Int. J. Advance Soft Compu. Appl, Vol. 7, No. 1, March 2015
ISSN 2074-8523

Attaining Reliability and Energy Efficiency in Cloud Data Centers Through Workload Profiling and SLA-Aware VM Assignment

Frank Elijorde, and Jaewan Lee

Institute of Information and Communication Technology,
West Visayas State University, Iloilo City, Philippines
e-mail: frank@kunsan.ac.kr
Department of Information and Communication Engineering,
Kunsan National University, Gunsan, South Korea
e-mail: jwlee@kunsan.ac.kr

Abstract

To attain the desired reliability and energy efficiency of a cloud data center, trade-offs should be carried out between system performance and power consumption. Although the deployment of energy-efficient hardware is a crucial step, getting rid of underutilized servers is a far more effective approach. Achieving this in a cloud data center would require monitoring resource utilization as well as managing the quality of service and the profitability of a cloud infrastructure. In this study, we propose a resource provisioning technique which involves clustering of similar VMs in a cloud datacenter based on their workload profile. We then combine it with an SLA-aware approach for VM assignment in order to improve the efficiency of virtual resource provisioning and increase the quality of service. The proposed approach was compared with other techniques and as shown by the results, performance and energy efficiency is improved.

Keywords: *Cloud Computing, Cloud Data Centers, Service Level Agreement, Green Computing, Resource Provisioning.*

1 Introduction

As a new paradigm, cloud computing has become a favorable approach for provisioning computing services. Using virtualization as enabling technology, the cloud can support dynamic resource consolidation and environment isolation. In a data center, the cloud service provider should guarantee the availability of resources required by each virtual machine (VM) at any moment it is in demand [1]. Whenever a VM's demand is not satisfied, it is said to suffer a SLA violation;

such occurrence is considered a downside of server consolidation, which leads to penalties that equates to monetary losses for the provider [2].

Reducing the energy consumption of data centers is a challenging issue caused by the rapid growth of computing applications. Unfortunately, the average server utilization in many data centers is found to be low, estimated to be only between 5% and 15% [3]. Underutilized resources would further add up to this waste because an idle server often consumes more than 50% of its peak power [4],which means that a number of servers at low utilization consume more energy than fewer servers at high utilization. The cause of the unreasonably high energy consumption is not just the quantity of computing resources and the power inefficiency of hardware, but most importantly the inefficient utilization of these resources. In a bid to resolve this, oversubscription of cloud services has become an appealing solution to further optimize cloud efficiency and utilization. In the cloud, resource demand is considered more unpredictable as compared with traditional IT environments. Therefore, it is necessary to monitor resource utilization as well as to manage the quality of service and the profitability of a cloud infrastructure.

To address the issue, we aim to improve the scalability of monitoring and management of cloud infrastructures by taking into consideration the similarity between VM behaviors. In contrast to known practice, we look at VMs not as individual entities but those which possess similar characteristics. Thus, we propose a resource provisioning technique which involves clustering of similar VMs in a cloud datacenter based on their workload profile. This would enable the cloud system to appropriately categorize the incoming requests which in turn will provide a more accurate estimate of computing resources to be consumed by the workload. By combining it with an SLA-aware approach for VM assignment, the efficiency of virtual resource provisioning is further improved by keeping SLA violations low. With regards to energy consumption, our proposed approach showed significant improvement by reducing idle resources.

2 Related Work

2.1 Resource Provisioning in Cloud Data Centers

Existing server provisioning solutions can be broadly classified into predictive and reactive solutions [5]. In predictive provisioning, it is assumed that there is a predictable and stable pattern in demand. An approach for workload prediction based on surrogate models to provide an autonomic controller is presented in [6]. Multi-dimensional surrogate models are employed for approximation and performance profiling of applications using a utility function, which is used afterwards to support the controller decision making. A work in [7] deals with an energy-aware dynamic VM consolidation system aimed at web applications whose SLAs are defined in terms of the response time. Weighted linear regression

is applied to predict the future workload and proactively optimize the resource allocation.

Reactive provisioning, on the other hand, allocates resources in short intervals in response to workload changes. A common approach in this solution is to use reactive control loops that are based on feedback control [8]. However, the drawback of control-based method is that the anomaly must first occur before carrying out corrective measures. Purely workload-driven service replication policies have been shown effective on platforms which apply simple control actions such as turning service replicas on and off [9]. Nonetheless, such policies may have issues in the optimization of the trade-off between cost and performance in a cloud due to the lack of consideration of the variability in VMs' performance and billing contracts.

2.2 Resource Management in Virtualized Cloud Environments

The virtual resource management in a cloud environment has been studied with goals such as QoS awareness, performance isolation and differentiation as well as higher resource utilization. The autoscaling approach of Amazon [10] is the customer side auto-scaling management component, which enable customers to apply their own rules for the capacity management of their cloud resources. The method is directed towards the customer side post-processing of capacity tuning rather than the provider side. In [11], applications are subjected to a strict enforcement of reservation of resources in order to determine if the given set of reservation parameters satisfies the time constraints for execution; otherwise, the parameters are modified accordingly. A framework for SLA management based on multi-objective optimization was presented in [12]. They propose comprehensive SLA management approach that uses event processing for monitoring and enables dynamic provisioning of virtual machines onto the physical infrastructure. In [13], they presented a cloud infrastructure that combines on-demand allocation of resources with opportunistic provisioning of cycles from idle cloud nodes which aims to improve the utilization of Infrastructure Clouds.

2.3 Resource Oversubscription in the Cloud

In cloud computing, a cloud is said to be oversubscribed (or overbooked) when the total customer requests for resources is beyond the actual physical capacity [14]. The practice of CPU overbooking is studied in [15] by applying predictive risk analysis. The history of CPU usage of VMs is analyzed to establish a one-sided tolerance limit which represents a threshold of risk for overcommiting a set of VMs based on the probability of overload and SLA violations. In [16], a multi-layer Neural Network is used to predict patterns of resource usage by studying historical data for the current workload. Combined with the Optimal Allocation Limit (OAL), the techniques are used to define an over-allocation algorithm

which overbooks the total available resources. Altogether, the focus is on the importance of overbooking for greener computing. The framework for VM placement proposed in [17] involves monitoring and profiling of applications to predict their behavior and type of resource usage. The best location for the application to be deployed is determined via a smart overbooking scheduler. In [18], they consider a cloud with overcommitted processor power. They introduce a VM migration algorithm which identifies the ideal VMs for migration based on evaluation of their workload degree of correlation which are migrated to optimal PMs as selected by the algorithm.

3 Efficient Workload Classification and VM Consolidation

3.1 System Architecture

In Fig. 1, the overview of the proposed scheme is presented with the components that compose its architecture. The proposed approach starts by processing varying workload data composed of client requests for compute resources such as memory and CPU. Second, the workload data are subjected to cluster analysis in order to identify data patterns that are similar to each other. Third, using the cluster generated from the previous phase, the client requests are re-classified by assigning them to a more appropriate class with corresponding amount of resources to be provisioned. Finally, client requests are granted with the required virtual resources with specific consideration on the actual demand.

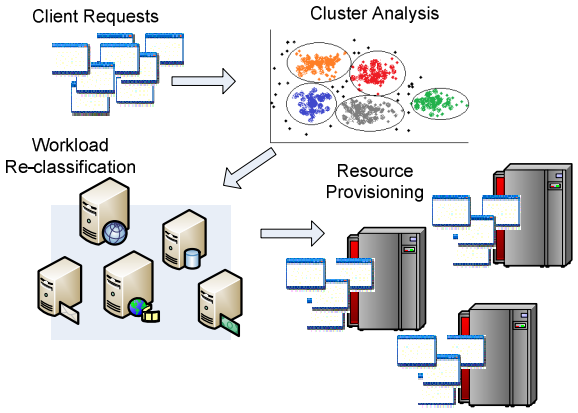


Fig. 1 Overview of the proposed scheme

3.2 Cluster Analysis and Workload Re-classification

Clustering is an unsupervised learning of unlabeled data, and such property has separated it from classification, where the class-prediction is done on unlabeled

data after a supervised learning on pre-labeled data. Cluster analysis is the organization of a collection of patterns represented as a vector of measurements into clusters based on similarity. Gathered around a center called the centroid, patterns belonging to the same cluster are considered similar to each other compared to those within a different cluster.

In this work, K-means Clustering [19] is used to perform classification of incoming workload. Each client request is assigned to the closest centroid. To do this, a distance function that quantifies the perception of closeness for the specific data under consideration is needed. The function used is the Euclidean Distance defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{m} (x^i - y^i)^2} \tag{1}$$

where $x = (x_i, \ldots x_m)$ and $y = (y_i, \ldots y_m)$ are two input vectors with m quantitative features. In the Euclidean distance function, all features contribute equally to the function value. We determine the optimal number of clusters by using the Silhouette [20] method on the cluster data. The silhouette value for each point is a measure of its similarity to points in its own cluster compared to points in other clusters. Thus, the quality of the classification depends on the silhouette value. It is defined as:

$$s(i) = \frac{b(i) - w(i)}{\max\{b(i), w(i)\}} \tag{2}$$

with

$$b(i) = \min\{B(i,k)\} \tag{3}$$

where $w(i)$ is the average distance from the ith point to the other points in its own cluster, and $B(i, k)$ is the average distance from the ith point to points in another cluster k. This measure ranges from +1, indicating points that are very distant from neighboring clusters, through 0, indicating points that are not distinctly in one cluster or another, to −1, indicating points that are probably assigned to the wrong cluster. The silhouette of a cluster is a plot of the $s(i)$, ranked in decreasing order, for all objects i in the cluster.

Prior to subjecting the data set to cluster analysis, we need to select a number of suitable parameters that would reflect the behavior of the client workloads. In this work, we used cpu requested, memory requested, cpu used, and memory used. Because of their correlations, they can provide clues about the actual behavior of clients which is useful towards resource provisioning.

An example of clustering is depicted in Fig. 2, in which points belonging to a cluster are assigned to a corresponding class. Commonly, the clients requesting compute resources in a cloud data center have already been pre-assigned with a certain VM template based on their subscriptions. As such, compute resources are reserved for each class of client VM which makes over-allocation imminent. By re-classifying the incoming workload, each client request is given the actual amount of resources as reflected by its behavior. This is opposite to the practice in which a cloud system passively responds by provisioning resources according to a fixed VM template.

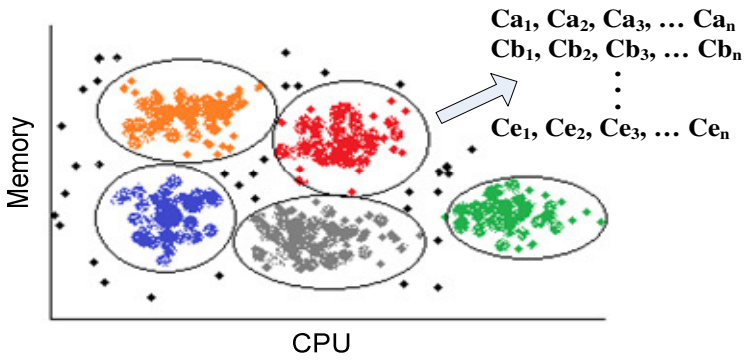


Fig. 2 Clustering of workload traces using K-means.

After the workload trace has been clustered and re-classified, the data subset from each cluster is then sorted according to their arrival time. Subsequently, each client request from the data set (e.g. {Ca_1, Cb_1, Cc_1, Cd_1, Ce_1 … Ca_n, Cb_n, Cc_n, Cd_n, Ce_n}) is placed in a queue prior to submission to the resource manager. In Fig. 3, it is shown that a client request is picked up from each cluster and included in the queue which will be forwarded to the cloud system's resource manager. From there, compute resources are dynamically provisioned according to the actual demand.

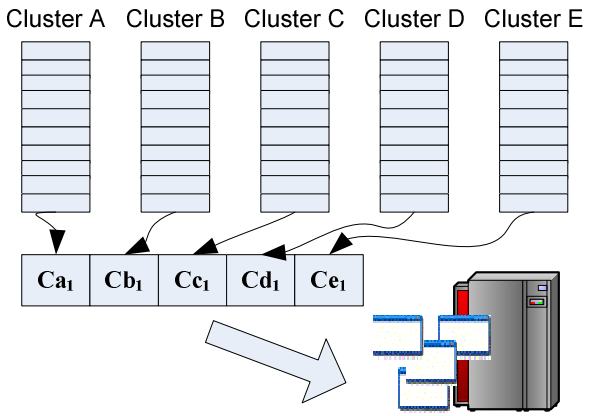


Fig. 3 Queuing of re-classified client requests.

3.3 SLA-aware VM Assignment

At this point, another important consideration is the approach for selecting the server that will host the VM to be launched. For this reason, we put forward a scheme for VM assignment that imposes SLA awareness.

SLAs provide customers with a sense of security by providing a level of assurance that their requested resources will be available and operational when they need them. When a VM's demand is not satisfied, it is said to suffer a SLA violation. Even one SLA violation can be costly to CSPs and so developing a policy that considers SLA constraints is crucial for efficient VM consolidation. The SLA metrics that we considered are the length of time an SLA violation occurred, and the ratio of the actual resource volume allocated to the volume requested. Combining the metrics, we express the SLA violation as:

$$SLA_V = \sum_{i=1}^{N} (T_e - T_s)_i * \sum_{i=1}^{N} \left(\frac{V_{allocated}}{V_{requested}}\right)_i \quad (4)$$

In Equation 4, $(T_e - T_s)_i$ is the difference between the end time and start time of an SLA violation, and $\left(\frac{V_{allocated}}{V_{requested}}\right)_i$ is the actual portion of the requested resource that has been allocated to VMs *i* to *N* respectively. Equation 4 is then subjected to equation 5 in order to derive the mean SLA violation of a host for a given data set *N*:

$$\mu SLA_V = \frac{1}{N}\sum_{i=1}^{N} SLA_{V_i} \tag{5}$$

Aside from the capability to host a given VM, another factor is to consider the server's adherence to the SLA. Our idea is to deploy high-volume VMs to hosts with good reputation based on the level of SLA violations they have encountered. Thus, hosts with lower SLA violation are more likely chosen to handle a VM with higher demands. The proposed approach is shown in Algorithm 1.

```
Algorithm: SLA-Aware VM Assignment
Input: VMRequest, ActiveHosts
Output: VMAssigment //VM Assigment to hosts
1.  Sort(ActiveHosts, utilization)
2.     //sort hosts, decreasing utilization
3.  Sort(VMRequest, volume)
4.     //sort VMs, decreasing volume
5.  For each vm in VMRequest {
6.     BestSLAV ← Max
7.     AssignedHost ← null
8.     For each host in ActiveHosts {
9.        if host.canSupport(vm) {
10.    curSLAV ← GetSLAHistory(host)
11.    if curSLAV < BestSLAV {
12.    BestSLAV ← curSLAV
13.             VMAssigment.update(vm,host)
14.      Break // goto next vm
15.    }
16.    }
17.    }
18. }
19. Return VMAssigment
```

Algorithm 1. The VM assignment strategy.

The servers capable of hosting the VMs to be consolidated are listed in decreasing order according to their current utilization level. It aims to find a host that can provision the resources for the requested VM while at the same time leaving the least unallocated resource. The VMs are also sorted according to their volume size, giving priority to those with higher requirements in acquiring the better hosts. After the host and VM lists have been created, the algorithm starts to traverse the *VMRequest* and keeps track of the current SLA violation level and host designation for the given VM. Each active host will then be checked if it can support the VM; if so, the algorithm will check further if it has a lesser SLA violation history than the previous one. The same process is done for each VM

until a suitable host is found. This procedure is repeated until all the VMs have been deployed to a host.

3.4 VM Migration Strategy

What comes next is the load balancing and mitigation process. Right after an overloaded host has been identified, it is necessary to decide which VM needs to be migrated to a less loaded host. Instinctively, we can simply choose to migrate either an under-provisioned or over-provisioned virtual machine. However, simply selecting a heavily utilized VM can disrupt the system and affect service delivery due to the resulting overhead. Furthermore, merely choosing an underutilized VM at the moment may not fully reflect its usage precedents prior to its selection. A VM which currently appears underutilized, may suddenly have huge spikes in resource consumption which raises the possibility of disrupting the host to which it is migrated. To minimize the overhead resulting from migration, we propose an approach which considers the VM's resource consumption pattern. The resource consumption of a VM is defined as the volume v of the resources actually consumed:

$$v = \sum (w * \frac{resource\_consumed}{resource\_allocated}) \tag{6}$$

In equation 6, the volume is derived by summing up the fractions of resources actually consumed by the VMs which are multiplied with corresponding weights. The weight values are assigned depending on the type of the VM machine to be provisioned. For example, a VM for serving compute intensive applications would give more weight to CPU while a transactional database server would require more weight for network bandwidth. From this, the volume set is defined as $\{v_1, v_2,\ldots, v_N\}$ composed of the VM's resource consumption accumulated on a given period. Finally, the mean volume μ of a VM is derived:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} v_i \tag{7}$$

The calculations are then used in the VM selection process. As shown in Algorithm 2, the process starts by acquiring the list of hosts that need to migrate VMs which is then sorted according to decreasing utilization levels. Each host will then have their respective VM lists traversed, in which the variables

curVolume and *minVM* are updated in each iteration should the function *GetMeanVolume(vm)* generate a value which is less than the current one. The update process goes on until the algorithm has inspected all VMs, which in return appends the VM with the lowest mean volume to the migration list. The same process is carried out for all of the remaining hosts. Finally, the VM selection routine is terminated and the final VM migration list is returned.

```
Algorithm: Minimum Mean Volume
1.  Input: HMigList, //host migration list
2.  Output: VMList //VM migration list
3.  Sort(HMigList, utilization)
4.    //sort hosts, decreasing utilization
5.  For each host in HMigList {
6.     curVolume ← Max
7.     For each vm in host{
8.        vmVolume ← GetMeanVolume(vm)
9.        if vmVolume < curVolume {
10.          curVolume ← vmVolume
11.          minVM=vm
12.       }
13.       VMList.Add(minVM)
14.    }
15. }
16. Return VMList
```

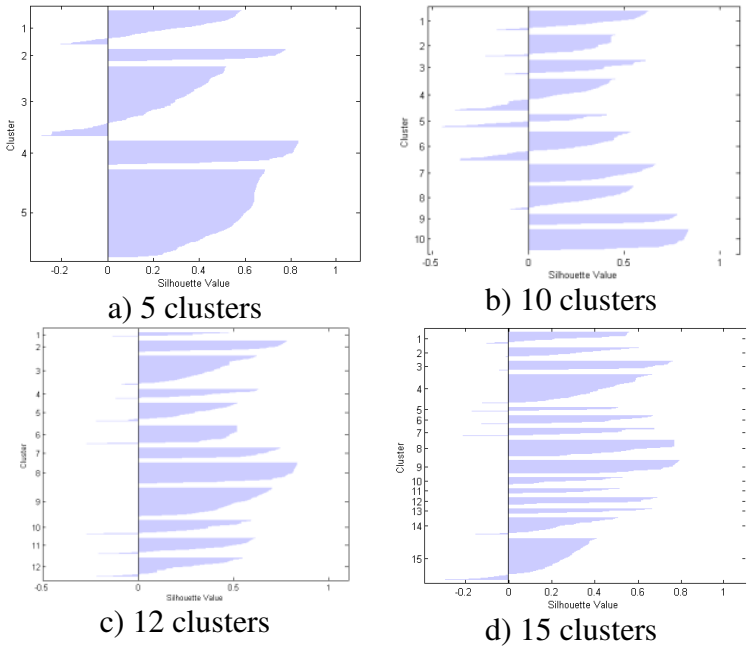
Algorithm 2. The VM Migration Strategy.

4 Simulation and Evaluation

4.1 Simulation Setup

To simulate the client requests in a datacenter, we utilized a workload trace file [21]. The log captured from October 1994 thru September 1996 consists 201, 387 jobs and contains detailed information about resource requests and use, including memory and CPU. It is found in [22] that the distribution for the whole log is distinctly modal, with several values that are extremely common coming from different flurries. After the flurry-related data is removed, the underlying distribution can be characterized as lognormal. Furthermore, the data preprocessing is initially done to filter unsuccessful jobs that were failed or aborted as well as resource utilization values that are negative. The number of clusters was determined by applying the Silhouette method on the cluster data derived from the trace file, whereas a higher silhouette value equates to better classification. In Fig 4, different values were experimented on, and 5 is found to be the best cluster number for the dataset. Although some negative silhouette values were present, they are too low and could be considered as outliers in the

dataset. As compared with other cluster numbers, setting it to 5 yields a higher *average silhouette value*.



a) 5 clusters b) 10 clusters

c) 12 clusters d) 15 clusters

Fig. 4 Cluster silhouettes

To evaluate our proposed approach, we performed a simulation which emulates the cloud computing paradigm. CloudSim toolkit [23], a simulation framework made in Java is used for the said purpose. The simulated data center is based on realistic models of VM instances and host machines composed of 1463 VM instances and 500 hosts. We utilized 4 types of VM instances with specifications based on the Amazon EC2 instance types [24] in Table 1:

Table 1: VM instances specification

Instance Type	**CPU** (1 compute unit = 1.0 Ghz)	**RAM** (GB)
M1 Small Instance	1 core with 1 EC2 Compute Unit	1.7
M1 Medium Instance	1 core with 2 EC2 Compute Units	3.75
M1 Large Instance	2 cores with 2 EC2 Compute Units each	7.5
High-CPU Medium Instance	2 cores with 2.5 EC2 Compute Units each	1.7

Next, two types of servers were considered: a) Fujitsu PRIMERGY RX300 S7 (8 cores, Intel Xeon E5-2660 2.2 GHz processor, 16GB RAM) b) IBM System X3500 M4 (8 cores, Intel Xeon E5-2680 2.7 GHz processor, 16GB RAM). The 500 host machines were equally distributed to the server types with specifications and power consumptions derived from [25] and [26] shown in Table 2. The production of multi-core CPUs and improved virtualization led to the production of modern servers equipped with large amounts of memory, which begins to dominate their power consumption [27]. Furthermore, the recent hardware advancement and the complexity of modeling power consumption by modern multi-core CPUs makes building precise analytical models a complex research problem [28]. This is same reason we utilize real data on power consumption provided by the results of the SPECpower benchmark instead of using an analytical model of power consumption by a server. Finally, the simulation is conducted in a period equivalent to a 24-hour operation of a data center.

Table 2: Server power consumption at varying loads

VM Instance	**Target Load (%)**										
	100	90	80	70	60	50	40	30	20	10	0
RX300 S7	255	217	187	156	134	117	105	94.9	85.3	74.9	53.1
X3500 M4	247	233	217	196	169	142	123	107	94.7	85.7	56.6

4.2 Evaluation

For the evaluation, we compare the performance of our work against other methods in terms of *Energy Consumption, VM Migrations, SLA Violation, Host Shutdowns,* and *Energy x SLA*. We acknowledge that deploying a VM deployment strategy alone is not enough to compensate for the QoS and energy efficiency demands of a complex infrastructure such as a cloud data center. This is the reason we paired SLA-Aware VM Assignment (SAVMA) with a clustering approach for workload re-classification; hence, R-SAVMA.

As shown in Table 3, R-SAVMA has shown significant improvement over the default approach and is therefore adopted and compared with the following methods: a) Threshold-Based (THR) approach, which requires setting the upper limit for host utilization and keeping the total CPU utilization below such threshold b) Random Selection (RS), which keeps the utilization level of hosts below the upper threshold by randomly selecting a number of VMs and migrating it to less loaded hosts. c) Non Power-Aware (NPA) policy, which does not employ energy efficient techniques and assumes 100% CPU host utilization thereby consuming maximum power at any given instance. d) Dynamic Voltage

and Frequency Scaling (DVFS), which uses dynamic voltage scaling to reduce the energy consumption of hosts.

Table 3: Comparison between SAVMA and R- SAVMA approach

Algorithm	SAVMA	R-SAVMA
Energy Consumption (kWh)	57.12	54.54
VM Migrations	17683	15891
SLA Violation %	1.62	1.44
Host Shutdowns	1640	1905
Energy X SLA	92.5344	78.5376

In Fig. 5, the evaluation result for the number of host shutdowns is shown. As can be seen, the NPA and DVFS were not able to shut down as many servers compared with their counterparts. Their poor performance is attributed to the lack of power-awareness, which is an important feature for a resource provisioning mechanism. On the other hand, THR has the most number of host shutdowns at around 2000, while those of R-SAVMA and RS were at 1900 and 1700 respectively. In a data center, idle resources that are left running still consume a certain amount of energy which adds up to the cost of operation. This initial result for R-SAVMA is further justified in the succeeding performance metrics.

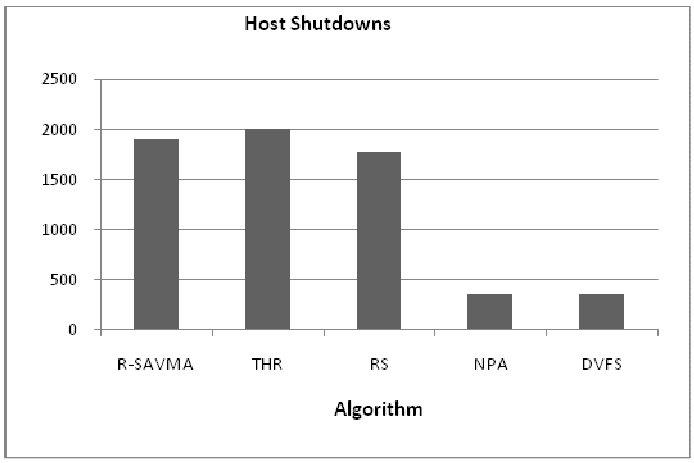


Fig. 5 Number of host shutdowns

The respective energy consumptions of the aforementioned methods are compared in Fig. 6. As shown, the results conform to that of the previous one in which the non-energy-efficient technique suffered the most with a staggering energy consumption of around 2800kWh while that of DVFS is improved at around 200kWh. For the other three approaches including ours, the results are almost similar at around 60kWh. Based on the result, we further confirm that minimizing the amount of idle resources on a cloud data center is indeed a huge contributor in the reduction of its energy consumption.

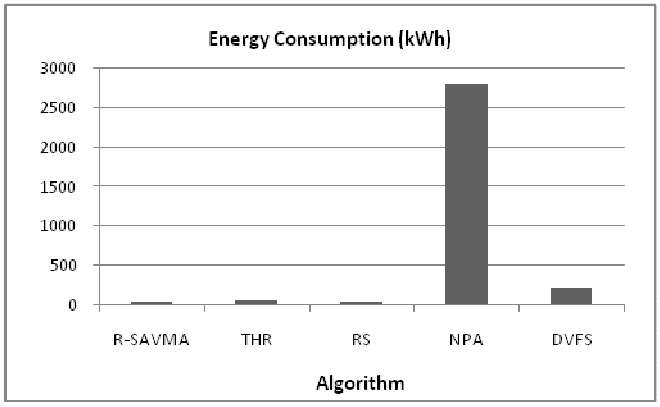


Fig. 6 Energy consumption

In the following evaluation results, it should be noted that they are not applicable to NPA and DVFS. The reason behind this is their inability to dynamically optimize resource allocation with regards to the reduction of SLA violations and energy consumption. In Fig. 7, it shows that the technique based on thresholds has the most number of VM migrations which is attributed to its static approach for triggering VM migration. In the case of R-SAVMA, its VM migration rate is a little more than 15000 while that of RS is about 17000.

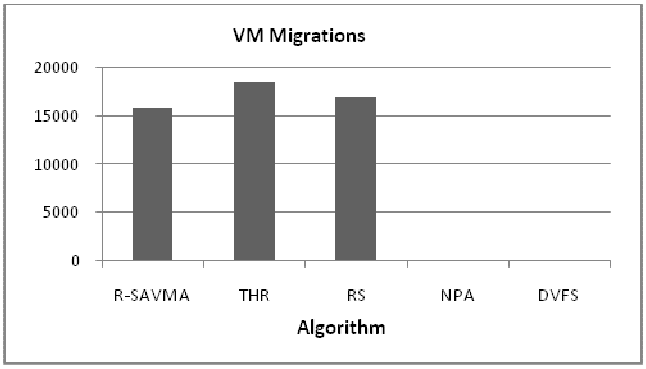


Fig. 7 Number of VMs migrated

In Fig. 8, the evaluation result for the SLA violation rate is presented. As shown, R-SAVMA has the lowest SLA violation rate at about 1.4% which means it was able to provision the requested resources more than 98% of the entire simulation period. In the case of THR, its SLA violation rate is close to 1.5%, while that of RS is at 1.7%. Due to the random nature of the RS approach, is does not guarantee that the selected VM for migration will always result to an optimal host machine utilization level, thereby making it unpredictable. In the case of the threshold-based approach, it is a matter of fine-tuning the system to determine the best threshold value for a particular scenario. These concerns were addressed by our proposed approach at the client request level as well as within the resource provisioning layer itself.

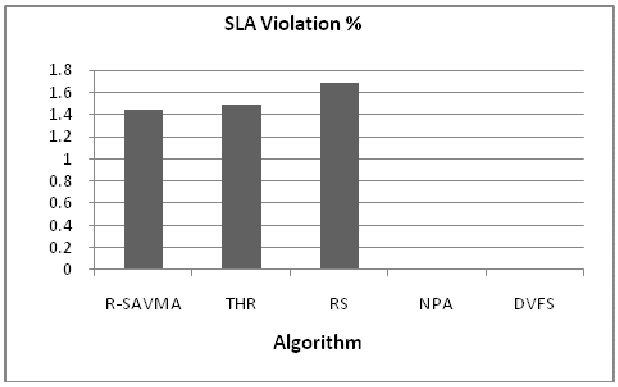


Fig. 8 SLA violation rate

Finally we put together the Energy Consumption and SLA violation rate of the methods presented in order to evaluate their performance in setting the balance between energy efficiency and quality of service. Fig. 9 shows that the R-SAVMA approach has the lowest Energy X SLA value at around 78. This is quite expected in the sense that R-SVMA has the least occurrence of SLA violations as well as the lowest energy consumption.

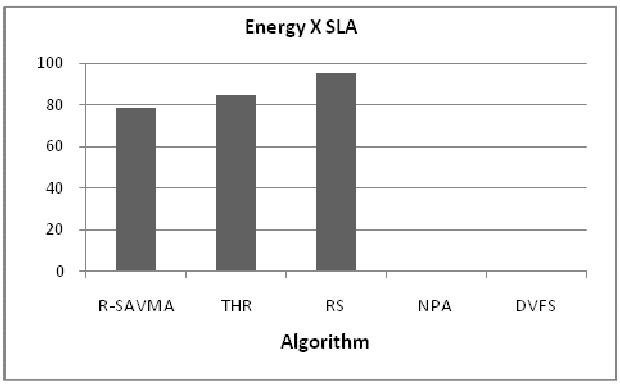


Fig. 9 Energy Consumption and SLA

5 Conclusion

In this study, the scalability of monitoring and managing a cloud data center is improved by taking advantage of workload data which is descriptive of a VM's behavior. By treating VMs as entities which possess similar characteristics, clustering was employed to categorize their resource consumption, thus, indicating their workload profile which provides an accurate estimate of the required computing resources. It is found that setting the classes of VMs to a higher number does not really characterize the collective behavior of a datacenter's incoming workloads. To further aid in the efficient provisioning of virtualized resources, an SLA-aware approach for VM assignment is also utilized for finding the appropriate hosts for workloads that are already classified. As for the efficient utilization of compute resources, techniques for migration of VMs and shutting down of idle servers were considered. In general, the proposed approach was compared with other techniques using threshold-based resource monitoring and management, VM migration by random selection, as well as the traditional non-power aware schemes. As shown by the results, SLA violations were kept low and at the same time energy efficiency is improved, thereby setting a good balance between performance and energy efficiency.

References

[1] Dillon, T., Wu, C., Chang, E. (2010). Cloud computing: Issues and challenges. in Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on, pp. 27 –33.

[2] Eyraud-Dubois, L., Larcheveque, H. (2013). Optimizing Resource allocation while handling SLA violations in Cloud Computing platforms. Parallel & Distributed Processing (IPDPS), IEEE 27th International Symposium on. pp. 20-24.

[3] U. S. Environmental Protection Agency (2007). Report to congress on server and data center energy efficiency public law 109-431. Technical report, EPA ENERGY STAR Program.

[4] Gandhi, M., Harchol-Balter, R., Das, C. (2009). Optimal power allocation in server farms. In SIGMETRICS. pp. 157–168.

[5] Gandhi, Y. Chen, D. Gmach, M. Arlitt, M. Marwah, M. (2011). Minimizing data center sla violations and power consumption via hybrid resource provisioning. In Proceedings of IEEE IGCC 2011, pp. 49-56.

[6] Toffetti, G., Gambi, A., Pezzè, M., Pautasso, C. (2010). Engineering Autonomic Controllers for Virtualized Web Applications. 10th international Conference on Web engineering.

[7] Guenter, B., Jain, N., Williams, C. (2011). Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. in Proc. of

the 30st Annual IEEE Intl. Conf. on Computer Communications (INFOCOM). pp. 1332–1340.

[8] Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer,M., Tantawi, A. (2007) An Analytical Model for Multi-tier Internet Services and its Applications. in ACM Transactions on the Web.

[9] Lin, M., Wierman, A., Andrew, L. L. H., Thereska, E. (2011). Dynamic Right-sizing for Power-proportional Data Centers. In Proceedings of IEEE INFOCOM.

[10] Amazon AutoScaling. http://aws.amazon.com/autoscaling/.

[11] Tirado, J. M., Higuero, D., Isaila, F., Carretero, J. (2011). Predictive data grouping and placement for cloud-based elastic server infrastructures. In 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.

[12] Antonescu, A-F., Robinson, P., Braun, T. (2013). Dynamic SLA management with forecasting using multi-objective optimization. Integrated Network Management 2013 IFIP/IEEE International Symposium on. pp.457- 463.

[13] Marshall, P., Keahey, K.. Freeman, T. (2011). Improving utilization of infrastructure clouds. In 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.

[14] Householder, R., Arnold, S., Green, R. (2014). On Cloud-based Oversubscription. International Journal of Engineering Trends and Technology (IJETT). vol. 8 no. 8. 2014.

[15] Ghosh, R., Naik, V.K. (2012) Biting Off Safely More Than You Can Chew: Predictive Analytics for Resource Over-Commit in IaaS Cloud. Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. pp.25-32.

[16] Moreno, I. S., Xu, J. (2012). Neural Network-Based Overallocation for Improved Energy-Efficiency in Real-Time Cloud Environments. IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. pp. 119-126.

[17] Tomás, L., Tordsson, J. (2013). Improving cloud infrastructure utilization through overbooking. In Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference.

[18] Zhang, X., Shae, Z.Y., Zheng, S., Jamjoom, H. (2012). Virtual machine migration in an over-committed cloud. Network Operations and Management Symposium (NOMS). pp. 192-203.

[19] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proc of the 5th Berkeley symposium on mathematical statistics and probability.

[20] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics. pp. 53–65.

[21] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/l_lanl_cm5/index.html.

[22] Feitelson, D. G., Tsafrir, D. (2006). Workload sanitation for performance evaluation. In IEEE Intl. Symp. Performance Analysis of Systems and Software, pp. 221-230.

[23] CloudBus. (2011). CloudSim: a toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience. pp. 23–50.

[24] Amazon EC2 Instance Types. http://aws.amazon.com/ec2/instance-types

[25] Standard Performance Evaluation Corporation. http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110209-00353.html

[26] .Standard Performance Evaluation Corporation. http://www.spec.org/power_ssj2008/results/res2010q2/power_ssj2008-20100315-00239.html

[27] Minas, L., Ellison, B. (2009). Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers. Intel Press: Hillsboro.

[28] Beloglazov, A., Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurrency Computat.: Pract. Exper., 24: 1397–1420.