World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:12, 2007

Performance Evaluation of Neural Network Prediction for Data Prefetching in Embedded Applications

Sofien Chtourou, Mohamed Chtourou, and Omar Hammami

***Abstract*—**Embedded systems need to respect stringent real time constraints. Various hardware components included in such systems such as cache memories exhibit variability and therefore affect execution time. Indeed, a cache memory access from an embedded microprocessor might result in a cache hit where the data is available or a cache miss and the data need to be fetched with an additional delay from an external memory. It is therefore highly desirable to predict future memory accesses during execution in order to appropriately prefetch data without incurring delays. In this paper, we evaluate the potential of several artificial neural networks for the prediction of instruction memory addresses. Neural network have the potential to tackle the non-linear behavior observed in memory accesses during program execution and their demonstrated numerous hardware implementation emphasize this choice over traditional forecasting techniques for their inclusion in embedded systems. However, embedded applications execute millions of instructions and therefore millions of addresses to be predicted. This very challenging problem of neural network based prediction of large time series is approached in this paper by evaluating various neural network architectures based on the recurrent neural network paradigm with pre-processing based on the Self Organizing Map (SOM) classification technique.

***Keywords*—**Address, data set, memory, prediction, recurrent neural network.

I. Introduction

EMBEDDED systems are widespread in numerous applications and support increasingly complex applications. These embedded software applications are diverse and exhibit varying behavior. This variability come from the nature of the applications or the data they process but also from the supporting hardware components composing embedded systems. This variability runs against stringent real time constraints and therefore it should be smooth out or remove when possible.

Cache memories are small memories used by microprocessors to store temporal data and code in order to avoid accessing the central memory. Cache memory access from an embedded microprocessor might result in a cache hit where the data is available or a cache miss and the data need to be fetched with an additional delay from an external memory. It is therefore highly desirable to predict future memory accesses during execution in order to appropriately prefetch data without incurring delays. In this paper, we evaluate the potential of several artificial neural networks for the prediction of instruction memory addresses.

This paper is organized as follows. In the next section, we present related work performed in prediction based on artificial neural networks, recurrent neural networks and work on optimizing prediction results. In section III, we briefly introduce recurrent neural networks applied on predicting time series. The section IV shows prediction results while using a single recurrent neural network. In section V, we propose a hybrid prediction scheme. Finally, section VI we conclude.

S. Chtourou. is with the National Engineering School of Sfax, Sfax, 3038 Tunisia. He is now with Ecole Nationale supérieure de techniques avancées. (corresponding author to provide phone: 33-(0)1-45525425; fax: 33-(0)145528327; e-mail: chtourou@ ensta.fr).

M. Chtourou. is with the National Engineering School of Sfax, Sfax, 3038 Tunisia (corresponding author to provide phone: 216.74.274.088 ; fax: 216.74.275.595; e-mail: mohamed.chtourou@enis.rnu.tn).

O. Hammami. is with the Ecole Nationale Supérieure de Techniques Avancées, Paris, 75739 France (corresponding author to provide phone: 33-(0)1-45525424; fax: 33-(0)145528327; e-mail: hammami@ ensta.fr).

II. Related Work

Data prefetching is an important research topic in traditional computer architecture studies [1-9][12] especially for multimedia workload. However, most proposed techniques follow very simple schemes with very limited adaptivity. Data prefetching remains an open issue in the general case. Our work raises several issues: (1) which neural network based technique is the most appropriate for large (millions of elements) time series? What is the maximum number of elements, which can be predicted at any time step? What should be the history used to predict the future memory accesses?

To the best of our knowledge no paper has ever addresses the issue of neural network based prediction on large time series from memory addresses.

In our work, we focus on optimization of neural network performance to predict the next instruction addresses. In order to improve prediction results, many works propose novel architectures for neural network based prediction. Parlos and al [11] propose a novel recurrent architecture based on multilayer percepton model with a modified learning algorithm. Owens and al [28] present a comparative study between many neural architectures based prediction notably the feedforward NAR (Nonlinear AutoRegressive) model and the fully recurrent architecture. The optimization of the neural network architecture is a limited solution to learn large and multi-variant data sets. In [13], the authors introduce a novel algorithm that trains a neural network to identify chaotic dynamics from a single measured time series. In this work, tested time series present chaotic dynamics but the length of the different time series is

limited. The work presented in [14] defines a flexible neural network for forecasting time series application. The paper introduces a flexible feature to a single neural network to deal with the complexity of the tested time series. In [16], the paper deals with real world application. It presents a locally recurrent multiplayer network to cope the complexity of the time series and to improve performance of the neural model.

III. Multi-Step Ahead Prediction based on RNN

Time series forecasting is one of the most important and interesting problems in several computing applications. Its importance comes from the fact that it has wide ranging applications including intelligent control systems. The prediction results are based on knowledge of some aspect of previous system behavior. The complexity of time series determines the prediction model to implement. The traditional forecasting method is the statistical models. The powerful method for time series prediction is the ANN. In this paper we implement RNN to predict instruction addresses of a running application. Recurrent networks represent a dynamic system composed by many states that evolve according to a number of nonlinear equations. In recent years, a large variety of recurrent neural networks applied to solve a single step-ahead prediction problem have been explored. We notice the (SRN) Simple Recurrent Network and (SRNSC) Simple Recurrent Network with Shortcut Connections [15]. The recurrent inputs of the SRN architecture are connected to the hidden neurons. The SRNSH architecture is a SRN model with connections between input and output neurons. Other recurrent neural networks architectures are presented in [11] for the multi-step ahead prediction, which are the Nonlinear Autoregressive with exogenous input (NARX) model [10,18] and a dynamic recurrent network. The connections in dynamic recurrent networks are composed by feed forward links, recurrent links (connection between neurons of the hidden layer) and cross-talk links (connection between neurons of the output and the hidden layer). The Fig. 1 shows the architecture of the NARX model.

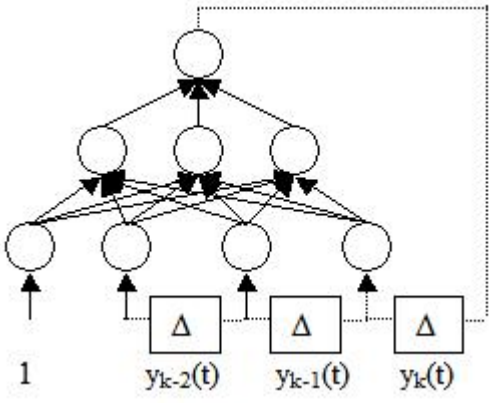

Fig. 1 Recurrent neural network model (NARX)

The formulation of the single step ahead prediction (SS prediction) performed on the y time series is:

$$\hat{y}(k+1)=p(y(k),y(k-1),....,y(k-N)) \quad (1)$$

N is the length of the input vector of the predictor p, y(k) is the k^{th} symbol of the y time series. $\hat{y}(k+1)$ is the $(K+1)^{th}$ time series estimated value.

The equation (1) becomes in the multi (n) step ahead prediction (MS prediction), $\hat{y}(k+n)$ is the $(K+n)^{th}$ time series estimated value:

$$\hat{y}(k+n)=p(\hat{y}(k+n-1),\hat{y}(k+n-2),....,y(k+n-N)) \quad (2)$$

In this work, we focus on NARX model (Fig.1) to predict a huge data set with high variance level.

Having selected the model structure to be used in MS prediction, we can now proceed to formulate the learning algorithm [10]. In this formulation, we allow N step ahead prediction. For the RNN represented in Fig. 1, the following objective function is optimized:

$$J=\frac{1}{2}\sum_{i=k+1}^{k+N}[y^e(k+i)-y^d(k+i)]^2=\frac{1}{2}\sum_{i=k}^{k+N}Ai \quad (3)$$

With $y^e(k+i)$ is the estimated output by the recurrent neural network, $y^d(k+i)$ is the desired one.

The network weights θ and biases are updated using the following gradient descent rules:

$$\frac{\partial J}{\partial \theta}=\sum_i \frac{\partial Ai}{\partial y^e(k+i)}\cdot\frac{\partial y^e(k+i)}{\partial \theta} \quad (4)$$

$$\frac{\partial J}{\partial \theta}=[y^e(k+i)-y^d(k+i)]\cdot \left[\sum_{j=1}^{n}\frac{\partial y^e(k+i)}{\partial y^e(k+i-j)}\cdot\frac{\partial y^e(k+i-j)}{\partial \theta}+\frac{\partial y^e(k+i)}{\partial \theta}\right] \quad (5)$$

With n is the number of recurrent input in the first layer, θ is the set of weights of the recurrent neural network. The weights correction of the recurrent neural network is made at the end of each prediction window with:

$$\theta_{new}=\theta_{old}-\varepsilon\cdot\sum_{i=1}^{N}\frac{\partial A_i}{\partial \theta} \quad (6)$$

IV. Instruction Memory Adresses Prediction based on Single RNN

Experimental results presented in this part of the paper are performed on instruction addresses data detected by the Pin tool [20]. This tool is designed for the instrumentation of programs running on the Linux environment. It provides a rich API that abstracts away the underlying instruction set. Pin tool is a (JIT) Just in Time compiler. The input to this compiler is a regular executable application. While using the Pin tool, we can perform Instruction instrumentation (instruction count, instruction address trace, memory reference trace). The Pin tool intercepts the execution of the first instruction of the executable and generates new code. The generated code execution is identical to the original one, but it ensures the control over branch operation. When a branch exits the sequence, Pin generates more code to describe the branch effect and continues execution. The Fig. 2 presented bellow shows the trace of the 10000 first integer

values of the instruction addresses for three different Linux applications (LS, CP, GZIP) running on Pentium IV processor. The instruction addresses allocated by different running application illustrate the allocation dynamics based on different distant locality.

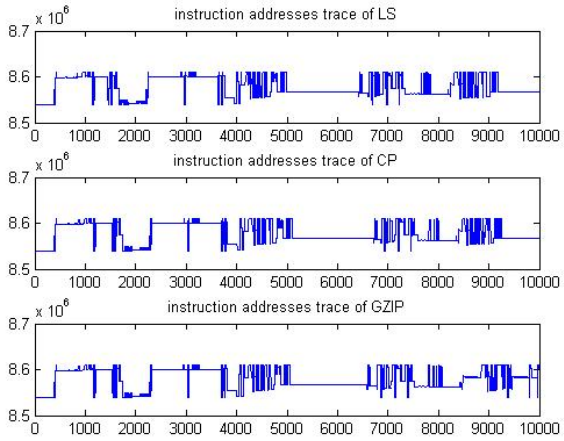

Fig. 2 Instruction memory addresses traces

The training data set has a large data span since memory addresses after the conversion into decimal values, goes from 2140644 to 134565638 for the LS application, from 8539104 to 134547942 for the CP application and from 8539104 to 134555070 for the GZIP application. The different instruction addresses represents large data set. Its components are dispersed in a large addressing sector.

The appropriate parameters for the RNN, which include the number of hidden neurons, the number of entry neurons, the training algorithm, the prediction window and the transfer function:

- Number of entry neurons: 10
- Number of Hidden neurons: 10
- Transfer functions: the sigmoid function
- Prediction window: 1
- Training algorithm: error back propagation learning rule

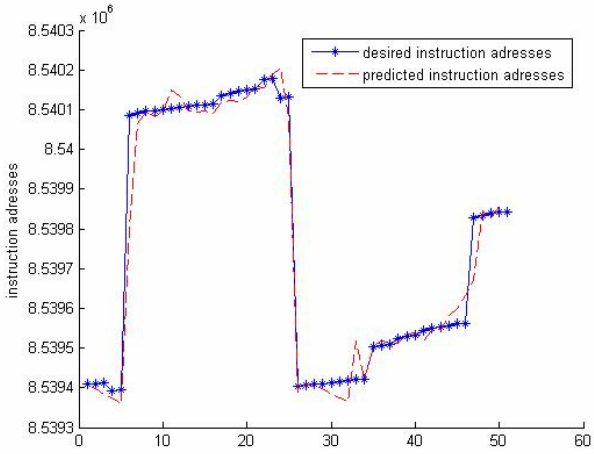

Fig. 3 Instruction addresses prediction on partial learning data

The Fig. 3 shows that when we focus on a small part (300 components) of the LS instructions addresses trace, the one step-ahead prediction results are similar to the desired time series components.

The partial learning data set components are located at the same addressing sector between 8539294 and 8540178. This zone is defined as the learning sector. If we present a time series in the same learning sector to test the RNN performance, the predicted time series behavior is similar to the desired one. If the tested time series contains many components that reside out of the learning sector, the RNN shows wrong prediction results. The Fig. 4 shows the prediction results given by the learned RNN while presenting a new test time series components extracted from a new instruction addresses locality.

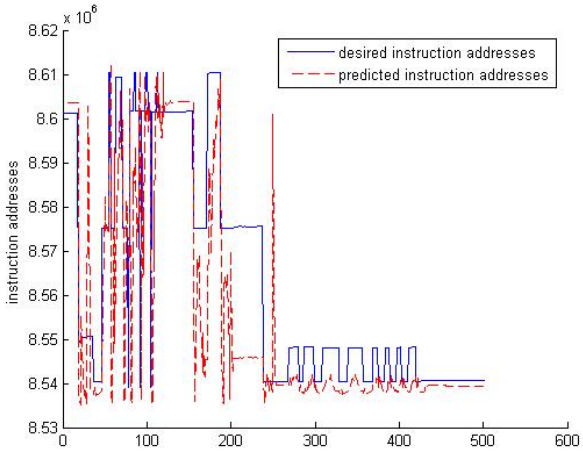

Fig. 4 Prediction results of the testing data set

Embedded applications use many kind of instruction addresses prediction system to detect branch operation in a running application. If a branch operation is detected, the processor loads the new addressing page into the cache memory. So, to perform the embedded applications performance, we look to predict the next instruction addresses page. In the next paragraph we focus on predicting the next instruction addresses page. We fix the page width at 4KB. The partial learning data set contains 1000 different pages of memory. If we adopt the new instruction addresses time series, we eliminate small addresses variations in the same locality. The Fig. 5 shows the prediction results of 400 components of the learning data set.

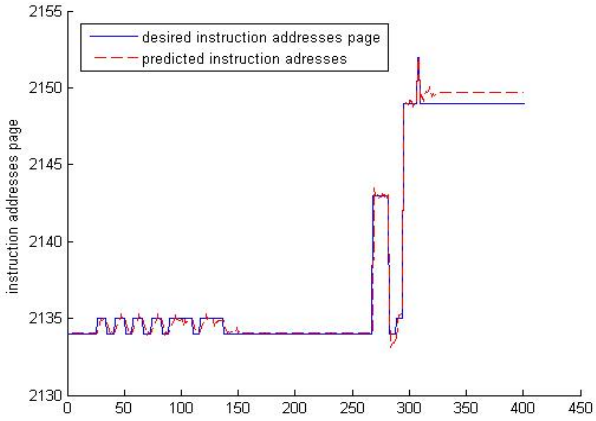

Fig. 5 Learning results of the instruction addresses pages data set

The generalization experiment made on a new test time series shows that the predicted instruction addresses pages

respect the global behavior of the addressing operation. The RNN is equal to the number of input neurons of the 1D SOM. The number of local predictors is defined by the width of the SOM output layer. In our scheme, we use an incremental approach to fix the number of sub-classes in the learning data set.

Fig. 6 shows the prediction results on a new time series composed by 600 instruction addresses pages. The artificial neural networks based prediction on instruction addresses pages perform better then the prediction of the instruction addresses while using the same neural network.

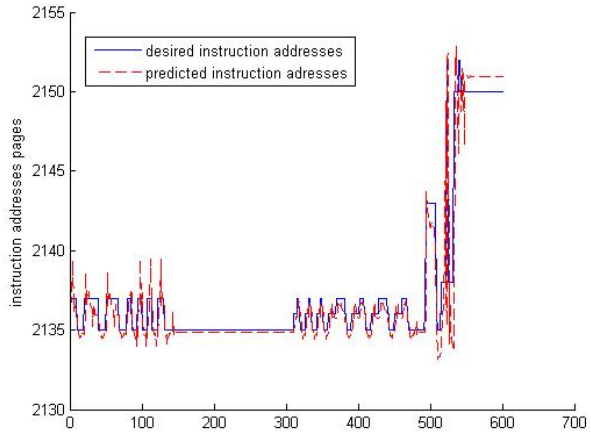

Fig. 6 Prediction results of the instruction addresses pages

We observe that the performance of the prediction based on RNN is affected by the variance of the learning components. In the next paragraph we propose a hybrid neural network to predict multiple locality time series.

V. Hybrid Neural Network for the Instruction Memory Adresses Prediction

Locality of references [1] of memory accesses is the foundation of cache memories and states that data is accessed at any time in a limited address space range even if during the whole program execution a very large memory address space have been accessed. This important property suggests the use of per-locality local predictor [19]. The hybrid neural network proposed in this paper is composed of two stages: (1) a classification stage and (2) a prediction stage. The main idea of this scheme is to define multiple local predictors to forecast a large and multi-variant data set. The proposed prediction scheme contains a single 1D SOM [17] network and several recurrent neural networks. Every output in the SOM is connected with a single recurrent neural network (Fig. 7).

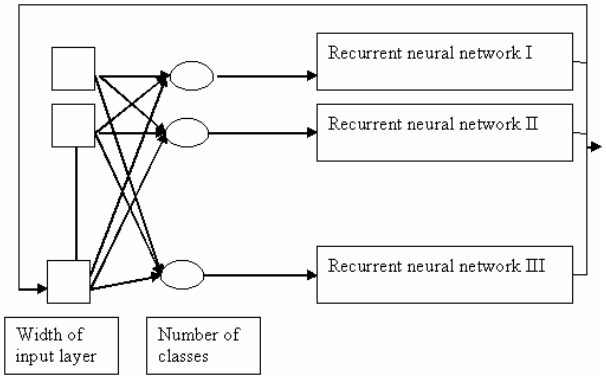

Fig. 7 Hybrid neural network for large time series prediction

The number of hidden neurons in each recurrent neural network is fixed at 10. The number of input neurons of each

The Table I describes comparison results of various learning and generalization approaches while using the proposed scheme to predict 100000 instruction addresses. The Mean Square Error (MSE) of the normalized data sets measures the performance of the prediction of this hybrid neural network.

TABLE I
Learning and Generalization Performance

Sub-classes	Learning MSE (LS)	Generalization MSE (CP)	Generalization MSE (GZIP)
1	2.8662	10.7405	24.1532
3	2.8690	10.7196	21.4572
6	2.8691	9.2355	24.7295
10	3.1025	50.6963	15.9570

The learning performance decreases while we add local predictors. The Fig. 8 shows partial learning results (2000 components) of 100000 vectors. The learning results show that the predictors perform better on components at the same addresses page. As one can easily expect, the predictors are inefficient during transitions.

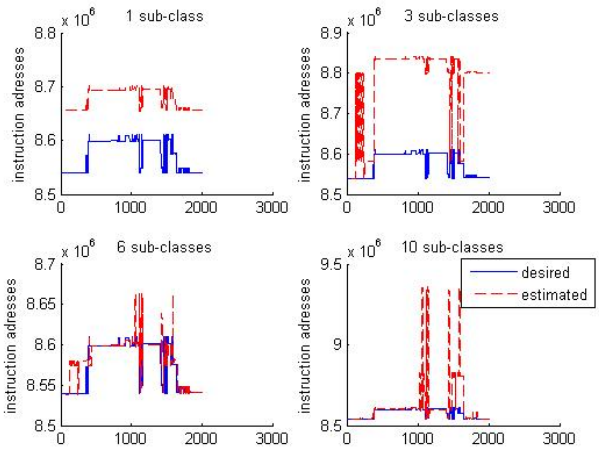

Fig. 8 Hybrid neural network prediction results

VI. Conclusion and Future Work

Complex system on chip executes large embedded applications with real time requirements. Increasingly applications exhibit variability in their behavior as demonstrated by their memory accesses. Therefore, correctly predicting memory accesses could help improve real time constraints. A hybrid artificial neural network for training large and chaotic time series and predicting instruction addresses of a running C application is suggested in this paper. Experimentations reveal that the presented neural model improves address prediction performance on the same address page but the performance decreases in high dynamics zone. Many parameters will be discussed in future works such as the prediction windows in multi-step ahead prediction, the optimal number of local predictors sub-classes to improve prediction performance. To the best of our knowledge few papers have addressed the challenging

issue of large time series (several millions) forecasting and much work remain to be done.

We also plan to extend the current work to program phases [21-27] in which elements to be predicted are computation phases representing a larger granularity of computation.

REFERENCES

[1] J.Hennesy and D.Patterson, "Computer Architecture A Quantitative Approach", Third Edition, Morgan Kauffman Pub. 2002.

[2] Vander Wiel, S.P. Lilja, D.J. "When caches aren't enough: data prefetching techniques", Computer Volume 30, Issue 7, July 1997 Page(s): 23-30.

[3] R. Piccardi, M. Prati, A. "Neighbor cache prefetching for multimedia image and video processing", Cucchiara, Multimedia, IEEE Transactions on Volume 6, Issue 4, Aug. 2004 Page(s): 539 – 552.

[4] Jung-Hoon Lee; Seh-woong Jeong; Shin-Dug Kim; Weems, "An intelligent cache system with hardware prefetching for high performance", Computers, IEEE Transactions on Volume 52, Issue 5, May 2003 Page(s): 607 – 616.

[5] Jiwei Lu; Chen, H.; Rao Fu; Wei-Chung Hsu; Othmer, B.; Pen-Chung Yew; Dong-Yuan Chen; "The performance of runtime data cache prefetching in a dynamic optimization system", Microarchitecture, 2003. MICRO-36. Proceedings 36th Annual IEEE/ACM International Symposium on 2003 Page(s): 180 – 190.

[6] Bin Wu; Kshemkalyani, A.D.; "Objective-Optimal Algorithms for Long-Term Web Prefetching", Computers, IEEE Transactions on Volume 55, Issue 1, Jan. 2006 Page(s): 2 – 17.

[7] Junghee Lee; Chanik Park; Soonhoi Ha; "Memory access pattern analysis and stream cache design for multimedia applications", Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific 21-24 Jan. 2003 Page(s): 22 – 27.

[8] Oliver, R.L; Teller, P.J.; "Dynamic and adaptive cache prefetch policies", Performance, Computing, and Communications Conference, 2000. IPCCC '00. Conference Proceeding of the IEEE International 20-22 Feb. 2000 Page(s): 509 – 515.

[9] D.Joseph and D.Grunwald, "Prefetching Using Markov Predictors", IEEE transactions on computers, vol.48, no.2, February 1999.

[10] M. Basso, L. Giarré, S. Groppi, and G. Zappa," NARX Models of an Industrial Power Plant Gas Turbine", IEEE transactions on control systems technology, vol. 13, no. 4, july 2005.

[11] A. G. Parlos, O.T. Rais, A.F. Atiya, "Multi-step-ahead prediction using dynamic recurrent neural networks", Neural Networks 13 (2000) 765,786.

[12] D. A. Jiménez, "Fast Path-Based Neural Branch Prediction", Proceedings of the 36th internantional symposium on microarchitecture, 2003.

[13] R. Bakker, J. C. Schouten, C.Van den Bleek, C. L. Giles, "Neural Learning of Chaotic Dynamics: The errror Propagation Algorithm", IEEE world conference on computational intelligence, p.2483, 1998.

[14] Y. Chen, B. Yang, J. Dong, A. Abraham, „Time series forecasting using flexible neural tree model ", Information sciences 174, 219-235, 2005.

[15] Stephan K. Chalup, Alan D. Blair, "Incremental training of first order recurrent neural networks to predict a context-sensitive language", Neural Networks 16 (2003) 955–972.

[16] T.G. Barbounis, J.B. Teocharis, "Locally recurrent neural networks for long-term speed and power prediction", Neurocomputing 69 2006, 466-496.

[17] Teuvo Kohonen: Self-Organizing Maps and Learning Vector Quantization for Feature Sequences. Neural Processing Letters: 151-159 (1999).

[18] T. Lin, C. L. Giles, B. Horne, S.Y. Kung, "A Delay Damage Model Selection Algorithm for NARX Neural Networks", IEEE transaction on signal processing, vol. 45, No. 11, November 1997, P 2719-2730.

[19] Jorg D. Wichard and Maciej Ogorzalek, "Time Series Prediction with Ensemble Models", IJCNN'04, Busdapest 2004.

[20] http://rogue.colorado.edu/Pin/index.html.

[21] Lau, J.; Schoenmackers S.; Calder, B.; "Transition phase classification and prediction", High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on 12-16 Feb. 2005 Page(s): 278 – 289.

[22] Sherwood, T.; Perelman, E.; Hamerly, G.; Sair, S.; Calder, B.; "Discovering and exploiting program phases", Micro, IEEE Volume 23, Issue 6, Nov.-Dec. 2003 Page(s): 84 – 93.

[23] Lau, J.; Sampson, J.; Perelman, E.; Hamerly, G.; Calder, B.; "The Strong correlation Between Code Signatures and Performance", Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on March 20-22, 2005 Page(s): 236 – 247.

[24] Lau, J.; Perelman, E.; Hamerly, G.; Sherwood, T.; Calder, B.; "Motivation for Variable Length Intervals and Hierarchical Phase Behavior", Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on March 20-22, 2005 Page(s): 135 – 146.

[25] Lau, J.; Schoemackers, S.; Calder, B.; "Structures for phase classification", Performance Analysis of Systems and Software, 2004 IEEE International Symposium on – ISPASS 2004 Page(s): 57 – 67.

[26] Sherwood, T.; Sair, S.; Calder, B.; "Phase tracking and prediction", Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on 9-11 June 2003 Page(s): 336 – 347.

[27] Calder, B.; Grunwald, D.; "Next cache line and set prediction", Computer Architecture, 1995. Proceedings. 22nd Annual International Symposium on 22-24 Jun 1995 Page(s): 287 – 296.

[28] Owens, A.J.; "Empirical Modeling of Very Large Data Sets Using Neural Network", Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Volume 6, 24-27 July 2000 Page(s): 302 - 307.