SLA-based Virtual Machine Management for Heterogeneous Workloads in a Cloud Datacenter

Saurabh Kumar Garg[a], Adel Nadjaran Toosi[b], Srinivasa K. Gopalaiyengar[c], Rajkumar Buyya[b]

[a]*Faculty of Engineering and ICT*
Department of Computing and Information Systems
The University of Tasmania, Australia
[b]*Cloud Computing and Distributed Systems Laboratory*
Department of Computing and Information Systems
The University of Melbourne, Australia
[c]*Department of Computer Science and Engineering*
M. S. Ramaiah Institute of Technology (MSRIT)
Bangalore, Karnataka, India

Abstract

Efficient provisioning of resources is a challenging problem in cloud computing environments due to its dynamic nature and the need for supporting heterogeneous applications. Even though VM (Virtual Machine) technology allows several workloads to run concurrently and to use a shared infrastructure, still it does not guarantee application performance. Thus, currently cloud datacenter providers either do not offer any performance guarantee or prefer static VM allocation over dynamic, which leads to inefficient utilization of resources. Moreover, the workload may have different QoS (Quality Of Service) requirements due to the execution of different types of applications such as HPC and web, which makes resource provisioning much harder. Earlier work either concentrate on single type of SLAs (Service Level Agreements) or resource usage patterns of applications, such as web applications, leading to inefficient utilization of datacenter resources. In this paper, we tackle the resource allocation problem within a datacenter that runs different types of application workloads, particularly non-interactive and transactional applications. We propose an admission control and scheduling mechanism which not only maximizes the resource utilization and profit, but also ensures that the QoS requirements of users are met as specified in SLAs. In our experimental study, we found that it is important to be aware of different types of SLAs along with applicable penalties and the mix of workloads for better resource provisioning and utilization of datacenters. The proposed mechanism provides substantial improvement over static server consolidation and reduces SLA violations.

Email addresses: `skgarg@utas.edu.au` (Saurabh Kumar Garg), `adeln@student.unimelb.edu.au` (Adel Nadjaran Toosi), `kgs@csse.unimelb.edu.au` (Srinivasa K. Gopalaiyengar), `rbuyya@unimelb.edu.au` (Rajkumar Buyya)

Keywords: Cloud Computing, Service Level Agreement, Virtual Machine Migration, Resource Management

1. Introduction

Cloud computing has led to a paradigm shift where enterprises, rather than maintaining their own infrastructure, started to outsource their IT and computational needs to third party service providers [1]. The clouds are large scale outsourcing datacenters that host thousands of servers which can run multiple virtual machines (VMs) simultaneously. Therefore, they host a wide range of applications and provide users with an abstraction of unlimited computing resources on a pay-as-you-go basis.

While there are several advantages of these virtualized infrastructures such as on-demand scalability of resources, there are still issues which prevent their widespread adoption [2] [3]. In particular, for a commercial success of this computing paradigm, cloud datacenters need to provide better and strict Quality of Service (QoS) guarantees. These guarantees, which are documented in the form of Service Level Agreements (SLAs), are crucial as they give confidence to customers in outsourcing their applications to clouds [4]. However, current cloud providers give only limited performance or QoS guarantees. For instance, Amazon EC2 [5] offers only guarantees on availability of resources, not on performance of VMs [6][7].

Resource provisioning plays a key role in ensuring that cloud providers adequately accomplish their obligations to customers while maximizing the utilization of the underlying infrastructure. An efficient resource management scheme would require dynamically allocating each service request the minimal resources that are needed for acceptable fulfillment of SLAs, leaving the surplus resources free to deploy more virtual machines. The provisioning choices must adapt to changes in load as they occur, and respond gracefully to unanticipated demand surges. For these reasons, partitioning the datacenter resources among the various hosted applications is a challenging task. Furthermore, current cloud datacenters host a wider range of applications with different SLA requirements [6][8][9]. For instance, transactional applications require response time and throughput guarantees, while non-interactive batch jobs[1] are concerned with performance (e.g. completion time) [10]. Resource demand of transactional applications such as web applications tend to be highly unpredictable and bursty in nature [11][12], while demand of batch jobs can be predicted to a higher degree [13][14]. Hence, the satisfaction of complex and different requirements of competing applications make the goal of a cloud provider to maximize utilization while meeting different types of SLAs far from trivial.

Traditionally, to meet SLA requirements, over-provisioning of resources to meet worst case demand (i.e., peak) is used. However, servers operate most of

[1]In this paper, we use non-interactive batch jobs, batch application, HPC application and non-transactional application interchangeably.

the time at very low utilization level which leads to waste resources at non-peak times [15]. This over-provisioning of resources results in extra maintenance costs including server cooling and administration [16]. Some companies such as Amazon [5] are trying to utilize such slack of resources in the form of spot "instances" by renting them out at much lower rate but with low performance guarantees. Similarly, many researchers tried to address these issues by dynamic provisioning of resources using virtualization, but they focused mainly on scheduling based on one specific type of SLA or application type such as transactional workload. Although computationally intensive applications are increasingly becoming part of enterprise datacenters and cloud workloads, still research considering such applications is in infancy. Today, most of the datacenters run different types of applications on separate VMs without any awareness of their different SLA requirements such as deadline, which may result in resource under-utilization and management complexity.

To overcome these limitations, we present a novel dynamic resource management strategy that not only maximizes resource utilization by sharing resources among multiple concurrent applications owned by different users, but also considers SLAs of different types. We handle scheduling of two types of applications, namely, compute intensive non-interactive jobs and transactional applications such as Web server, each having different types of SLA requirements and specifications. Our strategy makes dynamic placement decisions to respond to changes in transactional workload, and also considers SLA penalties for making future decisions. To schedule batch jobs, our proposed resource provisioning mechanism predicts the future resource availability and schedules jobs by stealing CPU cycles, which are under-utilized by transactional applications during off-peak times.

The rest of the paper is organized as follows: Section 2 presents the related work in the area of resource provisioning in cloud datacenters. Section 3 describes the system model and its components in detail. We formally define the problem to be solved in Section 4. The algorithms used for hybrid resource provisioning are discussed in Section 5. The evaluation parameters, testbed and performance related issues are elaborated in Section 6. Finally, Section 7 concludes the paper and discusses future research directions.

2. Related Work

There are several works that relate to our research focus particularly in the area of dynamic resource provisioning and allowing mixed/heterogeneous workloads within a cloud datacenter. We broadly classify the works with respect to dynamic resource provisioning such as scheduling mixed workloads, SLAs, and auto-scaling of applications. The comparison of the proposed work with most important existing ones, with respect to various parameters, is summarized in Table 1. The details of the related works are discussed below.

Meng et al. [17] proposed a joint-VM provisioning approach by exploiting statistical multiplexing among the workload patterns of multiple VMs, so that the unused resources of a low-utilized VM is borrowed by other co-located VMs with high utilization. Zhang et al. [18] designed an approach to quickly reassign the resources for a virtualized utility computing platform using "ghost" virtual machines (VMs), which participate in application clusters, but do not handle client requests until needed. These works concentrate on fixed number of VMs, while we consider variable amount of incoming workload.

Vijayaraghavan and Jennifer [19] focused on the analysis and resource provisioning for workloads management with considerable network and disk I/O requirements. The management workloads scale with an increase in compute power in the datacenter. Singh et al. [20] argue that the workload in Internet applications is non-stationary and consider the workload mix received by a Web application for their mix-aware dynamic provisioning technique. Our paper also considers non-interactive applications. Goiri et al. [6] define a unique resource-level metric (i.e., SLA) for specifying finer level guarantees on CPU performance. This metric allows resource providers to dynamically allocate their resources among the running services depending on their demand. In contrast to the proposed work, they do not handle multiple types of SLAs and SLA penalty-related issues.

Steinder et al. [21] take advantage of virtualization features to co-allocate heterogeneous workloads on one server machine, thus reducing the granularity of resource allocation. Ejarque et al. [22] developed a working prototype of a framework for facilitating resource management in service providers, which allows both cost reduction and fulfillment of the QoS based on SLAs. In contrast, our work concentrates on handling multiple types of SLAs both for High Performance Computing (HPC) and Web based workloads with a new admission control policy. Quiroz et al. [8] presented a decentralized and robust online clustering approach for a dynamic mix of heterogeneous applications on clouds, such as long running computationally intensive jobs, bursty and response-time sensitive requests, and data and IO-intensive analytics tasks. When compared to our approach, the SLA penalties are not considered. Sotomayor et al. [23] developed a lease management architecture called Haizea, that implements leases as VMs, leveraging their ability to suspend, migrate, and resume computations and to provide leased resources with customized application environments. Again, this paper does not consider the issues of SLAs and QoS.

Wang et al. [34] evaluated the overhead of a dynamic allocation scheme in both system capacity and application-level performance relative to static allocation. They also provided implications and guidelines for a proper feedback controller design in dynamic allocation systems. In our work, the idea of dynamic allocation is extended for multiple types of workloads including HPC and Web. Sahai et al. [28] proposed a technique where guarantees have to be specified in terms of SLAs that have to be monitored and assured. In contrast, we propose architecture for specifying and monitoring SLAs to achieve the above. Van et al. [29] proposed a SLA-aware virtual resource management for cloud infrastructures, where an automatic resource manager controls the virtual environment which decouples the provisioning of resources from the dynamic placement of virtual machines. Even though the paper fulfills the SLA and operating costs, it does not deal with SLA penalty related issues. Carrera et al. [12] developed

Table 1: Summary of Comparison of the Proposed Work with Existing Literature

Parameter	Related Works	Our Work
Admission Control	None	√
Quality of Service (QoS)	{ [22][7][24][25][26][27] }	√
Service Level Agreement (SLA)	{ [28][26] [29][24][30][31][32][33] }	√
Dynamic Resource Provisioning	{[17][20][34][35][24][25] [30][36][31][32][26]}	√
SLA Penalty	[31][32][33][33][37][27]	√
Autoscaling	[25][30][36][31]	√
Mixed/Heterogeneous Workloads	{[21][8][12][38] [33][37][27] }	√

a technique that enables existing middleware to fairly manage mixed workloads both in terms of batch jobs and transactional applications. The aim of this paper is towards a fairness goal while also trying to maximize individual workload performance. But our aim is to efficiently utilize the datacenter resources while meeting the different types of SLA requirements of the applications.

Hue et al. [35] developed an efficient and effective algorithm to determine the allocation strategy that results in the smallest required number of servers. Paton et al. [38] describes how utility functions can be used to make explicit the desirability of different workload evaluation strategies, and how optimization can be used to select between such alternatives. In our paper, these ideas are mapped in a different manner to handle multiple types of SLAs with dynamic workloads. Nathuji et al. [7] proposed a QoS-aware control framework that tunes resource allocations to mitigate performance interference effects. These works only dealt with one type of application workloads and the number of applications is constant during resource allocation. On the other hand, our work focuses on designing approaches for handling multiple types of SLAs with admission control to allow more submission of applications.

There are other several works that focus on dynamic resource provisioning in cloud. Casalicchio et al. [24] proposed five resource allocation policies with workload prediction model to determine the allocation or deallocation of new resources. The work focuses on web-based applications not on batch type workloads. Sharma et al. [25] presented a dynamic provisioning system called Kingfisher which exploits differences in pricing and elasticity mechanisms for resource selection in order to optimize the incurred cost. In this work, single application provisioning is considered; SLA and mixed workload are not considered. Bonvin et al. [30] proposed a decentralized approach to dynamically adapt cloud resources for different applications so that they can meet their SLA requirements. The focus again was on web applications and considered public cloud. Fito et al. [36] proposed a web hosting system called CHP which provides scalable web services using cloud resources in outsourcing economic model. In our work, we have considered in-house cloud resources rather than external resources. Zhu et al. [31], similar to our work, presented a datacenter architecture for hosting multi-tier web applications using virtualized resources with an aim to satisfy user's SLA and maximize overall profit of IaaS providers. Wang et al. [32] considered a similar scenario of a service provider with several clients having individual requirements and SLAs. They proposed a resource management method to maximize profit of service provider by optimizing the resources allocated to applications executing MPI or MapReduce. This work does not consider the requirements of web applications. Casalicchio et al. [39] proposed a near optimal algorithm for allocating virtual resources to its client based on hill-climbing. The scenario here is slightly different than others as they also consider renting resources from other cloud providers in peak times. In contrast, we consider only in-house resources. Qui et al. [26], focusing on Online Transaction Processing systems, proposed an online algorithm for provisioning resources dynamically to minimize response time to end users. They utilized neural network based technique to predict user demand on the system. Islam et al. [40] proposed a Neural Network and Linear Regression based prediction model which is used to develop various resource provisioning strategies for dynamically allocating resources for upcoming demand. Minarolli et al. [41] presented a distributed Artificial Neural Network based resource allocation to optimize the trade-off between the conflicting objectives of satisfying applications QoS requirements and reducing power costs. These works again focussed on specific applications, not heterogeneous workloads, which are considered in our proposed work. Pfitscher et al. [42] presented a complimentary work and thus proposed a model to measure memory usage of virtual machines allocated to the customer application.

Antonescu et al. [33] proposed a prediction-based method of virtual machine allocation in order to maximize profit and SLA satisfaction. Xiao et al. [37] proposed a dynamic resource allocation system which optimizes the resource allocated to virtual machines based on application demand. Kim et al. [27] presented a virtual machine allocation mechanism that takes into account the correlated information among various virtual machines and thus minimizes energy usage. Even though these works consider performance profile of different types of virtual machines, they do not consider auto-scaling requirements of applications.

The main **contributions** of this paper lie within the design of an efficient admission control and scheduling mechanism to maximize utilization of cloud datacenters with the following salient features:

- adaptive admission control and dynamic resource provisioning facility that maximizes resource utilization considering SLAs of different types,
- consideration of multiple types of SLAs based on application requirements,
- integration of mixed/heterogeneous workloads (such as non-interactive and transactional applications) for better utilization of resources,
- variable penalties depending on the type of SLA violation, and

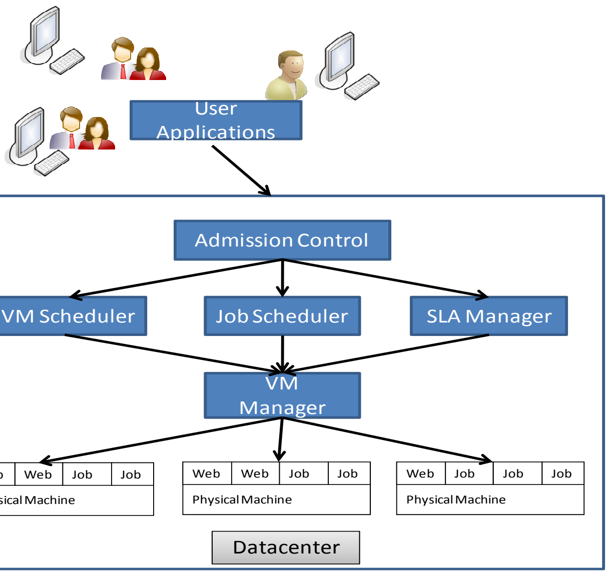


Figure 1: Datacenter Model

- support for auto-scaling of resources in order to satisfy SLAs and thus, cope with peak time demands.

3. System Model

3.1. Datacenter Model

We consider a virtualized datacenter with identical physical machines that can support the execution of multiple and heterogeneous VMs. It is assumed that each server is interconnected with a high-speed LAN network and high bandwidth link to the Internet. Figure 1 shows the key components involved in scheduling of an application on a VM. The *admission control* component decides if the requested VM (for an application) can be allocated and the QoS requirements can be met with a given number of available resources. If an application is accepted for execution, SLA is signed and penalty rates are negotiated with the user. The *VM scheduler* decides which server this new application (also known as VM) should be executed and thus generates execution schedules for all the accepted applications (VMs). The *VM manager* initiates a VM and allocates it to a server having the required capacity. The *job scheduler* schedules the applications on this newly initiated VM. The *SLA manager* monitors current SLAs and service level for each accepted application. It is assumed that the datacenter will receive two types of application workloads, i.e., transactional and non-interactive batch jobs. Since both applications have different QoS requirements, different charging models are used. Transactional applications are offered a set of VMs with varying capacity and hourly charges to allow the user to choose as per his/her requirements. The auto-scaling facility can also be provided if resource demand exceeds the allocated VM size. It is assumed in our model that combined resources promised for all the applications sharing the host will not exceed the total available resources such as bandwidth, memory and CPU. To run a batch job, the datacenter charges users based on the time taken to execute the job on a lowest capacity VM. In the next section, we discuss two types of application workloads considered in this work along with their SLA definitions.

3.2. Application Models and SLA

We have modeled the resource allocation to map the mix of two types of user applications. The transactional applications/workloads include Web applications, whose resource demand can vary with time based on incoming requests to the application. On the other hand, for the non-interactive workloads, we model the HPC compute intensive bag of task applications. “Bag of tasks” are a common type of parallel applications highly used in a variety of scientific applications, such as parameter sweeps, simulations, fractal calculations, computational biology. Most of these scientific applications include independent single-machine jobs (tasks) grouped into a single bag of tasks [43]. It is assumed that there is no data communication between each task.

The SLA model that is used as the basis for determining the VM capacity is discussed in detail below. For scheduling purposes, we consider a discrete-time scenario in which time is divided into intervals with identical length (T). During each interval, the system checks the requirements of each application and their SLAs, and allocates resources accordingly.

SLA model for transactional workload

A user gives QoS requirements in terms of response time rt_i of each transactional application i. This response time requirement can be translated to CPU capacity α_i needed to achieve this response time [12]. The SLA is violated if the capacity C_t allocated to the application is less than the required capacity at time t. A penalty q will incur whenever there is any violation of SLA. Thus, the net amount the user needs to pay at the end of the period (t_1, t_2) would be: $r * \alpha_i * (t_2 - t_1) - Q(Penalty\_Type)$, where r is amount of money charged by the provider per unit capacity per unit time for the provisioned capacity and $Q(Penalty\_Type)$ is the total penalty that must be credited to the user by the provider for the SLA violation according to the agreed type of penalty. Here, the user can choose three types of penalties [44]:

- **Fixed Penalty:** For fixed penalty, $Q(.)$ is independent of the time interval for which service is not provided as per the SLA. The fixed penalty q ($Q(fixed\ penalty) = q$) is charged whenever the cloud provider fails to fulfill the agreed capacity demand. For example, given that there are two time intervals $(t1, t2)$ and $(t3, t4)$ in which the provider fails to provide the agreed capacity, q will be charged for each of them independent of the interval length.

- **Delay-Dependent Penalty:** The penalty is proportional to the delay incurred by the service provider in returning the capacity. If q' is the agreed penalty per unit time in the SLA and t_1 and t_2 are the time instants between which the capacity for the current demand is less than the reserved capacity (according to the SLA), then the penalty to the service provider is calculated as: $Q(delay\ dependent) = (q' \times (t_2 - t_1))$.
- **Proportional Penalty:** This is also a form of delay-dependent penalty, where the penalty to be credited to a user is proportional to the difference between the user's provisioned capacity C_1 and its current allocation C_2. If q'' is the agreed penalty rate per unit capacity per unit time, t_1 and t_2 are the respective times when the capacity C_1 was requested and capacity C_2 allocated, then the penalty is given as: $Q(Proportional) = (q'' \times (t_2 - t_1) \times (C_1 - C_2))$

The SLA also contains the information regarding the user's choice of opting for the auto-scaling facility. If the user chooses this facility, when demand for resources increases beyond the initial requested amount, more resources are allocated to meet this spike. They will be automatically reduced when demand decreases below a threshold. Let $\Delta_c(t)$ be the extra CPU capacity that must be allocated to the application at time t to handle the spike in resource demand such that response time remains below the agreed rt_i on the SLA. If $\Delta_c(t) \geq 0$, then a VM of size $\Delta_c(t)$ will be initiated. The cloud provider can charge an additional amount for offering such flexibility.

SLA model for non-interactive batch jobs

Deadline and the amount of CPU time allocated is used as QoS requirements for batch jobs. Deadline is a commonly used QoS parameter defined by various works. In the context of cloud computing, these jobs also require performance based guarantees. Since the performance of allocated VMs can vary with the usage of datacenter, we define SLA as the p amount of CPU Cycles (calculated in terms of Millions of Instructions (MIs)) to be provided by the cloud provider before the deadline d in order to ensure successful completion of the job. Let db be the delay with respect to deadline before allocating p CPU cycles. For example, a user specify in the SLA that he requires resources to be allocated to the job equivalent to X number of CPU cycles before the deadline d. If the job has utilized the required resources (i.e., X CPU cycles) at time t and t is larger than d then the delay is $t - d$, otherwise it is zero. If the provider fails to complete the given job within the deadline, the following penalty applies: $Q(batch) = y \times db$, where y is the penalty rate.

4. Problem Statement

The aim of cloud service providers is to maximize the utilization of their datacenters by efficiently executing the user applications using minimal physical machines. Let n be the number of applications (virtual machines) to be executed in the datacenter at time t and K be the required number of physical machines with CPU capacity C_k to run these applications. Let $c_i(t)$ be the CPU capacity allocated to the virtual machine running application i at time t and α_i is the promised CPU capacity specified in the SLA for transactional applications. For batch jobs, $p_{rem}(i)$ is the remaining number of CPU cycles in MIs that need to be executed before the deadline d_i. The problem can be formulated as an optimization problem at time t as follows:

$$\begin{aligned}
&\text{minimize } K \\
&\text{subject to} \\
&\quad \sum_{i=1}^{n} c_i(t) x_{ik} \leq C_k \text{ where } 0 \leq k \leq K, \\
&\quad c_i(t) \leq \alpha_i \ \forall \ i \in \{\text{Transactional application}\} \\
&\quad c_i(t) \geq \frac{p_{rem}(i)}{d_i - t} \ \forall \ i \in \{\text{Batch jobs}\}
\end{aligned} \tag{1}$$

x_{ik} is 1 if application i is running on physical machine k, otherwise it is zero. The above optimization problem has two types of constraints: a) due to the capacity of the physical machine and b) due to CPU capacity promised for a transactional application and batch application. Given the online nature of the problem and the variation of applications' requirements with time, it is not trivial to solve the above optimization problem. Moreover, this mapping problem maps to the bin-packing problem which is known to be NP-hard. Hence, we use a heuristic approach to solve the above mapping problem.

5. Admission Control and Scheduling Policy

As discussed in the earlier sections, we consider the requirements of two different application workloads before accepting new requests and also while serving the accepted one. The main idea is to monitor the resource demand during the current time window in order to make decision about the server allocations and job admissions during the next time window. Datacenter resource allocation is monitored and reconfigured in regular intervals. At a given point of time, it is assumed that if a host is idle for a certain amount of time or not running any application, then it will be switched-off. In each scheduling cycle[2], admission control and scheduling can perform three functions:

- Admission Control: It makes the decision to accept or reject a new application (transactional or batch) based on present and future resource availability.

[2]Scheduling cycle occurs at the begining of every slot T.

- SLA Enforcement: The resource demand of each application is monitored by the SLA Manager based on agreed QoS level guaranteed in the SLA. In this step, dynamic resource reconfiguration is done to handle the demand bursts of transactional applications and to meet SLA requirements.

- Auto-Scaling: If the resource demand of any application exceeds the requested (reserved) capacity in the SLA, a new VM instance can be initiated based on the resource availability and an auto-scaling threshold specified in the SLA by the VM Manager.

All the above operations depend on the forecasting module which predicts the future resource availability and the expected resource demand of each application. We will discuss the above three functions and methodologies in the following sections. The *batch job queue* component represents the queue of VMs (corresponding to each batch job) that are waiting for execution.

The *batch job queue* is sorted based on a *threshold time* up to which a batch job can be delayed without any penalty. Let d be the deadline, MI be the CPU cycles (Millions of Instructions (MI) [45]) required for completion of job, and C_{min} be the Million of Instruction Per Second (MIPS) [45] of the smallest standardized VM offered by the cloud provider, then the threshold time $threshTime(i)$ for batch job i is calculated as below.

$$threshTime(i) = d - \frac{MI}{C_{min}}$$

Assuming that there is no delay due to I/O, threshold time is an approximation of the delay that can be tolerated in the execution of jobs and is used to decide whether to allocate the resources to the job at a given time or not.

5.1. Forecasting model

Artificial neural networks are computational models inspired by working of neurons on the brain that are capable of learning. They can compute values from inputs through the complex network. They consist of sets of numerical parameters (Weights) that are tuned by a learning algorithm and can approximate non-linear functions. The weights represent connection strengths between neurons and are activated during training and prediction.

Artificial Neural Network (ANN) has proven itself as an eminent method for irregular series and multiple-period-ahead forecasting [46][47]. Such neural network techniques have also been used and well studied in previous works for distributed application scheduling in the context of Grids [48]. In this paper, we have used ANN to forecast future CPU utilization for VMs of transactional applications based on past data. It is important to note that it is not our goal to determine the best prediction technique here. However, our contribution is that if the resource requirements of an application are well predicted, we can maximize the resource utilization considering different SLA requirements of different applications. In addition, our scheduling policy is designed to be robust and tolerant towards incorrect predictions of the forecasting model. We

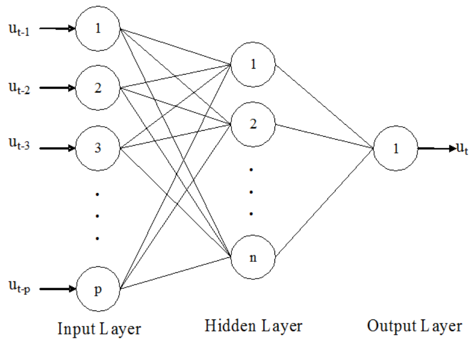


Figure 2: The 3-layer feed-forward artificial neural network

considered CPU utilization as an univariate time series with equally spaced time intervals of 5 minutes. The term univariate time series refers to a time series that consists of set of values over time of a single quantity observed sequentially over equal time increments. Our analysis of the CPU utilization in the used traces showed that 5 minutes interval is small enough to capture the CPU utilization variations for our purpose. Our forecasting ANN is modeled with three layers, one input layer, one output layer and one hidden layer between the inputs and output as shown in Figure 2.

The forecasting model has been formed based on the multi-layer feed-forward neural network. The standard *Back Propagation* (BP) algorithm with minimum *Root Mean Square Error* (RMSE) is used to train the neural network. RMSE is a measure to quantify differences between forecasted value and the true value of the quantity being estimated. It is calculated as follows:

$$RMSE = \sqrt{1/n \sum_{i=1}^{n} (\hat{y}_i - y_i)^2} \tag{2}$$

where $\hat{y}_i$ and y_i are the predicted value and the true value of the ith elements of the sample set respectively and n is size of the sample set.

A *hyperbolic tangent sigmoid* function is used as the transfer function for all hidden layer neurons and a *linear transfer* function is used for all output layer neurons. All neural networks are fully connected, that is, all input nodes are connected to all the hidden layer nodes, and all hidden layer nodes are connected to all linear output nodes. As observed from Figure 2, the network is a single output neural network that actually performs the following mapping from the inputs to the output: $u_t = f(u_{t-1}, u_{t-2}, ..., u_{t-p})$, where u_t is the utilization at time t, p is the dimension of the input vector (the number of past observed utilization) used to predict the future, and f in general is a nonlinear function determined by the network structure and its weights. Since web applications usually require long-running VMs, we assume that there is enough available data to train the neural networks at training phase. Each VM is considered as a web application and a neural network was created for that.

In the experiments[3], a time series vector of CPU utilization for one week for each VM is given as an input and CPU utilization of that VM is predicted for one day ahead. To avoid the overhead of training and forecasting, this process was done in offline mode before running the experiments. The inputs of the network are the past observations of the VM CPU utilization and the output is the forecasted value. Each input pattern is composed of a moving window of fixed length along the series. One of the challenging issues in using neural network for univariate time series forecasting is how to determine the most suitable number of input and hidden nodes in the network. In order to find the most suitable number of nodes in input layer (window length) and hidden layer, in the training phase combination of different number of nodes in input and hidden layer has been created and the performance of the network has been tested through validation data. To be precise, we trained the network with initial 80% of the one week input data and tested it with the remaining 20%. Optimal network structure is selected based on the minimum RMSE of the test result. Each network was trained for 1000 epochs when the learning rate was set at 0.1. Early stopping method has been used to solve the over-fitting problem.

In order to demonstrate the accuracy of the prediction model, a comparative analysis is done between a commonly used forecasting technique, i.e., generalized autoregressive conditional heteroskedasticity (GARCH) and the adopted ANN. GARCH has been widely used in financial and econometric modeling and analysis to characterize time series data. The trained ANN for a sample VM was used to predict the next two hours utilizations and the result for actual and predicted values by ANN and GARCH have been plotted in Figure 3 for every 5 minutes interval. Numbers in x-axis indicate the index of the 5 minutes intervals (e.g. 3 stands for minute 15).

From the prediction plots, it can be observed that the proposed ANN model performs better, when compared to other linear models like GARCH [46, 47]. Figure 3(a), (d) and (e) show that the forecast using ANN model is quite close to the actual data, although it starts deviating with time, while in Figure 3 (b) and (f), predicted values by ANN model almost overlapped with actual values. In contrast, for Figure 3 (c) and (e), we find that the predicted utilization is most of the times above the actual utilization. The reason for such different behavior of ANN model for different machines is the change in the CPU utilization of training and actual CPU utilization data in the prediction window. In the cases where the change is large, we find that ANN model prediction deviates significantly from the actual value. However, we can notice that not only the deviation is not very high in most of the cases but also ANN model slightly overpredicts the CPU utilization. From Figure 3, it can be closely observed that the prediction is near to accuracy and in very few cases the ANN model predicts lower utilization when compared to original values and in some cases (e.g.,

[3]Details about the workload used in this experiment are given in Section 6.1.

Figure 3 (c)) GARCH performs better than ANN model. The ANN model prediction overestimates the actual value which is better for our scheduling scheme, as under-estimation may lead to unnecessary SLA violations. In addition, even though 100% accuracy cannot be achieved through any forecasting models, nonlinear models such as ANN serve the purpose better when compared to many other statistical methods.

5.2. Admission Control and Scheduling

In this process, the decision is made on whether an application can be accepted and executed based on the available resources and QoS requirements of the application. To estimate how much resources will be available, we used ANN forecasting model (described in the previous section) which predicts future demand of all accepted applications. If free resources on a host are sufficient to run the application, it is accepted. The scheduling of the application is based on the following simple but effective principle:

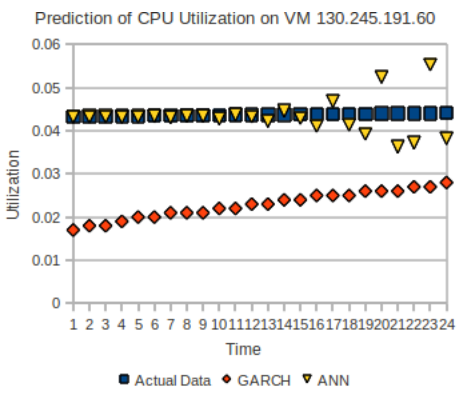
"*In the initial allocation, for transactional applications, reserved resources are equivalent to their peak demand. For non-interactive jobs, the scheduler tries to allocate slack of resources remaining on a host which is running a transactional application. At regular intervals, consolidates those VMs deployed on hosts which are not running any non-interactive jobs*".

The job scheduler always tries to run non-interactive jobs with transactional applications whose resource demand is highly dynamic. This strategy aims to minimize the SLA violations and network overhead of VM migration. It is clear that, this strategy differs from general approaches used in previous works where, during migration, multiple type of SLAs are not considered.

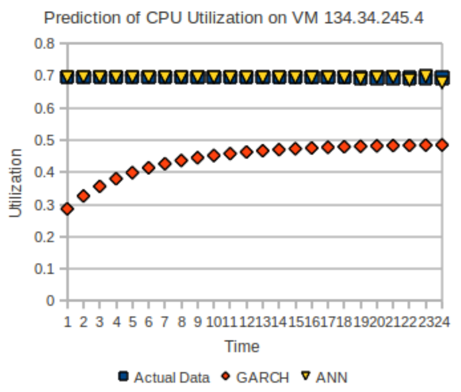
Each type of application has different resource demand characteristics and different QoS requirements. The details of admission control and scheduling for each of them are described below:

Non-Interactive Job: The QoS requirements and SLAs for this type of applications are already discussed in previous sections. Thus, in this section we discuss directly the methodology used for admission control. The datacenter accepts request for execution of a non-interactive batch job only when it can be allocated with the required amount of CPU capacity before its deadline. The steps used in this process are described in Algorithm 1.

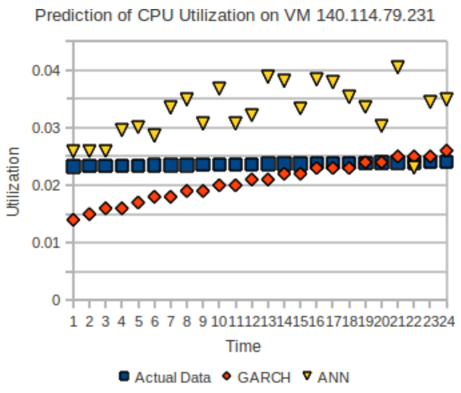
At any moment, when a job arrives for execution, the datacenter has some servers running the workload and others which are switched-off to save power. Thus, the VM scheduler first checks the resource availability of active servers where a Web application is already hosted. The resource availability on each host is calculated based on the forecasted server demand. To know whether the job can be successfully executed within its deadline, the start and completion time of each job is estimated based on resource availability on each host. Figure 4 shows the four possible ways a job can be scheduled. If the job can be completed before the deadline on any host, the job is accepted by the datacenter for execution, otherwise it is rejected. SLA is signed between both the user and provider.



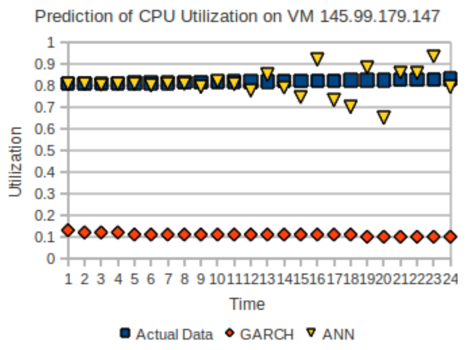
(a)



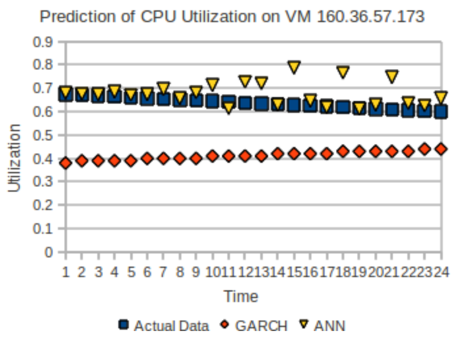
(b)



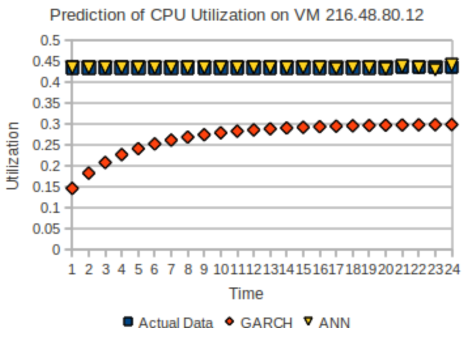
(c)



(d)



(e)



(f)

Figure 3: Comparison of prediction accuracies of GARCH and ANN. Numbers in x-axis indicates the index of each 5 minutes interval for the period of 2-hour prediction.

Algorithm 1 Admission Control for Non-Interactive Jobs

Input: Request for Execution of a non-interactive batch Job

Output: Accept/Reject the job with respect to available capacity versus deadline

Notation: Exp_St_j and Exp_Et_j are the expected start and finish time of the non-interactive job on Server S_j, $Deadline$ is the deadline of the submitted job, and $threshTime$ is the time by which job must start execution to finish before deadline.

```
1: for All host $j$ Which are $ON$ do
2:     Forecast the demand for the server resources by other hosted applications
3:     Calculate $Exp\_St_j$ // Expected Start Time
4:     Calculate $Exp\_Et_j$ // Expected Finish Time
5: end for
6: $\forall$ j, Calculate: $Exp\_ET = \min \{Exp\_Et_j\}$
7: if $Exp\_ET < Deadline$ & $threshTime > 0$ then
8:     Find the $S_j$ with minimum $Exp\_St_j$ & hosted Web VM and deploy New
       HPC VM
9:     If not found, deploy New HPC VM along with existing HPC VM
10:    if $Exp\_St > Current\_Time$ then
11:        Queue up the Job in the Batch Job Queue
12:    else
13:        A VM is Initiated on the Physical Host w.r.t $Exp\_Et$ by VM manager.
14:    end if
15: else
16:    Execute the Job by Initiating the VM on a Server in Switched Off Mode,
       if its deadline can be met.
17:    if No server is available then
18:        reject the request
19:    end if
20: end if
```

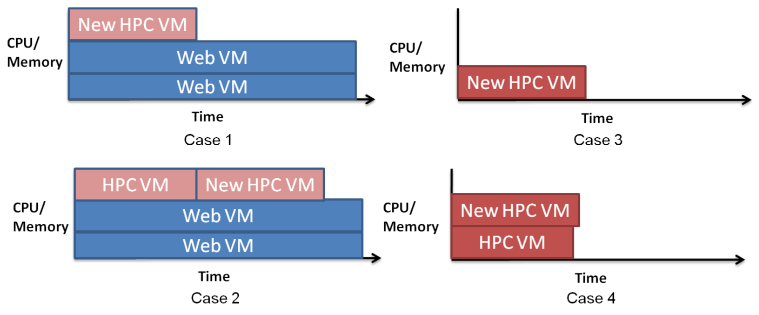


Figure 4: HPC Job Scheduling
(*HPC VMs allocated static amount of resources are indicated by red color, other HPC VMs are indicated by light red color.*)

For scheduling, a VM based on QoS requirements of a given job is created. The VM is deployed on a server (S_j) where the expected start time (Exp_St) of job is minimum and with an already hosted Web VM. Thus, for execution in the new HPC VM, it will utilize the slack resources on the server and such VM is termed as *dynamic HPC VM*. These HPC VMs are allowed to migrate and the resources allocated to them are also dynamic. In this case, if on the given penuthere is already one HPC VM running, then the new VM is deployed after the completion of the job on the current VM (as demonstrated in Figure 4 case 2); otherwise, the new VM is deployed immediately (as demonstrated in Figure 4 case 1).

If no such server is available and the threshold time of job is greater than the next scheduling interval, then the deployment of VM is delayed and the job is inserted into the batch job queue. This is done to exploit errors in the resource demand forecast. If the actual demand of the resources is less than the forecasted one, some HPC jobs can finish before their estimated completion time. Thus, the new VM can be deployed without switching on new servers. If the threshold time of the job is less than the next scheduling interval, then either it is scheduled on a server running only HPC VMs (as demonstrated in Figure 4 case 4), or on a new server which is switched on (as demonstrated in Figure 4 case 3). In this case, the VM deployment is static. Therefore, these VMs are not allowed to migrate, and the resources allocated to them will not be changed until the agreed amount (as in SLA) of CPU cycles is used for its execution. These VMs are called *static HPC VM* indicated by a different color in Figure 4 (case 3 and 4). This is done to avoid the effects of rescheduling VMs on the network and on the other VMs hosted on the server.

Transactional Applications: As discussed previously, a user who wants to run a transactional application can ask for VMs of different standard sizes offered by the cloud providers. Let the user requests a VM with capacity C_k. A request is accepted when the VM scheduler can schedule the VM with capacity C_k on any server assuming all hosted Web VMs are running at 100% utilization

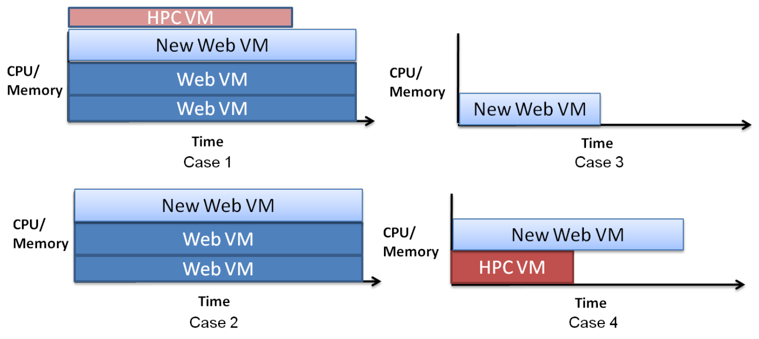


Figure 5: Transactional Application Scheduling

and without considering resources used by dynamic HPC VMs. The Web VM is scheduled based on the best-fit manner. As Figure 5 shows, there are again four possibilities to deploy the new Web VM.

Case 1: If new Web VM is deployed on a server hosting both a dynamic HPC VM and Web VMs, then the future resources available to the dynamic HPC VM will get affected. This scarcity of resources will delay the completion time of HPC job. Thus, the HPC VM is paused and rescheduled (migrated) to other servers if the HPC job is missing its deadline after the deployment of the new Web VM. The rescheduling of HPC job is done by the VM scheduler in such a way that the minimum penalty occurs due to SLA violation.

Case 2 - 4: In these cases, while scheduling a new Web application, the full utilization of resources by other VMs is considered. Therefore, there will not be any perceptible effect on the execution of other VMs. It can be noted that in Case 4, since a static HPC VM (denoted by red color) is hosted, the available resources on the server for executing the new Web application will be the amount of resources unused by the HPC VM.

5.3. SLA Enforcement and Rescheduling of VMs

To fulfill SLAs, the regular SLA monitoring of each application is required, since our initial allocation and admission control is based on forecasting. The forecasting model only gives approximate future resource demand and thus there may be a time when SLAs are violated. The steps involved during SLA enforcement process is given in Algorithm 2.

In every scheduling cycle,the VM scheduler performs the following actions: a) enforce SLAs and b) schedule the jobs from batch job queue, and c) consolidation.

For SLA enforcement, the resource demand for each transactional application until next scheduling cycle is recalculated (Algorithm 2: Line 1-2). If any Web VM requires less than the forecasted (currently allocated) capacity, then the extra resources are allocated to the HPC VM running on the same host (Algorithm 2: Line 3-5 and 20-21). Otherwise, if the Web VM requires more

Algorithm 2 SLA Enforcement and Rescheduling

Input: Current Utilization of VMs and Current Resource Demand
Output: Decision on Capacity Planning and Auto-scaling
Notations: VM_{web-i}: VM running Transactional (Web) Applications; $CurResDemand(VM_{web-i})$: Current Resource Demand; $CurAllocResVM_{web-i}$: Current Allocated Capacity; $ReservedRes(VM_{web-i})$: Reserved VMs Capacity Specified in SLA; VM_{hpc-i}: VM running HPC Application

```
1: for Each $VM_{web-i}$ do
2:   Calculate the current resource demand $CurResDemand(VM_{web-i})$
3:   if $CurResDemand(VM_{web-i}) < CurAllocResVM_{web-i}$ then
4:     Reduce the resource capacity of $VM_{web-i}$to match the demand
5:   else
6:     if $CurResDemand(VM_{web-i}) \leq ReservedRes(VM_{web-i})$ then
7:       Increase the resource capacity of $VM_{web-i}$ to match the demand
8:       Reduce correspondingly the resource capacity allocated to HPC application ($VM_{hpc-i}$ ) on the same server
9:     else
10:      if SLA contains Auto-scaling Option then
11:        VM manager initiates new VMs and offload the application demand to new VMs
12:      end if
13:    end if
14:  end if
15: end for
16: for Each Batch Job $VM_{hpc-i}$ do
17:   if slack resources available on the server where HPC VM is running then
18:     Allocate the slack resources
19:   end if
20:   Recompute the estimated finish time of the job
21:   Reschedule the Batch Job VM if missing the deadline.
22: end for
```

resources than allocated ($\leq$ promised capacity in SLA), then the resources allocated to the VM are increased to match the demand (Algorithm 2: Line 6-7). Correspondingly, the resources allocated to the HPC VM will be reduced (Algorithm 2: Line 8). If the Web VM requires more resource capacity than specified in the SLA, the decision is made based on whether the user has opted for auto-scaling or not (Algorithm 2: Line 10-15). This process is repeated for each transactional application.

After that, for each HPC job, its VM capacity is increased if some slack resources is available on the server where the VM is hosted (Algorithm 2: Line 17-18). Based on allocated resources to the HPC VM, the job completion time is recomputed (Algorithm 2: Line 20). If any HPC job is expected to miss its deadline, its corresponding VM is migrated and scheduled on another server using strategies discussed in the previous section (Algorithm 2: Line 21). The process is repeated for each HPC job (VM) in batch job queue. The VM manager consolidates Web VMs which are running on servers having no HPC VM. For consolidation, VM manager uses a greedy algorithm where VMs are sorted in decreasing order of CPU utilization and mapped to physical machines in a first-fit manner [49]. If due to consolidation, some Web VMs are short of resources, SLA can be violated. In this case, HPC VMs will be migrated to a server where the minimum penalty occurs due to SLA violation.

6. Performance Evaluation

We simulated a datacenter that comprises 1500 physical nodes. Simulation approach gives advantage of repeating the experiments under a similar environment. Thus, it allows the comparison of different scheduling strategies. In addition, it is considerably difficult to obtain workloads for several applications from commercial cloud providers as they are considered in this work. The CloudSim toolkit [49] has been chosen as a simulation platform since it allows the modeling of virtualized environments with on-demand resource provisioning and management.

Each node is modeled to have one CPU core with performance equivalent to 4000 Million Instructions Per Second (MIPS), 16 GB of RAM, 10 GB/s network bandwidth and 1 TB of storage. We consider four different types of VMs with 1000, 2000, 2500 or 3500 MIPS. The smallest instance has 1 GB of RAM, 100 Mb/s network bandwidth and 1 GB of storage. The CPU MIPS ratings are similar to different Amazon EC2 instance sizes. Users submit requests for provisioning of 500 heterogeneous VMs. Each VM is randomly assigned a workload trace from one of the servers from the workload data as described in the following section. The pricing for each VM instance is the same as used by Amazon EC2 for different sizes of VM. Even though only four types of VMs are considered, our model can be easily extended for other types of VM instances.

6.1. Workload Data

To make precise conclusions from a simulation study, it is important to conduct experiments using real workload traces. For our experiments, we used two different workload data, each for transactional and non-interactive applications. For transactional data, data is collected from CoMon [50], a monitoring infrastructure for PlanetLab[4]. The data contains the CPU utilization, memory and bandwidth usage of more than one thousand servers located at about 500 places around the world. The data has been collected for every five minutes during the period between 22nd and 29th of July 2010. The data is interpolated to generate CPU utilization for every second. The data satisfy our requirements of transactional application and thus have some peak utilization levels and very low off-peak utilization level: the average CPU utilization is below 50%.

For non-interactive batch jobs, the LCG workload traces from Grid Workload Archive (GWA) [51] are used. Since this paper focuses on studying cloud users with non-interactive applications, the GWA meets our objective by providing workload traces that reflect the characteristics of real applications running on one VM. From this trace, we obtain the submission time, requested number of VMs, and actual runtime of applications. Since workload traces do not contain any data on deadline and penalty rates specified in SLAs, for our evaluation, these are generated using uniform distribution which is commonly used in the literature [52, 53]. The deadline of a non-interactive application i is given by: $SubTime(i) + ExeTime(i) + ExeTime(i) * \lambda$, where $SubTime$ is the submission time and $ExeTime$ is the execution time. λ is the urgency factor derived from uniformly distribution (0,2).

6.2. Performance Metrics

The simulations ran for 10 hours of each workload category to determine the resource provisioning policy that delivers the best utilization, the least number of SLA violation and VM migrations, and accepts the maximum number of user requests. We have observed the performance from both user and cloud provider perspectives. From the provider perspective, two metrics are necessary to compare the policies: number of hosts utilized, and revenue generated. From the user perspective we have compared the number of SLA violations and the number of users accepted.

We have compared our resource provisioning policy Mixed Workload Aware Policy (MWAP) against two other well-known strategies used in current datacenters:

1. Static Approach (*also known as* Traditional): In this approach, during the whole VMs life cycle, an application will be allocated the capacity of the server as specified in the SLA. Thus, VM allocation will not change with time.
2. VM Migration and Consolidation Approach (*aka* withMigration): This strategy is used in various papers to address the problem of maximizing the utilization of datacenters [54]. In this approach, many VMs are consolidated in one server based on their usage. If an application demands more

[4]http://comon.cs.princeton.edu

Table 2: Effect of SLA consideration on provider and user parameters

Policy	Revenue (Batch Jobs)	Revenue Transactional	VM Migrations	SLA Violation (Transactional)	Batch Job Completed
Static	481848	647700	0	0	285
withMigration	31605	623370	267	26	160
MWAP	475732.8	647700	69	0	284

server capacity, either it is migrated or the capacity of server assigned to the VM running the application is increased. Since VM migration, in general, does not consider the SLA requirements of non-interactive applications, for fair comparison the VM capacity allocated to each such application does not change with time. This reduces the chance of batch applications to miss their deadline.

6.3. Analysis of Results

Although several experiments are conducted by varying different parameters, in this section, we discuss only the key and most relevant results of our evaluation. All the results are summarized in Figure 6 and 7, and Table 2 and 3.

6.3.1. Effect on Datacenter Utilization

Since our main objective is to maximize the utilization of the datacenter, first we compare all the techniques based on their effectiveness in maximizing the datacenter utilization. The datacenter utilization is indicated by the number of hosts which are used for a given workload. Figure 6 shows how the number of hosts utilized varied with time to meet the SLA requirements of applications and complete them successfully. It can be noticed that the number of VMs utilized by MWAP policy remains constant with time and utilized about 60% less servers on average. The reason for such a large difference is that MWAP tries to run VMs of batch jobs by using unutilized resources of VMs running Web applications. With more batch job submissions, the number of servers used by the static approach is increased from almost zero to 200. This is due to static allocation of server capacity to VMs based on SLA requirements. In this case, with the time the number of active servers becomes almost constant since enough servers are available to meet the resource demand of incoming non-transactional workload. In the case of withMigration resource provisioning policy, there is an increase in the number of servers between Time=300 to Time=400 due to high volume of batch job submissions. The latter due to consolidation, withMigration policy reduces the number of server utilized to about 83. This clearly shows the importance of considering resource usage pattern of different types of applications, which can result in efficient datacenter utilization. Thus, the datacenter can simultaneously serve more users with same server capacity.

6.3.2. Effect on Revenue Generated and SLA Violation

In general, the most important thing for a cloud provider is the profit generated by serving VM request of users. Secondly, the cloud provider wants to

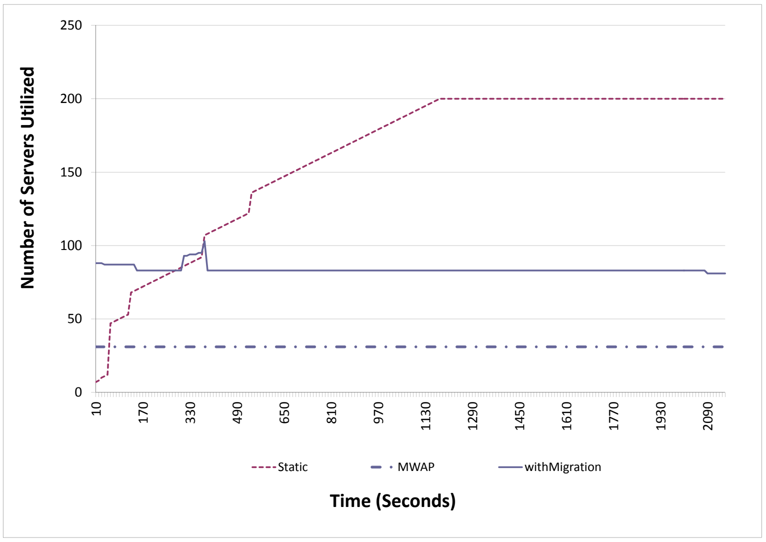


Figure 6: Effect on datacenter utilization

Table 3: Effect of different type of SLA penalties

Penalty Rate	Fixed		Delay Dependent		Proportional	
	Penalty($)	SLA Violation	Penalty($)	SLA Violation	Penalty($)	SLA Violation
Low	87.78063157	6	146.3010526	10	292.6021052	10
Medium	219.4515789	6	365.7526316	10	731.5052631	10
High	438.9031579	6	731.5052631	10	1463.010526	10

satisfy as many users as possible by meeting their SLA requirements. Table 2 gives the details of revenue generated for each type of applications, i.e., batch jobs and transactional applications. The revenue generated from Static policy and the proposed policy MWAP are similar because of zero number of violations both for transactional and batch jobs. The withMigration policy results in about 26 SLA violations due to the migration delays, which results in lower revenue. WithMigration policy also results in very low batch job revenue. The reason behind this is migration delays which result in SLA penalty. Therefore, the consideration of SLA penalty with VM migration and consolidation plays an important role in dynamic resource provisioning; otherwise cloud provider will incur huge revenue loss.

6.3.3. Migration Overhead and Batch Job Completion

It is well known that VM migration is not free and it always incur some network and CPU overhead. In this section, we show the number of migrations that MWAP performs in order to meet the SLA requirements in comparison to withMigration approach. It can be observed from Table 2 a significant reduction in VM migration by the MWAP policy, which results in almost 75% less migrations. WithMigration policy tries to optimize the utilization by migrating and consolidating the underutilized VMs, which results in very high number of migrations. The migration overhead causes unnecessary delays in batch job execution which results in almost 45% (Table II : Batch Job Completed) successful completions before deadline. This problem can further increase if the number of VMs initiated is not constant, which is accounted in our MWAP by using intelligent admission control policy.

6.3.4. Effect of SLAs Types

In this section, we present further results on the importance of considering different types of SLA penalties (requirements) along with dynamic provisioning of VMs within a cloud datacenter. Since there is no SLA violations noticed in the case of MWAP, we conducted the experiments using withMigration policy to understand the role of different types of SLA penalties (fixed, delay dependent and proportional) in resource allocation. For each application, Table 3 summarizes the results with variation of penalty rate (q) from low to high. The low penalty rate (q_{low}) is generated using a uniform distribution between (0, 1). The medium penalty rate is 2.5*q_{low} and the high penalty rate is 5*q_{low}. The proportional penalty incurs almost 50% more in comparison to other penalties. As the penalty rate varies, the total penalty incurred becomes more and more prominent. In withMigration policy, there is no consideration of different types of SLA penalties, as it results in more number of SLAs with delay-dependent and proportional penalty, and this further enhances the penalty. Thus, while doing resource allocation, the provisioning policy should take into account these penalty types and give priority to the applications with low penalty rates.

6.3.5. Effect of Deadline Urgency

In this section, we evaluate how deadlines of HPC (non-interactive) applications affect the performance of our SLA-based scheduling algorithm MWAP. Even though in real cloud datacenters the number of servers is almost unlimited, for this experiment we limited it to 30 and number of Web applications 40. Fixing these numbers allowed us to only observe the effects from deadline urgency. We varied the urgency factor λ of jobs from low (20) to very very high (.002) by decrementing factor of 10. Figure 7 shows how number of VM migration and SLA violation increases with user's urgency to execute the job. The results for low urgency are not presented since no violation was observed for that value. As the urgency level increases, from medium to high, the number of SLA violations drastically increases from about 30 to 120. But, after certain threshold, it reaches saturation. The reason for such a trend is that due to limited number of servers, our scheduler could not initiate new VMs without missing any new deadline. To avoid SLA violations, the scheduler tries to migrate the VMs from one server to another, which results in an increase in the number of migrations.

7. Conclusions and Future Directions

In this paper, we discussed how current cloud datacenters are facing the problem of underutilization and incurring extra cost. They are being used to

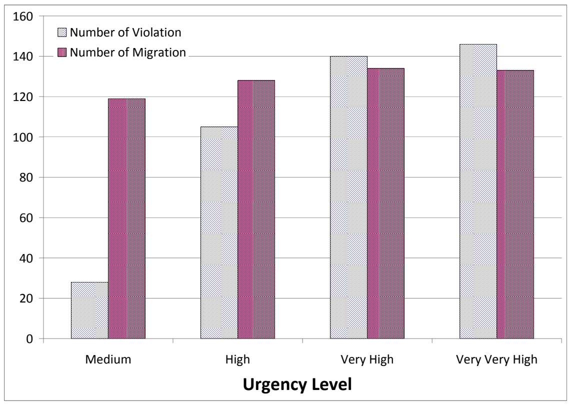


Figure 7: Effect of Deadline Urgency

run different types of applications from Web to HPC, which have different QoS requirements. This makes the problem harder, since it is not easy to predict how much capacity of a server should be allocated to each VM. Therefore, in this paper, we proposed a novel technique that maximizes the utilization of datacenter and allows the execution of heterogeneous application workloads, particularly, transactional and non-interactive jobs, with different SLA requirements. Our approach dynamically assigns VMs in such a way that SLA signed with customer is met without any penalty. The paper also described how the proposed technique can be easily integrated with the admission control and facilities such as auto-scaling offered by cloud providers. By extensive performance evaluation, it is demonstrated that the proposed mechanism MWAP reduces the number of servers utilized by 60% over other strategies like consolidation and migration with the negligible SLA violation. Our proposed mechanism MWAP performs reasonably well and is easily implementable in a real cloud computing environment.

The key reason here is that MWAP is able to manage different workload and exploits their usage patterns and QoS requirements to obtain efficient utilization of datacenter resources. Thus, we demonstrate that for designing more effective dynamic resource provisioning mechanisms, it is a must to consider different types of SLAs along with their penalties and the mix of workloads for better resource provisioning and utilization of datacenters; otherwise, it will not only incur unnecessary penalty to cloud providers but can also lead to under utilization of resources. This motivates further enquiry into exploration of optimizing the resource provisioning techniques by extending our approach to other type of workloads such as workflows and parallel applications. We also intend to include multiplexing strategies to remove interference from other applications when multiple VMs are consolidated on the same server. In future, we also plan to extend our model by considering multi-core CPU architectures as well as network and memory conflicts.

Acknowledgements

This work is partially supported by Linkage Project funding from the Australian Research Council (ARC) and CA Australia. This paper is a substantially extended version of our previous short conference paper presented at the 11th international Conference on algorithms and Architectures for Parallel Processing (ICA3PP 2011) [55]. Srinivasa's visit to the University of Melbourne was supported by a fellowship from the Government of India.

References

[1] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems 25 (6) (2009) 599–616.

[2] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, An early performance analysis of cloud computing services for scientific computing, Delft University of Technology, PDS-2008-006.

[3] M. Vouk, Cloud computing Issues, research and implementations, in: Proceedings of 30th International Conference on Information Technology Interfaces, Croatia, 2008.

[4] C. Yeo, R. Buyya, Service Level Agreement based Alloc. of Cluster Resources: Handling Penalty to Enhance Utility, in: Proceedings of the 7th IEEE International Conference on Cluster Computing, Boston, USA, 2005.

[5] J. Schneider, Amazon ec2: http://aws.amazon.com/ec2/.

[6] I. Goiri, F. Julià, J. O. Fitó, M. Macías, J. Guitart, Resource-level qos metric for cpu-based guarantees in cloud providers, in: Proceedings of 7th International Workshop on Economics of Grids, Clouds, Systems, and Services Naples, Italy, 2010.

[7] R. Nathuji, A. Kansal, A. Ghaffarkhah, Q-clouds: managing performance interference effects for qos-aware clouds, in: Proceedings of the 5th European conference on Computer systems (EuroSys 2010), Paris, France, 2010.

[8] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, N. Sharma, Towards autonomic workload provisioning for enterprise grids and clouds, in: Proceedings of 10th IEEE/ACM International Conference on Grid Computing, Melbourne, Australia, 2009.

[9] B. Sotomayor, K. Keahey, I. T. Foster, Combining batch execution and leasing using virtual machines, in: Proceedings of the 17th International ACM Symposium on High-Performance Parallel and Distributed Computing, Boston, USA, 2008.

[10] P. Balaji, P. Sadayappan, M. Islam, Techniques for Providing Hard Quality of Service Guarantees in Job Scheduling, R. Buyya and K. Bubendorfer (eds), ISBN: 978-0470287682, Wiley Press, Hoboken, New Jersey, USA.

[11] J. Hellerstein, F. Zhang, P. Shahabuddin, An approach to predictive detection for service management, in: Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, 1999. Distributed Management for the Networked Millennium, Boston, MA, U.S.A., 1999.

[12] D. Carrera, M. Steinder, I. Whalley, J. Torres, E. Ayguadé, Enabling resource sharing between transactional and batch workloads using dynamic application placement, in: Proceedings of the ACM/IFIP/USENIX 9th International Middleware Conference, Leuven, Belgium, 2008.

[13] G. R. Nudd, S. A. Jarvis, Performance-based middleware for grid computing, Concurrency - Practice and Experience 17 (2-4) (2005) 215–234.

[14] M. Smith, M. Schmidt, N. Fallenbeck, T. Doernemann, C. Schridde, B. Freisleben, Secure on-demand grid computing, Future Generation Computer Systems 25 (3) (2009) 315–325.

[15] L. Barroso, U. Holzle, The Case for Energy-Proportional Computing, Computer 40 (12) (2007) 33–37.

[16] J.-K. Kim, H. J. Siegel, A. A. Maciejewski, R. Eigenmann, Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling, IEEE Transactions on Parallel and Distributed Systems 19 (11) (2008) 1445–1457.

[17] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, D. Pendarakis, Efficient resource provisioning in compute clouds via vm multiplexing, in: Proceedings of the 7th International Conference on Autonomic Computing, Washington, USA, 2010.

[18] W. Zhang, H. Qian, C. Wills, M. Rabinovich, Agile resource management in a virtualized data center, in: Proceedings of Ist joint WOSP/SIPEW International Conference on Performance Engineering, California, USA, 2010.

[19] V. Soundararajan, J. Anderson, The impact of mngt. operations on the virtualized datacenter, in: Proceedings of the 37th Annual International Symposium on Computer Architecture, France, 2010.

[20] R. Singh, U. Sharma, E. Cecchet, P. Shenoy, Autonomic mix-aware provisioning for non-stationary data center workloads, in: Proceedings of the 7th International Conference on Autonomic Computing, Washington, USA, 2010.

[21] M. Steinder, I. Whalley, D. Chess, Server virtualization in autonomic management of heterogeneous workloads, SIGOPS Oper. Syst. Rev. 42 (1) (2008) 94–95.

[22] J. Ejarque, M. de Palol, n. Goiri, Í F. Julià, J. Guitart, R. M. Badia, J. Torres, Exploiting semantics and virtualization for sla-driven resource allocation in service providers, Concurrency - Practice and Experienc 22 (5) (2010) 541–572.

[23] B. Sotomayor, K. Keahey, I. Foster, Combining batch execution and leasing using virtual machines, in: Proceedings of the 17th International Symposium on HPDC, Boston, MA, USA, 2008.

[24] E. Casalicchio, L. Silvestri, Mechanisms for sla provisioning in cloud-based service providers, Computer Networks 57 (3) (2013) 795–810.

[25] U. Sharma, P. Shenoy, S. Sahu, A. Shaikh, A cost-aware elasticity provisioning system for the cloud, in: Proceedings of 31st International Conference on Distributed Computing Systems (ICDCS),Minneapolis, Minnesota, USA, 2011.

[26] X. Qiu, M. Hedwig, D. Neumann, Sla based dynamic provisioning of cloud resource in oltp systems, in: Proceeding of 2012 Workshop on E-Life: Web-Enabled Convergence of Commerce, Work, and Social Life,Shanghai, China, 2012.

[27] J. Kim, M. Ruggiero, D. Atienza, M. Lederberger, Correlation-aware virtual machine allocation for energy-efficient datacenters, in: Proceedings of the Conference on Design, Automation and Test in Europe, Ghent, Belgium, 2013.

[28] A. Sahai, S. Graupner, V. Machiraju, A. v. Moorsel, Specifying and monitoring guarantees in commercial grids through sla, in: Proceedings of the 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, 2003.

[29] H. N. Van, F. D. Tran, J.-M. Menaud, Sla-aware virtual resource management for cloud infrastructures, in: CIT '09: Proceedings of the 2009 Ninth IEEE International Conference on Computer and Information Technology, Xiamen, China, 2009.

[30] N. Bonvin, T. G. Papaioannou, K. Aberer, Autonomic sla-driven provisioning for cloud applications, in: Proceedings of the 11th IEEE/ACM international symposium on cluster, cloud and grid computing, Newport Beach, CA, USA, 2011.

[31] Z. Zhu, J. Bi, H. Yuan, Y. Chen, Sla based dynamic virtualized resources provisioning for shared cloud data centers, in: Proceedings of 2011 IEEE International Conference on Cloud Computing (CLOUD),Washington DC, USA, 2011.

[32] C. Wang, J. Chen, B. B. Zhou, A. Y. Zomaya, Just satisfactory resource provisioning for parallel applications in the cloud, in: Proceedings of 2012 IEEE Eighth World Congress on Services (SERVICES), Honolulu, HI, 2012.

[33] A.-F. Antonescu, P. Robinson, T. Braun, Dynamic sla management with forecasting using multi-objective optimization, in: Proceeding of 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 2013.

[34] Z. Wang, X. Zhu, P. Padala, S. Singhal, Capacity and performance overhead in dynamic resource allocation to virtual containers, in: Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, 2007.

[35] Y. Hu, J. Wong, G. Iszlai, M. Litoiu, Resource provisioning for cloud computing, in: CASCON '09: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, Ontario, Canada, 2009.

[36] J. O. Fito, Í. Goiri, J. Guitart, Sla-driven elastic cloud hosting provider, in: Proceedings of 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pisa, Italy, 2010.

[37] Z. Xiao, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment, Parallel and Distributed Systems, IEEE Transactions on 24 (6) (2013) 1107–1117.

[38] N. W. Paton, M. A. T. Aragão, K. Lee, A. A. A. Fernandes, R. Sakellariou, Optimizing utility in cloud computing through autonomic workload execution, IEEE Data Eng. Bull. 32 (1) (2009) 51–58.

[39] E. Casalicchio, D. A. Menascé, A. Aldhalaan, Autonomic resource provisioning in cloud systems with availability goals, in: Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, Miami, FL, USA, 2013.

[40] S. Islam, J. Keung, K. Lee, A. Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Future Generation Computer Systems 28 (1) (2012) 155–162.

[41] D. Minarolli, B. Freisleben, Distributed resource allocation to virtual machines via artificial neural networks, in: Proceedings of 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Turin, Italy, 2014.

[42] R. J. Pfitscher, M. A. Pillon, R. R. Obelheiro, Customer-oriented diagnosis of memory provisioning for iaas clouds, ACM SIGOPS Operating Systems Review 48 (1) (2014) 2–10.

[43] A. Iosup, D. Epema, Grid computing workloads: Bags of tasks, workflows, pilots, and others, IEEE Internet Computing 15 (2) (2010) 19–26.

[44] R. Garg, H. Saran, R. Randhawa, M. Singh, A sla framework for qos provisioning and dynamic capacity allocation, in: Proceedings of the 10th IEEE International Workshop on Quality of Service, Dalian, China, 2002.

[45] S. Benchmarks, Standard Performance Evaluation Corporation, Manassas, VA, USA.

[46] K. Srinivasa, K. Venugopal, L. Patnaik, An efficient fuzzy based neuro-genetic algorithm for stock market prediction, Hybrid intelligent system 3 (2) (2006) 63–81.

[47] E. Azoff, Neural network time series forecasting of financial markets, John Wiley & Sons, Inc. New York, NY, USA, 1994.

[48] E. Dodonov, R. de Mello, A novel approach for distributed application scheduling based on prediction of communication events, Future Generation Computer Systems 26 (5) (2010) 740–752.

[49] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41 (1) (2011) 23–50.

[50] K. Park, V. Pai, CoMon: a mostly-scalable monitoring system for PlanetLab, ACM SIGOPS Operating Systems Review 40 (1) (2006) 65–74.

[51] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, D. J. Epema, The grid workloads archive, Future Generation Computer Systems 24 (7) (2008) 672–686.

[52] B. Yang, X. Xu, F. Tan, D.-H. Park, An utility-based job scheduling algorithm for cloud computing considering reliability factor, in: Proceedings of International Conference on Cloud and Service Computing (CSC'11), Hong kong, 2011.

[53] S. K. Garg, R. Buyya, H. J. Siegel, Time and cost trade-off management for scheduling parallel applications on utility grids, Future Generation Computer Systems 26 (8) (2010) 1344–1355.

[54] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems, Advances in Computers, M. Zelkowitz (editor), ISBN 13: 978-0-12-012141-0, Elsevier.

[55] S. Garg, S. Gopalaiyengar, R. Buyya, Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter, in: Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing, Melbourne, Australia, 2011.