A Resource Prediction Model for Virtualization Servers

Sayanta MALLICK Gaétan HAINS

December 2011
TR–LACL–2011–3

Laboratoire d’Algorithmique, Complexité et Logique (LACL)
Université de Paris–Est Créteil (UPEC), Faculté des Science et Technologie
61, Avenue du Général de Gaulle, 94010 Créteil cedex, France
Tel: (+33)(1) 45 17 16 47, Fax: (+33)(1) 45 17 66 01
Laboratory for Algorithmics, Complexity and Logic (LACL)
University of Paris-East Créteil (UPEC)


S. MALLICK, G. HAINS
A Resource Prediction Model for Virtualization Servers

© S. Mallick, G. Hains, SOMONE, December 2011
A Resource Prediction Model for Virtualization Servers

Sayanta Mallick\textsuperscript{1,2}  \quad Gaëtan Hains\textsuperscript{1}

\textsuperscript{1} LACL, Université Paris-Est
61, Avenue du Général de Gaulle, 94010 Créteil, France

\textsuperscript{2} SOMONE
Cité Descartes, 9, rue Albert Einstein 77420 Champs sur Marne France

smallick@somone.fr \quad gaetan.hains@u-pec.fr

Abstract

Monitoring and predicting resource consumption is a fundamental need when running a virtualized system. Predicting resources is necessary because cloud infrastructures use virtual resources on demand. Current monitoring tools are insufficient to predict resources usage of virtualized system. So without proper monitoring, virtualized systems can have down time, which can directly affect cloud infrastructure.

We propose a new modelling approach to the problem of resource prediction. Models are based on historical data to forecast short-term resource usages. We present here in detail our three prediction models to forecast and monitor resources. We also show experimental results by using real-life data and an overall evaluation of this approach.
Contents

1 Introduction 5

2 Background and State of the Art 6
   2.1 Methodologies Used in Published Papers 6
   2.2 Markov Chain Based Model 6
   2.3 Service Level Objectives and Monitoring Data Analysis 7
   2.4 Anomaly Detection and Performance Monitoring 8

3 Markov Chains 9
   3.1 Stochastic Matrix 9
   3.2 Distribution Vector of Markov Chain 9
      3.2.1 Stationary Distributions for Markov Chains 9
   3.3 Criteria for "Good" Markov chains 9

4 Modelling CPU Usage with State Distribution Model 9
   4.1 Defining the States of State Distribution Model 10
   4.2 Computing State Distribution 10

5 Modelling CPU Usage with Maximum Frequent State 10

6 Modelling the CPU Usage with Markov Chain 10
   6.1 Defining the States of the System 11
   6.2 Computing the Transition Probability 11
   6.3 Building the Transition Probability Matrix 11
   6.4 Markov Chain Stochastic Matrix in Virtualized CPU Usage 12
   6.5 State Transition Diagram of CPU Resource Utilization 12
   6.6 Computing the Distribution Vector of the System 13

7 Calculating Prediction Error of Probability Distribution 13

8 Observation from Data and Experiments 14
   8.1 Experimental Results 14
   8.2 Data Sources 14
   8.3 Experimental Setup 14
   8.4 Results 15
      8.4.1 Quality of Predictions 21
      8.4.2 Alert Forecasts 22

9 Conclusion and Future Work 23

10 Acknowledgement 24

References 24
1 Introduction

This research focuses on the analysis of infrastructure resource log data and possible resource prediction associated to virtual infrastructure. The main problem discussed here is associated in many mission critical systems that uses virtual and cloud infrastructure.

Our resource prediction system is able to foresee resource bottleneck problems for the near future and sometimes more. A system administrator or VM manager using it would be able to react promptly and apply required corrections measure to the system.

We have collected resource log data recorded from virtual infrastructure usages at SOMONE. The data contains various monitoring metrics related to utilization of CPU, memory, disk and network.

The monitoring metrics values are collected averaged in 1 minute, 10 minute, 1 hour, 1 day time intervals from resource log data. These data are spread over hourly, daily, weekly, monthly and yearly time spans. These data are analysed for predicting resource usage. They are input to our prediction system.

Resource and capacity bottleneck in the virtual infrastructure can be happen due to several reasons including:

1. virtual resource vs physical resource utilisation.
2. offered load (an exceptionally high load is offered to the system).
3. human errors (e.g., configuration errors in set-up of capacity values).

Some capacity indicator values can evaluate directly a property. This property helps in resource monitoring. For example physical disk have certain capacity measurement proportional to virtual disk capacity measurement. The measurement of the physical disk gives directly information about whether a virtual disk should be added or removed in order to prevent system faults. In some cases we may not be able to measure directly the changes of states or it would be difficult to measure or to set-up a correct capacity value (e.g. system configuration settings). In that case we would use a prediction system to predict the system usage of virtual infrastructure.

In this research study we have done off-line predictions based on pre-recorded resource usage log. In reality the main objective is to come up with a real-time capacity prediction. To adapt to the real-time situation additional requirement and constrains would be added on the off-line prediction algorithm. For example, in on-line situation higher requirements on the performance of the prediction algorithm and on the amount of historical data available for the algorithm would exist. However, these issues are out of our scope in this preliminary research study.

In addition to the prediction of usages from real data, we have interviewed system monitoring experts that are working on the virtual infrastructure monitoring on a daily basis. Based on those discussions we have developed our practical use cases, for which our proposed algorithms and prototype could provide practical solutions.

Our focus here is on virtual resource prediction and monitoring from usage log data. However our work is related to IT infrastructure monitoring, telecom monitoring (various types of network, alerts, events handling prediction of alerts interfaces and networks elements), performance modelling, stochastic modelling and computer security (intrusion detection and prevention systems).
2 Background and State of the Art

In this section we explore some of the scientific approaches on prediction, monitoring and analysis so far used on server log data. We will provide a snapshot of scientific methodologies used in prediction, forecasting from usage log data. We discuss research methodologies in server monitoring, virtual infrastructure monitoring, prediction and performance modelling.

2.1 Methodologies Used in Published Papers

System-level virtualization is increasing popular in current data center technologies. It enables dynamic resource sharing while providing an almost strict partitioning of physical resources across several applications running on a single physical host. However, running a virtualized data center at high efficiency and performance is not an easy task. Inside the physical machine, the virtual machine monitor needs to control the allocation of physical resources to individual VMs. Different publish papers address the issues of modelling virtual capacity and resources. In early works Markov chains have been used in predicting software performance and testing[11]. In statical software testing the black box model has two important extensions. In a first step, sequences of defects in performance are stochastically generated based on a probability distribution. Then the probability distribution represents a profile of actual or anticipated use of the software. Second, a statistical analysis is performed on the test history. This enables the measurement of various probabilistic aspects of the testing process.

2.2 Markov Chain Based Model

Researchers has worked on determining the capacity supply model[6] for virtualized servers. In this model first a server is model as a queue based on a Markov chain. Then the effect of server virtualization will be analysed based on capacity supply with the distribution function of the server load. Guet al presented a new stream-based mining algorithm to achieve online anomaly prediction. Different types of anomaly detection, anomaly prediction requires to perform continuous classification on future data based on stream mining algorithm [1]. The stream mining algorithm presented in this paper shows the way of application naive Bayesian classification method and Markov models to achieve the anomaly prediction goal. The proposed system continuously collects a set of runtime system-level and application-level measurements. Markov chain also has been used for prediction in wireless networks performance. Katsaroset al presented information-theoretic techniques for discrete sequence prediction[3]. The article presented here addresses the issues of location and request prediction in wireless networks in a homogeneous fashion. The model then characterizes them in discrete sequence prediction problems. Pre-recorded system log data provides monitoring information about system behaviour. System log data analysis and symptom matching from large volume data is a challenging task on virtual infrastructure. Collecting, analysing, monitoring large volume of monitoring log data is important for monitoring and predicting system behaviour. Virtual IT infrastructure differs from physical IT infrastructure. Virtual infrastructure resource usage is dynamic in nature. It’s also difficult to monitor and predict its capacity resources. In our proposed system we used Markov chain based prediction model to predict resource usage.
2.3 Service Level Objectives and Monitoring Data Analysis

Holubet al presents an approach to implement run-time correlation of large volumes of log data. The system also proposed to do symptom matching of known issues in the context of large enterprise applications [2]. The propose solution provides automatic data collection, data normalisation into a common format. The system then do run-time correlation and analysis of the data to give a coherent view of system behaviour at real-time. This model also provides a symptom matching mechanism that can identify known errors in the correlated data on the fly. Predicting System Anomalies [9] in real world is a challenging task. We need to predict virtual resources anomalies in order to know virtual capacity problems. Tanet al presented an approach aims to achieve advance anomaly prediction with a certain lead time. The intuition behind this approach is that, system anomalies often manifest gradual deviations in some system metrics before system escalates into the anomaly state. The propose system monitor various system metrics called features (e.g., CPU load, free memory, disk usage, network traffic) to build a feature value prediction model. The proposed model use discrete-time Markov chain schemes. In the next step, system also includes statistical anomaly classifiers using tree augmented naive Bayesian (TAN) learning method. Proposed system only used for monitoring physical infrastructure rather than a virtual or cloud infrastructure. In order to use virtualization techniques properly virtual machines need to be deployed in physical machines fast and flexible way. These virtual machine needs to be automatically or manually monitor to meet service level objectives (SLOs). An important monitoring parameter in SLOs is high availability of virtualized systems. A high availability model in virtualized system has been proposed [4]. This model constructed on non-virtualized and virtualized two hosts system models using a two-level hierarchical approach. In this model fault trees are used in the upper level and homogeneous continuous time Markov chains (CTMC) are used to represent sub-models in lower level. In cloud computing, virtualized systems are distributed over different location. In most distributed virtualized systems provides minimal guidance for tuning, problem diagnosis, and decision making to monitor performance. In this aspect a monitoring system was been proposed. Stardust[10] is a monitoring system that replaces traditional performance counters with end-to-end traces of requests and allows efficient querying of performance metrics. These traces can inform key administrative performance challenges by enabling performance metrics. Such performance metrics are extraction of per-workload, per-resource demand information and per-workload latency graphs. This paper also proposes idea of experience building. In this model, a distributed storage system is used for end-to-end online monitoring. Distributed system has diverse system-workloads and scenarios. This model showed that such fine-grained tracing can be made efficient. This can be useful for on and off-line analysis of system behaviour. These proposed methodologies can be incorporate in other distributed systems such as virtualization based cloud computing framework. Cloud computing infrastructure contains a large number of virtualized system, applications and services with constantly changing demands for computing resources. Today’s virtualization management tools allow administrators to monitor current resource utilization of virtual machines. However, it is quite challenging to manually translate user-oriented service level objectives (SLOs), such as response time or throughput. Current virtualization management tools unable to predict demand of capacity resources. Prediction of capacity resources is very important task for the availability of cloud computing infrastructure. An article has been presented to have an adaptive control system. The proposed system automate task of tuning resource allocations and maintains service level objectives [7]. It also focuses on maintaining the expected response time for multi-tier web applications. In
this system there is a control system. This control system is capable of adjusting resource allocation for each VM so that the application’s response time matches the SLOs. The propose approach uses individual tier’s response time to model the end-to-end performance. It helps stabilize application’s response time. This paper also presents an automated control system for virtualized services. The propose system suggests that it is possible to use intuitive models based on observable response time incurred by multiple application tiers. This model can be used in conjunction with a control system to determine the optimal share allocation for the controlled VMs. Propose system helps maintain the expected level of service response time while adjusting the allocation demand for different workloads level. Such behaviour allows administrator to simply specify the required end-to-end service-level response time for each application. This method helps managing the performance of many VMs.

2.4 Anomaly Detection and Performance Monitoring

Cloud computing system is a service driven architecture. The cloud services are based on internet and web. In modern era web-based vulnerabilities represent a substantial portion of the security loop holes of computer network. This is also true for cloud computing infrastructure. In order to detect known web-based attacks, misuse detection systems are equipped with a large number of signatures. Unfortunately, it is difficult to keep up to date with the daily disclosure of web-related vulnerabilities, and, in addition, vulnerabilities may be introduced by installation-specific web-based applications. Therefore, misuse detection systems should be complemented with anomaly detection systems[3]. This paper presents an intrusion detection system that uses a number of different anomaly detection techniques to detect attacks against web servers and web-based applications. The system correlates the server-side programs referenced by client queries with the parameters contained in these queries. The application-specific characteristics of parameters allow the system to perform detailed analysis and produce a reduced number of false positive alerts. The system generates parameter profiles automatically associated with web application data (e.g., length and structure of parameters). This paper introduces a novel approach to perform-anomaly detection, using as input HTTP queries containing parameters. The work presented here is novel in several ways. First of all, to the best of our knowledge, this is the first anomaly detection system specifically tailored to the detection of web-based attacks. Second, the system takes advantage of application-specific correlation between server-side programs and parameters used in their invocation. Third, the parameter characteristics (e.g., length and structure) are learned from input data. Ideally, the system will not require any installation-specific configuration, even though the level of sensitivity to anomalous data can be configured via thresholds to suit different site policies. The proposed system has been tested on data gathered at Google Inc. and two universities in the United States and Europe. Propose model will work in the focus on further decreasing the number of false positives by refining the algorithms developed so far, and by looking at additional features. The ultimate goal is to be able to perform anomaly detection in real-time for web sites that process millions of queries per day with virtually no false alarms. This proposed model can be useful for reducing number of false monitoring alerts in virtualization environment.
3 Markov Chains

An Finite Markov chain is an integer time stochastic process, which consisting of a domain D of \( m > 1 \) states \( \{s_1, \ldots, s_m\} \) and which has the following properties

1. An \( m \) dimensional initial probability distribution vector \( (p(s_1), \ldots, p(s_m)) \).

2. An \( m \times m \) transition probabilities matrix \( M = M(s_i)(s_j) \) \[8\]

\[ M(s_i)(s_j) = \text{Pr}[\text{transition from } s_i \rightarrow s_j] \]

3.1 Stochastic Matrix

Stochastic matrix is used to describe the transitions of a Markov chain. A right stochastic matrix is a square matrix each of whose rows consists of non-negative real numbers, with each row summing to 1. \( M \) is a stochastic Matrix: \( \sum_t m_{st} = 1 \)

3.2 Distribution Vector of Markov Chain

The initial distribution vector \( (u_1, \ldots, u_m) \) which defines the distribution of \( X_1 \) \( \rho(X_1 = s_i) = u_i \). After one move, the distribution is changed to \( X_2 = X_1 M \), where \( M \) is the stochastic Matrix.

3.2.1 Stationary Distributions for Markov Chains

Let \( M \) be a Markov Chain of \( m \) states, and let \( V = (v_1, \ldots, v_m) \) be a probability distribution over the \( m \) states. where \( V = (v_1, \ldots, v_m) \) is stationary distribution for \( M \) if \( VM = V \). This can be only happen if one step of the process does not change the distribution, Then \( V \) is a stationary or steady state distribution.

3.3 Criteria for "Good" Markov chains

A Markov Chains is good if the distributions \( X_t \) has the steady state distribution and following characteristics, where \( t \rightarrow \infty \). \( X_t \) Converge to a unique distribution, independent of the initial distribution. In that unique distribution, each state has a positive probability

4 Modelling CPU Usage with State Distribution Model

In this section we describe how to create the simple state based probability model for predicting resource utilization. The model can used for predicting a set of resource utilization metrics i.e. CPU, Network, Memory and Disk Usage. The model uses data gathered from Somone virtual infrastructure. The infrastructure running natively VMware virtualization solution on the top of real hardware. For now we will show our simple model on state based probability model for predicting CPU usage.

CPU usage measurement unit is "CPU usage percentage". "CPU usage percentage" measurement level varies from 0 % to 100 %. We partition the measurement level in intervals. We define the each interval as states of our model. Let be \( s_1, s_2, \ldots, s_n \) are the states of our system.

Our model use experimental data from SOMONE virtual infrastructure. These experimental data has four collection intervals i.e day, week, month, and year. Each collection interval has
is own collection frequency. For example Monthly collection intervals has collection frequency of one day. One month data are rolled up to create one data point for every day. As a result is 30 data points each month. Similarly we can define collection frequency for hour, day, week and month collection intervals.

4.1 Defining the States of State Distribution Model

We described above that CPU usage measurement unit is CPU usage percentage. CPU usage percentage varies from 0 to 30% in our collected data. We partition the CPU usage level from 0 to 30% from our experimental data. In our collected data we don’t have CPU usage level more than 30%. Let be x1, ….. , xn are the different level of CPU usage.

state x1 = \( \epsilon [0, 5] \)% CPU usage
state x2 = \( \epsilon [5, 10] \)% CPU usage
state x3 = \( \epsilon [10, 15] \)% CPU usage
state x4 = \( \epsilon [15, 20] \)% CPU usage
state x5 = \( \epsilon [20, 25] \)% CPU usage
state x6 = \( \epsilon [25, 30] \)% CPU usage

we observed the state x1, x2, x3,..., xn percentage CPU usage of at discrete time t1, t2,….., tn. Where t is each day of the month.

4.2 Computing State Distribution

We compute the state based probability vector distribution for each state. Our data don’t contain higher levels of observation than 30%. Our process to compute the state based probability of x1 state \( Pr(x1) \), we compute total no number of events going from \( x1 \rightarrow x1...xn \) divided by total number of possible events. which is describe in following equation state based probability vector distribution \( Pr(x1) = n(x1)/n(S) \). where \( n(x1) = \) total no number of events going from \( x1 \rightarrow x1...xn \), \( n(S) = \) total number of possible events

Similarly we can compute the state based probability vector distribution of each states.

5 Modelling CPU Usage with Maximum Frequent State

We discussed on section §4.2 about how to compute state based based probability vector distribution for each state. In this section we will discuss about how to we model CPU usage with maximum frequent state. First we compute the state based probability vector distribution of CPU usage \( Pr(x1),....,Pr(xn) \). Which we have been discussed on section §4.2. In order to calculate maximum frequent state of probability vector distribution, we compute \( S = \max Pr(x1),....,Pr(xn) \), where S is the predicted maximum frequent state of CPU Usage and \( Pr(x1),....,Pr(xn) \) state based based probability vector distribution of CPU usage x1, x2, x3,..., xn state. Similarly we can compute Maximum Frequent State of Memory, Network, Disk usages.

6 Modelling the CPU Usage with Markov Chain

A virtual Infrastructure has different components such as CPU, Memory, IO, Disk and Network. Each component has different metrics to monitor resource utilization. Example CPU
have CPU usage (average), usage (MHz) metrics to monitor. Each metric has a unit of measurement. For example CPU usage metric’s measurement is "CPU usage percentage". "CPU usage percentage" measurement level varies from 0 % to 100 %. We partition the measurement level in intervals. We define the each interval as states of our system. Let be s1, s2, ..., sn are the states of our system. We have collected experimental data from SOMONE virtual infrastructure. These experimental data has four collection intervals i.e Day, Week, Month, and Year. Each collection interval has collection frequency. For example Yearly collection intervals has collection frequency of one day. One month data are rolled up to create one data point every day. As a result is 365 data points each year. Similarly we can define collection frequency for Day, Week, Month collection intervals. For now we show our Markov Chain model for CPU Usage prediction.

6.1 Defining the States of the System

CPU usage metric’s measurement unit is CPU usage percentage. CPU usage percentage varies from 0 to 30 % in our collected data. We partition the CPU usage level from 0 to 30 % from our experimental data. Let be s1, ..., sn are the different level of CPU Usage.

state s1 = \{0, 5\} % CPU Usage
state s2 = \{5, 10\} % CPU Usage
state s3 = \{10, 15\} % CPU Usage
state s4 = \{15, 20\} % CPU Usage
state s5 = \{20, 25\} % CPU Usage
state s6 = \{25, 30\} % CPU Usage

we observed the state s1, s2, s3, ..., sn percentage CPU usage at discrete time t1, t2, ..., tn. Where t is each day of the year

6.2 Computing the Transition Probability

We compute the transitional probability for each pair of states. Our data don’t contain higher levels of observation than 30%. For example to compute the transitional probability from s1 → s2 we compute total number of successive events going from s1 → s2 divided by the total number of s1 events. which is describe in following equation

Transitional probability s1 → s2 = \( Pr[X_{t+1} = s2 | X_t = s1] \). Similarly we can compute transitional probability for other pair of states.

6.3 Building the Transition Probability Matrix

We build an \( m \times m \) transition probability matrix \( M = M(s_i)(s_j) \)

where \( M = M(s_i)(s_j) = Pr[\text{transition from } s_i \rightarrow s_j] \)

we transition matrix with our data

\[
\begin{pmatrix}
\text{state} & s1 & s2 & s3 & s4 & s5 & s6 \\
s1 & 0.8564 & 0.1436 & 0 & 0 & 0 & 0 \\
s2 & 0.1628 & 0.8072 & 0.012 & 0.0060 & 0.120 & 0 \\
s3 & 0 & 1 & 0 & 0 & 0 & 0 \\
s4 & 0 & 1 & 0 & 0 & 0 & 0 \\
s5 & 0 & 0 & 0 & 0 & 0 & 0 \\
s6 & 0.25 & 0 & 0 & 0.25 & 0 & 0.5
\end{pmatrix}
\]
State 5 has 0 transition probabilities to other states. State 5 not a reachable state from other states. So we remove s5 from our system so our new transition matrix becomes:

$$\begin{pmatrix}
\text{state} & s1 & s2 & s3 & s4 & s5 \\
s1 & 0.8564 & 0.1436 & 0 & 0 & 0 \\
s2 & 0.1628 & 0.8072 & 0.012 & 0.0060 & 0.120 \\
s3 & 0 & 1 & 0 & 0 & 0 \\
s4 & 0 & 1 & 0 & 0 & 0 \\
s5 & 0.25 & 0 & 0 & 0.25 & 0.5 \\
\end{pmatrix}$$

Now state s6 = state s5 = 25 to 30 CPU Usage Level

6.4 Markov Chain Stochastic Matrix in Virtualized CPU Usage

M is stochastic matrix:

$$\begin{pmatrix}
\text{state} & s1 & s2 & s3 & s4 & s5 \\
s1 & 0.8564 & 0.1436 & 0 & 0 & 0 \\
s2 & 0.1628 & 0.8072 & 0.012 & 0.0060 & 0.120 \\
s3 & 0 & 1 & 0 & 0 & 0 \\
s4 & 0 & 1 & 0 & 0 & 0 \\
s5 & 0.25 & 0 & 0 & 0.25 & 0.5 \\
\end{pmatrix}$$

A right stochastic matrix is a square matrix each of whose rows consists of non-negative real numbers, with each row summing to 1

So if we add s11 + s12 = 1 Similarly if we add each row the sum will be 1. Therefore it satisfy the stochastic properties of Markov Chain.

![Figure 1: CPU Resource Utilization State Transition Diagram](image)

6.5 State Transition Diagram of CPU Resource Utilization

A Markov Chain system can be illustrated by means of a state transition diagram, which is a diagram showing all the states and transition probabilities. The information from the
stochastic matrix is shown in the form of transition diagram. The Figure (1) is a transition
diagram of CPU resource utilization that shows the five states of CPU resource utilization and
the probabilities of going from one state to another. The state transition diagram is necessary
to find if the all states of the systems are strongly connected. The strong connected state
transition diagram is important for the stability of Markov Chain prediction system. It can
help us to understand the behaviour prediction system in long run.

6.6 Computing the Distribution Vector of the System

A distribution vector is a row vector with one non-negative entry for each state in the system.
The entries can be represent the number of individuals in each state of the system. For compu-
ting the distribution vector we do the following. Let M be Markov Chain of s states, let V
is an initial distribution vector. The initial distribution vector (v1......vs) which defines the
distribution of V1 (p(V1 = si) = vi). After one move, the distribution is changed to V2 = V1M,
where M is the stochastic Matrix. We compute initial distribution vector over the s states of
CPU usage level. Which is based on our first entry of our historical data. Suppose we have
5 states of CPU usage s1, s2, s3,.....,s5. Our first entry our historical data fall on state s2
then the initial distribution vector V = [0,1,0,0,0]. M is the stochastic matrix virtualized
CPU usage §6.4. We multiply the initial distribution vector with stochastic matrix M to get
the distribution vector after 1 step is the matrix product VM. The distribution one step later,
obtained by again multiplying by M, is given by (VM)M = VM². Similarly, the distribution
after n steps can be obtained by multiplying V by Mn. In our monthly prediction system
of CPU resource usage we compute upto 30 time steps, which donated as n. Each time step is the
average prediction of CPU usage of each day. Our prediction of CPU usage is a distribution
vector over n dimensional space.

7 Calculating Prediction Error of Probability Distribution

In above section we discuss that the predicted CPU usage is the distribution over n dimen-
sional space. To calculate the prediction error of probability vector distribution, we use Euclidean
distance of n dimensional space. This space difference is calculated between predicted and
observed data. where x predicted data of the probability vector distribution of CPU usage over
n dimensional space and y observed data of the probability vector distribution of CPU usage
over n dimensional space. Where x = (x1, x2, ..., xn) and y = (y1, y2, ..., yn) are points in
Euclidean n dimension space.

We compute ỹ in the following way from our observed data.

ỹ is CPU probability vector distribution over n dimensional space in our observed data.
CPU usage metric’s measurement unit is CPU usage percentage. CPU usage percentage varies
from 0 to 30 % in our observed data. We partition the CPU usage level from 0 to 30 % from
our observed data. We don’t have more than 30 % CPU usage level in our observed data.

Let be y1, ..., yn are the different level of CPU Usage.

state y1 = ε [0,5] % CPU Usage
state y2 = ε [5,10] % CPU Usage
state y3 = ε [10,15] % CPU Usage
state y4 = ε [15,20] % CPU Usage
state y5 = ε [15,20] % CPU Usage
state $y_6 = d[15,20]$% CPU Usage
we observed the state $y_1, y_2, y_3, \ldots, y_n$ state percentage CPU Usage at discrete time $t_1, t_2, \ldots, t_n$. Where $t$ is each day of the month.

The Euclidean distance between $x$ and $y$ are the position of points in a Euclidean n-space. So, $x$ and $y$ are Euclidean vector. Then the distance from $x$ to $y$ is given by

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ are points in Euclidean n-space.

In error compare method we want to compare the prediction result with our observation. Our prediction result are generated in probability vector distribution of states. So the probability vector distribution is an n-dimensional state space. So compute the distance or error between the predicted and observed states we use

$$\bar{x} = \text{predicted}, \bar{y} = \text{observed}$$

$$d_1 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

### 8 Observation from Data and Experiments

#### 8.1 Experimental Results

In our experiments we have used a time unit of 1 day and we have discretized the monitoring data by this time unit. We have collected Memory, Disk, Network and CPU Usage monitoring parameter as a daily average from SOMONE virtual infrastructure. The plot of relative error vs time unit represents daily relative error of prediction in Memory, Disk, Network and CPU Usage of a month. In these experiments we predict average daily Memory, Disk, Network, CPU usage using the monitoring usage data of previous month. Then we compare our prediction with real-time usage of SOMONE virtual infrastructure to find out relative error of prediction.

#### 8.2 Data Sources

We use SOMONE virtual infrastructure for collecting monitoring data. We have tested our markov chain based model with monitoring traces from three different virtualized servers. Each virtual servers has its own usage particularities. Virtualized servers consists of VMWare ESXi version 3.5, ESX version 3.5(Enterprise edition) and ESX version 4. We use VSphere and Nagios client to access and monitor these virtual machine.

#### 8.3 Experimental Set up

In this section we will show some observations, with plots illustrating the phenomena, made during the experiments. The analysed data has the following characteristics:

All experiments use monitoring traces of the virtual infrastructure traces of the same form of input data i.e. a time unit of 1 day and resources are CPU, Memory, Network and Disk used across time span of 30 days. Each usage value designate average value of each day. From our experiment we predict average resource usages of each day with time span of prediction of 30 days. The results are displayed under the form of a relative error vs time unit. Which represents daily error of resource usage prediction.
Our SOMONE virtualized based cloud system contains following IT infrastructure. That consists of 3 physical server. We installed VMware bare metal embedded hypervisor on the top each physical server.

First physical server is Dell SC440 series. The server has an Intel Xeon Dual Core Processor 2.13 GHz with 4 GB memory and 500GB * 2 hard drive on RAID1. It contains embedded hypervisor system with VMware ESXi 3.5. The main purpose of this virtual server is to maintain critical tools of SOMONE IT Infrastructure. Critical tools are VPN server, DNS server, Intranet, DHCP(private agents on LAN), OCS and GLPI for managing IT infrastructure assets. This server also contains virtual center for managing the SOMONE virtual infrastructure.

Second physical server is Dell T100 series. The server has an Intel Xeon Quad Core Processor 2.4 GHz with 8 GB of memory and 1TB * 2 hard drive on RAID1. It contains embedded hypervisor with VMware ESX 3.0 enterprise edition. This virtual server is used for testing different monitoring tools like Nagios, BMC patrol, Centrion, Zabbix, Syslog. This virtual server is also used for production, development and configuration of SOMONE tools. This server provides hands on lab for SOMONE R and D team.

Third physical server is Dell T300 series. The server has an Intel Xeon Quad Core Processor 2.83 GHz with 24 GB of memory and and 1TB * 2 hard drive on RAID1 embedded hypervisor with VMware ESX 4.0. This virtual server is used to simulate IT Infrastructure. This physical server contains virtual machines and containers.

1. The data is from several virtualization servers. Some of the servers are application, system, web servers and others server hosts different monitoring systems for testing and simulating different monitoring environment.

2. Data is available from 15 consecutive months (small changes in the system have been made between the months and thus the results from different months are not directly comparable).

3. System log data contains originally almost 300 monitoring parameters. However, only a subset of important monitoring parameters was chosen for the analysis and experiments. These important monitoring parameters is suggest by SOMONE monitoring and virtualization experts. SOMONE monitoring experts has several years experience on system and data center monitoring.

4. The monitoring data is stored in constant time intervals, usually 10 minute, 30 minute, 1 hour, 12 hour and 1 day which spread on hourly, daily, weekly, monthly and yearly time span.

5. The monitoring data can’t able to show which instances of data contain troublesome for system behaviour.

6. The monitoring parameters used in predicting resource usage in virtual infrastructure included are CPU usage, Memory usage, Network usage and Disk usage.

8.4 Results

We had done several prediction tests from our 3 different prediction models. Which are 1. State Based Probability Model 2. Maximum Frequent State Model 3. Markov Chain Model. We have shown the results of predictions monitoring parameters i.e. CPU, Memory, Network and
Disk usage in probability distribution. We displayed the results of our prediction in terms of relative error. Relative error graph shows the relative error of probability distribution of CPU, Memory, Network and Disk usage with days and hour time scale. Relative error of probability distribution scale are measured in 0 to 100% scale.

Relative error of probability distribution of CPU usage with time scale of 30 days

In our prediction models we used monitoring traces from virtualization servers as historic data. The historic data was from November 2010. This data was used to predict CPU usage probability distribution of next month i.e. December 2010. We used back-testing techniques to check the accuracy of our prediction. The results of this accuracy is shown in terms of relative error. Less the relative error more the accuracy of prediction. Experimental results are shown in terms of relative error. A zoom into the plot of relative error is shown in figure (2). In the figure (2) X axis is time scale of 30 days and Y axis is the error of probability distribution of CPU usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

Relative error of probability distribution of Memory usage with time scale of 30 days

In our prediction models we used monitoring traces from virtualization server as historic data. The historic data was from March 2010. This data was used to predict Memory usage probability distribution of next month i.e. April 2010. We used back-testing techniques to check the accuracy of our prediction. The results of this accuracy is shown in terms of relative error. Less the relative error more the accuracy of prediction. Experimental results are shown
in terms of relative error. A zoom into the plot of relative error is shown in figure (3). In the figure (3) X axis is time scale of 30 days and Y axis is the error of probability distribution of Memory usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

**Relative error of probability distribution of Disk usage with time scale of 30 days**

In our prediction models we used monitoring traces from virtualization server as historic data. The historic data was from April 2010. This data was used to predict Disk usage probability distribution of next month i.e. May 2010. We used back-testing techniques to check the accuracy of our prediction. The results of this accuracy is shown in terms of relative error. A zoom into the plot of relative error is shown in figure (4). In the figure (4) X axis is time scale of 30 days and Y axis is the error of probability distribution of Disk usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

**Relative error of probability distribution of Network usage with time scale of 30 days**

In our prediction models we used monitoring traces from virtualization server as historic data. The historic data was from September 2010. This data was used to predict Network usage probability distribution of next month i.e. October 2010. We used back-testing techniques to check the accuracy of our prediction. The results of this accuracy is shown in terms of relative error. Less the relative error more the accuracy of prediction. Experimental results are shown

Figure 3: Relative Error of Memory Usage Prediction
Figure 4: Relative Error of Disk Usage Prediction

Figure 5: Relative Error of Network Usage Prediction
in terms of relative error. A zoom into the plot of relative error is shown in figure (5). In the figure (5) X axis is time scale of 30 days and Y axis is the error of probability distribution of Disk usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

**Relative error of probability distribution of CPU usage with time unit of minute**

In our prediction models we used monitoring traces from virtualization server as historic data of hourly CPU usages. The historic data was used to predict CPU usage probability distribution of next 1/2 hour. We used back-testing techniques to check the accuracy of our prediction. The results of this accuracy is shown in terms of relative error. Less the relative error more the accuracy of prediction. A zoom into the plot of relative error is shown in figure (6). In the figure (6) X axis is time unit of minute and Y axis is the error of probability distribution of CPU usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

**Relative error of probability distribution of Memory usage with time unit of minute**

In our prediction models we used monitoring traces from virtualization server as historic data of hourly Memory usages. The historic data was used to predict Memory usage probability distribution of next 1/2 hour. We used back-testing techniques to check the accuracy of our prediction. The results of this accuracy is shown in terms of relative error. Less the relative error more the accuracy of prediction. Experimental results are shown in terms of relative error. A zoom into the plot of relative error is shown in figure (7). In the figure (7) X axis is time unit of minute and Y axis is the error of probability distribution of Memory usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

**Relative error of probability distribution of Network usage with time unit of minute**

In our prediction models we used monitoring traces from virtualization server as historic data of hourly Network usages. The historic data was used to predict Network usage probability distribution of next 1/2 hour. We used back-testing techniques to check the accuracy of our
Figure 7: Relative Error of Memory Usage Prediction

Figure 8: Relative Error of Network Usage Prediction
prediction. The results of this accuracy is shown in-terms of relative error. Less the relative error more the accuracy of prediction. Experimental results are shown in terms of relative error. A zoom into the plot of relative error is shown in figure (8). In the figure (8) X axis is time unit of minute and Y axis is the error of probability distribution of Network usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

**Relative error of probability distribution of Disk usage with time unit of minute**

In our prediction models we used monitoring traces from virtualization server as historic data of hourly Disk usages. The historic data was used to predict Disk usage probability distribution of next 200 minutes. We used back-testing techniques to check the accuracy of our prediction. The results of this accuracy is shown in-terms of relative error. Less the relative error more the accuracy of prediction. Experimental results are shown in terms of relative error. A zoom into the plot of relative error is shown in figure (9). In the figure (9) X axis is time unit of minute and Y axis is the error of probability distribution of Disk usage. Relative error of State Based Probability Model, Maximum Frequent State Model and Markov Chain Model are shown in red, sky blue, green dots in graph respectively.

### 8.4.1 Quality of Predictions

We tested our proposed prediction models using CPU, Memory, Disk, Network usage data. We summarized quality of predictions in the table below.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Time scale</th>
<th>Prediction model</th>
<th>Safe look ahead time</th>
<th>Relative error</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU usage</td>
<td>days</td>
<td>maxFreqState</td>
<td>20 days</td>
<td>0%</td>
<td>maxFreqState model can predict 20 days of CPU usage with 0% relative error</td>
</tr>
<tr>
<td>Memory usage</td>
<td>days</td>
<td>maxFreqState</td>
<td>22 days</td>
<td>0%</td>
<td>maxFreqState model can predict 20 days of Memory usage with 0% relative error</td>
</tr>
<tr>
<td>Disk usage</td>
<td>days</td>
<td>maxFreqState</td>
<td>2 days</td>
<td>0%</td>
<td>maxFreqState model can predict 2 days of Disk usage with 0% relative error</td>
</tr>
<tr>
<td>Network usage</td>
<td>days</td>
<td>Markov chain</td>
<td>3 days</td>
<td>17%</td>
<td>Markov chain model can predict 3 days of Network usage with 17% relative error</td>
</tr>
<tr>
<td>CPU usage</td>
<td>minutes</td>
<td>maxFreqState</td>
<td>5 mins</td>
<td>0%</td>
<td>maxFreqState model can predict 5 mins of CPU usage with 0% relative error</td>
</tr>
<tr>
<td>Memory usage</td>
<td>minutes</td>
<td>maxFreqState</td>
<td>4 mins</td>
<td>0%</td>
<td>maxFreqState model can predict 4 mins of Memory usage with 0% relative error</td>
</tr>
<tr>
<td>Disk usage</td>
<td>minutes</td>
<td>maxFreqState</td>
<td>2 mins</td>
<td>0%</td>
<td>maxFreqState model can predict 2 mins of Disk usage with 0% relative error</td>
</tr>
<tr>
<td>Network usage</td>
<td>minutes</td>
<td>maxFreqState</td>
<td>7 mins</td>
<td>0%</td>
<td>maxFreqState model can predict 7 mins of Network usage with 0% relative error</td>
</tr>
</tbody>
</table>

### 8.4.2 Alert Forecasts

Knowing the quality of our prediction we now return to our observed resource usage data and estimate the usefulness of our models to predict alerts. Any parameters overshooting the level of 80% triggers a critical alerts. Our models can be used to forecast those critical alerts and thus give system administrators / system users some advanced warning. We consider how this can happen for each observed parameters.

CPU: our data shows no critical alerts but if any alert seen, our CPU maxFreqState prediction model would forecast them 20 days (resp. 5 mins) in advance with 0% prediction error.
Memory: the data shows no critical alerts but if any alerts seen, our Memory maxFreqState prediction model would forecast them 22 days (resp. 4mins) in advance without error.

Disk: our data contains 6 (resp. 5) Disk alerts at 80 % of max. Disk usage and our Disk maxFreqState prediction model can forecast them 2 days (2 minutes) in advance with 0 % error. This would be useful advance warning system to users and avoid possible risk of Disk crashes.

Network: our data contains 1 (resp. 1) alert at 83% (resp. 85%) of maximum Network usage and the Markov chain prediction model would forecast them 3 days (7 mins) in advance with a probability computed as follows. The Markov predicted level is 83% ± 17% error margin (resp.85% ± 17%) i.e. an interval of predicted network usage [69% ... 97%] (resp [70% ... 99%]) and the fraction of this interval intersecting the alert zone we treat as possibility for prediction. As a result the alert in the 1st case is given probability \( \frac{97-80}{97-69} = \frac{17}{28} \) (resp. \( \frac{99-80}{99-70} = \frac{19}{29} \)) i.e. 61% (resp. 65%). So the alert forecast for network level critical alert would be impossible without the Markov model.

9 Conclusion and Future Work

In this paper, we present three new resource usage prediction models. They use a set of historic data to predict the system usage in a minute, hour, daily and monthly manner. Our experiment has shown that each model has good results for specific resource prediction when applied to immediate future. This makes the evaluation of the model to be context dependent. On the hand our models treat historical data from black-box system without known the details of underlying hardware. This makes our model independent of any hardware and vendor specification.

There are clear differences in resource prediction between our different prediction models. Some resource prediction models behave nicely such as memory usages prediction in Figure(3). On the other hand, other prediction model are not so stable also exhibit rather irregular behaviour such as disk usage prediction Figure (4). First 3 days of monthly network usages can be predicted with Markov Chain based prediction model (5). Short term prediction with the interval 5 mins can be predicted with Maximum Frequent State model Figure (6, 7, 8, 9). In our experiment we can see, some models can have a constant increase of prediction error of the resources such as Markov Chain model for CPU usage prediction Figure( 2,3).

An important benefit of prediction based resource monitoring is the ability to forecast resource usages of virtualized servers. It can help cloud infrastructure provider to adapt virtualized resource on demand. This model has great implications on cost saving for resource consolidation, monitoring and resource utilization. Our model can forecast under-utilized resources, which can be used to consolidate existing under-utilized resources for an efficient use. On the other hand our model can also forecast over-utilized resources. Over-utilized resources can be adjusted properly to avoid system down time. The proposed model is able to do short-term and possibly long-term forecasting, which can help system administrator to monitor resources in advance. Long-term forecasting can help system managers for capacity planning. The current experiments should also be confirmed by more extensive datasets. As future work directions we will be building cost and billing model considering virtualization and cloud environment. Moreover we will be developing prediction models based on application and services monitoring metrics. Further ahead, we will be looking into ways how to predict alerts by combining system, application and services prediction models.
10 Acknowledgement

The first author thanks SOMONE S.A.S for an industrial doctoral scholarship under the French government’s CIFRE scheme. He also gratefully acknowledges research supervision, enthusiasm, involvement and encouragement of Prof. Gaétan Hains, LACL, Université Paris-Est Créteil (UPEC) and Mr. Cheikh Sadibou DEME, CEO of SOMONE. Both authors thank SOMONE for access to the virtual infrastructure on which we have conducted the experiments.

References


