Reliability Enhancement of Multi-Core Processors using Machine-Learning Techniques

Monir Zaman*, Ali Ahmadi* and Yiorgos Makris*

*Department of Electrical Engineering, The University of Texas at Dallas, Richardson, TX 75080, USA

***Abstract*—As a result of technology scaling, power density of multi-core chips increases and leads to temperature hot-spots which accelerate device aging and chip failure. Moreover tremendous efforts to reduce power consumption by employing low-power techniques decreases the reliability of new design generation. In this work, we first discuss the state-of-the-art methods for predicting workload dynamics and we compare their performance. We, then, introduce a prediction method based on Support Vector Regression (SVR), which accurately predicts the workload behavior several steps ahead. To evaluate the effectiveness of our approach, we use UltraSPARC T1 processor along with Sun Solaris operating system. We incorporate OS and architectural level sensors and knobs and our preliminary results show our predictive method achieve higher accuracy.**

I. Introduction

Due to technology scaling, power density of multi-core chips increases which leads to temperature hot-spots causing device to age faster. On the other hand, aggressive design techniques which tend to lower power consumption result in vulnerable devices and decrease reliability of new design generations. Researchers have introduced proactive thermal management approaches to prevent the thermal problem before is arises. In a proactive system, various forecasting methods are used to predict the dynamics of the workload through operating system-level or architectural-level information. The main engine of such proactive methods is prediction, which needs to provide accurate forecasting of parameters of interest multiple steps ahead into the future.

In this work, we use Support Vector Machines (SVM) to forecast workload characteristics based on past and current measurements. We experiment with a multi-core UltraSPARC T1 processor, running workload from the PARSEC benchmark suite and collecting available performance counter values. Our results show that the SVM-based technique outperforms other prediction models which have been widely used.

The remainder of this paper is organized as follows: we start by reviewing related work in section II. Then we discuss existing predictors in section III followed by elaborate discussion of our proposed prediction technique in section IV. In section V, we show experimental results demonstrating the effectiveness of the proposed method and conclusions are drawn in section VI.

II. Previous Work

In this section, we discuss prior work in proactive multi-core power and thermal management. [1] shows in multi-core systems, with reasonable performance degradation, temperature-aware job scheduling is effective for thermal management. Similarly, by profiling a set of different frequencies for various temperatures during an off-line phase, Murali, et al. [2] proposed a technique to assign different operating frequencies to various cores in order to meet thermal constraints. In [3], the authors proposed a proactive thermal-aware scheduling technique. There they reduced the impact of performance overhead due to temperature variation. In [4], the authors proposed predictive scheduling and power management techniques by profiling the workload, predicting its future characteristics using the ARMA predictor and dynamically scheduling new workload.

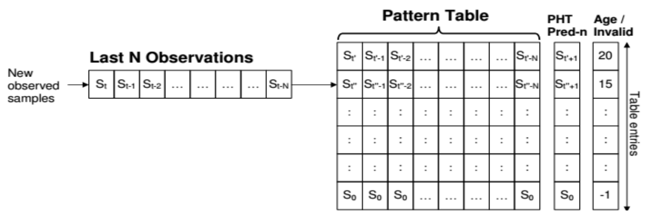


Fig. 1: History Table overview

III. Prediction Techniques

In this section, we briefly go over current state-of-the-art prediction techniques which have been used in the literature for predicting workload characteristics. Our main focus is on techniques that can be used in proactive systems for power or thermal management.

A. History Table Predictor

This simple table based predictor technique keeps track of the patterns to generate next sample result. This approach relies on repetitive application execution characteristics to produce a model for future behavior. An example of this predictor, called global phase history table [5], is depicted in Figure 1. It consists of a global register that tracks the last $\mathbf{n}$ observed samples, where $\mathbf{n}$ is the depth of table. The content of this register is used to index the pattern history table (HT). HT holds a certain number of previously encountered patterns, with their corresponding next sample prediction based on former experience. In case the content of the global register does not hit the HT, the last observed sample is predicted as the next sample (in this case HT predictor behaves like a last value predictor).

B. Autoregressive-Moving Average Predictor

We use Autoregressive-Moving-average (ARMA) mathematical model for identifying autocorrelation in time series data. This is widely used for time series prediction in various fields. One subset of ARMA models are the so-called autoregressive (AR) models which expresses a time series as a linear function of its past values. The order of the AR model reflects how many lagged past values are included. P-order autoregressive or AR(p) is shown in Equation-1:

$$\mathbf{u_t} + \sum_{i=1}^{\mathbf{p}} \alpha_\mathbf{i}.\mathbf{u_{t-i}} = \mathbf{e_t} \tag{1}$$

where $\mathbf{u_t}$ is the value of series at time $\mathbf{t}$, $\alpha_\mathbf{i}$ is the lag $\mathbf{i}$ autoregressive coefficient and $\mathbf{e_t}$ is called residual or noise. The residuals are assumed to be random (not autocorrelated) and normally distributed. By incorporating a linear regression of previous noise terms in the AR model we can build an ARMA model which is expressed in the following form:

$$\mathbf{u_t} + \sum_{i=1}^{\mathbf{p}} \alpha_\mathbf{i}.\mathbf{u_{t-i}} = e_t + \sum_{i=1}^{q} c_i.e_{t-i} \tag{2}$$

where the terms $\mathbf{c_i}$ are the moving average coefficients, and $\mathbf{p}$ and $\mathbf{q}$ represent the orders of the AR and the moving average (MA) parts of the model, respectively. The assumption of the ARMA model is that the modeled time series is stationary and has serial correlation in the data. In stationary process, change of probability distribution over time is negligible and the mean and variance are stable. The performance of the model is reduced significantly when the stationarity assumption is violated. In [6], the authors used adaptive ARMA model to predict some characteristics of workload and temperature for their proactive system.

IV. Proposed Prediction Approach

In this work, we use Support Vector Machines (SVM) as a regression technique to predict workload characteristics.

A. *Support Vector Machine*

SVM has two major applications, classification and regression. The SVM used as a regression predictor is called Support Vector Regression (SVR). One of the main characteristics of SVR is that instead of minimizing the training error, it attempts to minimize the generalized error bound so as to achieve good generalization performance.

In the time series forecasting problem (workload characteristic prediction is a form of time series forecasting), the objective is to process the previous $\mathbf{n}$ observations and predict the time series $\mathbf{k}$ step ahead into the future. Suppose we are given a time series $[\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_{t-1}}]$ and we want to predict the time series at time $\mathbf{t}$. To employ SVR for this model, we build our training set $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \dots, n\}$ in following format:

$$\mathbf{D} = \begin{cases} (\mathbf{x_1}, \mathbf{y_1}) = ([\mathbf{u_1}, \mathbf{u_2}, \dots \mathbf{u_n}], \mathbf{u_{n+k}}) \\ (\mathbf{x_2}, \mathbf{y_2}) = ([\mathbf{u_2}, \mathbf{u_3}, \dots \mathbf{u_{n+1}}], \mathbf{u_{n+1+k}}) \\ \dots \\ (\mathbf{x_m}, \mathbf{y_m}) = ([\mathbf{u_m}, \mathbf{u_{m+1}}, \dots \mathbf{u_{m-1+n}}], \mathbf{u_{m+n+k}}) \\ (\mathbf{m} + \mathbf{n} + \mathbf{k}) \leq (\mathbf{t} - \mathbf{1}), \end{cases} \tag{3}$$

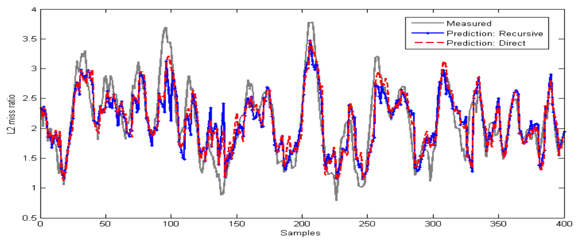


Fig. 2: SVR model comparison.

We can predict the time series $\mathbf{k}$ step ahead into the future at any time point after the SVR model is trained using the training set $\mathbf{D}$. The prediction horizon, $\mathbf{k}$, depends on the application and purpose of forecasting and typically is multi-step ahead. For example, in [6] the authors used 5 steps ahead prediction. For multiple steps ahead prediction, there are generally two different modeling approaches, **Recursive and Direct SVR**.

V. Results

A. *Experimental Setup*

We ran the experiments on the UltraSPARC T1 processor [7] running Solaris 10 operating system (OS). This processor has 8 cores, a unified L2 cache and a shared floating-point unit. Each core has a private data and instruction cache and a set of performance counters (Table I). We have used **cpustat** and **mpstat** [8] to access the information.

Precompiled scalable multi-threaded **ferret, facesim and blackscholes** from the PARSEC 2.1 benchmark [9] was used as our workload. We collected all performance counters at intervals of 90 ms during workload execution for all eight cores of the T1 processor.

IC_miss	Number of Instruction cache misses
DC_miss	Number of Data Cache misses
ITLB_miss	Number of Instruction TLB misses
DTLB_miss	Number of data TLB misses
L2_imiss	Number of instruction misses in L2
L2_dmiss_ld	Number of Load data misses in L2
FP_instr_cnt	Number of Floating point instructions
SB_full	Number of cycles spent due to SB full
Instr_cnt	Total number of instructions completed

TABLE I: Events tracked by performance counters

B. *Performance Counter Prediction*

First, we examine the performance of recursive and direct SVR models for workload characteristic prediction. Figure 2 presents the prediction of the L2-miss-ratio counter for both the recursive and the direct approach.We assume the collected data of each performance counter as an individual time series. Here, we demonstrate the prediction of L2 miss ratio, which provides an insight for power consumption. Both models are trained using 5000 samples and predict 5 steps into the future ($\mathbf{k} = \mathbf{5}$).

In terms of prediction accuracy, we compare SVR with the predictors that we described in section III. A simple and well-known method is averaging of last $\mathbf{n}$ values. We use simple

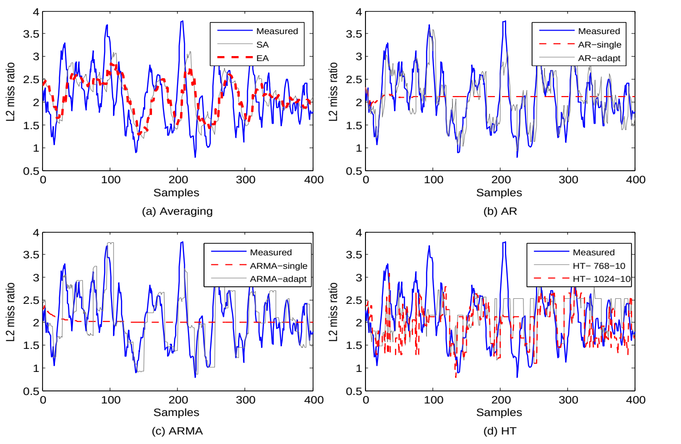


Fig. 3: Forecasting L2 miss ratio, five steps ahead using existing predictors

averaging (SA) and exponential averaging (EA) to forecast L2-miss-ratio five steps ahead into the future. Figure 3(a) shows the measured and predicted samples. Sharp variation of the performance counter results in poor predictions and produces significant forecasting error although the overhead of evaluating SA and EA techniques at runtime is almost negligible.

Next we fit an AR and an ARMA model to the training data of the performance counter and the trained models are used for prediction. Figure 3(b) illustrates the prediction performance of AR(21) for L2-miss-ratio when the model is used to forecast five steps ahead into the future. The blue line is the measured time values for L2-miss-ratio, while the red dotted curve shows the prediction when only a single AR(21) model is trained and used for prediction. It is seen here that the model performs well at the beginning but fails to keep up with the actual results prediction later on (from hereafter we call this model AR-single). To address this issue, The AR-single model needs to the retrained at the beginning of every prediction horizon and the model needs to be fitted with a data set which includes all measured samples up to this time point. We call this AR-adapt in the rest of the paper. The gray line in Figure 3(b) presents the prediction of the AR-adapt model which improves the quality of prediction significantly. Figure 3(c) shows the prediction performance of ARMA(4,2)-single and ARMA(4,2)-adapt. Like AR, the ARMA model also requires adaptation before forecasting. Although both AR-adapt and ARMA-adapt capture the dynamics of a target performance counter accurately, the overhead of adapting these models is significant. In [6], the authors reported that for the ARMA(p, q) model, up to the fifth order, it takes about 500 ms for computation and validation of the model. Therefore, these models are not appropriate candidates at runtime for a real-time workload monitoring and forecasting.

In section III, we described the history table (HT) model for workload characteristics prediction. Here we built 2 history predictors, HT-767-10 (table of size 768 with depth 10) and HT-1024-10. Figure 3(d) demonstrates the quality of prediction using HT.

Figure 4 presents the distribution of prediction error for forecasting L2-miss-ratio five steps ahead into the future. It can be seen that the proposed SVR approach not only generates a lower prediction error in the test data, but also results in tighter error bars than the previous models, indicating smaller variance of error. The mean prediction error is 10%, 11% and 13% for SVR-Direct, SVR-Recursive and AR-adapt models respectively. Note, that the SVR model is trained earlier using the history data and are used for prediction without adaption.

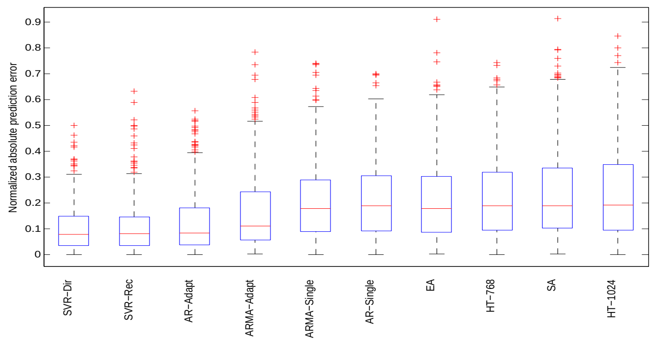


Fig. 4: Distribution of prediction error for forecasting five steps into the future over 400 samples.

Figure 5 shows the accuracy of our prediction algorithm. SVR-Direct model was used to predict all the available HW counters for all the executed workloads. It is worth mentioning, that the model was only trained by the Facesim workload and used to predict the counter for all the workloads.

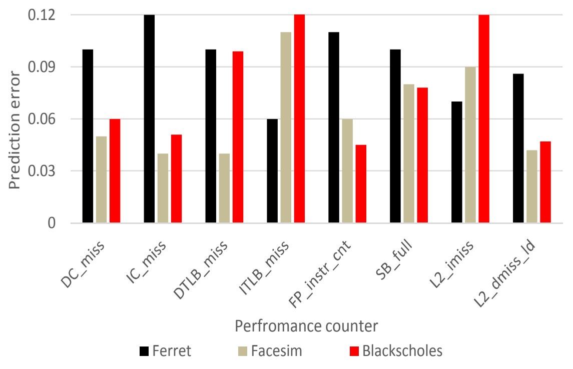


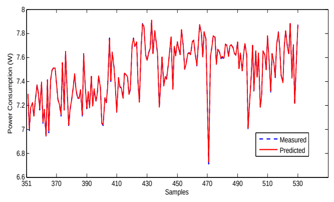
Fig. 5: Performance counter Prediction error for 3 workloads

C. Power and Temperature Prediction

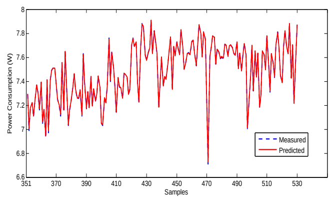
Our ultimate goal is to improve the reliability of circuits through intelligent workload scheduling and other circuit techniques. With that in mind, we need to predict the power and temperature (which reflects device aging) of the system and use that information to the intelligent pro-active system. Ideally, power and temperature need to be obtained directly from the circuit. However, due to lack of sensors or limitations in accessing them, thermal management systems in the literature typically acquire these measurements using simulators. In this work, we utilize our SVR model to predict power using performance counter values. We use 350 samples of averaged performance counter values along with their corresponding power to train a SVR model. Then the model is used to predict power from the value of the performance counters.

McPAT v1.0 [10] was used to compute power consumption of each core from its architectural trace. McPAT is modified to generate proper power traces for the thermal simulator HotSpot [11]. The input data for McPAT that were not generated by the performance counters, were assumed to take the default value. Performance counter data were averaged over a window of two seconds and fed to McPAT to obtain the power trace. Figure 6 shows the power consumption prediction for Core-1 for all the 3 workloads run.

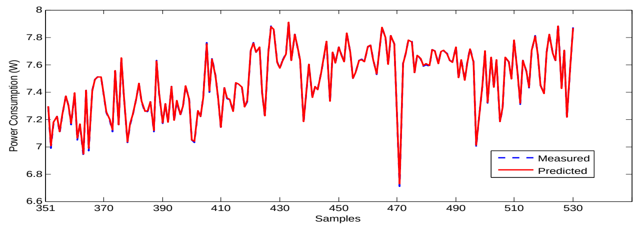
Afterwards, we feed the power trace to HotSpot to obtain the temperature trace. We modified the floor plan and thermal characteristics to reflect the UltraSPARC T1 processor. The temperature trace presents the effect of the reactive system of Solaris 10, wherein thread migration is employed in an attempt to cool down a core whose temperature has reached the threshold. Figure 7 shows the effect of temperature as simulated by Hotspot for the SPARC-T1 processor.



(a) ferret power prediction



(b) facesim power prediction



(c) blackscholes power prediction

Fig. 6: Power consumption prediction of core-1 during 3 workload execution

VI. Conclusion

In this paper, we have introduce the Support Vector Machines (SVM) modeling technique for accurate workload characteristics predictions. We also compare the SVM technique with different other widely used prediction engines and successfully shown that our method outperforms the existing ones by a decent margin. All of our experiments were run on the 8 core UltraSPARC T1 processor. The biggest advantage of the SVR method over other techniques is, this model does not need any retraining unlike AR-adapt or ARMA, which significantly reduces the training overhead in real-time scenario. We have also showed that this prediction engine can accurately predict

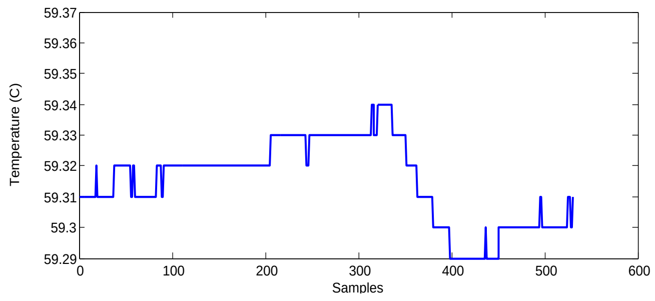


Fig. 7: Temperature of core-1 during workload execution

the power consumption as simulated by the McPAT power simulator. In future, we would like to extend this work by predicting the thermal characteristics of each core and make dynamic scheduling decision to increase lifetime of the system.

References

[1] A. Coskun, T. Rosing, and K. Whisnant, "Temperature aware task scheduling in mpsocs," in *Proceedings of the conference on Design, automation and test in Europe*, 2007, pp. 1659–1664.

[2] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. De Micheli, "Temperature control of high-performance multi-core platforms using convex optimization," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2008, pp. 110–115.

[3] A. Kumar, L. Shang, L. Peh, and N. Jha, "Hybdtm: A coordinated hardware-software approach for dynamic thermal management," in *Proceedings of the 43rd Annual Design Automation Conference*, 2006, pp. 548–553.

[4] A. Coskun, R. Strong, D. Tullsen, and T. Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proceedings of the ACM SIGMETRICS Performance Evaluation Review*, 2009, vol. 37, pp. 169–180.

[5] C. Isci, G. Contreras, and M. Martonosi, "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006, pp. 359–370.

[6] A. Coskun, T. Rosing, and K. Gross, "Proactive temperature balancing for low cost thermal management in mpsocs," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 250–257.

[7] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded sparc processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, 2005.

[8] R. McDougall, J. Mauro, and B. Gregg, *Solaris (TM) Performance and Tools: DTrace and MDB Techniques for Solaris 10 and OpenSolaris (Solaris Series)*, Prentice Hall PTR, 2006.

[9] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, 2009.

[10] L. Sheng, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 469–480.

[11] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proceedings of the ACM SIGARCH Computer Architecture News*, 2003, vol. 31, pp. 2–13.