Adaptive Predictor Integration for System Performance Prediction

Jian Zhang and Renato J. Figueiredo
Advanced Computing and Information Systems (ACIS) Laboratory
Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611, USA
{jianzh, renato}@acis.ufl.edu

Abstract

*The integration of multiple predictors promises higher prediction accuracy than the accuracy that can be obtained with a single predictor. The challenge is how to select the best predictor at any given moment. Traditionally, multiple predictors are run in parallel and the one that generates the best result is selected for prediction. In this paper, we propose a novel approach for predictor integration based on the learning of historical predictions. It uses classification algorithms such as k-Nearest Neighbor (*k-NN*) based supervised learning to forecast the best predictor for the workload under study. Then only the forecasted best predictor is run for prediction. Our experimental results show that it achieved 20.18% higher best predictor forecasting accuracy than the cumulative MSE based predictor selection approach used in the popular Network Weather Service system. In addition, it outperformed the observed most accurate single predictor in the pool for 44.23% of the performance traces.*

1 Introduction

Grid computing [11] enables entities to create a Virtual Organization (VO) to share their computation resources such as CPU time, memory, network bandwidth, and disk bandwidth. Predicting the dynamic resource availability is critical to adaptive resource scheduling. However, determining the most appropriate resource prediction model a priori is difficult due to the multi-dimensionality and variability of system resource usage. First, the applications may exercise the use of different type of resources during their executions. Some resource usages such as CPU load may be relatively smoother whereas others such as network bandwidth are bustier. It is hard to find a single prediction model which works best for all types of resources. Second, different applications may have different resource usage patterns. The best prediction model for a specific resource of one machine may not wok best for another machine. Third, the resource performance fluctuates dynamically due to the contention created by competing applications. Indeed, in the absence of a perfect prediction model, the best predictor for any particular resource may change over time.

This paper introduces a Learning Aided Adaptive Resource Predictor (LARPredictor), which can dynamically choose the best prediction model suited to the workload at any given moment. By integrating the prediction results generated by the best predictor of each moment during the application run, the LARPredictor can outperform any single predictor in the pool. It differs from the traditional mix-of-expert resource prediction approach in that it does not require running multiple prediction models in parallel all the time to identify the best predictors. Instead, the Principal Component Analysis *(PCA)* and classification algorithm such as k-Nearest Neighbor *(k-NN)* are used to forecast the best prediction model from a pool based on the monitoring and learning of the historical resource availability and the corresponding prediction performance.

The LARPredictor is inspired by the VMPlant [19] project, which provides automated cloning and configuration of Virtual Machines *(VMs)*. The virtual machines are highly configurable in terms of hardware and software. It is possible to adapt the machine configurations to the changing workload to exploit better resource allocation. The learning aided adaptive resource performance prediction can be used to support dynamic VM provisioning by providing accurate prediction of the resource availability of the host server and the resource demand of the applications that are reflected by the hosting virtual machines.

Our experimental results based on the analysis of a set of virtual machine trace data show:

1. The best prediction model is workload specific. In the absence of a perfect prediction model, it is hard to find a single predictor which works best across virtual machines

1-4244-0910-1/07/$20.00

which have different resource usage patterns.

2. The best prediction model is resource specific. It is hard to find a single predictor which works best across different resource types.

3. The best prediction model for a specific type of resource of a given VM trace varies as a function of time. The LARPredictor can adapt the predictor selection to the change of the resource consumption pattern.

4. In the experiments with a set of trace data, The LARPredictor outperformed the observed single best predictor in the pool for 44.23% of the traces and outperformed the cumulative MSE based prediction model used in the Network Weather Service system (NWS) [30] for 66.67% of the traces. It has the potential to consistently outperform any single predictor for variable workloads and achieve 18.63% lower MSE than the model used in the NWS.

The rest of the paper is organized as follows: Section 2 gives an overview of related work. Section 3 briefly introduces virtual machine concepts and presents the prototype of virtual machine resource prediction. Section 4 describes the linear time series prediction models used to construct the LARPredictor and Section 5 describes the learning techniques used for predictor selection. Section 6 details the work flow of the learning aided adaptive resource predictor. Section 7 discusses the experimental results. Section 8 summarizes the work and describes future direction.

2 Related Work

Time series analysis has been studied in many areas such as financial forecasting [22], biomedical signal processing [21], and geoscience [5]. In this work, we focus on the time series modeling for computer resource performance prediction.

In [6] and [7], Dinda et al. conducted extensive study of the statistical properties and the predictions of host load. Their work indicates that CPU load is strongly correlated over time, which implies that history-based load prediction schemes are feasible. They evaluated the predictive power of a set of linear models including autoregression (AR), moving average (MA), autoregression integrated moving average (ARIMA), autoregression fractionally integrated moving average (ARFIMA), and window-mean models. Their results show that the AR model is the best in terms of high prediction accuracy and low overhead among the models they studied. Based on their conclusion, the AR model is included in our predictor pool to leverage its performance.

To improve the prediction accuracy, various adaptive techniques have been exploited by the research community. In [32], Yang et al. developed a tendency-based prediction model that predicts the next value according to the tendency of the time series change. Some increment/decrement value are added/subtracted to the current measurement based on the current measurement and some other dynamic information to predict the next value. Zhang et al. improved the performance of tendency-based model by using a polynomial fitting method to generate predictions based on the data several steps backward [35]. In addition, in [20], Liang et. al proposed a multi-resource prediction model that uses both the autocorrelation of the CPU load and the cross correlation between the CPU load and free memory to achieve higher CPU load prediction accuracy. Vazhkudai et al. [27][28] used linear regression to predict the data transfer time from network bandwidth or disk throughput.

The Network Weather Service (NWS) [30] performs prediction of both network throughput and latency for host machines distributed with different geographic distances. Both the NWS and the LARPredictor use the mix-of-expert approach to select the best predictor at any given moment. However, they differ from each other in the way of best predictor selection. The prediction model used in the NWS system runs a set of predictors in parallel to track their prediction accuracies. A cumulative error measurement, Mean Square Error (MSE), is calculated for each predictor. The one that generates the lowest prediction error for the known measurements is chosen to make a forecast of future measurement values. Section 6 shows that the LARPredictor only uses parallel prediction during the training phase. In the testing phase, it uses the PCA and k-NN classifier to forecast the best predictor for the next value based on the learning of historical prediction performances. Only the forecasted best predictor is run to predict the next value.

The mix-of-expert approach has been applied to the text recognition and categorization area. The combination of multiple classifiers has been proved to be able to increase the recognition rate in difficult problems when compared with single classifier [15]. Different combination strategies such as weighted voting and probability-based voting and dimensionality reduction based on concept indexing are introduced in [16].

3 Virtual Machine Resource Prediction Overview

This section gives an overview of virtual machine resource prediction, including the virtual machine concepts in the context of resource prediction and the resource prediction prototype.

3.1 Virtual Machine

A "classic" virtual machine (VM) enables multiple independent and isolated operating systems to run on one physical machine, efficiently multiplexing system resources of the host machine [14]. It provides a secure and isolated environment for application execution [10]. Compared with

a physical machine, it is highly customizable in terms of hardware (such as CPU, memory, and disk space) and software (such as operating system and applications) and can be easily checkpointed and migrated to achieve host load balancing [3].

In a classic virtual machine, virtualization software, which is often referred to as the *Virtual Machine Monitor* (VMM), is placed between the underlying hardware (host) and conventional software (guest). The VMM manages both the hardware resources and the guest operating system (OS) and application programs compiled for that operating system. When the guest OS performs certain operations, such as a privileged instruction that directly involves the shared hardware resources, the operation is intercepted by the VMM, checked for correctness and performed by the VMM on behalf of the guest machine [25]. Thus monitoring of the guest VM's usage of host hardware resources can naturally be done at the VMM layer. When an application is scheduled to run on a dedicated virtual machine, which is called the *application-centric VM*, the VM guest's resource performance metrics, such as CPU load, memory usage, I/O activity, and network bandwidth utilization, collected by the VMM reflect the application resource usage and can be used to characterize the application workload [33][34]. In addition, the prediction of the VM resource usage reflects the future application resource demand.

Our learning based adaptive predictor can be generally used for the prediction of any time series including the VM resource performance. The prototype introduced in Section 3.2 can work with different types of virtual machines such as Xen [9] and VMware [1]. The prediction of the resource performance of VMs in a given time frame can be used to guide the dynamic VM provisioning in the VM management systems such as the VMPlant [19], the Virtual Workspace [18], and the Xenoserver [23].

3.2 Virtual Machine Resource Prediction Prototype

Our virtual machine resource prediction prototype, illustrated in Figure 1, models how the VM performance data are collected and used to predict the value for future time to support resource allocation decision-making.

A *performance monitoring agent* is installed in the VMM. In our implementation, VMware's ESX virtual machines are used to host the application execution and the *vmkusage* tool [29] of ESX is used to monitor and collect the performance data of the VM guests and host from the */proc* of the host server. The tool samples every minute, and updates its data every five minutes with an average of the one-minute statistics over the given five-minute interval. The collected data is stored in a Round Robin Database (RRD), the *performance database*. Table 1 shows the list of

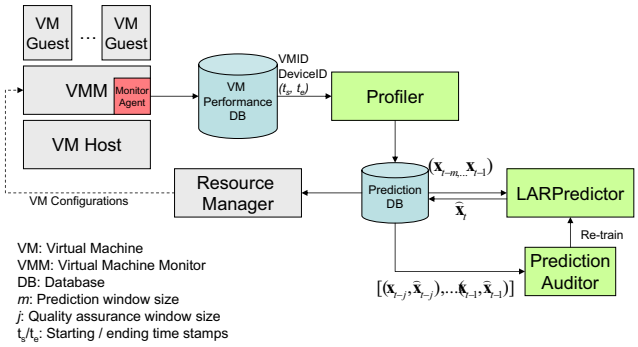

Figure 1. Virtual Machine Resource Usage Prediction Prototype

The *monitor agent*, which is installed in the *Virtual Machine Monitor (VMM)*, collects the VM resource performance data and stores them in the round robin *VM Performance Database*. The *profiler* extracts the performance data of a given time frame for the VM indicated by *VMID* and *deviceID*. The *LARPredictor* select the best prediction model based on learning of historical predictions, predicts the resource performance for time *t+1*, and stores the prediction results in the *prediction database*. The prediction results can be used to support the *resource manager* to perform dynamic VM resource allocation. The *Performance Quality Assuror* (QA) audits the LARPredictor's performance and orders re-training for the predictor if the performance drops below a predefined threshold.

performance features under study in this work.

The *profiler* retrieves the VM performance data, which are identified by *vmID*, *deviceID*, and a time window, from the RRD. The data of each VM device's performance metric form time series $(\mathbf{x}_{t-m+1}, \cdots, \mathbf{x}_t)$ with an identical interval, where m is the data retrieval window size. The retrieved performance data with the corresponding time stamps are stored in the *prediction database*. The [vmID, deviceID, timeStamp, metricName] forms the combinational primary key of the database.

The *LARPredictor* takes the time series performance data $(\mathbf{y}_{t-m}, \cdots, \mathbf{y}_{t-1})$ as inputs, selects the best prediction model based on the learning of historical prediction results, and predicts the resource performance $\hat{\mathbf{y}}_t$ of future time. The detail description of the LARPredictor's work flow is given in Section 6. The predicted results are stored in the *prediction DB* and can be used to support the resource manager's dynamic VM provisioning decision-making.

The *Prediction Quality Assuror (QA)* is responsible for monitoring the LARPredictor's performance in terms of MSE. It periodically audits the prediction performance by calculating the average MSE of historical prediction data stored in the prediction DB. When the average MSE of the audit window exceeds a predefined threshold, it directs the LARPredictor to re-train the predictors and the classifier using recent performance data stored in the database.

Performance Metrics	Description
CPU_Ready	The percentage of time that the virtual machine was ready but could not get scheduled to run on a physical CPU.
VCPUx	The percentage of physical CPU resources used by a virtual CPU. (ex.VCPU0)
Mem_Size	Current amount of memory in bytes the virtual machine has.
Mem_Swap	Amount of swap space in bytes used by the virtual machine.
Net_RX/TX	The number of packets and the MBytes per second that are transmitted and received by a NIC.
Disk_RD/WR	The number of I/Os and KBytes per second that are read from and written to the disk.

Table 1. Performance metric list

4 Time Series Models for Resource Performance Prediction

Time series is defined as an ordered sequence of values of a variable at equally spaced time intervals. A general linear process $\{Z_t\}$ is one that can be represented as a weighted linear combination of the present and past terms of a white noise process:

$$Z_t = a_t + \Psi_1 a_{t-1} + \Psi_2 a_{t-2} + \cdots \quad (1)$$

where $\{Z_t\}$ denotes the observed time series, $\{a_t\}$ denotes an unobserved white noise series, and $\{\Psi_i\}$ denotes the weights. In this paper, performance snapshots of virtual machine's resources including CPU, memory, disk, and network bandwidth are taken periodically to form the time series $\{Z_t\}$ under study.

Time series analysis accounts for the fact that those data points taken over time may have an internal structure (such as autocorrelation, trend, or seasonal variation) that should be accounted for. The purpose of time series analysis is generally two-fold: to understand or model the stochastic mechanism that gives rise to an observed series and to predict or forecast future values of a series based on the history of that series [4]. Time series analysis techniques have been widely applied to forecasting in many areas such as economic forecasting, sales forecasting, stock market analysis, communication traffic control, and workload projection. In this work, simple time series models, such as LAST, sliding-window average (SW_AVG), and autoregressive (AR), are used to construct the LARPredictor to support online prediction. However, the LARPredictor prototype may be generally used with other prediction models studied in [7][30][32].

LAST model: The LAST model predicts all future values to be the same as the last measured value:

$$Z_t = Z_{t-1} \quad (2)$$

SW_AVG model: The sliding-window average model predicts the future values by taking the average over a fixed-length history:

$$Z_t = \frac{1}{m} \sum_{i=t-m}^{t-1} Z_i \quad (3)$$

AR model: A p^{th}-order autoregressive process Z_t can be represented as follows:

$$Z_t = \Psi_1 Z_{t-1} + \Psi_2 Z_{t-2} + \cdots + \Psi_p Z_{t-p} + a_t \quad (4)$$

The current value of the series Z_t is a linear combination of the p latest past values of itself plus a term a_t, which incorporates everything new in the series at time t that is not explained by the past values. Yule-Walker technique is used in the AR model fitting in this work.

Generally, LAST performs better for smooth trace data and AR performs better for peaky data. In this paper, an approach to dynamically construct a resource predictor using multiple predictors such as LAST, AR, and SW_AVG is proposed to predict the VM resource performance.

The prediction performance is measured in *mean squared error* (MSE) [17], which is defined as the average squared difference between independent observations and predictions from the fitted equation for the corresponding values of the independent variables.

$$MSE(\hat{\boldsymbol{\theta}}) = E[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^2] \quad (5)$$

where $\hat{\boldsymbol{\theta}}$ is the estimator of a parameter $\boldsymbol{\theta}$ in a statistical model.

5 Algorithms for Prediction Model Selection

In the absence of a perfect generation model, the best resource prediction model varies with the machine workload. Learning algorithms are used to learn the relationship between the workload and suited prediction model. In this work, classification algorithms are used to forecast the best prediction model for a given workload based on the learning of historical predictions.

In this work, the k-NN classifier is used for best predictor selection. To reduce the dimension of the classification feature space, the Principal Component Analysis (PCA) technique is used. While we have chosen to use the k-NN algorithm due to its prior success in a large number of classification problems, such as handwritten digits and satellite image scenes, our methodology may be generally used with other types of classification algorithms.

5.1 k-Nearest Neighbor

The k-Nearest Neighbor (k-NN) classifier is *memory-based*. Its training data consist of the N pairs $(\mathbf{x}_1, p_1), \cdots, (\mathbf{x}_N, p_N)$ where p_i is a class label taking values in $1, 2, \cdots, P$. In this work, the P represents the number of prediction models in the pool. The training data are represented by a set of points in the feature space, where each point $\mathbf{x}_i$ is associated with its class label p_i. Classification of testing data $\mathbf{x}_j$ is made to the class of the closest training data. For example, given a test data $\mathbf{x}_j$, the k training data $\mathbf{x}_r, r = 1, \cdots, k$ closest in distance to $\mathbf{x}_j$ are identified. The test data is classified by using the majority vote among the k (an odd number) neighbors.

Since the features under study, such as CPU percentage and network received_bytes/sec, have different units of measure, all features are normalized to have zero mean and unit variance [26]. In this work, "closest" is determined by Euclidean distance:

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| . \tag{6}$$

The k-NN classifier can be applied to different time series without modification. To address the problem associated with high dimensionality, various dimension reduction techniques can be used in the data preprocessing.

5.2 Principal Component Analysis

The *Principal Component Analysis (PCA)* [8][26]is a linear transformation representing data in a least-square sense. The *principal components* of a set of data in $\Re^p$ provide a sequence of best linear approximations to those data, of all ranks $q \leq p$.

Denote the observations by $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$, and the parametric representation of the rank-q linear model is as follows:

$$f(\boldsymbol{\lambda}) = \boldsymbol{\mu} + \mathbf{V}_q \boldsymbol{\lambda}, \tag{7}$$

where $\boldsymbol{\mu}$ is a location vector in $\Re^p$, $\mathbf{V}_q$ is a $p \times q$ matrix with q orthogonal unit vectors as columns, which are called *eigenvectors*, and $\boldsymbol{\lambda}$ is a vector of q parameters, which are called *eigenvalues*. These eigenvectors are the principal components. The corresponding eigenvalues represent the contribution to the variance of data. Often there will be just a few $(= k)$ large eigenvalues and this implies that k is the inherent dimensionality of the subspace governing the data. When the k largest eigenvalues of q principal components are chosen to represent the data, the dimensionality of the data reduces from q to k.

In this work, the PCA is used to reduce the prediction input data dimensions. This helps to reduce the computing intensity of the subsequent classification process.

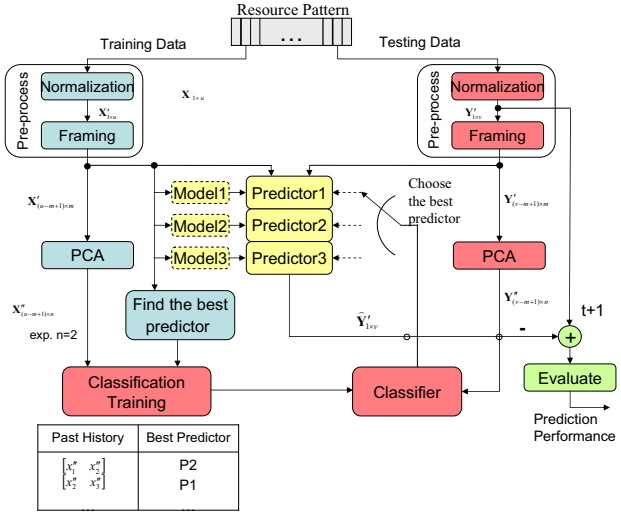

Figure 2. Learning Aided Adaptive Resource Predictor Workflow

The input data are *normalized* and *framed* with the prediction window size m. The *Principal Component Analysis (PCA)* is used to reduce the dimension of the input data from the window size m to $n(n < m)$. All prediction models are run in parallel in the training phase to identify the best predictor for each set of training data. The *classifier* is used to forecast the best predictor for the testing data based on the knowledge gained from the training data. Only the best predictor is used to predict the future value of the testing data.

6 Learning Aided Adaptive Resource Prediction

This section describes the work flow of the *Learning Aided Adaptive Resource Predictor (LARPredictor)* illustrated in Figure 2. The prediction consists of two phases: a training phase and a testing phase. During the training phase, the best predictors for each set of training data are identified using the traditional mix-of-expert approach. During the testing phase, the classifier forecasts the best predictor for the test data based on the knowledge gained from the training data and historical prediction performance. Then only the selected best predictor is run to predict the resource performance. Both phases include the data pre-processing and the Principal Component Analysis (PCA) process.

The features under study in this work, as shown in Table 1, include CPU, memory, network bandwidth, and disk I/O usages. Figure 3 illustrates how the features are processed to form the prediction database. Since the features have different units of measure, a data pre-processor was used to normalize the input data with zero mean and unit variance. The normalized data are framed according to the prediction window size to feed the PCA processor.

The PCA processor takes the pre-processed performance

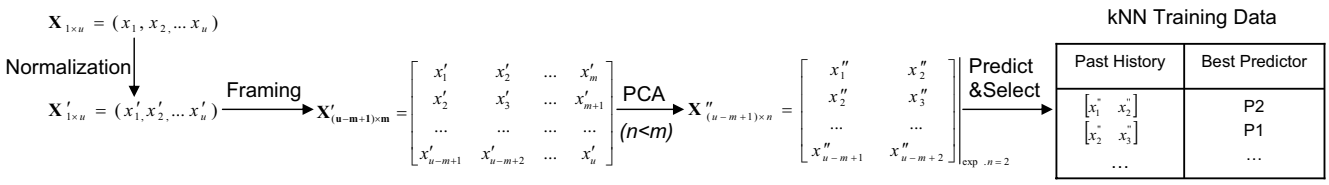

Figure 3. Learning Aided Adaptive Resource Predictor Dataflow

First, the u training data $X_{1\times u}$ is normalized to $X'_{1\times u}$ and subsequently framed to $X'_{(u-m+1)\times m}$ according to the predictor order m. The PCA processor is used to reduce the dimension of each set of training data from m to n before prediction. Then the predictors are run in parallel with the inputs $X''_{(u-m+1)\times n}$ and the one that gives the smallest MSE is identified as the best predictor to be associated with the corresponding training data in the prediction database. The dimension reduction of the testing data is similar to the training data's and is not shown here.

data as inputs and conducts the linear transformation of the data to select the principal components based on the predefined minimal fraction variance. For example, in the training phase, the PCA processor takes the pre-processed performance data $\mathbf{X}'_{(u-m+1)\times m}$ as inputs, where u is the total number of input data and m is the prediction window size. Then it conducts linear transformation of the input data and selects the first n ($n < m$) principal components to project the data into a n-dimensional feature space. In our implementation, the minimal fraction variance was set to extract exactly two principal components ($n = 2$). Therefore, at the end of process, the input data dimension gets reduced from the prediction window size m to n and the vector $\mathbf{X}''_{(u-m+1)\times n}$, is generated.

6.1 Training Phase

The training phase mainly consists of two processes: Prediction model fitting and best predictor identification. The training data with their corresponding best predictors are used for the k-NN classification in the testing phase.

The LAST and SW_AVG models do not involve any unknown parameters. They can be used for predictions directly. The parametric prediction models such as the AR model, which contain unknown parameters, require model fitting. The model fitting is a process to estimate the unknown parameters of the models. The Yule-Walker equation [4] is used in the AR model fitting in this work.

For window based prediction models, such as SW_AVG and AR, the PCA algorithm is applied to reduce the input data dimension. The naive mix-of-expert approach is used to identify the best predictor p_i for each set of pre-processed training data (exp.$(x'_i x'_{i+1} \ldots x'_{i+m-1})$). All prediction models are run in parallel with the training data and the one which generates the least MSE of prediction is identified as the best predictor p_i, which is a class label taking values in (LAST, AR, SW_AVG) to be associated with the training data. The u pairs of PCA-processed training data and the corresponding best predictors $[(x''_1, p_1), \cdots, (x''_u, p_u)]$ form the training data of the classifiers.

As a non-parametric classifier, the k-NN classifier does not have an obvious training phase. Its major task of this phase is to label the training data with class definitions.

6.2 Testing Phase

Similar to the training phase, the testing data are normalized using the normalization coefficient derived from the training phase and framed with the prediction window size m. Then the PCA is used to reduce the dimension of the preprocessed testing data $\left(y'_{t-m} y'_{t-m+1} \cdots y'_{t-1}\right)$ from m to n.

In the testing phase of k-NN classifier based LARPredictor, the Euclidean distances between all PCA–processed test data $\left(y''_{t-n}\ y''_{t-n+1}\ \ldots\ y''_{t-1}\right)$ and all training data $\mathbf{X}''_{(u-1+m)\times n}$ in the reduced n dimensional feature space are calculated and the k ($k = 3$ in our implementation) training data which have the shortest distances to the testing data are identified. The majority vote of the k nearest neighbors' best predictor will be chosen as the best predictor to predict $\hat{y'}_t$ based on the $(y'_{t-m}, y'_{t-m+1}, \cdots, y'_{t-1})$ in case of the AR model or the SW_AVG model and $\hat{y'}_t = y'_{t-1}$ in case of the LAST model. The prediction performance can be obtained by comparing the predicted value $\hat{y'}_t$ with the normalized observed value y'_t.

The testing phase differs from the training phase in that it does not require running multiple predictors in parallel to identify the one which is best suited to the data and gives the smallest MSE. Instead, it forecasts the best predictor by learning from historical predictions. The reasoning here is that these nearest neighbors' workload characteristics are closest to the testing data's and the predictor that works best for these neighbors should also work best for the testing data.

7 Empirical Evaluation

We have implemented a prototype for the LARPredictor including Perl and Shell scripts of the profiler to extract and profile the performance data from the round robin

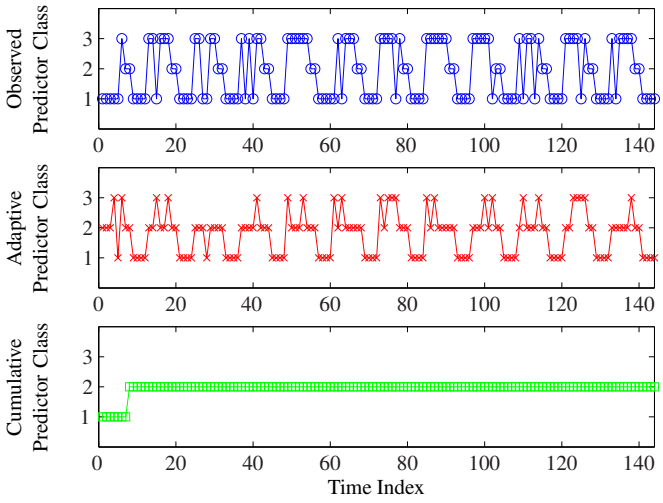

Figure 4. Best Predictor Selection for Trace VM2_load15
Predictor Class: 1 - LAST, 2 - AR, 3 - SW_AVG

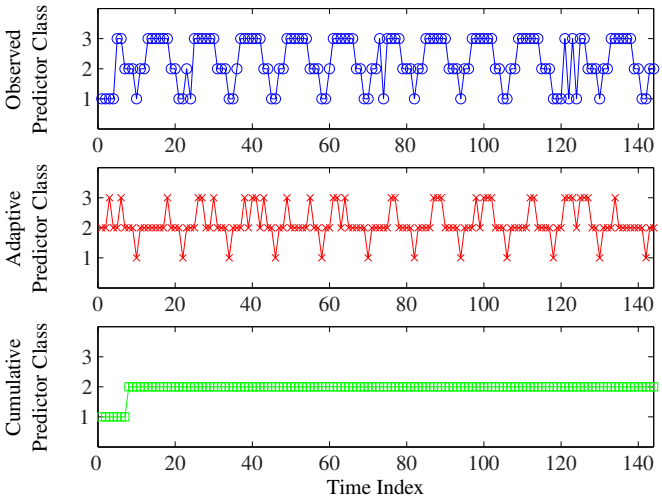

Figure 5. Best Predictor Selection for Trace VM2_PktIn
Predictor Class: 1 - LAST, 2 - AR, 3 - SW_AVG

performance database, and a Matlab implementation of the LARPredictor. This section evaluates the prediction performance of the LARPredictor using traces of five virtual machines as follows:

VM1: Hosts a web server, Globus GRAM/MDS and GridFTP services, and a PBS head node.

VM2: Hosts a Linux-based port-forwarding proxy for VNC sessions.

VM3: Hosts a WindowsXP based calendar.

VM4: Hosts a web server, a list server, and Wiki server.

VM5: Hosts a web server.

These virtual machines were hosted by a physical machine with an Intel(R) Xeon(TM) 2.0GHz CPU, 4GB memory, and 36GB SCSI disk. VMware ESX server 2.5.2 was running on the physical host. The *vmkusage* tool was run on the ESX server to collect the resource performance data of the guest virtual machines every minute and store them in a round robin database. The profiler was used to extract the data with given VMID, DeviceID, performance metric, starting and ending time stamps, and intervals. In this experiment, the performance data of a 24-hour period with 5-minute intervals were extracted for VM2, VM3, VM4, and VM5. The data of a 7-day period with 30-minute intervals of VM1 were extracted. During the 7-day period, total 310 jobs were executed varying with a mix of 93.55% short running jobs (1-2 seconds), 3.87% medium running jobs (2-10 minutes), and 2.58%long running jobs (45-50 minutes). The data of a given VMID, DeviceID, and performance metrics form a time series under study. The time series data were normalized with zero mean and unit variance.

7.1 Best Predictor Selection

This set of experiments illustrates the adaptive predictor selection of the LARPredictor. The k-NN classifier was used to forecast the best predictor among the LAST, AR, and SW-AVG for the workload understudy. Only the selected best predictor is used for performance prediction. VM2 was used in the experiments. Figure 4 shows the predictor selections for CPU fifteen minute load average during a 12 hour period with a sampling interval of 5 minutes. The top plot shows the observed best predictor by running three prediction models in parallel. The middle plot shows the predictor selection of the LARPredictor and the bottom plot shows the cumulative MSE based predictor selection used in the NWS. Similarly the predictor selection results of the trace data of network packets in per second is shown in Figure 5.

These experimental results show that the best prediction model for a specific type of resource of a given trace varies as a function of time. In the experiment, the LARPredictor can better adapt the predictor selection to the changing workload than the cumulative MSE based approach presented in the NWS. The LARPredictor's average best predictor forecasting accuracy of all the performance traces of the five virtual machines is 55.98%, which is 20.18% higher than the accuracy achieved by the cumulative MSE based predictor used in the NWS for the workload studied.

7.2 VM Performance Trace Prediction

This set of experiments is used to check the prediction performance of the LARPredictor. Section 7.2.1 shows the

Perf.Metrics	Predictors				
	P-LAR	LAR	LAST	AR	SW
CPU_usedsec	0.6976	0.9508	1.1436	***0.9456***	1.0352
CPU_ready	0.6775	0.9632	1.1699	***0.9579***	1.0333
Memory_size	0.2071	0.2389	***0.2298***	0.2379	0.4883
Memory_swapped	0.2071	0.2386	***0.2298***	0.2379	0.4883
NIC1_received	0.3981	***0.5436***	1.836	***0.5436***	0.9831
NIC1_transmitted	0.3776	***0.5845***	1.8236	***0.5845***	0.9829
NIC2_received	0.9788	***0.9912***	1.4392	0.9966	1.0397
NIC2_transmitted	0.3983	***0.5463***	1.8406	***0.5463***	0.9843
VD1_read	0.9062	1.0215	1.2849	***0.9754***	1.0511
VD1_write	0.7969	0.9587	1.1905	***0.9473***	1.0566
VD2_read	1	1.2156	1.4191	1.1536	***1.035***
VD2_write	0.662	0.9931	1.1572	***0.9929***	1.0292

Table 2. Normalized Prediction MSE Statistics for Resources of VM1

duration = 168 hours, interval = 30 minutes, prediction order = 16

Perform. Metrics	VM1	VM2	VM3	VM4	VM5
CPU_usedsec	AR	AR	AR	AR*	AR*
CPU_ready	AR	AR*	AR*	AR*	AR
Memory_size	LAST	AR*	AR*	LAST	AR*
Memory_swapped	LAST	AR*	NaN	LAST	AR*
NIC1_received	AR*	AR	AR*	AR	NaN
NIC1_transmitted	AR*	AR*	AR*	AR	NaN
NIC2_received	AR*	LAST	NaN	AR	SW_AVG
NIC2_transmitted	AR*	AR*	NaN	AR*	AR
VD1_read	AR	AR	NaN	AR	SW_AVG
VD1_write	AR	AR	NaN	SW_AVG*	AR
VD2_read	SW_AVG	AR	AR	AR*	NaN
VD2_write	AR	AR	AR*	AR*	AR

Table 3. Best Predictors of All the Trace Data

The predictors shown in the table have the smallest MSE among all the three predictors (LAST, AR, and SW_AVG). The "*" symbol indicates that the LARPredictor outperforms the best predictor in the predictor pool.

prediction accuracy of the k-NN based LARPredictor and all the predictors in the pool. Section 7.2.2 benchmarks the performance of the LARPredictors and the cumulative MSE based prediction model used in the NWS.

In the experiments, ten-fold cross validation were performed for each set of time series data. A time stamp was randomly chosen to divide the performance data of a virtual machine into two parts: 50% of the data was used to train the LARPredictor and the other 50% was used as test set to measure the prediction performance by calculating its prediction MSE.

7.2.1 Performance of k-NN based LARPredictor

The k-NN algorithm was used for classification in this experiment. In the training phase, the training data were used to derive the regression coefficients of the AR model. In addition, the three prediction models were run in parallel. The prediction error was calculated by comparing the predicted value with the observed value. For each prediction, the model that gave the smallest absolute value of the error was identified as the best predictor to be associated with the corresponding training data.

In the testing phase, the 3NN classifier was used to forecast the best predictors of the testing data. First, for each set of testing data of the prediction window size, the PCA was applied to reduce the data dimension from m, which was 5 or 16, to $n = 2$ in this experiment. Then the Euclidean distances between the test data and all the training data in the reduced feature space were calculated. The three training data which had the shortest distances to the testing data were identified and the majority vote of their associated best predictors was forecasted to be the best predictor of the testing data. At last, the forecasted best predictor was run to predict the future value of the testing data. The MSE of each time series was calculated to measure the performance of the LARPredictor. Table 2, shows the a sample (VM1) of the prediction performance of the LARPredictor with current implementation (LAR) and the three prediction models including LAST, AR, and SW_AVG (SW) for all resource performance traces of the five virtual machines. Also shown in these tables is the computed MSE for a perfect LARPredictor (P-LAR). The MSE of the P-LAR model shows the upper bound of the prediction accuracy that can be achieved by the LARPredictor. The MSE of the best predictor among LAR, LAST, AR, and SW_AVG is highlighted with ***italic bold*** numbers.

Table 3 shows the best predictor among LAST, AR, and SW_AVG for all the resource performance metrics and VM traces. The symbol "*" indicates the cases in which the LARPredictor achieved equal or higher prediction accuracy than the best of the three predictors. Overall, the AR model performed better than the LAST and the SW_AVG models.

The above experimental results show:

1. It is hard to find a single prediction model among LAST, AR, and SW_AVG that performs best for all types of resource performance data for a given VM trace. For example, for the VM1's trace data shown in Table 2, each of the three models (LAST, AR, and SW) outperformed the other two for a subset of the performance metrics.

2. It is hard to find a single prediction model that perform best consistently for a given type of resources across all the VM traces. In the experiment, only the AR model worked best for the predictions of CPU traces.

3. The LARPredictor achieved better-than-expert performances using the mix-of-expert approach for 44.23% of the workload traces. It shows the potential for the LARPredictor to outperform any single predictor in the pool and approach the prediction accuracy of the P-LAR by improving the best predictor forecasting / classification accuracy. How to further improve the predictor classification accuracy is a topic of our future research.

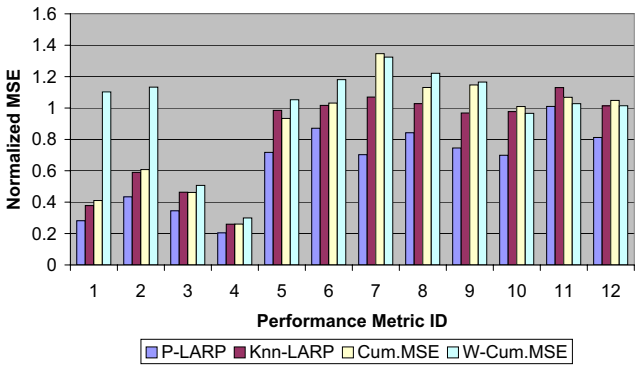

Figure 6. Predictor Performance Comparison (VM4)

1 - CPU_usedsec, 2 - CPU_ready, 3 - Mem_size, 4 - Mem_swap,
5 - NIC1_rx, 6 - NIC1_tx, 7 - NIC2_rx, 8 - NIC2_tx,
9 - VD1_read, 10 - VD1_write, 11 - VD2_read, 12 - VD2_write

7.2.2 Performance Comparison of the LARPredictors and the Cumulative MSE based Predictor

This section compares the prediction accuracy of the LARPredictors and the NWS predictor. Figure 6, shows an example (VM4) of the prediction accuracy of the perfect LARPredictor that has 100% best predictor forecasting accuracy (P-LARP), the k-NN based LARPredictor (Knn-LARP), the cumulative MSE of all history based predictor used in the NWS (Cum.MSE), and the cumulative MSE of a fixed window size (n=2 in this experiment) based predictor used in the NWS (W-Cum.MSE).

The experimental results show that without running all the predictors in parallel all the time, for 66.67% of the traces, the LARPredictor outperformed the cumulative MSE based predictor used in the NWS. The perfect LARPredictor shows the potential to achieve 18.6% lower MSE in average that the cumulative MSE based predictor.

7.3 Discussion

PCA is an optimal way to project data in the mean-square sense. The computational complexity of estimating the PCA is $O(d^2W) + O(d^3)$ for the original set of $W \times d$-dimensional data [2]. In the context of resource performance time series prediction, $W = 1$ and d is the prediction window size. The typical small input data size in this context makes the use of the PCA feasible. There also exist computationally less expensive methods [24] for finding only a few eigenvectors and eigenvalues of a large matrix; in our experiments, we use appropriate Matlab routines to realize these.

The k-NN does not have an off-line learning phase. Its "training phase" is simply to index the N training data for later use. Therefore, its training complexity is $O(N)$ both in time and space. In the testing phase, the k nearest neighbors of a testing data can be obtained $O(N)$ time by using a modified version of *quicksort* [31]. There are fast algorithms for finding nearest-neighbors [12][13] also.

Three simple time series models were used in this experiment to show the potential of using dynamic predictor selection based on learning to improve prediction accuracy. However, the LARPredictor prototype may be generally used with other more sophisticated prediction models such as these studied in [7][30][32]. Generally, the more predictors in the pool and the more complex the predictors are, it is more beneficial to use the LARPredictor because the classification overhead can be better amortized by running only single predictor at any given time.

8 Summary

The best prediction model varies with the types of resources and workload from time to time. We have developed a time series resource prediction model, LARPredictor, which can adapt the predictor selection to the changing workload. The k-NN classifieris used to forecast the best predictor for the workload based on the learning of historical load characteristics and prediction performance. The principal component analysis technique has been applied to reduce the input data dimension of the classification process. Our experimental results with the traces of the full range of virtual machine resources including CPU, memory, network and disk show that the LARPredictor can effectively identify the best predictor for the workload and achieve prediction accuracies that are close to or even better than any single best predictor.

Different predictors tend to work best for workloads with different characteristics. We plan to incorporate more prediction models such as those presented in [7][32] into the predictor pool to leverage their prediction power for different type of workload and develop a quantitative method to access the LARPredictor's applicability to time series predictions in other areas. We are also working on how to improve the best predictor forecasting accuracy to further improve the resource prediction performance. In addition, we plan to study the relationship between the computing complexity and the prediction performance.

Acknowledgement

This work was supported in part by a grant from Intel Corporation ISTG R&D Council and the National Science Foundation under grants: EIA-0224442, EEC-0228390, ACI-0219925, ANI-0301108, SCI-0438246 and SCI-0537455. Any opinions, findings and conclusions or recommendations expressed in this material are those of the

authors and do not necessarily reflect the views of Intel and NSF.

References

[1] http://www.vmware.com/pdf/esx25_admin.pdf.

[2] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, pages 245–250, 2001.

[3] C. Clark, K. Fraser, S. Hand, J. Hanseny, E. July, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. of NSDI'05*, Boston, MA, 2005.

[4] J. D. Cryer. *Time series analysis*. Duxbury Press, Boston, MA, 1986.

[5] K. Didan and A. Huete. Analysis of the global vegetation dynamic metrics using modis vegetation index and land cover products. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'04)*, volume 3, pages 2058–2061, 2004.

[6] P. Dinda. The statistical properties of host load. *Scientific Programming*, (7:3-4), 1999.

[7] P. Dinda. Host load prediction using linear models. *Cluster Computing*, 3(4), 2000.

[8] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, New York, NY, 2001. 2nd edition.

[9] P. B. et. al. Xen and the art of virtualization. In *Proc. of the 19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, USA, 2003.

[10] R. Figueiredo, P. Dinda, and J. Fortes. A case for grid computing on virtual machines. In *Proc. of 23rd ICDCS*, pages 550–559, May 19–22, 2003.

[11] I. Foster. The anatomy of the grid: enabling scalable virtual organizations. In *Proc. of First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 6–7, 2001.

[12] F. Friedman, J.H. Baskett and L. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, C-24(10):1000–1006, Oct. 1975.

[13] J. Friedman, J.H. Bentley and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

[14] R. P. Goldberg. Survey of virtual machine research. *IEEE Computer Magazine*, 7(6):34–45, June 1974.

[15] S. Gunter and H. Bunke. An evaluation of ensemble methods in handwritten word recognition based on feature selection. In *Proc. of ICPR-17*, volume 1, pages 388–392, Aug. 2004.

[16] G. Jain, A. Ginwala, and Y. Aslandogan. An approach to text classification using dimensionality reduction and combination of classifiers. In *Proc. of IRI*, pages 564–569, Nov. 2004.

[17] S. G. John O.Rawlings and D. A.Dickey. *Applied Regression Analysis*. Springer, 2001.

[18] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming Journal*, 13(4):265–276, 2005.

[19] I. Krsul, A. Ganguly, J. Zhang, J. Fortes, and R. Figueiredo. Vmplants: Providing and managing virtual machine execution environments for grid computing. In *Proc. of SC'04*, Washington, DC, Nov. 6–12, 2004.

[20] J. Liang, K. Nahrstedt, and Y. Zhou. Adaptive multi-resource prediction in distributed resource sharing environment. In *IEEE International Symposium on Cluster Computing and the Grid*, pages 293–300, 2004.

[21] P. Magni and R. Bellazzi. A stochastic model to assess the variability of blood glucose time series in diabetic patients self-monitoring. *IEEE Trans. Biomed. Eng.*, 53(6):977–985, 2006.

[22] I. Matsuba, H. Suyari, S. Weon, and D. Sato. Practical chaos time series analysis with financial applications. In *Proc. of 5th International Conference on Signal Processing*, volume 1, pages 265–271, Beijing, 2000.

[23] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: accountable execution of untrusted programs. In *Proc. of the Seventh Workshop on Hot Topics in Operating Systems*, pages 136–141, Rio Rico, AZ, 1999.

[24] L. Sirovich and R. Everson. Management and analysis of large scientific datasets. *Int. Journal of Supercomputer Applications*, 6(1):50–68, 1992.

[25] J. Smith and R. Nair. *Virtual Machines*. Morgan Kaufmann, June 2005.

[26] R. T. Trevor Hastie and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[27] S. Vazhkudai and J. Schopf. Predicting sporadic grid data transfers. *Proc. of HPDC*, pages 188–196, 2002.

[28] S. Vazhkudai, J. Schopf, and I. Foster. Using disk throughput data in predictions of end-to-end grid data transfers. In *In the 3rd International Workshop on Grid Computing*, Nov. 2002.

[29] V. white paper. Comparing the mui, virtualcenter, and vmkusage.

[30] R. Wolski. Dynamically forecasting network performance using the network weather service. In *Journal of cluster computing*, 1998.

[31] J. Yang, Y. Zhang and B. Kisiel. A scalability analysis of classifiers in text categorization. In *ACM SIGIR'03*, pages 96–103, 2003.

[32] L. Yang, J. M. Schopf, and I. Foster. Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environments. In *Proc. of SC'03*, page 31, Nov. 15-21, 2003.

[33] J. Zhang and R. Figueiredo. Application classification through monitoring and learning of resource consumption patterns. In *Proc. of IPDPS'06*, Rhodes Island, Greece, Apr. 25–29, 2006.

[34] J. Zhang and R. Figueiredo. Autonomic feature selection for application classification. In *Proc. of ICAC '06*, pages 43–52, 2006.

[35] Y. Zhang, W. Sun, and Y. Inoguchi. CPU load predictions on the computational grid *. In *Proc. of CCGRID'06*, volume 1, pages 321–326, May 2006.