**RC24474 (W0801-091) January 23, 2008**
**Computer Science**

# IBM Research Report

## Unified Prediction Method for Predicting Program Behavior

**Ruhi Sarikaya, Alper Buyuktosunoglu**
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

# Unified Prediction Method for Predicting Program Behavior

*Ruhi Sarikaya and Alper Buyuktosunoglu*
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{sarikaya,alperb}@us.ibm.com

## ABSTRACT

Dynamic management of computer resources is essential for adaptive computing. Adaptive computing systems rely on accurate and robust metric predictors to exploit runtime behavior of programs. In this study, we propose the Unified Prediction Method (UPM) that is system- and metric-independent for predicting computer metrics. Unlike ad-hoc predictors, UPM uses a parametric model and is entirely statistical and data-driven. The parameters of the model are estimated by minimizing an objective function. Choice of the objective function and the model type determines the form of the solution whether it is closed form or numerically determined through optimization. In this study two specific realizations of UPM are presented. The first realization uses mean squared error (MSE) objective function and the second realization uses accumulated squared error (ASE) objective function, in conjunction with autoregressive models. The former objective function leads to Linear Prediction and the latter leads to Predictive Least Square (PLS) prediction. The model parameters for these predictors can be estimated analytically. The prediction is optimal with respect to the chosen objective function. An extensive and rigorous series of prediction experiments for the instruction per cycle (IPC) and L1 cache miss (L1–miss) rate metrics demonstrate superior performance for the proposed predictors over the last value predictor and table based predictor on SPECCPU 2000 benchmarks.

## 1. INTRODUCTION

Intra and inter application variability motivate the design of adaptive computer systems where the system adapts to different requirements of the applications. In such an adaptive system, the underlying mechanism implicitly or explicitly utilizes the repetitive and recognizable patterns in the application. This repetitive intra-workload variability also refered as phase behavior may last successive time periods before making a transition to the next one. For each such phase in the application, predictors give the ability to predict and proactively adapt the computer system to the application needs.

Last value predictor is especially popular and has been widely used because of its simplicity. The last value predictor simply assumes that the next sample will be same as the current observed sample. Although this predictor is effective for stable application behavior, it gives inaccurate predictions for highly variable application behavior. For highly variable data, several other methods including table based predictor [13] and cross metric predictor [8, 13] are proposed to alleviate the shortcomings of the last value predictor. If all of the aforementioned ad-hoc methods were successful in predicting application behavior, then conceptually there would not be a new problem to solve in adaptive computer systems. However, most of these predictors center on the use of past values to predict next values without any mechanism with regards to how to use them in a principled and optimum way. In that respect, these predictors are not reliable and accurate. Furthermore, they are typically designed to work for a specific metric of interest, and may not generalize to other metrics. For example, the performance of table based predictor depends on such parameters as history vector length, quantization granularity and table depth. Typically, one has to tune these parameters for different metrics to achieve the best possible performance.

In this paper, we propose the Unified Prediction Method (UPM) for metric prediction problem. UPM is a generic prediction method where last value predictor, linear predictors and many other predictors can be considered as special cases of UPM. UPM is a principled, novel and practically viable approach to predict values of a metric within a application. UPM has also strong theoretical foundations in statistics as well as signal processing and it does not require offline or online calibration. As such, it is independent of the system and program being predicted. It applies to any metric of interest and performs accurate, reliable on the fly predictions.

Application of UPM consists of several steps. First, the model type (e.g. autoregressive, Markov) and the objective function (e.g. total squared error, accumulated squared error, likelihood) are selected. Next, the parameters of the model are estimated by minimizing the chosen objective function. Note that in general, the objective function itself is a function of model parameters and past observed samples. Later, the estimated model is used to predict the next value.

Our contributions can be summarized below. Note that this work expands our recent work [6] in several ways which are highlighted in the last three bullets:

- We develop UPM based on objective function minimization.
- We show that choice of the autoregressive models as the model type and mean squared error (MSE) as the objective function reduces UPM to Linear Predictor. Thus, we implement Linear predictor as a special case of UPM.
- We extend Linear Predictor concept and implement the Predictive Least Square (PLS) predictor by using mixture of autoregressive models and last value predictor with accumulated squared error (ASE) objective function. In this scheme, the prediction error is guaranteed to be less than or equal to that of the last value predictor within a small $\epsilon$ error term.
- We also compare the proposed predictors to a last value predictor and demonstrate substantially improved results over the last value predictor.
- We also compare these predictors to a table-based predictor. We demonstrate the overall superiority of PLS predictor over table based predictor both in terms of accuracy and consistency.
- We also investigate the impact of mixture size on the accuracy of PLS predictor.
- We perform a series of rigorous experiments comparing aforementioned methods on IPC and L1 miss rate using two different sampling and prediction intervals.

The experiments show that both Linear Predictor and PLS predictor outperform last value predictor such as last value. While PLS predictor significantly outperforms the last–value predictor for some subsets of the benchmarks, unlike previous studies [8, 13], it never provides worse (within an $\epsilon$ error term) prediction than the last value predictor. For example, table based predictor shows large variations in its performance in relation to last value predictor. As such, it suffers from consistency. PLS predictor, on the other hand, is consistent and generally provides better prediction accuracy than table based predictor with the exception of several benchmarks.

PLS predictor is also effective in overcoming the shortcomings of the Linear Predictor with negligible extra overhead. All our techniques are driven by data from processor performance counters. Both of the proposed techniques provide effective prediction capabilities which are practically viable through a software implementation over millisecond intervals.

In what follows, Section 2 presents the UPM based on objective function minimization. Section 3 introduces the Linear Predictor as a particular realization of the UPM. Section 4 describes PLS. Relevant prior work in program behavior characterization and prediction is summarized in Section 6. Detailed experimental results and discussions are provided in Section 7. Section 8 concludes the paper.

## 2. THE UNIFIED PREDICTION METHOD USING OBJECTIVE FUNCTION MINIMIZATION

The Unified Prediction Method (UPM) is a parametric model and is based on statistical framework. The first step for the application of UPM is to choose a parametric model $\theta$. The second step is to define an objective function. The objective function typically depends on both past data samples and model parameters. The model parameters are estimated by minimizing the objective function:

$$\hat{\theta}(t) = \underset{\theta(T_1^{t-1}),X_1^{t-1}}{\arg\min}\ \mathcal{O}(\theta(T_1^{t-1}), X_1^{t-1}) \qquad (1)$$

where $T_1^{t-1} = \{1, \ldots, t-1\}$ is the time from the begining of the observation to time $t-1$ at which the last sample is observed and $X_1^{t-1} = \{x[1], \ldots, x[t-1]\}$ are all past data samples observed up to time $t$. Note that the sample to be predicted arrives at time $t$. In general, $\theta(T_1^{t-1})$ are time dependent models that are estimated so far. $\mathcal{O}(\theta(T_1^{t-1}), X_1^{t-1})$ is the objective function to be minimized to estimate the prediction model parameters at time $t$. Finally, $\hat{\theta}(t)$ is the model to be used to predict the next sample arriving at time $t$. We cast the metric prediction problem as that of minimizing an objective function given the model parameters and the past observed samples. Eqn. 1 is sufficiently general in that, many commonly known objective functions (e.g. mean square error, likelihood, accumulated squared error) in conjunction with specific model classes (e.g. autoregressive models, Markov models) are considered specific cases of the objective function minimization. The objective function given in Eqn. 1 can be minimized either analytically or via numerical optimizations. The form of the solution is determined by the chosen objective function and the model type. For example, if the objective function is Mean Square Error (MSE) and the model type is autoregressive, there is a closed–form solution, which can be obtained analytically. Solutions through numerical optimizations may carry a large overhead and they may not have convergence guarantees. Therefore, we prefer objective function and model pairs for which analytical solutions can be obtained without significant computational overhead.

We consider two specific realizations of the UPM by concentrating on autoregressive models with mean squared error (MSE[1]) and accumulated squared error (ASE) as the objective functions.

## 3. LINEAR PREDICTION METHOD

The choice of model type and objective function are key for the UPM model. If we use autoregressive models with the MSE objective function, UPM reduces to the commonly known Linear Prediction method. Linear Prediction is a method for signal source modeling and is used in many applications such as speech signal processing, coding, power spectral estimation, sensor array processing, and optimal digital filtering. The objective of Linear Prediction is to form a model of a linear time–invariant digital system through observation of input and output sequences. Linear Prediction or more generally least–squares algorithms [15, 16] require very little information on the statistics of the input data and they have simple implementations. These are desirable properties that make Linear Prediction models suitable for metric prediction problem.

### 3.1 Mean Square Error As The Objective Function

MSE lends itself as an appealing objective function. There are numerous advantages of using MSE, including:

- Minimization of the MSE objective function leads to linear system of equations, which are straightforward to solve without significant cost. Any other choice of an objective function may lead to nonlinear system of equations that can be solved through numerical optimization. Nonlinear system of equations can be expensive to solve without convergence guarantees.
- MSE objective function favors many small errors over a few big ones that linearly add up to the same value, which is usually desirable in order to minimize the impact of the overshoots in prediction.
- MSE is the error "variance", which grows monotonically with standard deviation, so minimization of squared error also minimizes other statistical measures that are commonly used.

[1]As far as the derivation of the Linear Prediction parameters are concerned, minimization of total squared error, summed squared error, expected squared error or MSE are the same.

### 3.2 Minimizing MSE for Linear Prediction Parameter Estimation

Before formally describing the objective function minimization process, let us define some notations. We consider any computer metric as a signal, $x[t]$, where $t$ denotes the time at which signal is sampled. $x[t]$ can be approximated by a linear combination of its own past $p$ samples:

$$\begin{aligned} \hat{x}[t] &= \sum_{k=1}^{p} \theta_{p,k} x[t-k] \\ &= \sum_{k=1}^{p} a_k x[t-k] \end{aligned} \tag{2}$$

where $\hat{x}(t)$ is the predicted output, and $\theta_{p,k} = [a_1, \ldots, a_p]$ are model parameters, which are used to weigh past samples to generate an estimate of the next sample. A model that depends only on the previous outputs of the system is called an autoregressive model (AR). Autoregressive model finds the coefficients of a $p$th order linear predictor that predicts the current value of the real–valued time series $x[t]$ based on past samples. The model parameters, $a_k$ are assumed to be constant over the duration of the analysis window. The parameters of the Linear Predictor are estimated as follows. Given past $n$ samples of a metric, we like to compute estimates for $a_i$ that result in the best fit. One reasonable way to define "best fit" is in terms of MSE.

$$\begin{aligned} e_t &= x[t] - \hat{x}[t] \\ &= x[t] - \sum_{i=1}^{p} a_i x[t-i] \end{aligned} \tag{3}$$

where $e_t$ denotes the prediction error at time $t$. Fixing the model order $p$ and computing the total squared error over a window of $n$ observed past samples gives the following objective function:

$$\begin{aligned} \mathcal{O}_t(x, \hat{x}_p) &= \sum_{t=0}^{n} e^2[t] \\ &= \sum_{t=0}^{n} [x[t] - \sum_{k=1}^{p} a_k x[t-k]]^2 \end{aligned} \tag{4}$$

$\mathcal{O}_t(x, \hat{x}_p)$ is quadratic in each of $a_i$ and is convex. The minimum of $\mathcal{O}_t(x, \hat{x}_p)$ occurs when the derivative with respect to each of the parameters $a_i$ is zero. Therefore $\partial\mathcal{O}_t(x, \hat{x}_p)/\partial a_i = 0$ must give a minimum. By differentiating Eqn. 4 with respect to $a_i$ and setting equal to zero gives the set of $p$ equations. The $p$ equations are solved recursively [24]. One of the important advantages of the Linear Prediction technique is its computational efficiency. The lattice implementation of the technique require only $O(p)$ operations [14], which can be easily implemented in software.

### 3.3 Model Order Selection for Linear Prediction

The performance of Linear Prediction for a specific problem depends largely on the model order used. Picking the optimal model order, which also determines the number of past samples to use, is crucial for applying Linear Prediction. However, the ordinary Linear Prediction technique merely minimizes the mean error squares for a fixed model order and it does not provide means for model order selection. For almost all Linear Prediction applications, either the distribution of the input data is assumed to be known, or a fixed model order is arbitrarily chosen for all possible input data. Obviously, one can hardly imagine that any of the so-called optimum prediction techniques are able to perform “true” optimum prediction without a method for estimating the model order for the input data.

Model order selection is a long–standing problem in signal processing and statistics. There is a large body of research directed at this problem. There are several landmark contributions in this area including Akaikes information criterion (AIC) [1] and the minimum description length (MDL) [17]. Both of these methods require that the data distribution is Gaussian and are founded on asymptotic results. Their implementation requires two passes over the data, one to estimate the optimal parameters and a second to calculate the prediction errors, and as such they are not preferred for online applications.

## 4. PREDICTIVE LEAST SQUARE PREDICTION

Choosing the objective function as the “accumulated squared prediction error” and using a mixture of Linear Predictors reduces UPM to Predictive Least Squares (PLS) predictor. In [18], PLS principle is proposed as an application of the “predictive stochastic complexity” concept [17] for model order selection. The PLS principle gives a consistent estimate of number of parameters (model order) for the basic Gaussian regression process. The PLS principle gives the maximum lower bound for the number of binary digits needed for encoding the observed values of data. The PLS criteria represents the minimum accumulated squared prediction error, when the data is predicted sequentially one at a time. The criteria selects the model order as the one for which the sum of the squared prediction errors along the sequence is minimized. The model order that achieves the minimum (sequentially) accumulated squared prediction error is used for predicting the next sample. The PLS method estimates the model order and the predicted value at the same time. In other words, it does not perform multiple passes over the data.

The difference between the“accumulated squared prediction error”, which is the objective function for PLS, and the “total squared prediction error”, which is the objective function for ordinary Linear Prediction, lies both in the method used to compute the predictor coefficients and in the samples over which the error is computed. The total squared prediction error results from applying the fixed set of predictor coefficients obtained by minimizing the square prediction error over the same set of data. The accumulated squared prediction error, on the other hand, results from sequential application of a time–dependent set of predictor coefficients.

The problem of optimal model order selection for ordinary Linear Predictor is no longer an issue for PLS predictor. Instead of relying on a single Linear Predictor, PLS predictor uses a mixture of Linear Predictors with different model orders including the last value predictor. At each time $t$, PLS–predictor computes the total prediction error for each model in the mixture. It also keeps track of the accumulated prediction errors for each model in the mixture. For the prediction of the next sample, it finds the model that provides the smallest accumulated prediction error so far and uses it to predict the next sample. The flexibility and modeling power of PLS–predictor depends on mixture size and prediction ability of the individual models in the mixture. As the data behavior changes so may the model that best describes the data. PLS–predictor takes advantage of model switch and can rely on the prediction of the different models in different time intervals.

Interestingly, PLS predictor has a computational complexity that is not larger than that of the largest model order in the mixture. Efficient recursive lattice implementations can be used to generate all of the Linear Predictors at the computational price of only the largest model order. This is possible due to the fact that for a given model order all the models of smaller order are by–products of the model estimation process. The overall cost of PLS–predictor is negligible for the prediction task that is performed here.

### 4.1 Formulation of Predictive Least Square Prediction

Similar to ordinary Linear Prediction, the goal is to predict a sequence of $t$ observations $x[1], \ldots, x[t]$ in terms of its $p$ past observations, $x[t-1], \ldots, x[t-p]$. We can define "accumulated squared error" (ASE) objective function:

$$\mathcal{O}_p(t) = \sum_{i=1}^{t} (x[t] - \theta_{p,i}\mathbf{x}_{p,i})^2 \tag{5}$$

where $\mathbf{x}_{p,i} = [x_{i-1}, x_{i-2}, \ldots, x_{i-p}]$ and $\theta_{p,i}\mathbf{x}_{p,i}$ is an inner product of model parameters and a vector past $p$ observations. One can obtain a Linear Prediction filter of length $p$ by finding a parameter vector, $\theta_{p,i} = [a_1(i), a_2(i), \ldots, a_p(i)]^T$ to minimize the objective function $\mathcal{O}_p(t)$:

$$\min_p \sum_{i=1}^{t} (x[t] - \theta_{p,i}\mathbf{x}_{p,i})^2 \tag{6}$$

where $\theta_{p,i}$ is the least squares parameter estimate determined by data measurements up to time $i$. We define $\hat{x}_i = \theta_{p,t-1}\mathbf{x}_{p,i}$ as the "honest" least squares prediction of $x(t)$, where $\theta_{p,t-1}$ is the model parameter vector associated with the linear regression model.

It is worth mentioning that Eqn. 6 contains a second layer of minimization over the model order. Ordinary Linear Predictor however, operates with a fixed model order to minimize MSE. As such, PLS has more flexibility to switch between different model orders for each sample. Applying the PLS principle requires solving Eqn. 6 to obtain a parameter vector, $\theta_{p,t}$ for each $p$. Since the parameter vector is re-estimated for each data sample observed in time, $\theta_{p,t}$ is time dependent. As such, the model parameters are optimized only over the data available up to but not including the sample to be predicted. Given $\theta_{p,t}$, we can estimate the accumulated squared error:

$$\begin{aligned} \mathcal{O}_p(X_1^T) &= \sum_{i=1}^{t} (x_i - \theta_{p,i}x_{p,i})^2 \\ &= \sum_{i=1}^{t} (x_i - \hat{x}_i)^2 \end{aligned} \tag{7}$$

for all $p$ when $\hat{x}_t$ is the estimate of $x_t$. Then choose the $p = p_{min}(t)$ that minimizes $\mathcal{O}_p(X_1^t)$, and use it for prediction of the next sample, $x_{t+1}$, that can be estimated as an inner product of model parameters and past samples:

$$\hat{x}_{t+1} = \theta_{p_{min}(t),t}\mathbf{x}_{p_{min}(t),t+1} \tag{8}$$

The PLS predictor guarantees the least ASE without making any assumption on the distributions of $\{x\}$. It also gives a consistent estimate of the model order providing a minimum model order for Gaussian regression process [18, 19]. As it will be shown in Section 6.1 last–value predictor can be considered as a special case of Linear Predictor with $p = 1$. Therefore PLS predictor guarantees to perform better or equal to last value predictor or any other Linear Predictor in the mixture within an $\epsilon$ error term which is bounded by $c\log(n)/n$ [18, 25], where $c$ is a constant and $n$ is the length of the data sequence. Note that, the error term diminishes rapidly as number of data samples increases.

## 5. METHODOLOGY

We performed the experiments on an IBM POWER4 [22] server platform with the AIX5L operating system. The results present per-thread behavior running in multi-user mode on a lightly loaded machine. We have used the hardware performance counters, with a sampling tool that works on top of the AIX Performance Monitoring API (PMAPI). The sampler ties together counter values to a particular thread, including all library calls and system calls performed by that thread. The prediction is performed at two different time scales, approximately 50ms and 100ms, to quantitatively understand the effect of different time granularities on the prediction accuracy. All the experiments are performed with the SPECCPU 2000 suite using reference datasets. All benchmarks are compiled with XLC and XLF90 compilers with the base compiler flags.

## 6. RELATED WORK

Previous research on phase characterization and prediction focuses on various categories. A body of research concentrates on offline simulator or counter analysis to select representative execution points [10, 11, 20]. While these approaches largely deal with phase characterization, they do not perform predictions guided by online phase changes. Another group of approaches experiments with architectural dynamic optimizations guided by fine–grained phase information [2, 3, 7, 9, 21]. These techniques operate at fine grain phases and are not effective handling coarse grain phases. Some application-oriented work has been done to read hardware counters and use their information for techniques like DVS or thermal-aware OS scheduling [4, 5, 23] at the operating system level. All of these are reactive, while our approach shows the ability to predict and to proactively adjust system behavior. We implemented two of the previous methods, namely last value predictor and global history table predictor to compare against the proposed methods.

Duesterwald et al. [8] present the performance of various coarse-grained prediction schemes for performance counter metrics on two different IBM POWER platforms. They show the possibility of cross metric predictions and mention potential applications of such predictive techniques. While [8] focuses on near-term sample-to-sample predictions, Canturk et al. [12] has developed longer-term approaches incorporating duration prediction with gradient prediction schemes. Both of these methods center on the use of past values to predict the next values, but without any mechanism with regards to how to use the past values in a principled and optimum way. Our presented technique applies to any metric of interest and performs accurate, reliable on the fly "optimum" predictions. We claim that PLS predictor, while significantly outperforming the last–value predictor for some subsets of the benchmark data, it never provides worse prediction than the last–value predictor. That is, in some cases it may perform equally well as the last value predictor.

### 6.1 Last Value Predictor

One of the simplest yet effective metric predictor is the last value predictor. In this predictor, the next sample of metric is assumed to the last observed sample of that metric. In fact, last value predictor can also be considered as a specific realization of UPM with the following objective function:

$$\mathcal{O}(t) = x[t] - x[t-1] \tag{9}$$

and the model,

$$\theta(t) = \delta(t-1) \tag{10}$$

where $\delta(t) = 0,\ t \neq 0$ is the Dirac function and it can be considered as model $\theta(t)$. The next sample $\hat{x}$ to be predicted is,

$$\hat{x} = \delta(t-1)x[t] \tag{11}$$

Note that for a last–value predictor the model is fixed and it does not have any parameters to estimate.

### 6.2 Global Phase History Table Predictor

In addition to last value predictor, we also implemented the Global Phase History Table (GPHT) [13] predictor. GPHT predictor exploits repetitive phase behavior by looking not only at the last observed sample but a larger sequence of observed samples to predict the next sample. GPHT predictor resembles global branch history predictor [26]. Canturk et al. implement GPHT at the software layer for dynamic phase prediction. Later, dynamic phase prediction is used to trigger DVFS actuation.

GPHT method works as follows. We first decide on the size of the pattern history table (PHT), which is essentially a matrix. The rows of this matrix represent the distinct pattern vectors acquired from the past observed sample sequences. The length of these vectors is related to both the predictive power of PHT and the hit rate of finding a pattern in the PHT. If these vectors are too short, PHT has a weaker predictive power, but has a greater hit rate (finding a pattern in the table). On the other hand, if the vector is too long, it gets difficult to find the matching patterns in PHT. A PHT of 1024 by 8 is found to be a good tradeoff between these competing goals [13]. In other words, we keep 1024 distinct pattern vectors, each of which is 8 samples in length. For each of these vectors we also keep the corresponding next sample as the output of these vectors. Here in this study, we used the same table size as in [13]. However, unlike [13] we used 20 quantization levels instead of 6, since the dynamic range of IPC and L1-miss rates are larger than that of the memory accesses per instruction (MPI).

For each sample to be predicted, the GPHT method looks at the most recently observed 8 samples, and searches for the exact match of this vector in PHT. If there is a match, it takes the corresponding output for that vector. If there is no matching pattern in PHT and then it backs off to the last value predictor and takes the most recently observed sample as the prediction output. Most applications exhibit some amount of repetitive patterns due to the common loop-oriented and procedural execution style. Effectiveness of the table predictors relies on the extent of the repetitive patterns existing in the metric.

One of the disadvantages of table predictors is the quantization required to generate the table entries. The patterns in the table are quantized values for the real valued metric samples. The number of quantization levels should be decided carefully. On the one hand having too few quantization levels does not allow accurate prediction. On the other hand having too many quantization levels makes it difficult to find the matching vector, thus reduces the table hit rate. After several experiments, we decided to use 20 quantization levels for both IPC and L1-miss rate.

## 7. EXPERIMENTAL RESULTS AND DISCUSSIONS

We present an extensive set of experiments for Linear Predictors, PLS predictor and their comparison to last value predictor and table based predictor. We also investigate PLS predictor in-depth for its performance in relation to mixture size. All the predictors are applied to predict IPC and L1–miss rate of benchmarks in SPECCPU 2000 suite. Even though PLS predictor is applied to predict IPC and L1–miss rate, it has general applicability to predict any metric of interest with

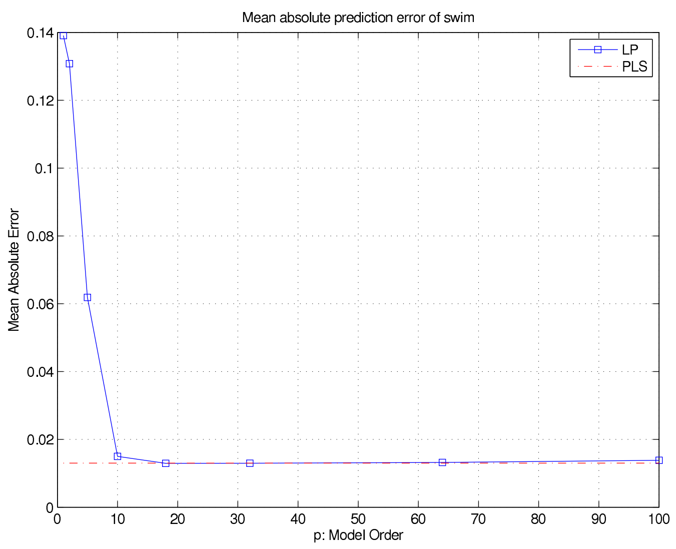

Figure 1: Mean absolute prediction error of swim IPC data with different model orders for LP and PLS predictors.

great accuracy. This section also provides the prediction accuracy results at different time scales. Note that, unless otherwise indicated, the figures in the following sections show results where prediction is performed at 100ms time scales.

Much of the past work formulated ad-hoc methods that are engineered to work on specific metric values. Typically, these methods may perform very well for a few benchmarks as compared to last value predictor. However for the rest of the bencmarks they do not provide better prediction accuracy than the last value predictor. As such, they lack the generality and fail to work on other metrics. We guarantee that the PLS predictor does not perform worse than the best predictor in the mixture including the last value predictor within an $\epsilon$ error term. Due to space limitations, mathematical proof of this assertion is omitted [18, 25]. The PLS-predictor also eliminates a major short-coming of Linear Prediction models, that is the correct model order for the best description of the sequential metric samples. With the PLS predictor, as the metric behavior changes so does the best model describing the metric value. On–the–fly determination of the best model order is guaranteed for the PLS predictor.

### 7.1 Metric Tracking Using Linear Prediction Models

The optimal model order for Linear Prediction depends on the specific benchmark. In this study, we consider the following model orders for Linear Prediction: $p \in \{2, 5, 10, 18, 32, 64, 100\}$. We do not have any particular reason to choose this set over others except for sampling the different model orders in a reasonable range. We also use the last value predictor both as a baseline

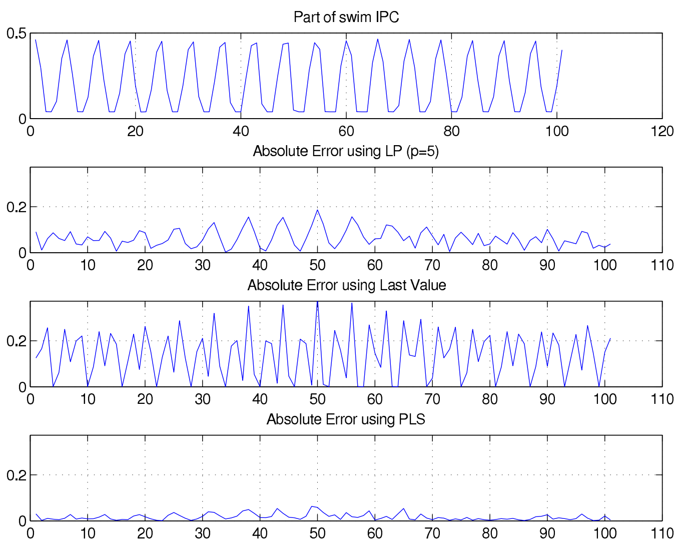

Figure 2: Absolute IPC prediction errors for different predictors over a segment of the swim execution.

to compare and as part of the mixture of models to be used by the PLS predictor. Using larger model orders results in modeling the finer structure of the metric values and may lead to worse performance, especially if the data behavior is frequently changing. Linear Prediction model orders of more than 100 are rarely used in practice. Likewise, using smaller model orders may not have enough flexibility to model the dominant structure of the metric values. Across many disciplines and applications Linear Prediction model orders of 5 to 30 are commonly used.

The mean absolute prediction errors for the prediction of IPC using "swim" data from the SPECCPU 2000 benchmark suite is presented in Figure 1. The solid line in the figure shows the errors using Linear Prediction models of different orders. In the figure, $p = 1$ is used for the last–value prediction. The dashed line corresponds to PLS predictor, which will be discussed in the next section. We observe from the figure that Linear Prediction model orders of $p = 30$ to $p = 70$ provides the lowest mean error for the swim IPC data. Using smaller model order may substantially increase the error and using higher model order may also result in modest degradations in the mean error. Last alue prediction does not perform as well as any of the Linear Prediction models for this benchmark.

In the first panel of Figure 2, we plot a segment of the swim IPC data. This segment of the data seems to be periodic with significant differences in value between consecutive samples. In the second panel, the corresponding absolute prediction error is plotted using Linear Prediction model with order 5 ($p = 5$). In the third panel, we plot the prediction error for the last value predictor. The large consecutive sample–to–sample dif-

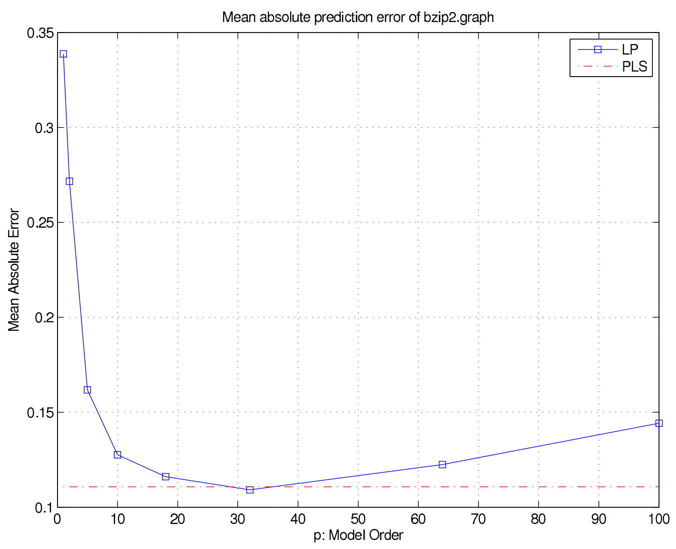

Figure 3: Mean absolute prediction error of bzip2_graph IPC with different model orders for LP and PLS predictors.

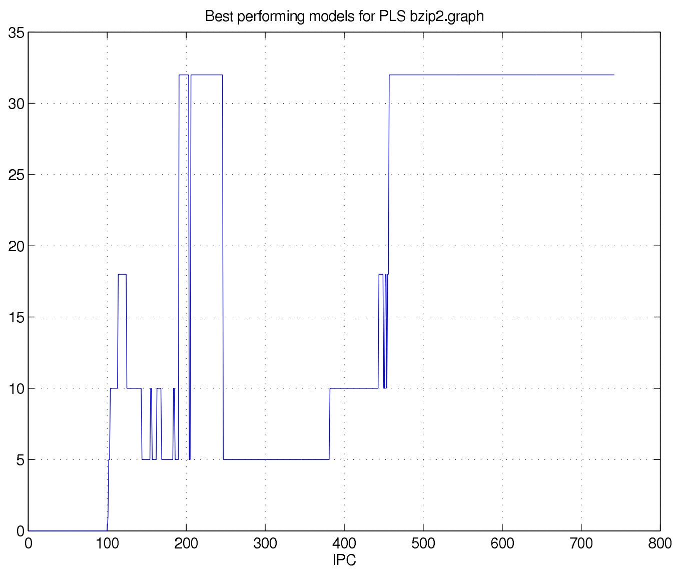

Figure 4: Model orders picked by PLS during the execution of bzip2_graph.

ferences explain the relatively poor performance of last–value predictor, which does not capture periodicity in the data, as compared to Linear Prediction model with order 5.

### 7.2 Metric Tracking Using PLS Predictor

The effectiveness of PLS predictor depends on the number and predictive power of the models in the mixture. We will present several examples of PLS prediction as compared to Linear Prediction models of different orders and last value prediction. In Figure 1, the dashed line corresponds to PLS prediction error. The PLS performance is as good as the best performing model in the mixture. That is, PLS performance is the same[2] as those for the Linear Prediction models of order $p = 32$ and $p = 64$. At the bottom panel of Figure 2, the prediction error using PLS predictor is plotted for a segment of the "swim" IPC data. PLS predictor is tracking the data very closely and makes significantly smaller prediction errors than the last value predictor and Linear Predictor with model order 5 (LP-5). It is worth mentioning that since we use Linear Prediction model with order 100, we need to have at least 100 past samples to estimate the model parameters. Even though Linear Prediction models of lower orders are active and make predictions during this time, we choose to start the prediction process when we observe at least 100 samples so that all models in the mixture are active. As the sequential data arrives the behavior of the data may change. Consequently, the best performing model order may change as well.

It is essential to point out the dependency betwen model order and Linear Predictor performance. Large Linear Prediction model orders provide always best performance in the "batch prediction" case, that is, if the entire data is observed. The large model orders provide the flexibility to describe the details in the part of data that is observed up to now. However, this comes at the expense of poor generalization for the unseen part of the data, which are the future samples to be predicted. Therefore, for sequential prediction large model orders do not always achieve the lowest prediction error. For example, Figure 3 shows the mean prediction error for different models in the mixture for the "bzip2_graph" IPC data. For this benchmark, Linear Predictor with model order 32 provides the lowest accumulated prediction error. Using Linear Prediction models of order less than 18 or more than 32 may result in increased prediction error. The last value predictor performs significantly worse as well, increasing the mean absolute prediction error by about 3-fold ($0.11 \rightarrow 0.34$). Figure 4 shows the model switch for the entire "bzip2_graph" data. First PLS switches between LP–5, LP–10 and LP–32, and finally settles for LP–5 for a while and then switches back to LP–10. Later PLS uses LP–18 and LP–32. In order to see the model switching impact on the prediction error, we zoom into the region between IPC samples of 320 and 420 and plot the results for last value, LP–5 and PLS–predictor in Figure 5. In the first half of the figure, PLS predictor and LP–5 have virtually identical prediction errors. As soon as the PLS predictor switches from LP–5 to LP–10 the PLS prediction error starts to become smaller than the LP–5 prediction error. Essentially, PLS–predictor uses the best model order to fit to the data in different segments of the execution.

[2]The performance difference is negligibly small.

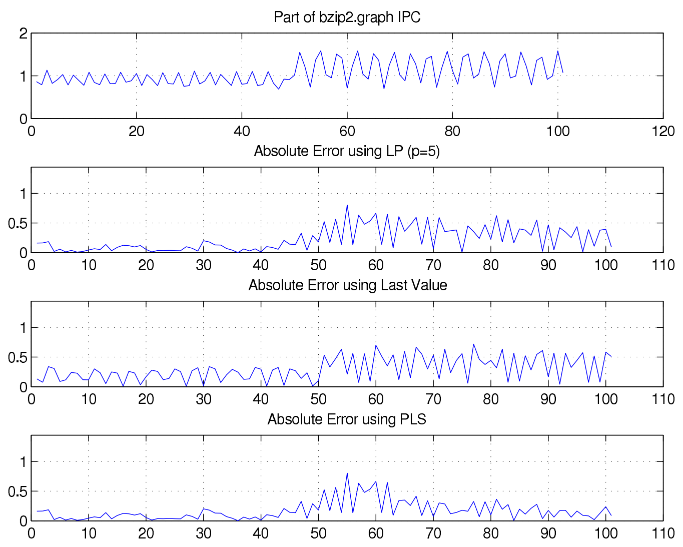

Figure 5: Absolute IPC prediction errors for different predictors over a segment of bzip2_graph execution.

There is no guarantee that any of the Linear Predictors can outperform the last value predictor. For example, for "ammp" data the last value predictor achieves the lowest accumulated predictive error. In Figure 6, we present the mean absolute prediction errors for the "ammp" IPC data using different model orders. Here, last value predictor has lower error than any of the Linear Prediction models. As such, the PLS predictor chooses the last value predictor for the entire run of the data. Finally, we plot the absolute prediction errors for the entire "applu" data in Figure 7. The PLS prediction error on the lowest panel demonstrates significant improvement in prediction performance compared to LP–5 and the last value predictor.

## 7.3 Impact of Mixture Size on PLS Predictor Performance

We mentioned that the performance of PLS predictor largely depends on the models used in the mixture pool. The larger the mixture pool gets, the better becomes the prediction results. Each benchmark can be described better with a Linear Predictor of a specific model order. For each benchmark, Linear Predictor of a specific model order may provide smallest prediction accuracy. In fact, for different phases of a benchmark, different model orders may provide best performance. In Figure 8, we provide mean percentage prediction errors averaged across all benchmarks for IPC with different mixture sizes. In the figure, PLS-1 is composed of two Linear Prediction models with orders 1, namely the last value predictor, and 2. Adding a second model alongside last value predictor immediately reduces the prediction error. PLS-3 contains models with order {1,2,5}. Using the past 5 samples, in addition to using the last

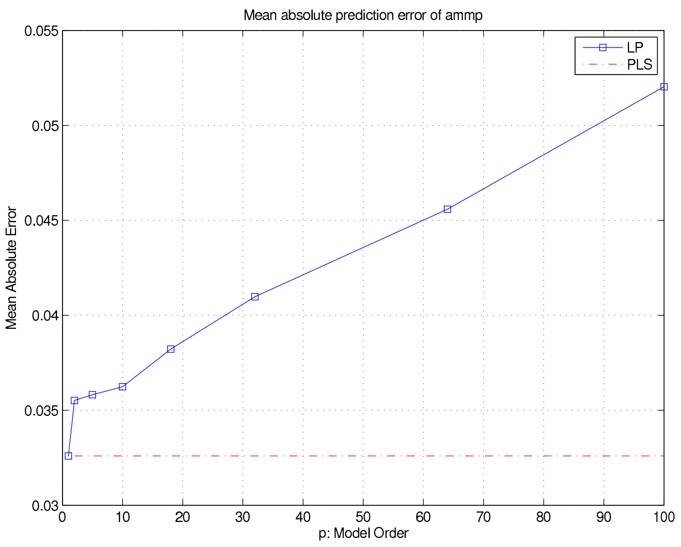

Figure 6: Mean absolute prediction error of ammp IPC data with different model orders for LP and PLS predictors.

two samples, to predict the next sample improves the prediction accuracy significantly as compared the last value predictor. Adding more models with larger model orders results in further reductions in prediction error. As expected the relative improvements with the addition of more models to the mixture become smaller. Linear Predictors with model order more than 100 are rarely used in practice. We could have used all 100 Linear Predictors instead of eight predictors, which are in PLS-7 mixture, but it would not bring significant improvement over PLS-7. Typically, models with larger model order are good in describing the fine structure in the segment of the data that is observed so far. However, this comes at the expense of poor generalization for the unseen future observations, which are to be predicted.

## 7.4 Summary of Results

We present prediction errors for 37 benchmarks using the last value predictor, LP–18, table based predictor and PLS predictor for IPC data sampled at 100ms in Figure 9. There are multiple ways of comparing different predictors. One crude way is just by counting for how many benchmarks one method outperformed the other. In such a comparison, we observe that LP–18 is better than the last value predictor in 17/37 cases, and last–value predictor is better in the remaining 20/37 cases. These ratios might suggest that LP–18 and last–value predictors have somewhat similar performance. Likewise, if we compare PLS-predictor with the last–value predictor, we see that in 27/37 cases PLS is better than the last–value predictor and in the remaining 10/37 cases they perform equally well. PLS predictor is better for 20/37 benchmarks as compared to table based predictor. Table based predictor is better

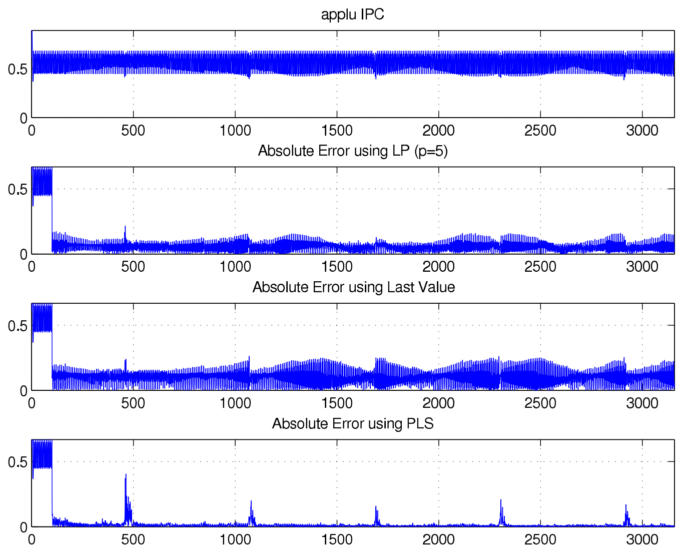

Figure 7: Absolute IPC prediction errors for different predictors over a segment of applu execution.

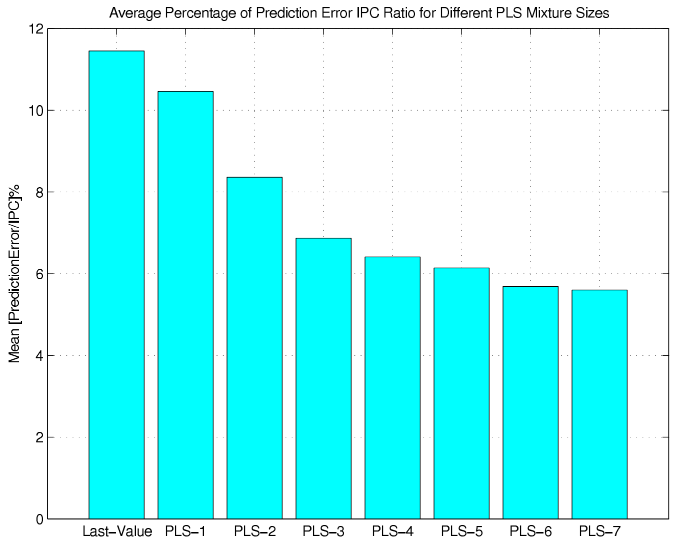

Figure 8: PLS performance on IPC for different mixture sizes. PLS-1={1, 2}, PLS-2={1, 2, 5}, PLS-3={1, 2, 5, 10}, PLS-4={1, 2, 5, 10, 18}, PLS-5={1, 2, 5, 10, 18, 32}, PLS-6={1, 2, 5, 10, 18, 32, 64}, PLS-7={1, 2, 5, 10, 18, 32, 64, 100}.

than PLS predictor on 6/37 benchmarks and they are equally good on 10/37 benchmarks. Even though, table based predictor is better than the last value predictor overall, unlike PLS predictor, for some benchmarks it provides significantly worse prediction accuracy than the last value predictor, as in the cases of *gzip_random, mcf_inp, perlbmkdiffmail* and *sixtrack_inp*.

It is important to have consistency in the predictor's performance. It is always preferable to perform better on as many benchmarks as possible, rather than performing very well on a few bencmarks and performing poorly on the rest of the benchmarks. For example, Table based predictor is slightly better than PLS predictor on *ammp, facerec, gzip_log, gzip_program* and *lucas*, but it is significantly worse than PLS predictor on *applu, bzip2_graph, bzip2_program,bzip2_source* and *mgrid*. In fact, the prediction error of table based predictor can become more than twice as large as that of the PLS predictor for some benchmarks. PLS predictor exhibits better consistency in its prediction accuracy than the table based predictor.

Even though Figure 9 is useful in giving an idea about the relative performances of the different predictors, a more accurate measure would be the normalized mean prediction error across all benchmarks. This measure is calculated in two steps. First, mean IPC, mean absolute prediction error and their ratios are calculated for each benchmark. Second, these ratios are averaged across all 37 benchmarks used in this paper. In Figure 10, we present the per predictor average [Prediction Error/IPC] ratios in percent for predictions performed at different time scales. This measure normalizes against the varying IPC amplitutes across different benchmarks and prevents the results of a particular benchmark dominating the overall results. We present prediction errors for the last value predictor, a set of Linear Predictors, table based predictor and PLS predictor for IPC data sampled at 100ms and 50ms in Figure 10. Using two different granularity level is important in order to see how models perform in modeling patterns at different time scales. In Figure 10 for 100ms, we see that the absolute prediction errors for the last value predictor is on average within 11.4% of the IPC values being predicted. The corresponding number for table based predictor is 7.7%. LP–18 performs the best among the linear predictors in which the prediction error is within 7.2% of the IPC values being predicted. Not surprisingly, the PLS–predictor achieves the best performance with 5.6%. This number also corresponds to a 50% reduction in prediction error of last value predictor. The PLS predictor achieves 5.2%, whereas the last–value predictor is on average within 8.6% of the IPC values being predicted with reduced time scales. As the time granularity of prediction and sampling of data reduces, the averaging effect (filtering effect) of longer time scales diminishes and last value predictor starts to improve relatively better in comparison to other predictors. The performance of table based predictor is about the same as that of LP–18 for both 100ms and 50ms data. PLS predictor is significantly better than the table based predictor.

In order to investigate how predictors perform on a different metric, we made predictions for the L1–miss rate. In Figure 11, we present the per predictor average [Pre-

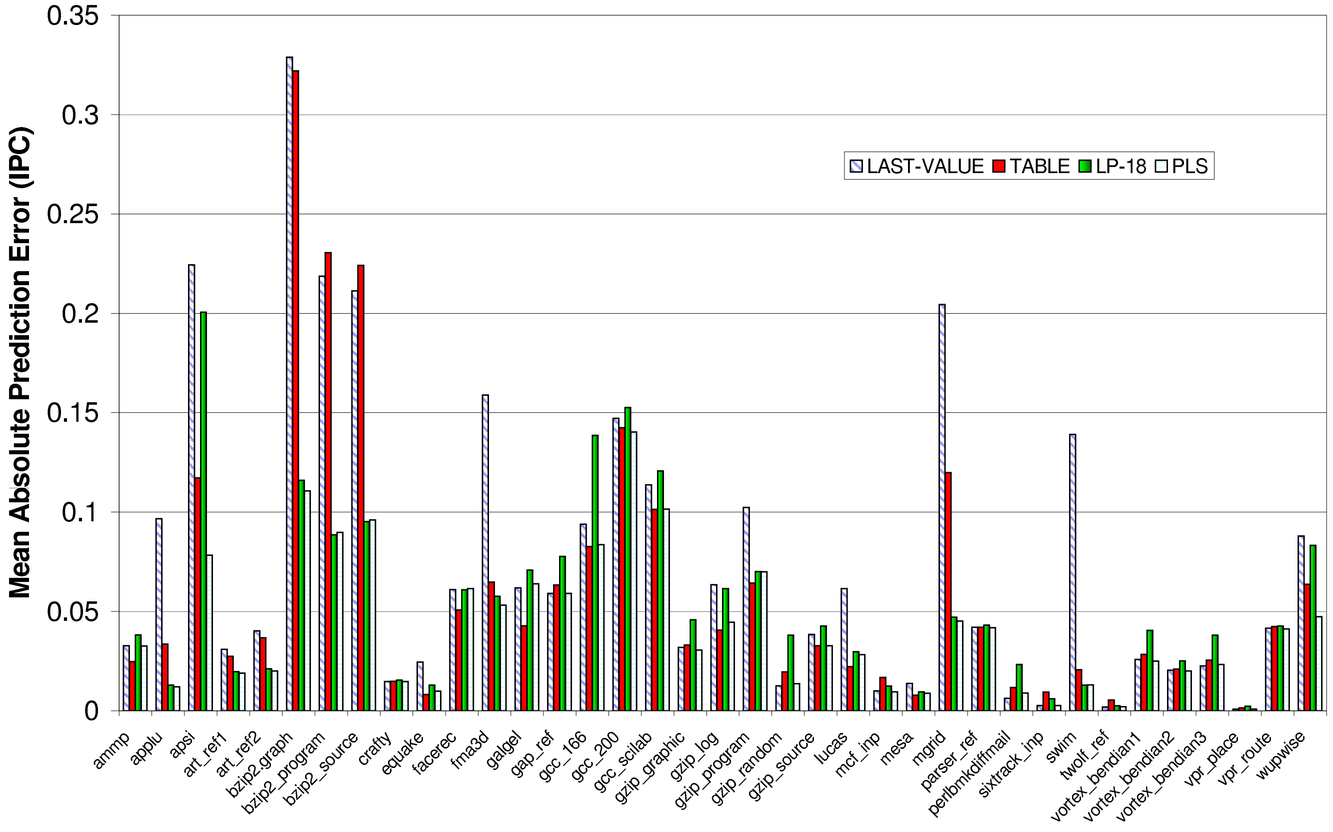

Figure 9: Mean absolute IPC prediction errors on SPECCPU 2000 suite.

diction Error/L1 miss rate] ratios in percent for predictions performed at different time scales. Figure 11 substantiates our claim that the PLS predictor is general in the sense that it can be applied to other metrics. In Figure 11, for 100ms, we see that the absolute prediction errors for the last value predictor is on average within 28% of the L1 miss rate values being predicted. The corresponding number for the table based predictor is 20%. Not surprisingly, the PLS predictor achieves the best performance with 16%.

## 8. CONCLUSIONS

We described a new metric prediction scheme called Unified Prediction Method (UPM). UPM is a parametric model based on stochastic framework and uses objective function minimization to estimate the parameters of the predictor. We implemented Linear Prediction and Predictive Least Squares (PLS) based predictor as special realizations of the UPM. The new predictors do not make any assumption about the distribution of metric being predicted. The PLS predictor can achieve the least accumulated squared prediction error while providing a consistent prediction across all benchmarks. We presented a series of experiments demonstrating superiority of PLS predictor as compared to the last value and table based predictors on the SPECCPU 2000 benchmark data.

## References

[1] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. AC-19, pp. 716-723, Dec. 1974.

[2] D. Albonesi, R. Balasubramonian, S. Dropsho, S. Dwarkadas, E. Friedman, M. Huang, V. Kursun, G. Magklis, M. Scott, G. Semeraro, P. Bose, A. Buyuktosunoglu, P. Cook and S. Schuster, "Dynamically tuning processor resources with adaptive processing," *IEEE Computer*, 36(12):43-51, 2003.

[3] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, S. Dwarkadas, "Memory Hierarchy Reconfiguration for Energy and Performance in General-Purpose Processor Architectures,", *International Symposium on Microarchitecture*, Dec. 2000.

[4] F. Bellosa, A. Weissel, M. Waitz, and S. Kellner, "Event-driven energy accounting for dynamic thermal management", *Workshop on Compilers and Operating Systems for Low Power*, Sept. 2003.

[5] K. Choi, R. Soma, and M. Pedram, "Dynamic voltage and frequency scaling based on workload decomposition", *International Symposium on Low Power Electronics and Design*, Aug. 2004.

[6] R. Sarikaya and A. Buyuktosunoglu, "Predicting program behavior based on objective function minimization," *IEEE International Symposium on Workload Characterization*, September 2007.

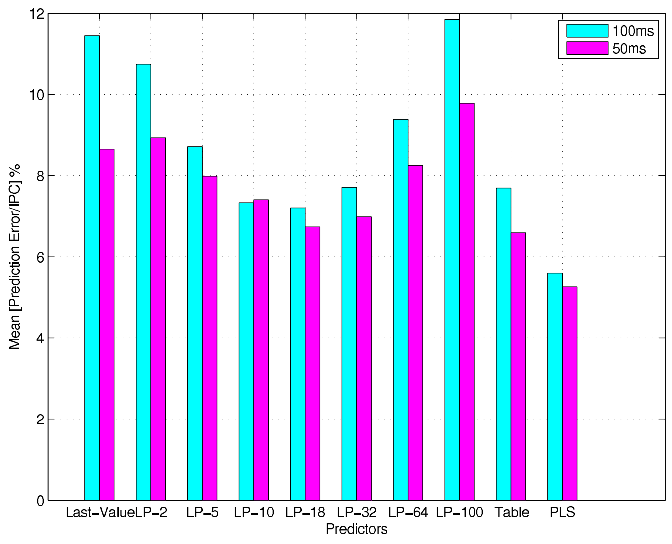

Figure 10: Mean [Prediction Error/IPC] ratios (%) across all benchmarks for different predictors with different time scales.

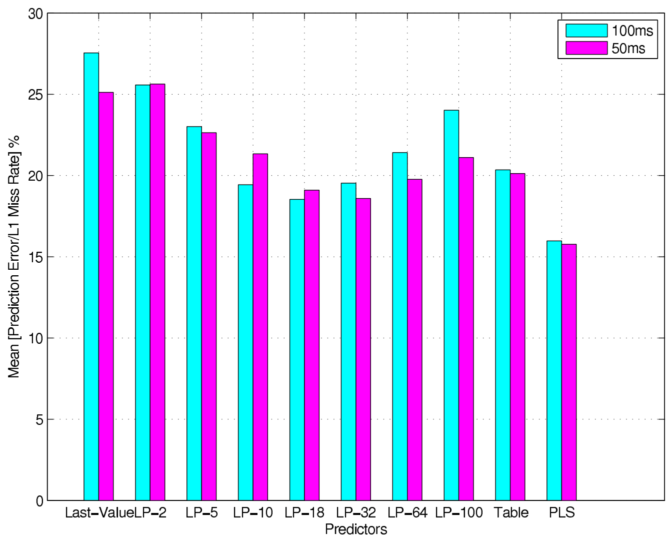

Figure 11: Mean [Prediction Error/L1 miss rate] ratios (%) across all benchmarks for different predictors with different time scales.

[7] A. Dhodapkar and J. Smith, “Managing multi-configurable hardware via dynamic working set analysis”, *In 29th Annual International Symposium on Computer Architecture*, June 2002.

[8] E. Duesterwald, C. Cascaval, and S. Dwarkadas, “Characterizing and predicting program behavior and its variability”, *In International Conference on Parallel Architectures and Compilation Techniques*, March 2003.

[9] M. Huang, J. Renau, and J. Torrellas, “Positional adaptation of processors: Application to energy reduction”, *International Symp. on Computer Architecture*, June 2003.

[10] C. Isci and M. Martonosi, “Identifying program power phase behavior using power vectors”, *IEEE International Workshop on Workload Characterization*, 2003.

[11] C. Isci and M. Martonosi, “Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data”, *36th International Symp. on Microarchitecture*, Dec. 2003.

[12] C. Isci, M. Martonosi, and A. Buyuktosunoglu. “Long-term Workload Phases: Duration Predictions and Applications to DVFS”, *IEEE Micro: Special Issue on Energy Efficient Design*, 25(5):39-51, Sep/Oct 2005.

[13] C. Isci, G. Contreras and M. Martonosi, “Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management”, *International Symposium on Microarchitecture*. Dec. 2006.

[14] D. T. L. Lee, M. Morf, and B. Friedlander, “Recursive least-squares ladder estimation algorithms, *IEEE Transactions on Circuits and Systems*, no. 6, pp. 467-481, June 1981.

[15] J. Makhoul, “Linear Prediction: A tutorial review”, *Proceedings IEEE*, 63(5):561-580, 1975.

[16] R.L. Plackett, “The discovery of the method of least squares”, *Biometrica*, 59:239-251, 1972.

[17] J. Rissanen, “Modeling by shortest data description”, *Automatica*, vol. 14, pp. 465-471, 1978.

[18] J. Rissanen, “A predictive Least-Squares Principle”, *IMA Journal of Mathematical Control & Information*, vol. 3, pp. 211-222, 1986.

[19] T-J. Shan, “On the predictive Least Squares Filtering,” *International Conference on Audio Speech and Signal Processing*, pp. 1312-1315, 1987.

[20] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, “Automatically characterizing large scale program behavior”, *In 10th International Conference Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[21] T. Sherwood, S. Sair, and B. Calder, “Phase tracking and prediction”, *28th International Symposium on Computer Architecture*, June 2003.

[22] J. M. Tendler, J. S. Dodson, S. Fields, H. Le and B. Sinharoy, “POWER4 system microarchitecture”, *IBM Journal of Res. and Dev.*, 46(1), 2002.

[23] A.Weissel and F. Bellosa, “Process cruise control: Event-driven clock scaling for dynamic power management”, *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, Grenoble, France, Aug. 2002.

[24] G. U. Yule, “On a method of investigating periodicities in disturbed series with special reference to Wolfer’s sunpot numbers”, *Phi. Trans. Royal Soc.*, 226-A:267-298, 1927.

[25] M. Wax, “Order Selection for AR Models by Predictive Least Squares”, *IEEE Transactions on Acoustic, Speech Signal Processing* vol. 36 no. 4, April 1988.

[26] T. Y. Yeh and Y. N. Patt, “Alternative implementations of two-level adaptive branch prediction”, *In 19th Annual International Symposium on Computer Architecture*, May 1992.