into a table, where *N* is some fixed integer, Stark proposed (Stark et al. 1998), using profiling information, to select the proper value of *N* for each branch, thus, achieving an extremely accurate branch prediction.

In (Chang et al 1997) was proposed a structure similar to the two-level adaptive scheme, named Target Cache (TC). Its history register recorded partial targets from a selected group of branches (the group may include all branches, indirect branches, conditional branches or calls/returns). Their path-based simulation results showed the dependence of indirect branch predictability on the type of correlation.

In (Driesen and Holzle 1998b) was performed another study about the predictability of indirect branches using filtering. The proposed filtering scheme improve the prediction ratio by isolating monomorphic and low entropy branches from a main body predictor and reducing the collision factor in the later. Their *cascaded predictor* included a GAp/Dual-path hybrid predictor as their major component and strict/leaky filter, implemented as a BTB–like structure. In (Driesen and Holzle 1999) the two-stage prediction was generalized to multistage prediction. At 1.5K entries a three-stage predictor reaches 94% accuracy, the hit rate of a hypothetical two-level predictor with an unlimited, fully associative prediction table. At 6K entries, accuracy increases to 95%, the limit achieved by an idealized twelve-stage cascaded predictor with an unlimited hardware budget.

In (Kalamatianos and Kaeli 1998) the authors applied the PPM algorithm to the task of indirect branch prediction. The PPM predictor shortens a history pattern bit by bit, and looks it up successively smaller stages. Each stage is half the size of its predecessor. The bits correspond to branch targets, so this scheme tests ever shorter path lengths. Kalamatianos combines a viable implementation of the PPM algorithm with dynamic per-branch selection of the path-based correlation. The maximum prediction accuracy reported is 90.53%.

III. INDIRECT JUMPS TARGET PREDICTION

Extending Target Cache structure with a confidence mechanism

The first considered predictor was a tagged Target Cache predictor inspired from that presented in (Chang et al. 1997). The Target Cache improves the prediction accuracy for indirect branches by choosing its prediction from the last *N* targets of the indirect branch that have already been encountered. When an indirect jump is fetched, both the PC and the *globalHR* (a history register that retains the behavior of last HRgLength conditional branches) are used to access the TC for predicting the target address. As the program executes, the TC records the target for each indirect jump target encountered. Our proposed scheme, for set selection, uses the least significant bits of the word obtained by hashing (XOR) the indirect jump's address (PC) and the *globalHR*. The most significant bits of the obtained output form the *Tag*. In the case of a hit in TC the predicted target consists in the corresponding address belonging to that TC set (field *Adr* from figure 1). In case of a misprediction, (the Tags coincide but the target addresses differ) after the indirect branch is resolved, the Target Cache entry is updated with its real target address. It is implemented a LRU replacement algorithm. We have implemented and simulated a *P*-way set associative TC, where *P*=1, 2, 4, 8 like that presented in figure 1. In the case of a miss in TC the prediction is considered wrong, it doesn't propose any value and it is added a new entry in the respective set updating with the proper *tag* and the proper *target,* accordingly with the specified replacement algorithm.

A common criticism for all the present two-level adaptive branch prediction schemes (applied either to conditional branches or indirect jumps) consists in the fact that they used insufficient global correlation information (Vintan and Egan 1999). There are situations when for the same static indirect branch and in the same *globalHR* (see figure 1) context pattern it's possible to find different targets. If each bit belonging to *globalHR* will be associated during the prediction process with its corresponding PC the current indirect branch's context becomes more precisely and therefore its prediction accuracy could be better. Next, we developed a *path based predictor*, through extending the correlation information according to the above idea (Vintan and Egan 1999). Thus, the first level of history of Target Cache predictor records the path (the conditional branches' addresses) leading to the current indirect jump. Extending the correlation information in this way suggests that at different occurrence contexts of a certain indirect jump it will access different sets from TC structure, reducing a significant amount of interferences and increasing the prediction accuracy. Compression of this complex information is possible and even necessary, taking into account the request of reasonable costs for these schemes. The hash function used is a simple XOR.

The next contribution follows up to improve indirect branch prediction accuracy by selectively ignoring some predictions. Therefore, we attached a confidence counter (degree of reliability) at every Target Cache location, together with a replacement mechanism based on *LRU*, *confidence* and on the both metrics superposition called by us *MPP* (performance potential minimum) – see figure 1. A confidence mechanism performs speculation control by limiting the predictions to those that are likely to be correct. A high confidence degree represents *the continuous correct predictability in a given history* of an indirect jump while the LRU field means its *activity degree* (how many times this branch was accessed). Both *Confidence* field and *LRU* field were implemented as saturating counter being represented on different number of bits. For exploiting

this subsection results we used some new metrics, proposed in (Deswet et al. 2002):

We call only here ***prediction accuracy*** **(*Ap*)** the probability that prediction generated by a high confident state to be correct. ***Usage*** represents the prediction performed degree, practically the number of cases when the automaton was in a high confident state (confidence is greater than a certain threshold and TC is ensured to make a prediction), reported to the total number of indirect jumps.

$$A_p = \text{Prob}(\text{correct prediction} \mid \text{High confidence}) = \frac{HC_{corr}}{HC_{corr} + HC_{ntcorr}} \quad (1)$$

$$Usage(\text{prediction performed degree}) = \frac{HC_{corr} + HC_{ntcorr}}{\text{Total\_indirect\_jumps}} \quad (2)$$

We define the ***predictors' overall performance*** metric, the product: $\mathbf{P = A_p \cdot Usage}$ (3)

From (1), (2) and (3) results that:

$$P = \frac{HC_{corr}}{\text{Total\_indirect\_jumps}} \; . \quad (3')$$

We determined how are influenced this metrics by a parameterized threshold.

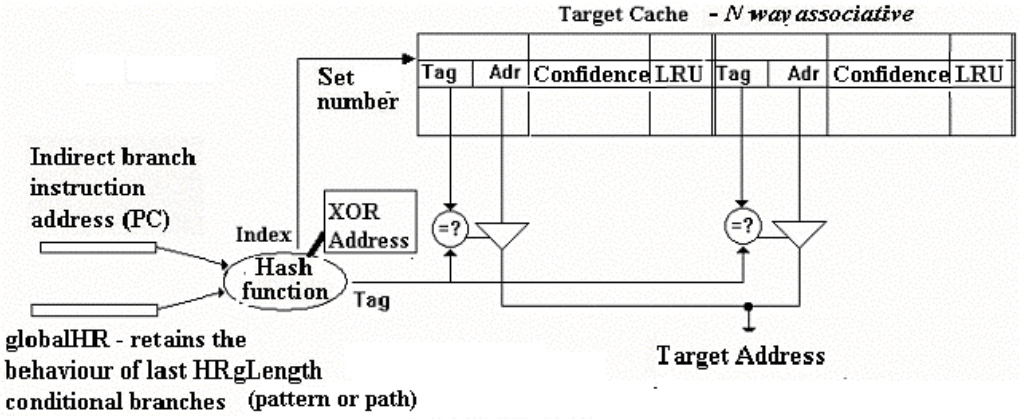

Figure 1. Extending Target Cache Structure with a Confidence Mechanism

Arity-based selection hybrid predictor

In a previous research (Florea et. al 2004), in order to have an ultimate context predictability metric of indirect branches we measured their target address value locality. The value locality concept was first introduced by Lipasti (Lipasti et al. 1996) and it represents the likelihood of the recurrence of a previously seen value within a storage location. Accordingly, in our case, we'll say that an indirect jump (call) target value is local if it belongs to the previous *K* dynamic target instances of that certain jump (call). Obviously, a great target locality degree involves great prediction accuracy, too. In other words, the value locality degree obtained for *K* dynamic target instances represents the maximum achievable prediction accuracy using a context predictor of order *K*. Therefore, our approach establishes an analogy between value prediction and, respectively, indirect jumps target prediction. Statistical results based on simulations have proved that indirect jumps targets are characterized by higher degree of value locality (over 90% for K≥4). The main causes for this phenomenon are: the compiler routines that resolve virtual function calls, inheritance and polymorphism from object-oriented programs, indirect calls through function pointers, switch/case statements, etc.

In the same research we find that a complete PPM predictor having an associative indirect jump value prediction table (JVPT) with 256 entries generates average prediction accuracy between 89.33% and 91.58% depending on the context used in simulation: *short history* (the last 32 targets) and a *rich history* (the last 256 targets), the search pattern varying descending from of 4 to 1. Taking into account PPM's complexity we tried to implement a simplified PPM or a confidence-base hybrid predictor having as components two contextual predictor of different order. Knowing that only a small amount of the path information leading up to a branch is needed for prediction (Stark et al. 1998) we tried to find what the optimum search pattern is when different contexts are used. Thus, we obtained that the markovian predictor has the best behavior in two different contexts: using a *short history* (the last 32 targets and a search pattern of 3) and a *rich history* (last 256 targets and a search pattern of 6).

The values obtained (orders of Markov predictors) we used in developing of a hybrid predictor with arity-based selection. We classified branches according to a dynamic measure, the number of different targets encountered in a program run, or branch *arity*. The arity of a branch we determined in a profiling run. The profile information table, completely associative and retaining the arity of every indirect jump, helps to select which component predictor will predict at every moment. The component predictors are: a *LastValue* predictor for monomorphic or duomorphic indirect jumps, and respectively, the best contextual predictor, previously determined, for polymorphic branches. Both LastValue and contextual predictor are completely associative and are indexed in the instruction fetch stage with indirect branch address (PC).

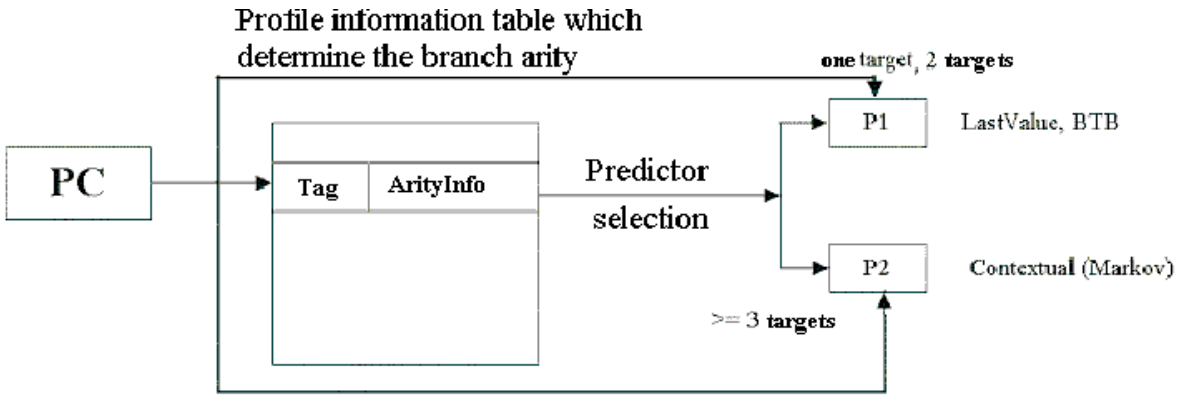

Figure 2. Arity-based Selection Hybrid Predictor

IV. SIMULATION METHODOLOGY AND EXPERIMENTAL RESULTS

We developed a cycle-accurate execution driven simulator derived from the *sim-outorder* simulator in the *SimpleScalar* tool set (Burger and Austin 1997). The baseline superscalar processor supports out-of-order instruction issue and execution. We modified it to incorporate the indirect jumps' predictors proposed in section 3 in order to measure target locality, and, respectively to predict targets for indirect jumps and calls.

To perform our evaluation, we collected results from different versions of SPEC benchmarks: 3 integer (*li, go, cc1*) and 4 floating point (*applu, apsi, fpppp, hydro*) SPEC'95 benchmarks. From the CINT SPEC2000 set, it was simulated 8 benchmarks (*gzip, b2zip, parser, crafty, gap, gcc, twolf* and *mcf*). We simulated some SPEC'95 benchmarks too in order to compare their behavior with that one involved by more recently SPEC2000. In other words, we intend to discover how these different benchmarks influence the indirect jumps predictors' micro-architectural features.

The number of instructions fast forwarded through before starting our simulations is 500 million. We used the `-fastfwd` option in SimpleScalar / PISA 3.0 to skip over the initial part of execution in order to concentrate on the main body of the programs. Results are then reported for simulating each program for 200 million committed instructions.

For improving indirect branch prediction accuracy the first tentative was to modify the native Target Cache predictor (Chang et al. 1997). In a previous work (Florea et al. 2004) it was studied the potential of native TC and determined the influence of conditional branches global history about prediction. Thus, without taking into account of conditional branches global history, the prediction accuracy produced by a native TC having 64 4-way associative sets is slightly less than 80%. Repeating the simulation process for a 256 set TC we obtained that the prediction accuracy is practically saturated. Anyway, the obtained results are smaller than those obtained using a complete PPM predictor (see section III). For benchmarks with a high number of targets generated by an indirect branch (cc1, li), through indexing the Target Cache with conditional branches global history the indirect branch prediction accuracy increases (in average until 16.93% and also, in cc1 particular case even 45%). Simulation results on 7 SPEC'95 benchmarks show that the optimum accuracy it is obtained by keeping the behavior (T/NT) of last 8 conditional branches encountered (*globalHR* – global history register on HRgLength bits), a pattern longer that this behaving as noise. Even if, in other researches is used a "long" history (Thomas et al. 2003) or "elastic" (variable length depending on every branch) (Stark et al. 1998, Tarlescu et al. 1997), since the simulation results on the native Target Cache structure (Florea et al. 2004) were optimum for a history length of 4 (in average on 7 SPEC'95 benchmarks) or 8 (in average on the 4 benchmarks rich in dynamic indirect jumps – apsi, cc1, li, hydro2d) we decided to continue the simulations using a fix value for *HRgLength* (4 or 8). Figure 3 and table 1 are presenting the quantitative benefits of extending context information that indexes TC structure.

For benchmarks with the largest number of indirect jumps, by extending the correlation **(PC_1, PC_2, …, $PC_{HRgLength}$)** and at the same level of complexity (Target Cache structure of the same size), the prediction accuracy increases (with **8.64%** for HRgLength of 4 and respectively, with **15.16**% for HRgLength of 8 – see table 1). Table 1 illustrates the prediction accuracy in average on apsi, cc1, li, hydro2d testing programs.

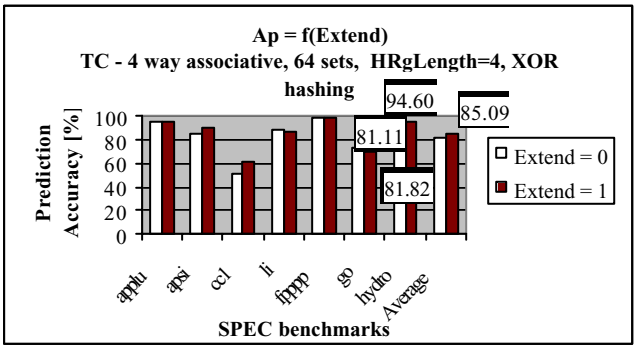

Figure 3. The Influence of Extended Correlation Information about Indirect Branches Prediction Accuracy

Table 1. Increasing the Indirect Jumps Prediction Accuracy by using a more precise Context on Benchmarks with the largest Number of Indirect Jumps, using a TC – 4 way associative; 64 sets

	HRgLength = 4	**HrgLength = 8**
Extend = 0	76.52%	74.56%
Extend = 1	83.13%	**85.86**% (respectively **88.21**% for a TC – **8 way** associative)

Although the improvement obtained by extending the correlation information is obvious, in average, the indirect branches prediction accuracy is still lower than that generated by a complete PPM predictor (**89.33%**). However, particularly there are also very good results: prediction accuracy obtained on hydro2d – 99.98% (HRgLength = 8; TC – 4 way associative; 64 sets, and using rich context PC_1, PC_2, …, $PC_{HRgLength}$), is equal with that exhibits by the complete PPM predictor.

Unpleased by previously results we tried to improve indirect branch prediction accuracy by selectively ignoring some predictions.

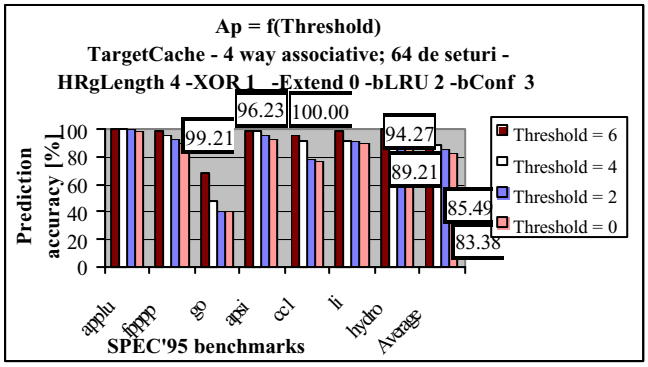

Figure 4. Prediction Accuracy (Ap) varying the Threshold, using a Confidence Mechanism

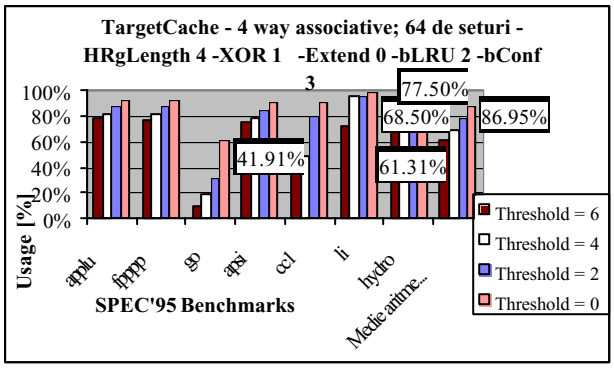

Figure 5. The Prediction performed Degree (the Indirect Jump Fraction having *Confidence>Threshold*)

Practically, the probability that prediction generated by a high confidence states ($\mathbf{A_p}$) to be correct significantly increases through reducing the cases when the structure makes a prediction (between 3.57% and 11.45%

depending the *threshold*). The disadvantage is that the percentage of cases in which is made a prediction dramatically decreases. The efficiency of extending correlation information is proved in this case once again (*predictors' overall performance* increases with 5.62% when is used a rich context to identify the current indirect branch, approaching by the PPM predictors performance). After laborious simulations the conclusion is that attaching a confidence counter the prediction accuracy is improved when this is less selective (see tables 2 and 3).

Table 2. The Influence of Associativity Degree on TC Predictor (with and without Confidence)
(TC - 64 sets -HRgLength 4 -XOR 1 -Extend 1 -bLRU 2 -bConf 3) - I

Associativity Degree (AD)	**Prediction Accuracy (without confidence mechanism)**	***Predictors' overall performance* (with confidence mechanism) - P**			
			Threshold = 1		**Threshold = 2**
AD = 2	78.15%	<	**79.39%**	>	77.91%
AD = 4	82.27%	<	**82.78%**	>	81.32%
AD = 8	82.35%	<	**85.13%**	>	83.87%

Table 3. The Influence of Associativity Degree on TC Predictor (with and without Confidence)
(TC - 128 de seturi -HRgLength 8 -XOR 1 -Extend 1 -bLRU 2 -bConf 3) - II

Associativity Degree (AD)	**Prediction Accuracy (without confidence mechanism)**	***Predictors' overall performance* (with confidence mechanism) - P**		
		Threshold=0	**Threshold=1**	**Threshold=2**
AD = 4	86.40%	**< 87.39%**	85.66%	84.98%
AD = 8	86.47%	**< 88.88%**	**> 87.17%**	> 86.51%

Increasing the associativity degree greater than 8-way, the predictors' overall performance became asymptotical. Thus, for a 8-way associative Target Cache predictor having 128 sets, keeping the behavior of last 8 conditional branches **P=88.97%**, only with 0.4 % under the accuracy provided by a PPM predictor (**89.33%**).

Since for a Target Cache 4-way associative the percentage of cases in which there are made replacements according to LRU principle is less than 1% (except *apsi* and *hydro* benchmarks) it results that, increasing the associative degree and implicit decreasing the percentage of conflict misses the influence of LRU field tend to become insignificant. Thus it might be implemented a trivial replacement algorithm (e.g. FIFO) having benefits about reducing Target Cache structure complexity. Also, the replacement mechanism from Target Cache based upon LRU principle has proved more efficient than that based upon confidence field minimum value (with 2.43%) and even with 0.34% better than the MPP algorithm. This leads to the idea that some indirect branches are replaced before attaining a minimum confidence and later they would prove correct predicted. Furthermore, although it was expected that MPP replacement algorithm to generate the highest prediction accuracy it seems that the minimum values of this metric (MPP) – possible 0 are influenced by the lower confidences (0 – for many times).

The next two figures exhibit what is the *optimum search pattern* when different contexts are used.

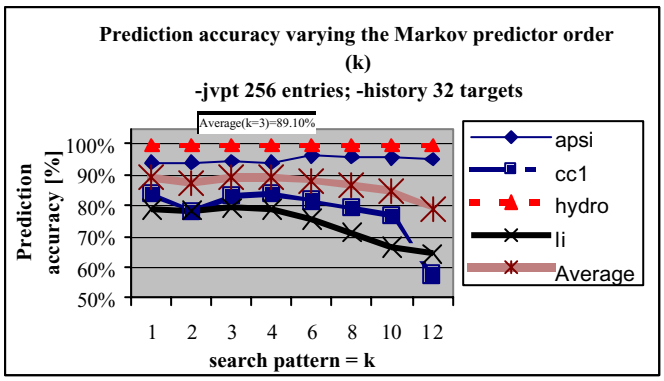

Figure 6. Obtaining the proper Order of Markov Predictor (*poor Context* – 32 targets)

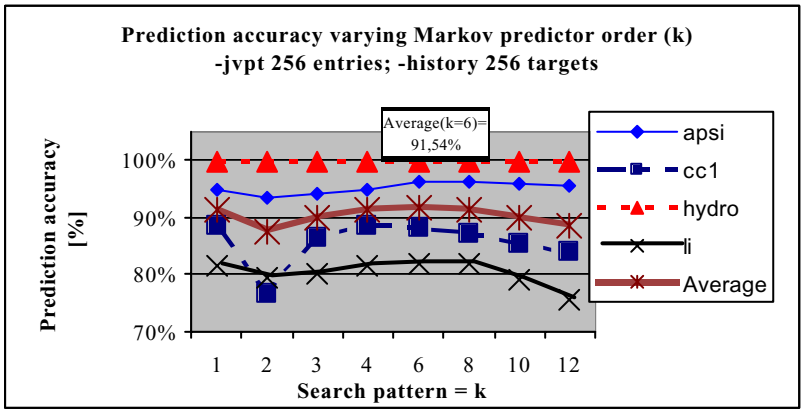

Figure 7. Obtaining the proper Order of Markov Predictor (*rich Context* – 256 targets)

More or less obviously the obtained results suggest that a longer history retained implies a higher prediction accuracy by increasing the search pattern (if the context would permit it could be seen a correlation between branches situated at a large distance in the dynamic instruction stream). The simulation results using a contextual (Markov) predictor showed that for a history of 32 targets the maximum prediction accuracy 89.10% it was obtained for a pattern of 3, while the history became longer the maximum value 91.54% was obtained with a pattern of 6 (Figure 7). This could suggest implementing a hybrid or cascaded indirect branch predictor with component having different path length. The results emphasize the researchers trends (Thomas et al. 2003) to keep in prediction process a very long history. The authors argue that for a branch under prediction, some of the correlated branches may have appeared at a large distance in the dynamic instruction stream. This can happen if two correlated branches are separated by a call to a function containing many branches.

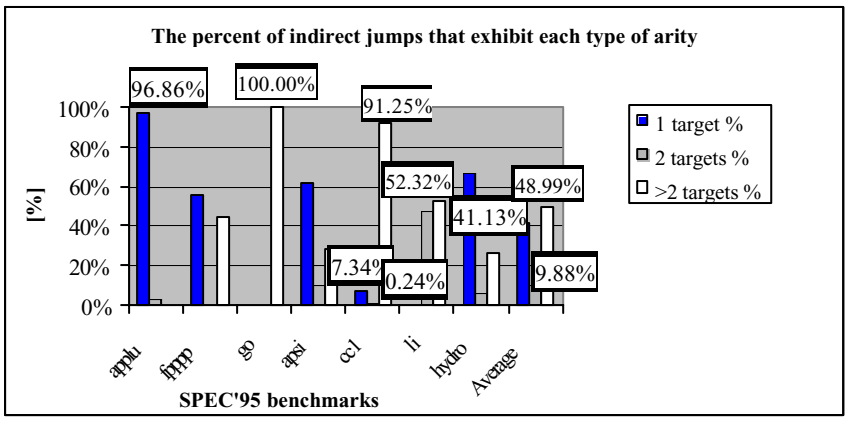

Figure 8. The Indirect Branches Arity –dynamically Point of View

Arity-based classification which classifies indirect branches according to the number of different targets permit us to implement a hybrid predictor having components a Last Value predictor (without history) for a monomorphic branches and the best previous determined contextual predictor for polymorphic jumps. Simulation results on SPEC'95 benchmarks with a large number of indirect branches exhibit 41.13% monomorphic branches, 9.88% duomorphics and 48.99% polymorphics. Our developed hybrid predictor with arity-based selection improves indirect branch prediction accuracy with percentages between 2.44% and 5.42% reaching in average 93.77%, comparable with that reported in literature (Driesen and Hoelzle 1999). Simulation results on SPEC2000 suite show that three of the eight simulated benchmarks (*gcc, crafty, twolf*) generate polymorphic indirect branches in proportion of 96%. In our opinion, the significant percentage of polymorphic indirect branches and higher targets entropy specific for some indirect jumps (see the ccl, li benchmarks) fundamentally limits the indirect jumps prediction accuracy.

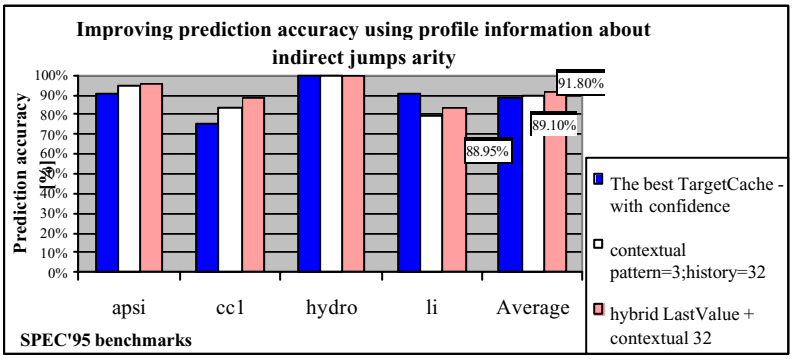

Figure 9. Indirect Branch Prediction Accuracy using a Hybrid Predictor with Arity-based Selection (*poor Context*)

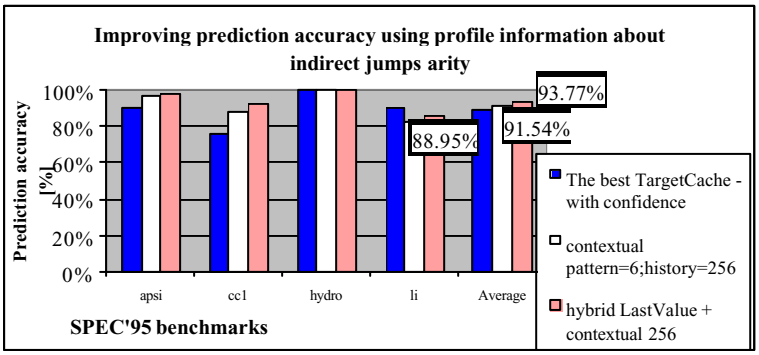

Figure 10. Indirect Branch Prediction Accuracy using a Hybrid Predictor with Arity-based Selection (*rich Context*)

V. CONCLUSIONS AND FURTHER WORK

Due to higher degree of target localities associated with indirect jumps, we predicted these indirect jumps and calls using some contextual value predictors, derived from the complete PPM predictor respectively the Target Cache predictor. The obtained results were better than those reported by other researchers that used more simplified context predictors. PPM predictor seems to be an almost ultimate limit of context target prediction, and, thus, a good frame for further deriving new practical prediction schemes.

In this sense, we tried to extend the context prediction information adding new correlations. This information, available during the instruction fetch stage in the pipeline, consists of global history register together with its corresponding PCs (one PC for each previous encountered branch (Vintan and Egan 1999)). Using this new correlation information together with the global history register, the current indirect jump's context becomes more precisely and therefore its prediction accuracy is showed to be better. Our first simulation results are encouraging; we showed that a scheme based on this principle performs better than a classical Target Cache scheme (Chang et al. 1997), at the same hardware complexity level. Also, we extend and improve the native Target Cache structure (Chang et al. 1997) with a confidence mechanism for improving the indirect jumps' prediction accuracy. The prediction accuracy generated by the new scheme proposed is only with 0.4 % under the accuracy provided by a PPM predictor, but our scheme is much simpler and feasible to be implemented in hardware. The best prediction accuracy was obtained using a hybrid predictor with arity-based selection that improves indirect branch prediction accuracy reaching in average to 93.77%, comparable with a more complex multi-stage cascaded predictor.

The excellent results obtained impose introducing and exploiting the hybrid and cascaded predictor in other computer architecture issues to increase the instruction and thread level parallelism: conditional branch prediction, instruction and register value prediction. As a further work we will try to replace the arity-based selection hybrid predictor with a simple neural network which will select dynamically between ordinary component predictors (Last Value, Target Cache, contextual, hybrid). Also we will study the feasibility of an indirect branch predictor correlated and decision trees based. Another solution could appear from development of some "*semantic predictors*", based on High Level Language applications' information that we prove being important related to indirect jumps generation (polymorphism, indirect function calls, etc.; see our investigations presented in (Florea et al. 2004)). This might be a completely new approach in branch prediction domain, where HLL semantics are often hidden. As far as the architecture designers are concerned, their proposed schemes could be more efficient if not only the object code from benchmarks ("*wear off by any semantic information*") is analyzed but they will also look "*higher*" towards high level sources of simulated programs.

REFERENCES

Burger D. and T. Austin. 1997. "The SimpleScalar Tool Set, Version 2.0", University of Wisconsin Madison, USA, Computer Science Department, *Technical Report #1342*, June, 1997.

Calder B. and D. Grunwald. 1994."Reducing Indirect Function Call Overhead in C++ Programs*"*, In 1994 *ACM Symposium on Principles of Programming Languages*, pages 397-408, January 1994.

Chang P.Y., E. Hao, Y.N. Patt. 1997. "Target Prediction for Indirect Jumps", *Proceedings of the Int'l Symposium on Computer Architecture*, 1997.

Deswet V., B. Goeman, K. Bosschere. 2002. "Independent hashing as confidence mechanism for value predictors in microprocessors", *Int'l Conf. EuroPar*, Augsburg, Germany, 2002.

Driesen K. and U. Holzle. 1998a. "Accurate Indirect Branch Prediction". *In Proceedings of the International Symposium on Computer Architecture*, pages 167-178, Barcelona, Spain, June 1998.

Driesen K. and U. Holzle. 1998b. "Improving Indirect Branch Prediction With Source- and Aritybased Classification and Cascaded Prediction". *Technical Report TRCS98-07*, Computer Science Department, University of California, Santa Barbara, 1998.

Driesen K. and U. Holzle. 1999. **"**Multi-stage Cascaded Prediction". *Euro-Par'99 Conference Proceedings*, Toulouse, France, September 1999.

Egan, C., G. Steven, P. Quick, R. Anguera, F. Steven, and L. Vintan. 2003. **"**Two-level branch prediction using neural networks", *Journal of Systems Architecture 49(12), Elsevier*, December, 2003.

Florea A., L. Vintan, D. Sima. 2002. "Understanding Value Prediction through Complex Simulations". *Proceedings of the 5th International Conference on Technical Informatics*, University "Politehnica" of Timisoara, Romania, October, 2002.

Florea A., L. Vintan, I.Z. Mihu. 2004. "Understanding and Predicting Indirect Branch Behavior". *Studies in Informatics and Control Journal: With Emphasis on Useful Applications of Advanced Technology*, March 2004, Vol.13, No. 1, Bucharest.

Kaeli D. and P. Emma. 1991. "Branch History Table Prediction of moving Target Branches due to Subroutines Returns", *Proceedings of the Int'l Symposium on Computer Architecture*, May 1991.

Kaeli D. and P. Emma. 1997. "Improving the Accuracy of History-Based Branch Prediction". *IEEE Transactions on Computer Architecture, 46(4):469-472*, April 1997.

Kalamatianos J. and D. Kaeli. 1998. *"*Predicting Indirect Branches via Data Compression". *In Proceedings of 31st International Symposium on Microarchitecture*, pages 272-281, Dec. 1998.

Lee J. and A. Smith. 1984. "Branch Prediction Strategies and Branch Target Buffer Design", *Computer 17:1*, January 1984.

Lipasti M., C. Wilkerson, P. Shen. 1996. "Value Locality and Load Value Prediction". In *17th ASPLOS International Conference VII*, SUA, 1996.

Mudge T., I.K. Chen, J.T. Coffey. 1996."Analysis of Branch Prediction via Data Compression". *Proceedings of the 7th International Conference on ASPLOS VII*, Cambridge, MA, USA, October 1996.

Sazeides Y. 1999. **"**An analysis of value predictability and its application to a superscalar processor". *PhD Thesis*, University of Wisconsin-Madison, 1999.

Seznec A. 2004. "Revisiting the Perceptron Predictor". *IRISA research reports, IRISA Editeur*, May, 2004.

Steven G.B., C. Egan, P. Quick, and L. Vintan. 1999. **"**Reducing Cold Start Misspredictions in Two-Level Adaptive Branch Predictors", *Proceedings of the Int'l CSCS-12 Conference*, Bucharest, May 1999.

Tarlescu M., K. Theobald, and G. Gao. 1997. **"**Elastic history buffer: A low cost method to improve branch prediction accuracy". *In Proceedings of the IEEE International Conference on Computer Design*, 1997.

Thomas R., M. Franklin, C. Wilkerson, J. Stark. 2003. "Improving Branch Prediction by Dynamic Dataflow-based Identification of Correlated Branches from a Large Global History". *The 30th Annual International Symposium on Computer Architecture*, San Diego, California, 2003.

Vintan L. and C. Egan. 1999. "Extending Correlation in Branch Prediction Schemes". *International Euromicro'99 Conference*, Italy, September 1999.

AUTHOR BIOGRAPHIES

ADRIAN FLOREA obtained an MSE (Computer Science) in 1997 and is a PhD student in (Computer Science) since 2000 from the University "Politehnica" of Bucharest, Romania. He is a lecturer of Computer Science and Engineering at the University ''Lucian Blaga ''of Sibiu, Sibiu, Romania. Adrian is an active researcher in the fields of High Performance Processor Design and Dynamic Branch Prediction. His Web-page could be found at http://webspace.ulbsibiu.ro/adrian.florea

LUCIAN VINTAN obtained an MSE (Computer Engineering) in 1987 and a PhD (Computer Science) in 1997, both from the University "Politehnica" of Timisoara, Romania. He is a Professor of Computer Science and Engineering at the University "Lucian Blaga" of Sibiu, Romania. Professor Vintan is an active researcher in the fields of High Performance Processor Design and Dynamic Branch Prediction and has collaborated with the Computer Architecture Research Group at the University of Hertfordshire since 1996. His Web-page could be found at http://webspace.ulbsibiu.ro/lucian.vintan