SLA-Based Resource Provisioning for Heterogeneous Workloads in a Virtualized Cloud Datacenter

Saurabh Kumar Garg, Srinivasa K. Gopalaiyengar, and Rajkumar Buyya

Cloud Computing and Distributed Systems Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{sgarg,raj}@csse.unimelb.edu.au

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 with different performance requirements. Currently, cloud datacenter providers either do not offer any performance guarantee or prefer static VM allocation over dynamic, which lead to inefficient utilization of resources. Earlier solutions, concentrating on a single type of SLAs (Service Level Agreements) or resource usage patterns of applications, are not suitable for cloud computing environments. In this paper, we tackle the resource allocation problem within a datacenter that runs different type of application workloads, particularly non-interactive and transactional applications. We propose admission control and scheduling mechanism which not only maximizes the resource utilization and profit, but also ensures the SLA requirements of users. In our experimental study, the proposed mechanism has shown to provide substantial improvement over static server consolidation and reduces SLA Violations.

1 Introduction

With the increasing popularity of Cloud computing, research centers and enterprises have started outsourcing their IT and computational needs to on-demand cloud services [3].The clouds are typically large scale virtualized datacenters hosting thousands of servers. 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 in clouds. In particular, for a commercial success of this computing paradigm, the cloud datacenters need to provide a better and strict Quality of Service (QoS) guarantees. These guarantees which are documented in the form of Service Level Agreement (SLA) are crucial, since only then the customers can be confident in outsourcing their jobs to clouds [19]. Resource provisioning plays a key role in ensuring that the cloud providers adequately accomplish their obligations to customers while maximizing the utilization of underlying infrastructure. An efficient resource management scheme would require to automatically allocate to each service request,

Y. Xiang et al. (Eds.): ICA3PP 2011, Part I, LNCS 7016, pp. 371–384, 2011.

the minimal resources 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 automatically is a challenging task. Current cloud datacenters hosts a wider range of applications with different SLA requirements[12]. The intrinsic differences among these different workloads further make the resource provisioning a challenging task [15]. First, the SLA requirements of different applications are different. The transactional applications require response time and throughput guarantee, while non-interactive batch job concerns performance (e.g. completion times). Second, the resource demand of transactional applications such as Web application tend to be highly unpredictable and busty in nature[4], while of batch jobs can be predicted to a higher degree[14]. Hence, the satisfaction of complex and different requirements of all applications makes the goal of utilization maximization while meeting different types of SLAs far from trivial.

Traditionally, to meet SLA requirements, over-provisioning of resources to meet worst case demand are used. However, since servers operate most of the time at very low utilization level, it leads to waste of resources in non-peak times. This over-provisioning of resources results in extra maintenance costs including server cooling and administration [8]. 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. Even though computationally intensive applications are increasingly becoming the part of enterprise datacenter, still research with mixed types of applications having different SLAs 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.

Therefore, in this paper, we present a novel dynamic resource management strategy that not only maximizes the 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 i.e., compute intensive non-interactive jobs and transactional applications, each having different types of SLA requirements and specifications. Our strategy makes dynamic placement decisions to respond to the changes in transactional workload, and also considers the SLA penalties for making future decisions. To schedule batch jobs, our proposed resource provisioning mechanism predicts the future resource availability and schedules the jobs by stealing CPU cycles, which are unutilized by the transactional applications.

2 Related Work

In this section, we compare our work with the most relevant ones. Meng et al. [9] proposed a joint-VM provisioning and borrowing approach by exploiting the statistical multiplexing among the workload patterns. Zhang et al. [20] designed

an approach to quickly reassign the resources for a virtualized utility computing platform using ghost virtual machines (VMs). These works concentrate on fixed number of VMs, while we have considered variable amount of incoming workload. Vijayaraghavan and Jennifer [16] focussed on the analysis and resource provisioning for the datacenters's management workloads which have considerable network and disk I/O requirements. Singh et al. [13] 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. In contrast, our work concentrates on handling multiple types of SLA both for High Performance Computing (HPC) and Web based workloads with a new admission control policy. Quiroz et al. [12] present a decentralized, 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 WS requests, and data and IO-intensive analytics tasks. When compared to our approach, the SLA penalty is not considered and it uses a static number of VMs. Wang et al. [18] evaluated the overhead of a dynamic allocation scheme in both system capacity and application-level performance relative to static allocation. In our work, the idea of dynamic allocation is extended for multiple types of workloads including HPC and Web. Carrera et al. [4] developed 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 type of SLA requirements of the applications.

The main **contributions** of this paper lies with the design of an efficient admission control and scheduling mechanism for Cloud datacenters with the following salient features: a) adaptive admission control and dynamic resource provisioning facility, b) considered multiple type of SLAs based on application requirements, c) integration of mixed/heterogeneous workloads (such as non-interactive and transactional applications) for better utilization of resources, and d) variable penalties depending on the type of SLA violation.

3 System Model

3.1 Datacenter Model

We consider a Cloud virtualized datacenter model. Each server is interconnected with a high-speed LAN network and high bandwidth link to the Internet. The key components involved in the process of scheduling an application on a VM are: admission control, VM manager, job scheduler and SLA manager. The admission control component decides whether 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 manager will initiate a VM and allocate it to a server having the required capacity. Job scheduler will schedule applications on this newly initiated VM. SLA manager monitors the current

SLAs and service level for each accepted application. We consider the 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 to allow the user to choose as per his requirements and they can be charged on hourly basis. The auto-scaling facility can also be provided if resource demand exceeds the VM size allocated. In the next section, we discuss the two types of application workloads considered in this work along with their SLA definitions.

3.2 SLA and Application Models

The transactional workloads include Web applications whose resource demand can vary with time. On the other hand, for the non-interactive workloads, we model the HPC compute intensive bag of task applications. Most of the scientific workloads include mostly independent single-machine jobs (tasks) grouped into single "bag of tasks" [6]. Thus, 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. We consider a discrete-time scenario in which time is slotted into intervals with equal length (T).

SLA Model for Transactional Workload: A user gives QoS requirements in terms of response time t_i with each transactional application i. This response time requirement can be translated to CPU power α_i needed to achieve this response time [4]. SLA will be broken if the capacity C_{t_i} allocated to the application is less than the required capacity at time t. A penalty q will incur if number of such violation increases beyond a threshold β_i. Thus, the net amount the user needs to pay at the end of the period (T_1, T_2) would be: $r * \alpha_i * (T_2 - T_1) - q$, if r is rate charged. Here, the user can choose three types of penalties:

- **Fixed Penalty:** The fixed penalty q is charged whenever the cloud provider fails to fulfil current capacity demand.
- **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 rate 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 SLA), then the service provider's penalty due to the user is calculated as: $q' * (T_2 - T_1)$.
- **Proportional Penalty:** This is also a form of delay-dependent penalty, where the penalty to be credited to a user which 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 was requested and allocated, then the penalty is given as: $q' * (T_2 - T1) * (C_1 - C_2)$.

SLA also contains the information regarding auto-scaling option. If the user choose this facility, when demand for resources increases beyond the initial requested amount, more resources will be allocated to meet this spike and

automatically reduced when demand is decreased to a threshold. The Cloud provider can charge an additional amount for offering such flexibility.

SLA Model for Non-interactive Batch Jobs: QoS requirements for batch jobs are deadline and the amount of CPU time allocated. These jobs require performance based guarantees. Since, the performance of VMs allocated can vary with the usage of datacenter and we define SLA as the p amount of CPU Cycles to be provided by the Cloud provider before the deadline d in order to ensure successful completion of job. Let db be the delay before allocating p CPU cycles. Then, if the provider fails to complete the given job, following penalty applies: $q = y * db$, where y is the penalty rate.

4 Admission Control and Scheduling Policy

As discussed in the earlier sections, we need to consider the requirements of two different application workloads before accepting the 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 decisions 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 certain amount of time or not running any applications, then it will be switched-off. In each scheduling cycle, admission control and scheduling can perform three functions:

- Admission Control: It decides to accept or reject a new application (transactional or batch) based on present and future resource availability.
- SLA Enforcement: The resource demand of each application is monitored based on agreed QoS level guaranteed in SLAs. 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 auto-scaling thresholds.

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 following sections. The *batch job queue* component represents the queue of VMs (corresponding to each batch job) which 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 the d be the deadline, MI be the CPU cycles (Millions of Instructions) required for completion of job, and C_{min} be the Million of Instruction Per Second(MIPS) of smallest standardized VM offered by the Cloud provider, then the threshold time $threshTime(i)$ for batch job i is calculated as : $threshTime(i) = d - \frac{MI}{C_{min}}$.

4.1 Forecasting Model

In this paper, we have used an Artificial Neural Network (ANN) based forecasting model, which is a multi-layer feedforward network. It had been shown in literature [17][1] that ANNs perform better than other linear models (e.g., regression models), specifically, for more irregular series and for multiple-period-ahead forecasting. Such neural network techniques has also been used and well studied in previous works for distributed application scheduling in context of Grids [5]. *It is important to note that it is not our goal here to determine the best prediction technique.* However, our contribution is that if the resource requirements of an application is 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 the incorrect prediction of the forecasting model.

In this paper, the standard Back Propagation (BP) algorithm is used to predict the CPU utilization. The forecasting model predicts one day utilization values of each VMs from one week data from the dataset, with minimum Root Mean Square Error (RMSE) network. Here, the ANN is modeled with one input layer, one output layer and variable hidden layers between the inputs and outputs. As all the nodes at each layer are interconnected by weights, a training algorithm is used to attain a set of weights that minimizes the difference between the predicted value and the actual output through the network. For a given resource usage of an application, the model can be tested for accuracy of prediction by varying the number of hidden layers, training set, and the learning rate. To avoid the overhead of forecasting, this process occur in offline mode.

The neural network considered in this paper, forecasts the CPU utilization with minimum RMSE network [10]. In the experiments[1], a time series vector of CPU utilization for one week is given as an input. The number of hidden layers is varied to tune the performance of the network and through iterations it was found to be optimum at the value of 5 hidden layers. 80% of the data is considered for training and at any point of time 24 periods of CPU utilization for every next 5 minutes is forecasted. The learning rate is found to be optimum at 0.1. The network generates a neural model with minimum MSE (Mean Square Error) and outputs the minimum of RMSE. The 8064 instances of CPU utilization (About one month data, recorded on every five minutes) taken from workload traces is given as the input to the model and the CPU utilization for next one day (288 instances) are predicted.

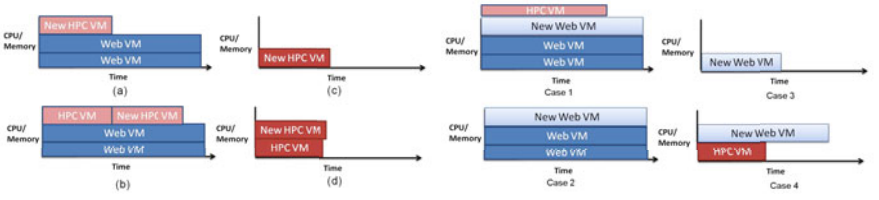
4.2 Admission Control and Scheduling

In this process, the decision 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

[1] Please see section 5 for the details of web application workload.

accepted applications. If free resources on a host are sufficient to run the application, it will be accepted. The scheduling of application is based on following simple but effective principle:"*In initial allocation, for transactional applications, reserve resources are equivalent to their peak demand. For non-interactive jobs, scheduler tries to allocate slack of resources remaining on a host which is running a transactional application. At regular intervals, consolidate those VMs deployed on hosts which are not running any non-interactive jobs*".

The 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 is quite different from general approaches used in previous works where, during migration, multiple types 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:



(a) Non-interactive (b) Transactional

Fig. 1. Job Scheduling Scenarios

Non-Interactive Job: The datacenter accepts the 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. At any moment, when a job arrives for execution, datacenter have some servers running the workload and others which are switched-off to save the power. Thus, the scheduler first checks the resource availability of active servers where a Web application is already hosted. The resource availability on each host are 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 jobs is estimated based on resource availability on each host. Figure 1(a) shows the four possible ways a given job can be scheduled. If the job can be completed before the deadline on any host then, the job is accepted by the datacenter for execution, otherwise it will be rejected. SLA is signed between both the user and the provider.

For scheduling, a VM image based on QoS requirement of a given job is created. The VM is deployed on a server (S_j) where the start time (Exp_St) of job is minimum and with an already hosted Web VM. Thus, for execution 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 host, there is already one HPC VM is running, then the new VM will be deployed after the completion of the job on the current VM (as demonstrated in Figure 1(a)).

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 job is queued up in the batch job Queue. This is done to exploit errors in the resource demand forecast. If actual demand of the resources is less than forecasted one, then some HPC job could finish before their estimated completion time. Thus, new VM can be deployed without switching on to the 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 the HPC VMs, or a new server which is switched on. In this case, the VM deployment will be static. Therefore, these VMs are not allowed to migrate, and the resources allocated to them will not be changed till the agreed amount (as in SLA) of CPU cycles are used for its execution. These VMs are called *static HPC VM* indicated by a different color in Figure 1(a). This is done to avoid the effects of rescheduling of VMs on the network and on the other VMs running in the server.

Transactional Applications: A user having a transactional application can ask for VMs of different standard sizes offered by the Cloud provider. Let the user request for a VM with capacity C_k. A request is accepted when the datacenter can schedule the VM with capacity C_k on any server assuming all hosted Web VMs are running at 100% utilization and without considering resources used by dynamic HPC VMs. The Web VM is scheduled based on the best-fit manner. As Figure 1(b) shows, there are four possibilities to deploy the new Web VM.

Algorithm 1. 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: Initiate 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**

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 will be paused and rescheduled (migrated) to other servers if the HPC job is missing its deadline after deployment of new Web VM. The rescheduling of HPC job is done in such a way that the minimum penalty occurs due to SLA violation.

Case 2 - 4: In these cases, since, while scheduling of 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 static HPC Vm (denoted by red color) is hosted therefore, the available resources on the server for executing new Web application will be the amount of resources unused by HPC VM.

4.3 SLA Enforcement and Rescheduling of VMs

To fulfil SLAs, the regular SLA monitoring of each application is required, since our initial allocation and admission control is based on the 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 1. In every scheduling cycle, scheduler will do following functions: a) enforce SLAs and b) schedule the jobs from batchjob Queue, and c) Consolidation. For SLA enforcement, the resource demand for each transactional application till next scheduling cycle is recalculated (Line 1-2). If any Web VM requires less than the forecasted (currently allocated capacity) then the extra resources are allocated to HPC VM running on the same host (Line 3-5 and 20-21). Otherwise, if the Web VM requires more resources than allocated ($\leq$ promised capacity in SLA), then the resources allocated to the VM are increased to match the demand (Line 6-7). Correspondingly, the resources allocated the HPC VM will be reduced (Line 8). If the Web VM requires resource capacity more than specified in SLA, then the decision is taken based on whether the user has opted for auto-scaling or not (Line 10-15). This process is repeated for each transactional application. After that, for each HPC job, their VM capacity is increased if some slack resources is available on the server where the VM is hosted (Line 17-18). Based on allocated resources to the HPC VM, the job completion time is recomputed (Line 20). If any HPC job which is currently running is expected to miss deadline, then its corresponding VM is migrated and scheduled on another server using strategies discussed in previous section (Line 21). Similar process is repeated for each HPC job (VM) in batch job queue. The scheduler consolidates Web VMs which are running on servers having no HPC VM. If due to consolidation, some Web VMs may be short of resources, in that case, SLA can be violated.

5 Performance Evaluation

We have simulated a datacenter that comprises of 1500 heterogeneous physical nodes. Simulation approach gives advantage of repeating the experiments under similar environment. Thus, it allows the comparison of different scheduling strategies. In addition, it is pretty difficult to gain workload for many real applications as they are considered in this work. Each node is modelled 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 have considered for different types of VMs with 1000, 2000, 2500 or 3500 MIPS. The smallest instance will be allocated 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. The 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 taken 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.

5.1 Workload Data

For our experiments, we have used two different workloads data, each for transactional and non-interactive applications. For transactional data, data is collected from CoMon [11], a monitoring infrastructure for PlanetLab (http: //comon.cs.princeton.edu). The data contains the CPU utilization, memory and bandwidth usage of more than thousand servers located at about 500 places around the world. The data has been collected for each five minutes during the period from the 22nd to 29th of July 2010. The data is interpolated to generate CPU utilization for every second. The data satisfy our requirement of transactional application and thus have some good number of 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) [7] are used. Since this paper focuses on studying the 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 submit 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. The deadline of a non-interactive application i is given by: $SubTime(i) + ExeTime(i) + ExeTime(i) * \lambda$, where $SubTime$ is submit time and ExeTime is execution time. λ is urgency factor derived from uniformly distribution(0,2).

5.2 Performance Metrics

The simulations have been run for 10 hours of each workload category to determine the resource provisioning policies that delivers the best utilization, the

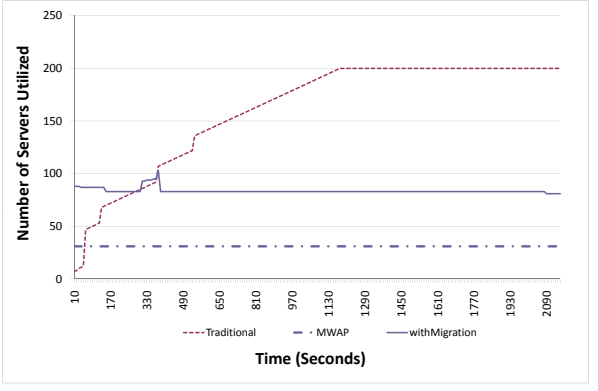


Fig. 2. Effect on Datacenter Utilization

least number of SLA violation and VM migrations, and accepted maximum user requests. Two metrics are used to compare the policies: number of hosts utilized and revenue generated. We have compared our resource provisioning policy MWAP (Mixed Workload Aware Policy) against two other following well known strategies used in current datacenters:

1. Traditional Approach (*aka* Traditional): In this approach, during the whole VMs life cycle, an application will be allocated the capacity of server as specified in 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 [2]. In this approach, many VMs are consolidated based on one server and their usage. If an application demands more server capacity, either it is migrated or capacity of server assigned to the VM running the application is increased. Since VMMigration does not consider the SLA requirements of non-interactive applications, for fair comparison the VM capacity allocated to each such applications does not change with time. This reduces the chance of batch application to miss its deadline.

6 Analysis of Results

Although several experiments are conducted by varying different parameters, due to space constraint, in this section, we discuss only the key results of our evaluation. All the results are summarized in Figure 2, Table 1 and 2.

Effect on Datacenter Utilization: Since our main objective is to maximize the utilization of 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 host which are used for a given workload. Thus, Figure 2, shows how the number of hosts utilized is varied with time to meet the SLA requirements of applications and complete them successfully. It can be noticed that number of VMs utilized by MWAP policy remains constant with time

and utilized about 60% less number of 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 traditional approach has 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, number of active servers become 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 T=300 to T=400 due to high volume of batch job submissions. The later 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 type of applications, which can result in efficient datacenter utilization. Thus, datacenter can simultaneously serve more users with same server capacity.

Table 1. Effect of SLA consideration on provider and user Parameters

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

Table 2. 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

Effect on Revenue Generated and SLA Violation: In general, the most important thing for a Cloud provider is, the profit generated by serving the VM request of users. Secondly, the Cloud provider wants is to satisfy as many users as possible by meeting their SLA requirements. Table 1 gives the details of revenue generated for each type of applications i.e., batch jobs and transactional applications. The revenue generated from Traditional policy and the proposed policy MWAP are similar because of zero number of violations both for transactional and batch jobs. While, withMigration policy results in about 26 SLA violation due to the migration delays which results in reduction of revenue. The withMigration policy also results in very low batch job revenue. The reason behind this is migration delays which results in SLA penalty. Therefore, the consideration of SLA penalty with VM migration and consolidations plays an important role in dynamic resource provisioning, otherwise Cloud provider will incur huge revenue loss.

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 will show the number of migrations MWAP required to perform in order to meet the SLA requirements in comparison to withMigration approach.

It can be clearly noticed from Table 1, the huge reduction in VM migration by MWAP policy which results in almost 75% less number of migrations. The with-Migration 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 1 : Batch Job Completed) of 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.

Effect of SLAs Types: In this section, we present the 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 have 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 2 summarizes the results with variation of penalty rate (q) from low to high. The proportional penalty incur almost 50% more in comparison to other penalties. As the penalty rate is varied, the total penalty incurred becomes more and more prominent. Since, in withMigration policy, there is no consideration of type 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.

7 Conclusions and Future Directions

In this work, we have proposed a novel technique that maximizes the utilization of datacenter and allows the execution of heterogenous 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 can be met without any penalty. The paper, also describes how the proposed technique can easily integrate with the admission control and facilities like autoscaling offered by the Cloud providers. By extensive performance evaluation, it is demonstrated that, the proposed mechanism MWAP reduces number of servers utilized by 60% over other strategies like consolidation and migration with negligible SLA violation. Our proposed mechanism MWAP performs reasonably well and is easily implementable in a real Cloud Computing environment.

Therefore, we demonstrate that for designing more effective dynamic resource provisioning mechanisms, it is important 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 provider 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. In future, we also plan to extend our model by considering the multi-core CPU architectures as well as network and memory conflicts.

References

1. Azoff, E.: Neural network time series forecasting of financial markets. John Wiley & Sons, Inc., New York (1994)
2. Beloglazov, et al.: A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. In: Zelkowitz, M. (ed.) Advances in Computers. Elsevier, Amsterdam (2011) ISBN 13: 978-0-12-012141-0
3. Buyya, et al.: Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. FGCS 25(6), 599–616 (2009)
4. Carrera, D., Steinder, M., Whalley, I., Torres, J., Ayguadé, E.: Enabling resource sharing between transactional and batch workloads using dynamic application placement. In: Issarny, V., Schantz, R. (eds.) Middleware 2008. LNCS, vol. 5346, pp. 203–222. Springer, Heidelberg (2008)
5. Dodonov, E., de Mello, R.: A novel approach for distributed application scheduling based on prediction of communication events. FGCS 26(5), 740–752 (2010)
6. Iosup, A., Epema, D.: Grid computing workloads: Bags of tasks, workflows, pilots, and others. IEEE Internet Computing 99(PrePrints) (2010)
7. Iosup, et al.: The grid workloads archive. FGCS 24(7), 672–686 (2008)
8. Kim, J.K., Siegel, H.J., Maciejewski, A.A., Eigenmann, R.: Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling. IEEE Trans. Parallel Distrib. Syst. 19(11), 1445–1457 (2008)
9. Meng, et al.: Efficient resource provisioning in compute clouds via vm multiplexing. In: Proc. of the 7th Intl. Conf. on Auton. Comp., Washington, DC, USA (2010)
10. Mohammadi, S., Abbasi-Nejad, H.: Forecasting With Matlab. 129.3.20.41/eps/prog/papers/0505/0505001.pdf (2005)
11. Park, K., Pai, V.: CoMon: a mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Operating Systems Review 40(1), 65–74 (2006)
12. Quiroz, et al.: Towards autonomic workload provisioning for enterprise grids and clouds. In: Proc. of 10th IEEE/ACM Intl. Conf. on Grid Comp., USA (2009)
13. Singh, et al.: Autonomic mix-aware provisioning for non-stationary data center workloads. In: Proc. of the 7th Intl. Conf. on Autc. Comp., USA (2010)
14. Smith, et al.: Secure on-demand grid computing. FGCS 25(3), 315–325 (2009)
15. Sotomayor, et al.: Combining batch execution and leasing using virtual machines. In: Proc. of the 17th Intl. Sym. on HPDC, Boston, MA, USA (2008)
16. Soundararajan, et al.: The impact of mngt. operations on the virtualized datacenter. In: Proc. of the 37th Ann. Intl. Sym. on Comp. Arch., France (2010)
17. Srinivasa, et al.: An efficient fuzzy based neuro-genetic algorithm for stock market prediction. Intl. Jnl. of Hyb. Intelligent Sys. 3(2), 63–81 (2006)
18. Wang, et al.: Capacity and performance overhead in dynamic resource allocation to virtual containers. In: Proc. of the 10th IFIP/IEEE Intl. Symp. on Intgd. Net. Mangt., Munich, Germany (2007)
19. Yeo, C., Buyya, R.: Service Level Agreement based Alloc. of Cluster Resources: Handling Penalty to Enhance Utility. In: Proc. of the 7th IEEE Intl. Conf. on Cluster Comp., Boston, USA (2005)
20. Zhang, et al.: Agile resource management in a virtualized data center. In: Proc. of Ist Joint WOSP/SIPEW Intl. Conf. on Perf. Eng., California, USA (2010)