Workload Prediction in Cloud Datacenters Based on User Behavior Modeling

Undergraduate Thesis

Submitted in the partial fulfillment of the requirements of
BITS F421T Thesis

By

Pratyush Kar
2013A7TS029P

Under the supervision of

Prof. Sundar B.
Professor, Dept. of Computer Science and Information Systems,
Birla Institute of Technology and Science, Pilani

Birla Institute of Technology and Science, Pilani (Rajasthan)

November 22, 2016

Acknowledgements

I am grateful to the Dean, Academic Research Division and Head of the Department, CSIS BITS-Pilani for giving me this opportunity to take up the course BITS F421T, Thesis. I sincerely thank my thesis supervisor Prof. Sundar B. for guiding me throughout the course of this thesis. His valuable guidance has given me a great learning experience and helped me achieve my goals. I thank Dr. Virendra S. Shekhawat for his enlightening discussions. His guidance has been of great help in understanding the problem and working out the implementation methods. Finally I also thank the staff of CSIS, BITS-Pilani, for their immense help in logistic matters.

Certificate

This is to certify that the thesis entitled, **Workload Prediction in Cloud Datacenters Based on User Behavior Modeling** and submitted by **Pratyush Kar**, ID No. **2013A7TS029P** in partial fulfillment of BITS F421T (Thesis) embodies the work done by him under my supervision.

Prof. Sundar B.
Professor,
Dept. of Computer Science and Information Systems,
Birla Institute of Technology and Science, Pilani

Abstract

Thesis Title: Workload Prediction in Cloud Datacenters Based on User Behavior Modeling

Supervisor: Prof. Sundar B., Professor, Dept. of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani

Semester: First **Session:** 2016-17

Name of Student: Pratyush Kar **ID No.:** 2013A7TS029P

Abstract:

Workload characterisation can be used to predict future resource requirements in a Cloud Data Centre (CDC) environment. Understanding the workload behaviour or pattern plays a key role in optimal resource provisioning and may prove vital in order to meet the stipulated service level agreements (SLAs). A better workload prediction algorithm prevents the CDC from under or over-provisioning of resources. This is critical for energy conservation in the datacenter and maintaining the Quality of Service (QoS) for the end user.

User behaviour in the form of task submission has shown a strong correlation to the workload level in the datacenter. This thesis identifies features that are representative of user behaviour and discusses methods of modelling this behaviour. These models are then utilised to predict future CPU and memory workloads in the datacenter.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Majority of the applications that are in existence nowadays are based on cloud platforms. Various applications have diverse Quality of Service (QoS) aspects, such as availability, reliability and performance. These aspects are stipulated in the Service Level Agreements (SLAs) negotiated between the client and cloud service provider. Failure to comply with the QoS requirements would incur SLA violations resulting in loss of revenue for the service providers [1]. On an average a user does not use the entire amount of resources that she has requested from the cloud service provides. The providers capitalise on this behaviour by rolling out more number of virtual machines (VMs) than they physically possess at cheaper rates, relying of the fact that most clients applications will not run at their peak requirements. The service providers offer on-demand performance by dynamic provisioning of resources. Gaining a thorough understanding of the workload behaviour of a production cloud data centre (CDC) is important for the datacenter's ability to elastically scale up and down the provisioned resources. Workload characterisation can be used to predict future resource requirements which help in capacity planning and better resource utilisation.

Workload characterisation is typically performed by using two different approaches – either trace based or model based methodologies. The model based methodology is preferred over trace based one because it is agnostic to the underlying system on which the trace is recorded on. Furthermore, due to the limited number of production quality traces, a model based approach is far superior to the trace based approaches which require frequent tweaking to make them compatible to new datacenter environments. Most cloud datacenter workloads are a mix of heterogenous applications. Building a unified model that can predict the future resource usage of these diverse applications is an extremely challenging

task. These tasks show varied behaviour in periodicity, burstiness and repeating patterns. Therefore it is important to break these large number of diverse applications into homogeneous clusters and build predictive models for each of the individual clusters.

Users are responsible for driving the volume and behaviour of various tasks. The submitted tasks may be batch jobs that may run for a prolonged duration in the background or they may be user-interactive tasks that require active input from the user. For each submitted task the user also mentions the requested amount of CPU and memory resources. Identifying patterns in such requests may prove vital in clustering the diverse application behaviour. This thesis fills this void by developing ideas that help in modelling user behaviour and utilises these models for developing a mechanism for resource prediction.

1.2 Objectives

The three main objectives of this thesis are as follows:

- identify parameters based on the diversity of submitted tasks to describe user behaviour
- analyse the predictability of various resource (CPU, Memory) and identify the presence of time patterns in the usage of various resource types
- implement models for prediction of future CPU and memory usage based on historical data

Chapter 2

Related Work

The analysis of workload patterns in cloud data centre (CDC) environments has been an active research problem in cloud computing and has been addressed previously in [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Recent research in dynamic resource allocation, explore some interesting techniques and heuristics for predicting workload patterns in advance. Typically these techniques can be broadly classified into – *trace-based* and *model-based* methodologies. This section talks about some of the most relevant approaches.

2.1 Trace Based Approaches

Workload prediction has typically been done by using trace based techniques. Traces are resource consumption metrics recorded over a prolonged period of time from a particular datacenter infrastructure. We try to identify behavioural patterns in the task submission and resource consumption in order to predict future resource requirements. This is extremely vital for dynamic resource provisioning in a datacenter. In this section we describe the most relevant state-of-the-art approaches.

Wang et al. [13] discuss techniques of obtaining coarse-grain statistics from the workloads of Cloud computing Hadoop clusters (first version of the Google tracelog). The main objective of this work is to classify tasks and jobs based on duration of execution. Zhang et al. [3] present a study that discusses the feasibility of using mean values of task waiting time, CPU usage, memory and disk consumption. The data used by them is not available freely and is a trace of 6 Google clusters running for a span of five days. Mishra et al. [4] identify workload characteristics to divide the submitted tasks into multiple clusters. They further identify qualitative boundaries for task clusters and reduce the number of clusters by merging adjacent clusters. The analyses data consists of trace from five

Google clusters across a period of 4 days. Kavulya et al. [5] present a statistical analysis of job completion times in MapReduce traces. The data used for analysis is a ten month trace of the M45 supercomputing cluster. Aggarwal et al. [6] provide techniques to analyse Hadoop jobs in Yahoo! clusters. The trace used in this analysis spans a period of 24 hours and contains data from storage usage of over 11686 jobs. The dataset only contains metric data from the storage system of Yahoo!'s distributed file system and hence this work neglects critical resources like CPU and memory usage. Solis et al. [7] perform an extensive analysis on the Google Cluster tracelog by identifying parameters that describe both user and task behaviour. Using k-means clustering they divide both users and task into representative clusters. The analysis done on these clusters is two pronged – first cluster centroids are used to describe the overall behaviour of a particular cluster and secondly statistical distribution models are fitted onto the distribution data to build a model of the resource usage. Shen et al. [8] explore business critical workloads which represent a very different behaviour than the MapReduce workloads of the Google tracelog. They collect traces from the distributed datacenter of Bitbrains using the monitoring and management tools provided by VMware. The characterisation workload is done by capturing parameters/metrics at – VM level (CPU and memory usage), Networking level, I/O and storage level. They perform distribution analysis on the capture trace and evaluate the correlation between different resource types. Based on this they identify the short term predictability of resource usage. The observations from this workload variant do not match the profile of the Google cluster workloads. Business-critical applications show a higher variability in CPU and memory usage than the Google cluster, where the workloads are relatively stable.

Table 2.1: Overview of Trace Based Approaches

Authors	**Trace Size**	**Analysis Methodology**	**Analysed Components**	**Analysed Parameters**
Smith [2]	7 hours	Coarse-grain	Task	Task duration
Zhang [3]	30 days (5 day sample)	Coarse-grain	Task	Task resource usage
Mishra [4]	4 days	Cluster centroids	Task	Task resource usage
Kavulya [5]	10 months	Coarse-grain	Task	Task duration
Aggarwal [6]	24 hours	Cluster centroids	Task	Task disk usage
Solis [7]	29 days	Cluster centroids & distribution analysis	User & Task	User resource estimation & task resource usage
Shen [8]	1 month	Distribution & predictability analysis	Task	Task resource, I/O & network usage

Although trace based approaches are more natural and try to model the real data recorded from large production clusters, the tracelog is heavily dependent on the underlying system architecture. A model based approach, which is agnostic to the underlying system, is more feasible to build a prediction framework that is

not specific to a particular set of application mix and datacenter organisation.

2.2 Model Based Approaches

As stated earlier high quality traces are essential in building an efficient workload prediction framework. However, the utility of a tracelog is limited to the system on which they are recorded. Model based approaches is an alternative to this approach. Different approaches exist in modelling workload patterns and are fundamentally different. Broadly, we can categorise these proposed designs into – *in-breadth* and *in-depth* modelling. In-breadth approach strives to model system-centric measures like CPU, memory, network usage, I/O requests, etc. In-depth approach on the other hand tracks a particular user request through multiple system levels.

In-Breadth Modelling

This approach attempts to track workloads in different parts of the system. Techniques span from modelling CPU and memory usage to modelling network and disk I/O subsystems.

Storage forms a major part of a cloud datacenter. There has been extensive work done on storage usage based production traces [14], [15]. Other approaches propose to model storage usage characteristics using a state diagram model [16]. Gulati et al. [17] propose to model storage workload based on a number of features – *seek distance*, *I/O sizes*, *read/write ratio* and *number of I/Os*. Other approaches try to eliminate the need of system specific traces by utilising a general purpose storage model [18], [19].

Characterising the CPU loads is extremely important for dynamic provisioning of VMs. Abrahao et al. [20] propose a trace-based approach to identify representative features for CPU usage analysis. They analyse 12 application during the course of 2 weeks. Using this data as the input to PCA (Principal Component Analysis) they identify 3 features to characterise applications as noisy or intermittent. Other works include modelling CPU loads in web environments [21]. Huang et al. [22] focus on predicting CPU utilisation, in order to apply DVFS (Dynamic Voltage and Frequency Scaling) for stalling the processors during batch jobs in order to improve energy efficiency.

Network modelling is an important aspect of workload prediction because of its impact in large-scale applications. Feitelson et al. [23] present a detailed overview of distribution fitting using the Kolmogorov-Smirnov test to identify the distribution of submitted requests. Li [24] analysis jobs based on arrival rates, pseudoperiodicity, job size for CPU and network intensive applications. Joo et al.

[25] attempt to model network traffic by comparing two different network models – *infinte-source-based* model and a *SURGE-based* model.

In-Depth Modelling

In-depth modelling attempts to trace a request through multiple levels in the datacenter hierarchy. This allows us to model a more accurate model of the user behaviour. Liu et al. [26] capture the behaviour of the request through a 3-tier web application. The traces used for this purpose is a TPC-W benchmark which is a transactional web benchmark. Ganapathi et al. [27] identify multiple task features in MapReduce jobs and use Kernel Canonical Correlation Analysis (KCCA) to predict the execution time of submitted jobs.

Chapter 3

Dataset

The dataset used in this work was collected from the second version (ClusterData2011_2) of the Google MapReduce Cloud tracelog that spans a period of approximately one month [28, 29]. The log contains millions of records of jobs, tasks and server events represented in a tabular format. It also provides the CPU, memory and disk usage per task in a timestamp every 5 minutes.

3.1 Introduction

The cluster represented in this dataset is a set of machines, packed into racks and connected by a high bandwidth cluster network. A set of machines is termed as a cell and which are all typically, inside a single cluster. The users submit work to the cluster in the form of a job. A job can be comprised of one or more task, each accompanied by a set resource and scheduling requirements. The actual resource usage information is obfuscated as is presented in a normalised manner. Each task represents a Linux program possibly containing multiple processes which run onto a single virtual machine (VM).

3.2 Description

The majority of our analysis is focused on two data structures – *Task events table* and *Resource usage table*.

3.2.1 Task Events Table

The task events table contains the following attributes:

1. timestamp – in microseconds

2. missing info
3. job ID
4. task index – within job
5. machineID – unique 64 bit identifier
6. event type – scheduling event type
7. user name – opaque base64-encoded string
8. scheduling class
9. priority
10. resource request for CPU cores
11. resource request for RAM
12. resource request for local disk space
13. different-machine costraint

Job and task events indicate transitions between the different scheduling states. Each task event has a integer value that represents the type of the event. The state of the job and the individual tasks can always be determined by the event type. The diagram below shows the state transition diagram.

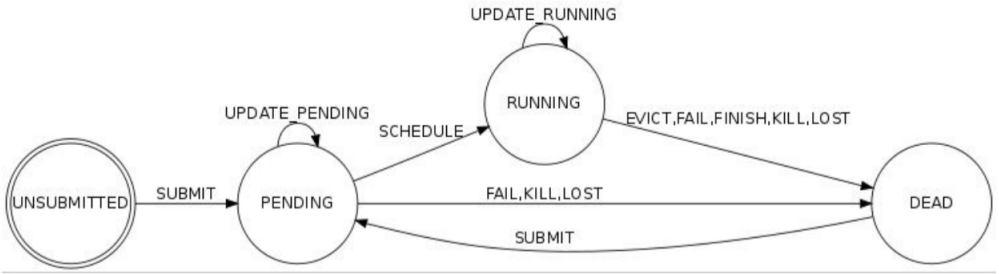


Figure 3.1: State transition diagram for jobs and tasks

The resource request represents the maximum amount of CPU, memory and disk space that a particular task is allowed to access. Tasks that use more than the requested resources are throttled (for CPU) or killed. The scheduler often over commits the resources on a machine. As a result at runtime, the scheduler may not have sufficient amount of resources to execute all the tasks at once. If the total amount of requested resources exceeds the amount of physical resources present in the machine, the scheduler will kill one or more lower priority processes. The runtime environment also sometimes permits tasks to use more resources than requested, if a resource is free tasks are permitted to use excess resource for small amounts of time.

3.2.2 Resource Usage Table

The task resource usage table contains the following attributes:

1. start time of the measurement period – in microseconds
2. end time of the measurement period – in microseconds
3. job ID
4. task index
5. machineID
6. mean CPU usage rate
7. canonical memory usage
8. assigned memory usage
9. unmapped page cache memory usage
10. total page cache memory usage
11. maximum memory usage
12. mean disk I/O time
13. mean local disk space used
14. maximum CPU usage
15. maximum disk I/O time
16. cycles per instruction (CPI)
17. memory accesses per instruction (MAI)
18. sample portion
19. aggregation time
20. sample CPU usage: mean CPU usage during a random 1s sample in the measurement period

Within each measurement period, measurements are sampled at 1 second intervals. The one second readings are averaged over the duration of the measurement period to obtain the aggregate value. CPU usage (also referred as the CPU usage rate) is measured in units of CPU core seconds per second. Both CPU and memory usage information is not available in absolute values. The values are normalised due to privacy concerns. Disk I/O time is measured using the *blkio* subsystem. However, the trace provides no information for the disk usage in the Google distributed file system. It only gives information of the local disk usage. Additionally, disk usage required for binaries and read-only, pre-staged runtime files are not included.

3.3 Tracelog Statistical Analysis

The Google Cloud tracelog contains trace of over 12000 servers, 25 million tasks and 930 users over the period of a month. The tracelog includes detailed data of task submission patterns, scheduling informations and physical resource usage on the virtual machines (VMs). The total size of the data is approximately 250GB.

Trace span	29 Days	**Num of servers**	12532
Num of tasks	25752951	**Avg tasks / day**	888032.72
Num of users	930	**Avg users / day**	153.20
Avg task length	61575043.48	**Avg tasks / user**	3981.06

Table 3.1: Dataset Overview

Due to the large size of the trace, it is infeasible to analyse the entire duration of the trace at the same time. Instead, a sampling methodology to collect the resource utilisation statistics is more suitable. We randomly choose a sample of moments in the duration of the trace and collect the aggregate statistics during those moments.

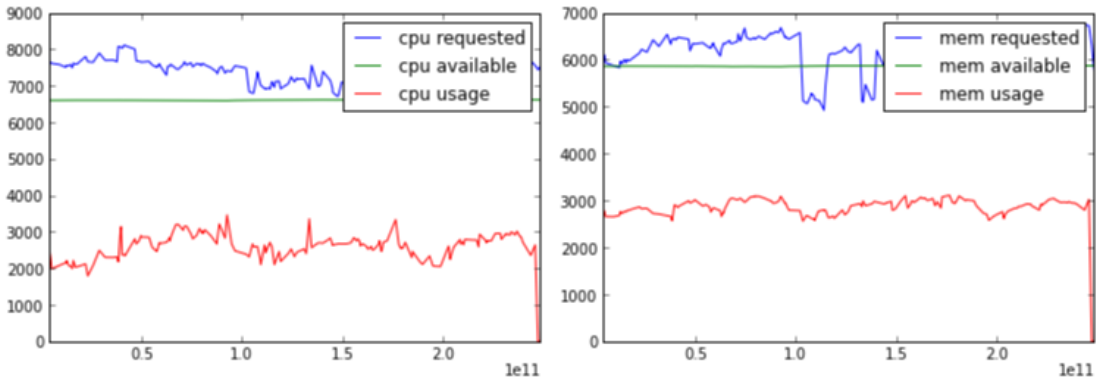


Figure 3.2: Total CPU and memory usage collected from the trace for a sample of 200 moments

While the CPU and memory have (on an average) been over-allocated by the cluster scheduler. The actual usage of CPU and memory although, hovers around 50% – slightly under for CPU and slightly over for memory. These observations seem consistent with the finding in the technical report by Reiss et al. [30].

Chapter 4

User Cluster Model

As stated earlier the user requests are primarily responsible for driving the workload in a cloud data center environment. Our methodology utilises this fact to build a workload model centered around accurately predicting the user behavior. We first divide the 930 unique users present in the Google Tracelog into clusters having common attributes like submission rate, requested memory, requested CPU and inter-arrival time between subsequent jobs.

Moreno et al. [7] identify this behavior using three important characteristics submission rate α, and requested amount of CPU β and Memory ϕ. They also define the task behavior using three essential parameters - task length χ, average resource utilisation for CPU γ and Memory π. The resultant cloud workload can be defined between a set of submitted tasks T and a set of unique users U. Where α, β and ϕ define each user profile u_i and χ, γ and π define each task profile t_i using the following equation.

$$U = \{u_1, u_2, u_3, ..., u_i\} \tag{4.1}$$

$$T = \{t_1, t_2, t_3, ..., t_i\} \tag{4.2}$$

$$u_i = \{f(\alpha), f(\beta), f(\phi)\} \tag{4.3}$$

$$t_i = \{f(\chi), f(\gamma), f(\pi)\} \tag{4.4}$$

We further build on this model and identify three characteristic parameters for user behaviour:-

- Submission Rate
- Average CPU Usage
- Average Memory Usage

4.1 K Means Clustering

The second step of our methodology is dividing the users into different clusters. This is essential in order to group users based on their attributes because users having similar behaviour are likely to exhibit common workload effects. k means clustering is a popular clustering approach that groups data points around the cluster centroid points while minimizing the total sum of squared errors for the generated clusters. One caveat of using this approach is that k, the resultant number of clusters, needs to be fixed before the clustering can be done. Hence, choosing an optimal value of k is essential for generating high quality resultant clusters. For our analysis we utilise the statistical method provided by Pham et al. [31]. The method quantifies the variability of the generated clusters and provides an estimate of the optimal value of k. The following equations summarise the conditions of the heuristic.

$$f(k) = \begin{cases} 1 & \text{if} \quad k = I \\ \frac{S_k}{\alpha_k S_{k-1}} & \text{if} \quad S_{k-1} \neq 0, \forall k > I \\ 1 & \text{if} \quad S_{k-1} = 0, \forall k > I \end{cases} \tag{4.5}$$

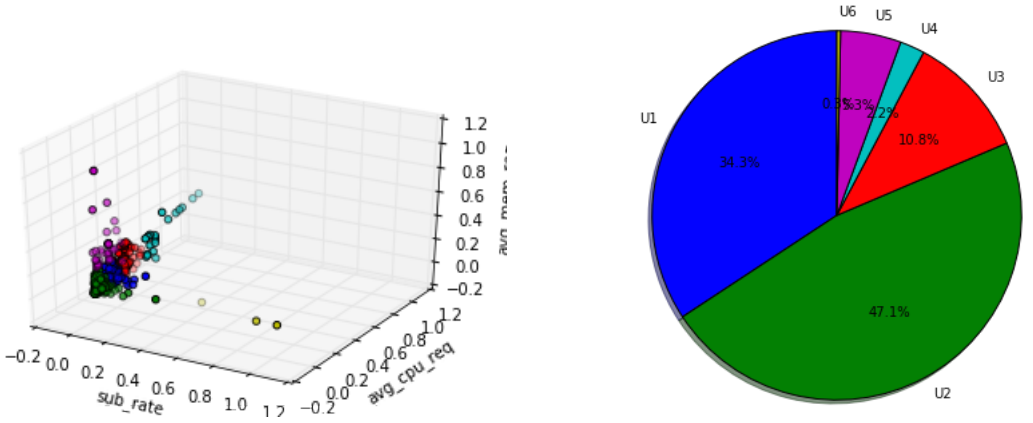
$$\alpha_k = \begin{cases} 1 - \frac{3}{4N_d} & \text{if} \quad k = 2, N_d > I \\ \alpha_{k-1} + \frac{1-\alpha_{k-1}}{6} & \text{if} \quad k > 2, N_d > I \end{cases} \tag{4.6}$$

Here $f(k)$ represents the total variability of the resultant clusters where k is the number of clusters in the k means algorithm. S_k is the sum of cluster distortions, N_d is the number of attributes of the data points and α_k is the weight factor of the based on the previous clustering. We run the clustering algorithm starting from 1 to the maximum desired number of clusters. At each value of k we calculate the variability $f(k)$. A value of k is suggested when the resultant variability is lower than or equal to 0.85.

$$f(k) <= 0.85 \tag{4.7}$$

We ran this heuristic on the Google Tracelog data sample to identify the number of user clusters and the optimal value of k was found to be 6.

4.2 Cluster Analysis



(a) User clusters (b) User population distribution

Figure 4.1: Clusterization of users

Figure 4.1 illustrates the final user clusters after the application of k means clustering algorithm. Majority of the user population is covered by U1 and U2 and have relatively low submission rate, average CPU and average memory usage. Three specific users, represented by U6, have significantly high submission rate than the rest and request large amounts of CPU and memory. U3 and U5 represent the CPU and memory intensive jobs respectively. U4 has a low submission rate but requests high CPU and moderately high memory resources. This behavior is characteristic of batch jobs that are submitted less frequently and run in the background. The coefficient of variance (C_v) is a strong indicator of the relative spread of the data. From the figure it can be seen that most of the users represented by U1 and U2 have a low resource usages and therefore have an extremely compact spread. The following table expresses the statistical properties in a systematic manner.

Table 4.1: Statistical Properties of User Clusters

		Submission Rate			**Requested CPU**			**Requested Memory**		
Cluster	**Population (%)**	***Mean***	***Stdev.***	***Cv***	***Mean***	***Stdev.***	***Cv***	***Mean***	***Stdev.***	***Cv***
U1	34.27	0.006	0.022	3.440	0.133	0.030	0.226	0.069	0.042	0.606
U2	47.14	0.006	0.022	3.918	0.041	0.026	0.638	0.035	0.032	0.912
U3	10.81	0.001	0.002	3.07	0.277	0.052	0.188	0.131	0.069	0.522
U4	2.16	0	0	2.91	0.631	0.154	0.244	0.202	0.106	0.525
U5	5.3	0.002	0.011	4.914	0.118	0.061	0.517	0.334	0.139	0.417
U6	0.32	0.819	0.183	0.224	0.041	0.017	0.41	0.046	0.016	0.351

Chapter 5

Prediction Model

Once we have divided the entire workload space into groups based on user behavior. The prediction problem essentially condenses to a time series prediction problem. The techniques described in this chapter are used frequently for time series analysis. We use Support Vector Regression (SVR) as our core prediction model. Since the time series is uni-variate we cannot use the traditional regression techniques. We need to use the previous values in the time series to predict the future values. We use the sliding-window technique to map the problem of resource usage prediction into a traditional regression problem. The sliding window approach maps a k sized window of the input vector x to the predicted value y that is r lags ahead.

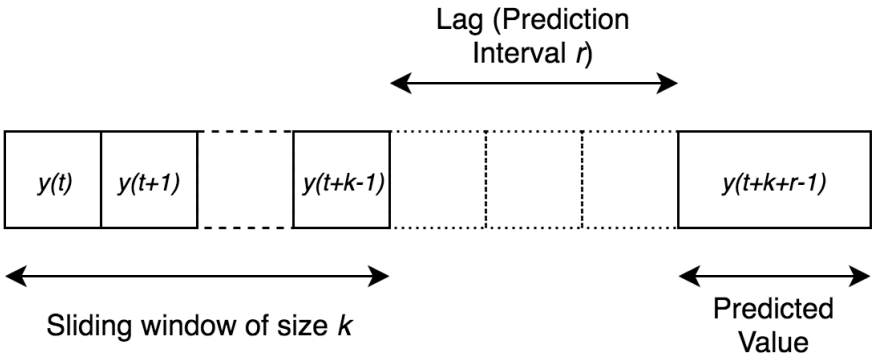


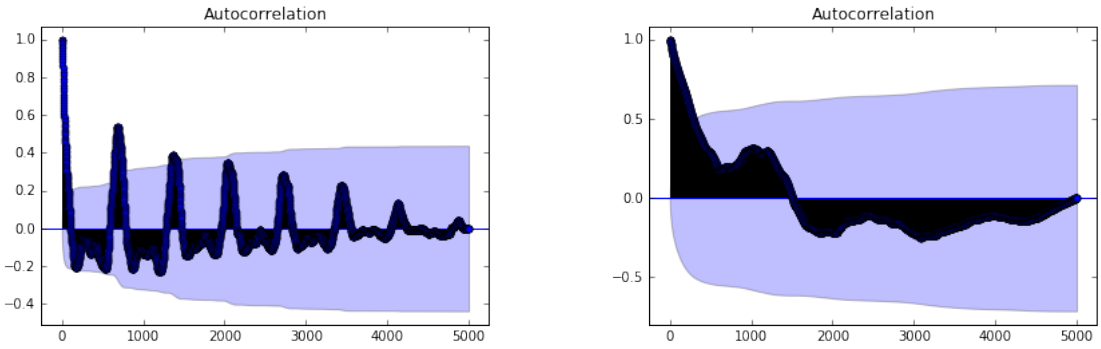
Figure 5.1: Sliding Window Technique

Figure 5.1 illustrates the basic idea of the sliding window technique. The sliding window interval of size k, $x = [y(t), y(t+1), ..., y(t+k-1)]$ serves as the input to the SVR model. The predicted value $y(t+k+r-1)$ is after a lag interval of size r which depends on the predictability of that particular resource.

5.1 Resource Predictability

Before we can train the SVR model for predicting the future resource usage for the different user clusters, we need to set an appropriate value of the prediction interval (lag) r. This value needs to be fixed based on domain knowledge and the predictability of the resource. At the minimum, we need to be able to set the lag r so that we can take preemptive measures when we observe a sudden increase/decrease in resource usage, for example booting up/reallocating virtual machines (VMs). This feature is extremely essential for implementing resource dynamic provisioning. To this end, we plot the autocorrelation function (ACF) for various time lags to identify the appropriate value of time lag. Autocorrelation of serial correlation is the similarity between the values of a signal and observations after a particular time lag.

$$ACF(r) = \frac{E[(Y_t - \mu)(Y_{t+r} - \mu)]}{\sigma^2} \tag{5.1}$$



(a) Strong daily patterns

(b) No daily patterns

Figure 5.2: ACF Plots for CPU usage

We plot the ACF plots for every user cluster and each resource type for lag values 0 to 7 days at 2 minute intervals. Figure 5.2 depicts two qualitatively different ACF plots of two separate user clusters. We see a strong daily patterns in the left ACF plot, hence the CPU usage for that user cluster can be predicted in well advance hence, we can set high value for the prediction interval. On the other hand the right ACF plot has a high ACF value for a small lag and the ACF value rapidly falls for further lags. This tells us that the CPU usage for that particular user cluster can only be predicted in the short-term. We see similar patterns for requested memory usage for various clusters.

5.2 SVR Model

Owing to the enormous size of the Google Tracelog dataset we employ a sampling technique to obtain the usage statistics of various resources like CPU and memory. Due to this the resultant time series is extremely noisy and unsuitable for applying the SVR prediction model. We need to extract the relevant signal from the noisy data but at the same time we must preserve the sudden spikes in resource usages. We achieve this using the Savitzky-Golay filter [32]. Savitzky-Golay filter improves the signal to noise ratio without completely flattening sudden peaks in the signal. This is done by fitting a low order polynomial in successive windows of the input signal. The following figure demonstrates the merits of Savitzky-Golay filter, other filters like moving average and erosion filters undermine the peaks in the noisy signal data, whereas Savitzky-Golay removes noise and preserves the peaks.

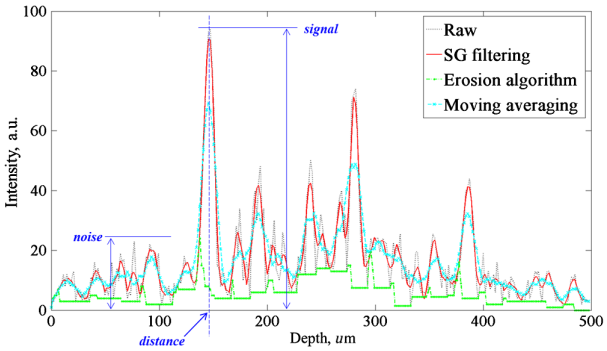


Figure 5.3: Savitzky-Golay Filter

Support Vector Regression (SVR) is an extension of the Support Vector Machine (SVM) for the regression problem. It essential has the same features as SVM but with an added error term. In this technique we try to find the hyperplane which minimises the total error term. The idea behind SVR is to find the optimal value of hyperplane parameters that maximise the margin of the decision boundary. The objective is to find optimal values for $y = \vec{w}.\vec{x} + b$.
Minimise:

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{5.2}$$

Constraints:

$$y_i - \vec{w}.\vec{x_i} - b \leq \epsilon + \xi_i \tag{5.3}$$

$$\vec{w}.\vec{x_i} + b - y_i \leq \epsilon + \xi_i^* \tag{5.4}$$

$$\xi, \xi^* \geq 0 \tag{5.5}$$

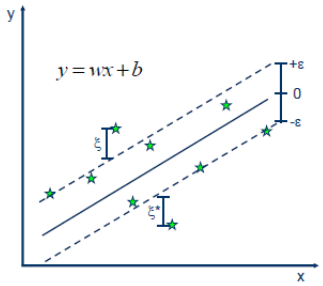


Figure 5.4: Support Vector Regression

The true power of SVR lies in the use of kernel functions. It can map the given data points to a higher dimensional space (possibly infinite) and apply SVR on the transformation without incurring a significant increase in the computation. This is done through the use of kernel functions K which allow dot product computation without actually transforming the original point to the higher dimensional space.

$$K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j) \tag{5.6}$$

The kernel function used in the implementation is the Radial Basis Function (RBF) given by the following equation:-

$$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2}, \gamma > 0 \tag{5.7}$$

We apply the aforementioned SVR model on each user cluster for every resource usage. The hyper parameters of the model are optimised using the grid search technique. Finally, models of all the user clusters (see Appendix A for individual user models) is combined into a single ensemble model. The following figure shows the performance of our prediction model for requested CPU usage using 40% of the data for training. The predicted value is shown in red while the actual values is shown in blue.

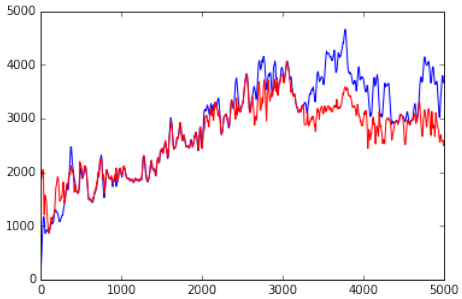


Figure 5.5: Prediction Model

5.3 Evaluation Criteria

We evaluate the fitness accuracy of our generated models based on a number of different metrics like PRED(25), R^2 Prediction accuracy, Mean Absolute Percentage Error (MAPE). In this section we describe the metrics and showcase our results.

Mean Absolute Percentage Error (MAPE)

The Mean Absolute Percentage Error is given by the following formula:-

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \tag{5.8}$$

Here y_i is the actual workload value while $\hat{y}_i$ is the predicted output. The lower the value of MAPE the better the performance of the prediction model.

PRED(25)

PRED(25) represents the number of values that have been predicted within a 25% error window of the actual value. It can be represented by the following equation:

$$PRED(25) = \frac{\text{No. of observations with relative error} \leq 25\%}{\text{No. of observations}} \tag{5.9}$$

The closer this value to 1.0 the better the prediction model.

R^2 Prediction Accuracy

R^2 Prediction Accuracy is a measure of the goodness of fit of the predicted curve to the actual curve. It can be represented by the following equation:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})} \tag{5.10}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, y_i is the actual workload value while $\hat{y}_i$ is the predicted output. The value of the R^2 prediction accuracy lies between the range $[0, 1]$. This metric determines how closely a model fits a given curve. The value of 1.0 represents a perfect fit by the forecasting model.

We used the k fold cross validation technique to identify the best prediction model for a given user cluster. The total training set was divided into k sets $y_1, y_2, ..., y_k$. The SVR model was trained by successively leaving out one set and training the model on the rest $k-1$ sets. The left out set is used as a validation

set to get an estimate of the generalisation accuracy. The length of the sliding window was set to a modest value of 10.
The following table shows the values of the error metrics for our prediction model.

Table 5.1: Prediction Accuracy of the SVR model

MAPE	R^2 **Prediction Accuracy**	**PRED(25)**
1.208	0.902	0.972

Chapter 6

Conclusion

In this work, we design a system that utilised user behavior modelling for the prediction of future workloads in a cloud data center environment. As a part of this system, we explore various methods of modelling the user behavior. In our system, we use k-means clustering algorithm to divide the users based on the attributes of the submitted jobs. This distinction allows use to train separated models for different clusters, thereby, giving the model flexibility to work in environments having a diverse application mix. Once we have segregated all the users in clusters we analyse the predictability of various resources using the autocorrelation plots, which tell us how far in advance can we accurately predict the future workload. We model the time series forecasting problem into a classical regression problem using the sliding window technique and use SVR build the regressor.

We have tested the aforementioned technique on a 7 day window of the Google Tracelog dataset and the preliminary results are promising. Such a system, when completely developed, will be an asset in developing efficient dynamic provisioning in cloud datacenters.

Chapter 7

Future Scope

The current mechanism only uses basic features like normalised values of requested CPU and memory resources, submission rate, inter-arrival time for the initial clustering of resources. While such features have shown to be useful in successfully clustering the different users, we need to identify complex features that capture the periodicity, burstiness of the user submissions. Incorporating such features into the k-means clustering algorithm will greatly increase the quality of the resultant clusters thereby, improving the performance of the overall system.

Secondly, in the existing system we are using the requested amount of resources (CPU and memory) as a method of distinguishing various user clusters. But, the requested resources are not a true reflection of the amount of resources a task is actually utilising. Although, the Google Tracelog has information about the task usage, the duration for which a task is running is extremely small compared to the trace duration. Thus, the actual usage statistics have an extremely bursty workload profile.

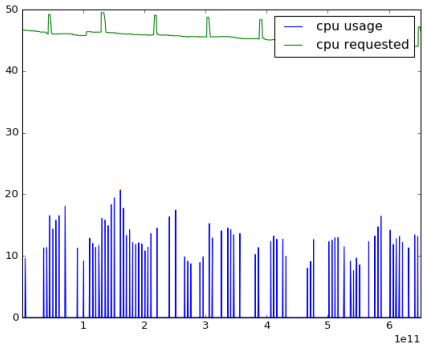


Figure 7.1: CPU requested vs CPU usage

As in any computing system, the amount of resources requested by a task

is not completely representative of the amount resource that is actually going to be used. Hence, including the usage statistics in the prediction system will significantly improve the accuracy of the system.

Bibliography

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] J. W. Smith and I. Sommerville, "Workload classification & software energy measurement for efficient scheduling on private cloud platforms," *arXiv preprint arXiv:1105.2584*, 2011.

[3] Q. Zhang, J. Hellerstein, and R. Boutaba, "Characterizing task usage shapes in google compute clusters," 2011.

[4] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: insights from google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[5] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An analysis of traces from a production mapreduce cluster," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 94–103, IEEE, 2010.

[6] S. Aggarwal, S. Phadke, and M. Bhandarkar, "Characterization of hadoop jobs using unsupervised learning," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 748–753, IEEE, 2010.

[7] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, Modeling and Simulation of Workload Patterns in a Large-Scale Utility Cloud," *IEEE Transactions on Cloud Computing*, vol. 2, pp. 208–221, apr 2014.

[8] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," *Proceedings - 2015*

IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015, pp. 465–474, 2015.

[9] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "An approach for characterizing workloads in google cloud to derive realistic resource utilization models," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pp. 49–60, IEEE, 2013.

[10] A. Bahga, V. K. Madisetti, *et al.*, "Synthetic workload generation for cloud computing applications," *Journal of Software Engineering and Applications*, vol. 4, no. 07, p. 396, 2011.

[11] A. Beitch, B. Liu, T. Yung, R. Griffith, A. Fox, D. A. Patterson, *et al.*, "Rain: A workload generation toolkit for cloud computing applications," *University of California, Tech. Rep. UCB/EECS-2010-14*, 2010.

[12] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available google cluster trace," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-95*, vol. 94, 2010.

[13] G. Wang, A. R. Butt, H. Monti, and K. Gupta, "Towards synthesizing realistic workload traces for studying the hadoop ecosystem," in *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 400–408, IEEE, 2011.

[14] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, "Characterization of storage workload traces from production windows servers," in *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, pp. 119–128, IEEE, 2008.

[15] S. Kavalanekar, D. Narayanan, S. Sankar, E. Thereska, K. Vaid, and B. Worthington, "Measuring database performance in online services: a trace-based approach," in *Technology Conference on Performance Evaluation and Benchmarking*, pp. 132–145, Springer, 2009.

[16] S. Sankar and K. Vaid, "Storage characterization for unstructured data in online services applications," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pp. 148–157, IEEE, 2009.

[17] A. Gulati, C. Kumar, and I. Ahmad, "Modeling workloads and devices for io load balancing in virtualized environments," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 3, pp. 61–66, 2010.

[18] O. Ozmen, K. Salem, M. Uysal, and M. Attar, "Storage workload estimation for database management systems," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pp. 377–388, ACM, 2007.

[19] J. Wilkes, "Traveling to rome: Qos specifications for automated storage system management," in *International Workshop on Quality of Service*, pp. 75–91, Springer, 2001.

[20] B. Abrahao and A. Zhang, "Characterizing application workloads on cpu utilization for utility computing," *Hewlett-Packard Labs*, 2004.

[21] J. P. Patwardhan, A. R. Lebeck, and D. J. Sorin, "Communication breakdown: analyzing cpu usage in commercial web workloads," in *Performance Analysis of Systems and Software, 2004 IEEE International Symposium on-ISPASS*, pp. 12–19, IEEE, 2004.

[22] S. Huang and W. Feng, "Energy-efficient cluster computing via accurate workload characterization," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 68–75, IEEE Computer Society, 2009.

[23] D. G. Feitelson, "Workload modeling for performance evaluation," in *IFIP International Symposium on Computer Performance Modeling, Measurement and Evaluation*, pp. 114–141, Springer, 2002.

[24] H. Li, "Realistic workload modeling and its performance impacts in large-scale escience grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 480–493, 2010.

[25] Y. Joo, V. Ribeiro, A. Feldmann, A. C. Gilbert, and W. Willinger, "Tcp/ip traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 2, pp. 25–37, 2001.

[26] X. Liu, J. Heo, and L. Sha, "Modeling 3-tiered web applications," in *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 307–310, IEEE, 2005.

[27] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pp. 87–92, IEEE, 2010.

[28] Google, "Google cluster data v2."

[29] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," *Google Inc., White Paper*, pp. 1–14, 2011.

[30] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, p. 84, 2012.

[31] D. T. Pham, S. S. Dimov, and C. Nguyen, "Selection of k in k-means clustering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 219, no. 1, pp. 103–119, 2005.

[32] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

Appendix A

Cluster CPU Usage Analysis

The data analysed in this study is a seven day snapshot from the 29 day Google Tracelog. It is evident from Figure A.1 and Figure A.2 the requested CPU workload has a general trend of increase. However, different user clusters exhibit different workload profiles. U1 and U2, which constituted the majority of the user population, have entirely different profiles in terms of periodicity. U1 has a general increasing trend having hourly patterns. Although the SVR model captures these hourly trends beautifully, it is unable to capture the sudden increase in CPU workload that occurs around day 5. U2 on the other hand exhibit strong daily patterns and that is modelled properly by the SVR predictor. As stated before U4 has an extremely low submission rate and has a high average CPU request. As a consequence of the low submission rate U4 has an extremely bursty CPU profile making it difficult to train the SVR predictor suitably. U6 has an extremely high submission rate but the number of jobs running at a time are negligible hence we can neglect its effects on the overall workload profile. The bursty nature of the U4's workload and sudden increase in U1's workload cause the sub-optimal behavior of the SVR predictor in the in the later stages of the seven day snapshot. This problem can be rectified by adding a linear regression component in the SVR predictor.

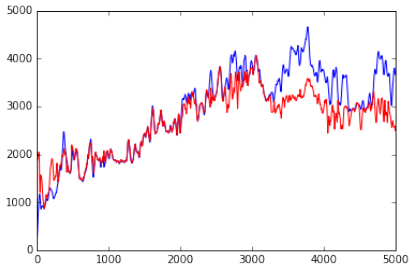
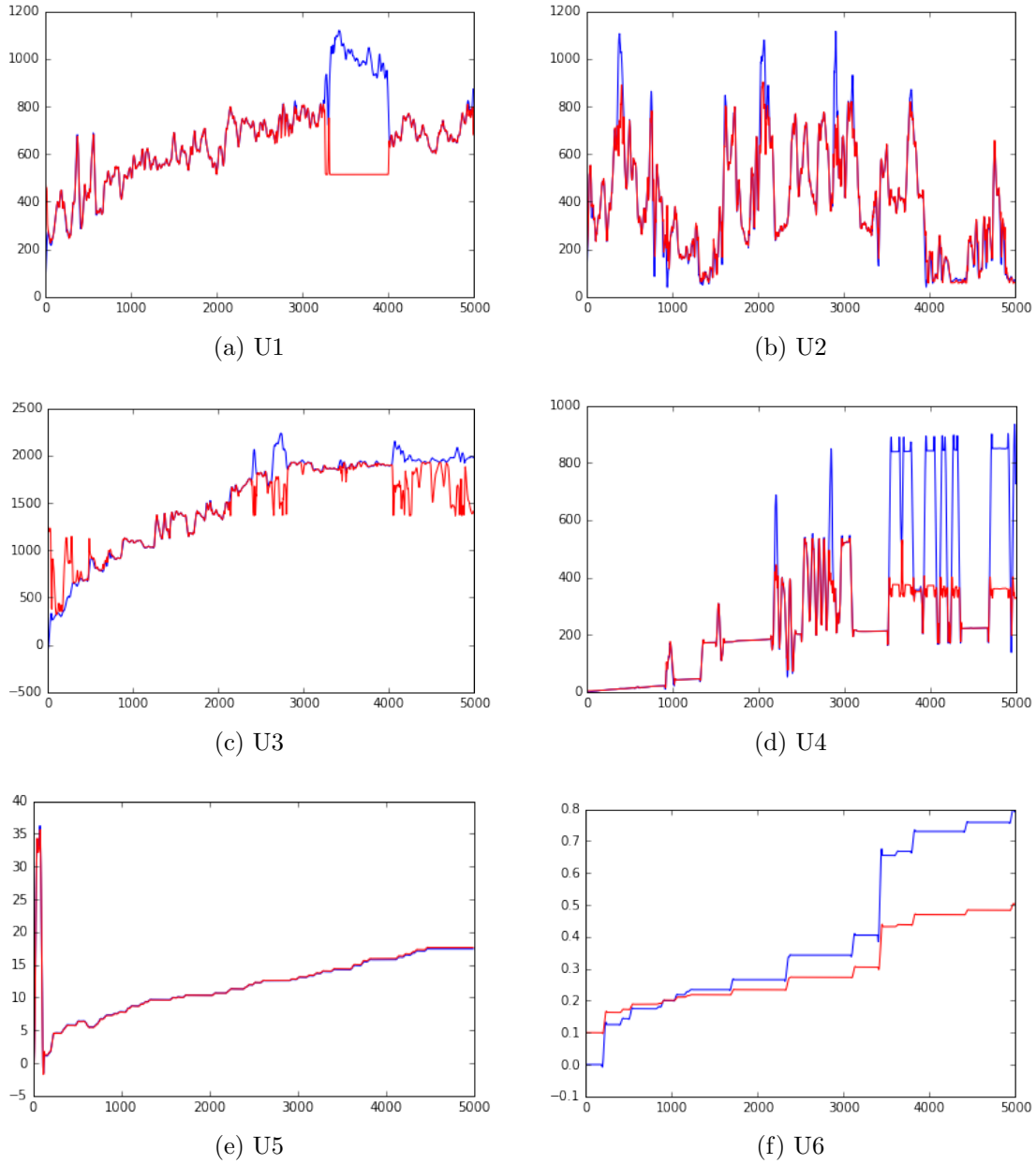


Figure A.1: Overall CPU Workload Profile



(a) U1

(b) U2

(c) U3

(d) U4

(e) U5

(f) U6

Figure A.2: CPU Prediction Models for User Clusters