Workload Characterization and Prediction: A Pathway to Reliable Multi-core Systems

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, intense efforts to reduce power consumption by employing low-power techniques decrease the reliability of new design generations. Traditionally, reactive thermal/power management techniques have been used to take appropriate action when the temperature reaches a threshold. However, these approaches do not always balance temperature and, as a result, may degrade system reliability. Therefore, to distribute temperature evenly across all cores, a proactive mechanism is needed to forecast future workload characteristics and the corresponding temperature, in order to make decisions before hot spots occur. Such proactive methods rely on an engine to precisely predict future workload characteristics. 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 several programs from the PARSEC benchmark suite on an UltraSPARC T1 processor running the Sun Solaris operating system and we extract architectural traces. Then, the extracted traces are used to generate power and thermal profiles for each core using the McPAT and Hot-Spot simulators. Our results show that the proposed method forecasts workload dynamics and power very accurately and outperforms previous prediction techniques.**

I. Introduction

Technology scaling increases the power density of multi-core chips and leads to temperature hot-spots which age the device faster and cause chip failure. On the other hand, aggressive design techniques, which tend to lower power consumption, result in vulnerable devices and decrease reliability of new design generations. Dynamic power and thermal management have been introduced to address this issue. These techniques are usually activated upon reaching a threshold and maintain the power or temperature below a critical level, often at the cost of significant performance loss. Similarly, in the multi-processor domains, thread migration has been introduced to achieve safe core temperature. This approaches are reactive, taking action after a specific event happens, which may limit options and effectiveness. Researchers have, therefore, also introduced proactive thermal management approaches to prevent the thermal problem before is arises. In proactive techniques, 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 utilize 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 benchmarking suite and collecting available performance counter values, such as cache misses, floating-point instructions, etc. Our results demonstrate that the SVM-based technique outperforms other prediction methods which have been used so far for workload monitoring, and forecasts future workload dynamics much more accurately.

The remainder of this paper is organized as follows: we start by reviewing related work in section II. In section III, we discuss existing predictors with their advantages and disadvantages. Then, in section IV, we introduce the new regression method. Experimental results demonstrating the effectiveness of the proposed method are presented in section V and conclusions are drawn in section VI.

II. Previous Work

In this section, we discuss prior work in multi-core power and thermal management. In general, two approaches have been followed for power and thermal management, namely reactive and proactive. Reactive policies take action when a target parameter reaches a threshold, while proactive techniques predict future values of parameters of interest and make decisions in advance, in order to avoid critical situations.

The concept of a Dynamic Thermal Management (DTM) system based on thermal measurements was introduced by Brooks and Martonosi [1]. One such DTM is clock-gating, which is implemented in Pentium 4 [2]. Therein, the thermal management policy stalls/freezes the operation of certain cores until their temperature drops below the safe limit. In Simultaneous Multithreaded (SMT) systems, several techniques proposed by Donald et al. [3] use the hardware event counters and schedule carefully process-intensive threads, so that hot-spots can be avoided. Towards reducing hot-spots, thread migration techniques are also used, wherein existing threads are migrated to another cooler core [4].

Temperature-aware job scheduling [5] in multi-core systems with reasonable performance degradation has also been shown effective for thermal management. Similarly, by profiling a set of different frequencies for various temperatures during an off-line phase, Murali, et al. [6] proposed a technique to assign different operating frequencies to various cores in order to meet thermal constraints. In [7], the authors proposed a proactive thermal-aware scheduling technique. Thereby, they reduced the impact of performance overhead due to temperature

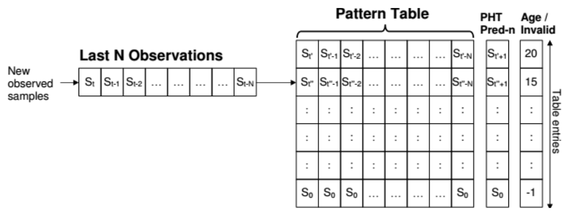


Fig. 1: History Table overview

variation. In [8], the authors proposed predictive scheduling and power management techniques by profiling the workload, predicting its future characteristics using the Autoregressive-Moving-average (ARMA) predictor and dynamically scheduling new workload. Therein, the authors compared various predictors and their results showed that ARMA outperforms other methods.

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. Last Value Predictor

A simple and effective way of predicting workload dynamics is the last value predictor. In this predictor, the next sample metric is assumed to be the same as the previously observed sample. This method can also be extended to make predictions based on a window of past history, where the next sample metric prediction can be expressed as $\mathbf{u_t} = \mathbf{f}(\mathbf{u_{t-1}}, \mathbf{u_{t-2}}, ..., \mathbf{u_{t-n}})$. The function $\mathbf{f}()$ can be a simple average (SA) function, which computes the mean value of the last $\mathbf{n}$ observations, an exponential averaging (EA) function, or a weighted averaging (WA) function, which assigns larger weights to recent observations. In general, this predictor is very popular due to its simplicity and its low cost. In slow varying applications, it performs reasonably accurately; however in case of rapidly varying applications its performance reduces dramatically.

B. History Table Predictor

Table-based predictors track previous patterns to deduce the next sample behavior. 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 [9], 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 the 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. The age entry in this structure is used to track the age of each pattern for a least recently used replacement policy when the table is full. 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 the HT predictor behaves as a last value predictor).

C. Autoregressive-Moving Average Predictor

Autoregressive-Moving-average (ARMA) is a mathematical model for identifying autocorrelation in time series. ARMA models are widely used in time series prediction in various fields. One subset of ARMA models are the so-called autoregressive (AR) models. An AR model 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. Equation 1 shows a p-order autoregressive, or AR(p), model:

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

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 ARMA model assumes the modeled time series is stationary and has serial correlation in the data. In a stationary process, change of probability distribution over time is negligible and the mean and variance are stable. Forecasting performance of this model significantly drops when the stationarity assumption is violated. In [10], the authors used the ARMA model to predict some characteristics of workload and temperature for their proactive system. They proposed to adapt the model when the ARMA model no longer fits the workload due to violation of the stationarity assumption.

ARMA modeling proceeds in three steps. The first step is identification, which consists of specifying the appropriate structure (AR, MA or ARMA) and order of the model. This can be done by looking at the autocorrelation function $((acf)$ or the partial $(acf$, or by using an automated iterative procedure to fit many models and use a goodness-of-fit statistic to select the best one. The second step is to estimate the coefficients of the model. Finally, the third check is checking the model to ensure that the residuals of the model are random and the estimated coefficients are statistically significant.

IV. Proposed Prediction Approach

As we mentioned earlier, the ARMA model is a linear regression of a time series based on its past values. Also it assumes that the workload characteristics are stationary over time, which does not necessarily hold true. In this work, we use Support Vector Machines (SVM) as a regression technique to predict workload characteristics, power, and temperature of cores, using previously observed values. First, we briefly describe the proposed regression model.

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 this section, we first give an overview of SVR, and then adapt it to the workload prediction problem at hand.

Suppose we are given a training set $\mathbf{D}$ of n data points, $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i)|i = 1, \dots, n\}$, where $\mathbf{x}$ denotes an input vector and $\mathbf{y}$ is the output. In SVR training, our goal is to find the optimal hyperplane $\mathbf{w}$, while the relationship function takes the following form:

$$\mathbf{f}(\mathbf{x_j}) = \mathbf{w^T}\phi(\mathbf{x_j}) + \mathbf{b} \tag{3}$$

where $\phi(\mathbf{x_j})$ represents the non-linear mapping of input features $\mathbf{x_j}$ into some high-dimensional space [11]. The SVR training is equivalent to solving the following optimization:

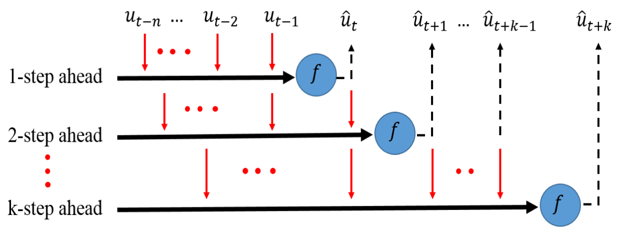
$$\begin{aligned} &\textbf{minimize} \ \frac{1}{2}\mathbf{w^T w} + C\sum_{j=1}^{r}(\xi_j + \xi_j^*), \\ &\textbf{subject to} \begin{cases} \mathbf{y_j} - \mathbf{f}(\mathbf{x_j}) \le \epsilon + \xi_\mathbf{j} \\ \mathbf{f}(\mathbf{x_j}) - \mathbf{y_j} \le \epsilon + \xi_\mathbf{j}^* \\ \xi_j, \xi_j^* \ge 0, \end{cases} \end{aligned} \tag{4}$$

where $\xi_\mathbf{j}$ and $\xi_\mathbf{j}^*$ are slack variables and ϵ is the parameter for the Insensitive Loss Function (ILF) with cost $\mathbf{C}$ [12].

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 $\mathbf{k}$ steps 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 $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i)|i = 1, \dots, m\}$ in the following format:

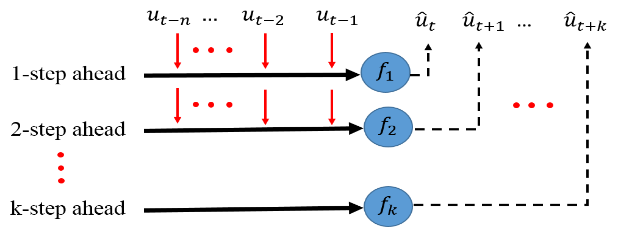
$$\mathbf{D} = \begin{cases} (\mathbf{x_1}, \mathbf{y_1}) = ([\mathbf{u_1}, \mathbf{u_2}, \dots \mathbf{u_n}], \mathbf{u_{n+k}}) \\ (\mathbf{x_2}, \mathbf{y_2}) = ([\mathbf{u_2}, \mathbf{u_3}, \dots \mathbf{u_{n+1}}], \mathbf{u_{n+1+k}}) \\ \dots \\ (\mathbf{x_m}, \mathbf{y_m}) = ([\mathbf{u_m}, \mathbf{u_{m+1}}, \dots \mathbf{u_{m-1+n}}], \mathbf{u_{m+n+k}}) \\ (\mathbf{m} + \mathbf{n} + \mathbf{k}) \le (\mathbf{t} - \mathbf{1}), \end{cases} \tag{5}$$

After the SVR model is trained using the training set $\mathbf{D}$, we can predict the time series $\mathbf{k}$ steps ahead into the future at any time point. The prediction horizon, $\mathbf{k}$, depends on the application and purpose of forecasting. As we mentioned earlier, our ultimate objective is to forecast workload behavior and to use this prediction to improve reliability and longevity of a system. Hence, it requires predictions of multiple steps ahead. For example, in [10] the authors used 5 steps ahead prediction. For multiple steps ahead prediction, there are generally two different modeling approaches, recursive and direct, which are briefly described next.



u_{t-n} ... u_{t-2} u_{t-1} $\hat{u}_t$ $\hat{u}_{t+1}$ ... $\hat{u}_{t+k-1}$ $\hat{u}_{t+k}$
1-step ahead
2-step ahead
k-step ahead
f

(a) Recursive approach



u_{t-n} ... u_{t-2} u_{t-1} $\hat{u}_t$ $\hat{u}_{t+1}$ ... $\hat{u}_{t+k}$
1-step ahead
2-step ahead
k-step ahead
f_1
f_2
f_k

(b) Direct approach

Fig. 2: SVR approaches for forecasting multiple steps ahead

B. Recursive Model

The recursive approach is a simple and straightforward modeling technique. In this method, a single one-step ahead model, i.e. $\mathbf{k} = \mathbf{1}$, is trained and repeatedly used for prediction until the desired horizon is reached. At each step, the output of the previous step is used as one of the inputs. The concept of recursive prediction is illustrated in Figure 2(a). As can be seen, this approach is simple to implement, yet the prediction error accumulates as the forecasting horizon increases.

C. Direct Model

Unlike the recursive approach, wherein a single model is trained and is used for all forecasting steps, in the direct approach an individual model is trained for each forecasting step and all the models have the same inputs. Figure 2(b) depicts this technique.

V. Results

A. Experimental Setup

The experimental results reported in this work are based on the UltraSParc T1 processor [13] runing a Solaris 10 operating system (OS). The T1 MPSoC has been manufactured in 90nm process with 8 cores, a unified L2 cache and a shared floating-point unit. Each core has a private data and instruction cache. This processor includes in its design a set of performance counters that allow tracking of a series of processor events. Table I lists the events which can be captured using performance counters in a T1 processor. T1 and Solaris 10 OS provide a set of tools for configuring and accessing these performance counters. The basic tools that we have used are **cpustat** and **mpstat** [14]. In terms of workload, we experimented with **Ferret**, **Facesim** and **Blackscholes** which are scalable multi-threaded applications from the PARSEC benchmark [15], and we collected all performance counters at intervals of 90 ms during workload execution for all eight cores of the T1 processor.

TABLE I: Events tracked by performance counters

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

B. *Performance Counter Prediction*

First, we examine the performance of recursive and direct SVR models for workload characteristic prediction. We assume the collected data of each performance counter as an individual time series, we use the first half of it as historical data to train our models and we forecast the future behavior of that counter for different forecasting horizons. Here, we demonstrate the prediction of L2 miss ratio, which provides an insight for power consumption. Figure 3(a) presents the prediction of the L2-miss-ratio counter for both the recursive and the direct approach. Both models are trained using 5000 samples and predict 5 steps into the future ($\mathbf{k = 5}$). As can be seen, the SVR models capture the dynamics of target performance counter almost exactly. While the performance of the two models is in the same range, the direct model requires more memory and computation time. Moreover, if we need model retraining for online adaption, the recursive approach has a single model while the direct method has multiple models which need to be retrained.

Now we compare SVR with the predictors that we described in section III, in terms of prediction accuracy and computation time. As mentioned before, a simple and well-known method is averaging. Averaging is a successful technique when the desired parameter changes slowly, however in case of rapid variation this predictor performs poorly. We use simple averaging (SA) and exponential averaging (EA) to forecast L2-miss-ratio five steps ahead into the future. Figure 4(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, because of its large prediction error, averaging is not a promising technique to forecast workload characteristics.

Next we fit an AR and an ARMA model to the training data of the performance counter and the trained models are used for

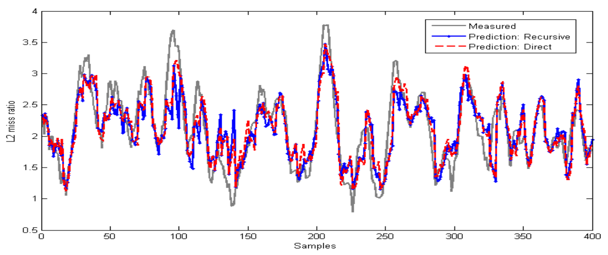


Fig. 3: Measured vs. five steps ahead forecast using SVR models

prediction. Figure 4(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. As can be seen, it performs reasonably at the very beginning, yet for the rest of the samples it predicts with huge discrepancy from the actual values (in the rest of the paper we call this model AR-single). In order to address this issue, the AR model needs to be trained at the beginning of forecasting, i.e. before predicting the next horizon, the model needs to be fitted with a data set which includes all measured samples up to this time point. In the rest of the paper, we call this model as AR-adapt, as it needs adaptation before forecasting. The gray line in Figure 4(b) presents the prediction of the AR-adapt model. The adaptation technique improves the quality of prediction significantly. Figure 4(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 [10], 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) as a method that has been used in the past for workload characteristics prediction. We built several history predictors (various table size and table depth) using the 5000 samples of training data to forecast five steps ahead into the future. The two tables with the lowest forecasting error are HT-767-10 (table of size 768 with depth 10) and HT-1024-10. Figure 4(d) demonstrates the quality of prediction using HT.

To evaluate the accuracy of each model, we compare the predicted value to the measured value of the performance counter at each time point and we capture the discrepancy using the normalized average absolute difference metric:

$$\delta = \frac{\mathbf{1}}{\mathbf{n_{TS}}} \sum_{j=1}^{\mathbf{n_{TS}}} \frac{\mid \mathbf{\hat{u}_j - u_j} \mid}{\mathbf{IQR(u)}} \qquad (6)$$

where $\mathbf{n_{TS}}$ is the number predicted samples, $\mathbf{\hat{u}_j}$ and $\mathbf{u_j}$ are the predicted and the actual values at time point $\mathbf{j}$, and $\mathbf{IQR(u)}$ is the interquartile range of the performance counter series.

In order to quantitatively assess the forecasting accuracy, we employ equation 6 to compute the discrepancy between predicted samples and actual ones. Figure 5 presents the distribution of prediction error for forecasting L2-miss-ratio five steps ahead into the future. We note 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, which indicates that the variance of error is also smaller. The mean prediction error is $\mathbf{10\%}$, $\mathbf{11\%}$ and $\mathbf{13\%}$ for SVR-Direct, SVR-Recursive and AR-adapt models

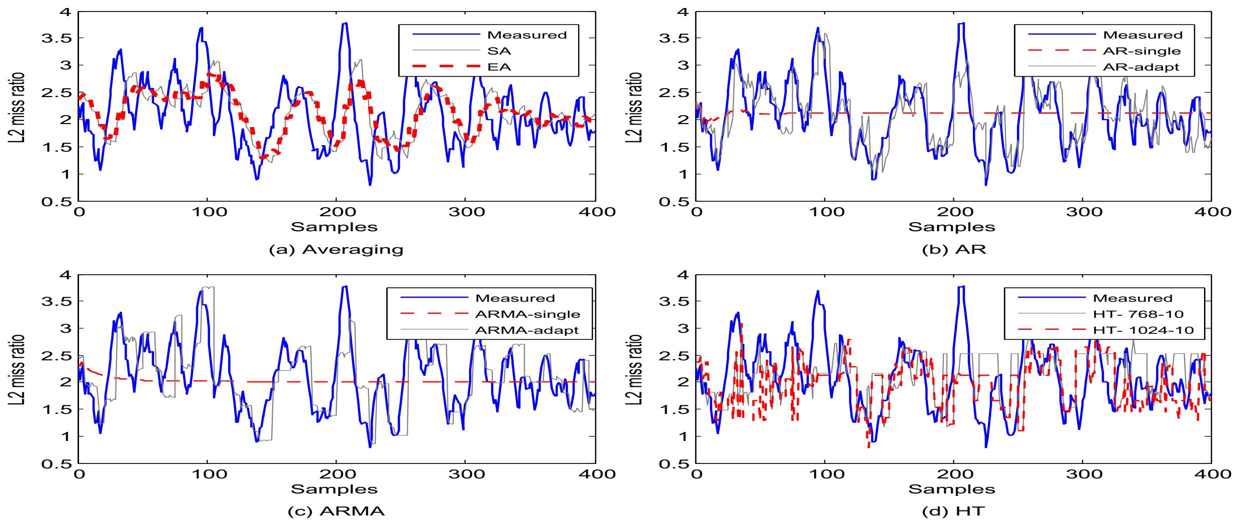
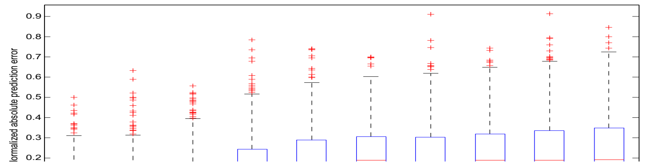


Fig. 4: Forecasting L2 miss ratio, five steps ahead using existing predictors



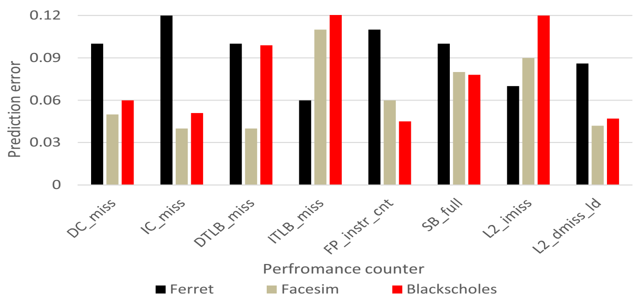


Fig. 6: Prediction error of all eight counters for three workloads

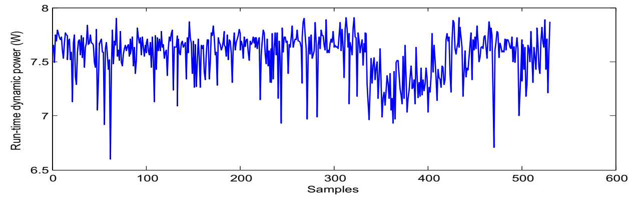
respectively. Note, that the SVR models are trained earlier using the history data and are used for prediction without adaption. Therefore, they show great potential for real-time workload monitoring and forecasting.

Figure 6 demonstrates the prediction error for all performance counters of Table I for three applications of PARSEC benchmark. For each performance counter, half of the data from one of the workloads is used to train the SVR model. Then, the trained model is used to predict future values of performance counters for the current workload as well as for the other two workloads. In this work, SVR models are trained using data collected for Facesim program. Therefor, the prediction error for this program is slightly lower than the other two programs.

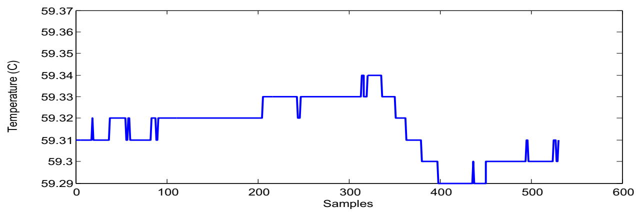
C. Power and Temperature Prediction

As mentioned earlier, within the context of our research the ultimate objective of workload prediction (and other proactive techniques) is to improve longevity of a system through workload scheduling and other circuit-level techniques. In this quest, beyond expected workload characteristics, the optimization functions needed to drive the adaptation algorithms require power and temperature estimation to reflect device stress and aging. Therefore, forecasting power and temperature is necessary to take the right action and prevent the thermal problem. Herein, we use the McPAT [16] power simulator to compute power consumption of each core from its architectural trace. Then, we feed the power trace to a thermal simulator to obtain the temperature trace. We used HotSpot [4] as the thermal modeling tool and modified the floorplan and thermal characteristics to reflect the UltraSPARC T1 processor. McPAT is modified to generate proper power traces for HotSpot. Figures 7(a) and 7(b) show the power and temperature trace for core 1 during execution of Ferret program. Performance counter data were averaged over a window of two seconds and fed to McPAT to obtain the power trace. 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.

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 of the Ferret application along with their corresponding power to train an SVR model. Then the model is used to predict power from the value of the performance counters. It should be noted that only a single SVR model is trained and used for all three programs. Figures 8(a), 8(b) and 8(c) show the power prediction for Ferret, Facesim and Blackscholes respectively. As one can

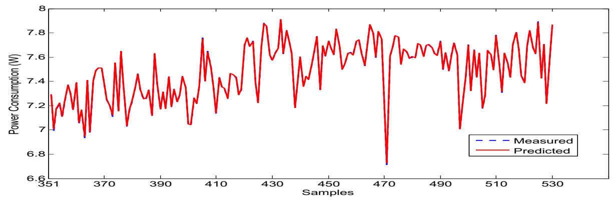


(a) Power consumption of core 1 during workload execution

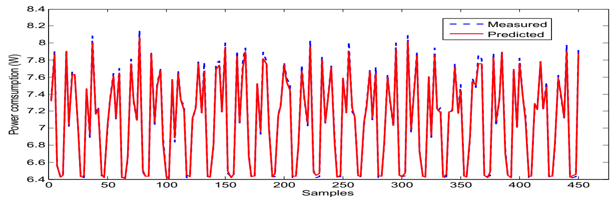


(b) Temperature of core 1 during workload execution

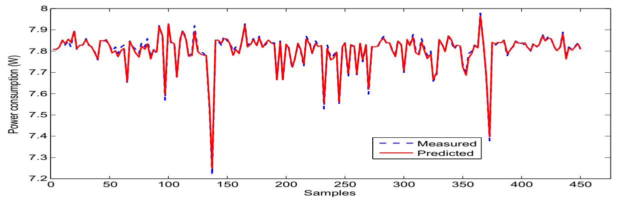
Fig. 7: Power and temperature profile of core 1 during workload execution



(a) Ferret application



(b) Facesim application



(c) Blackscholes application

Fig. 8: Power consumption prediction using performance counter data

see, the SVR model can predict the power consumption very accurately.

VI. Conclusion

We introduced Support Vector Machines (SVM) as a modeling method for accurately forecasting future workload characteristics which can be used to drive a proactive reliability management approach in multi-core systems. We provided comparison of our SVR-based method to other prediction techniques such as averaging, last value, history table and ARMA. All experiments were run on UltraSPARC T1 and data of performance counters were collected for several workloads. Our results show that our forecasting model outperforms existing methods in terms of prediction error. Average normalized absolute error for SVR is about **20**% less than the AR model. Another advantage of the SVR technique is its evaluation overhead. AR and ARMA models require continuous adaptation for accurate prediction, which increases the overhead and degrades the performance in real-time scenarios, while the SVR approach surpasses their performance without requiring retraining.

References

[1] D. Brooks et al., "Dynamic thermal management for high-performance microprocessors," in *Proc. IEEE International Symposium on High-Performance Computer Architecture*, 2001, pp. 171–182.

[2] S. Gunther et al., "Managing the impact of increasing microprocessor power consumption," *Intel Technology Journal*, vol. 5, no. 1, pp. 1–9, 2001.

[3] J. Donald et al., "Leveraging simultaneous multithreading for adaptive thermal control," in *Proc. of the Second Workshop on Temperature-Aware Computer Systems*, 2005.

[4] K. Skadron et al., "Temperature-aware microarchitecture," in *Proc. ACM SIGARCH Computer Architecture News*, 2003, vol. 31, pp. 2–13.

[5] A. Coskun et al., "Temperature aware task scheduling in mpsocs," in *Proc. Design, Automation & Test in Europe Conference*, 2007, pp. 1659–1664.

[6] S. Murali et al., "Temperature control of high-performance multi-core platforms using convex optimization," in *Proc. Design, Automation & Test in Europe Conference*, 2008, pp. 110–115.

[7] A. Kumar et al., "HybDTM: A coordinated hardware-software approach for dynamic thermal management," in *Proc. IEEE Design Automation Conference*, 2006, pp. 548–553.

[8] A. Coskun et al., "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proc. ACM SIGMETRICS Performance Evaluation Review*, 2009, vol. 37, pp. 169–180.

[9] C. Isci et al., "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," in *Proc. IEEE/ACM International Symposium on Microarchitecture*, 2006, pp. 359–370.

[10] A. Coskun et al., "Proactive temperature balancing for low cost thermal management in mpsocs," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 250–257.

[11] J. Gao et al., "A probabilistic framework for SVM regression and error bar estimation," *Machine Learning*, vol. 46, no. 1-3, pp. 71–89, 2002.

[12] A. Smola et al., "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[13] P. Kongetira et al., "Niagara: A 32-way multithreaded sparc processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, 2005.

[14] R. McDougall et al., *Solaris (TM) Performance and Tools: DTrace and MDB Techniques for Solaris 10 and OpenSolaris (Solaris Series)*, Prentice Hall PTR, 2006.

[15] C. Bienia et al., "PARSEC 2.0: A new benchmark suite for chip-multiprocessors," in *Proc. Workshop on Modeling, Benchmarking and Simulation*, 2009.

[16] L. Sheng et al., "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 469–480.