ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

클라우드 시스템에서 동적 임계치와 호스트 평판도를 기반으로 한 성능 및 에너지 중심 자원 프로비저닝☆

Performance and Energy Oriented Resource Provisioning in Cloud Systems Based on Dynamic Thresholds and Host Reputation

프랭크 엘리호데[1]　　이 재 완[2*]
Frank I. Elijorde　　Jaewan Lee

요　약

정의된 SLA의 QoS를 지키기 위해서, 클라우드 시스템은 동적인 사용 패턴에서 발생하는 변화무쌍한 작업 부하를 처리해야 한다. 서비스 관점이외에도 에너지 소비를 최소화 하는 것이 또한 새로운 관심사이다. 이는 클라우드 데이타 센터에서 가상화된 자원을 할당할 때 클라우드 제공자들은 에너지와 성능의 상관관계를 고려해야 한다. 본 논문에서는 호스트 컴퓨터의 작업부하 수준을 탐지하기 위해 동적 임계치를 기반으로 한 자원 프로비저닝 방안을 제시한다. VM선정 정책은 이주할 VM을 선택하기 위해 활용 데이터를 사용하며, VM 할당 정책은 서비스 평판도에 따라 VM들을 호스트에 지정한다. 시뮬레이션을 통해 연구결과를 평가하였으며, 시뮬레이션 결과 이주를 지원하지 않는 비 전력 방법뿐만 아니라 동적 임계치, 임의 선정 정책보다 성능이 우수함을 보였다.

주제어 : 클라우드 컴퓨팅, 자원 프로비저닝, 클라우드 데이타 센터 가상화, 그린 컴퓨팅

ABSTRACT

A cloud system has to deal with highly variable workloads resulting from dynamic usage patterns in order to keep the QoS within the predefined SLA. Aside from the aspects regarding services, another emerging concern is to keep the energy consumption at a minimum. This requires the cloud providers to consider energy and performance trade-off when allocating virtualized resources in cloud data centers. In this paper, we propose a resource provisioning approach based on dynamic thresholds to detect the workload level of the host machines. The VM selection policy uses utilization data to choose a VM for migration, while the VM allocation policy designates VMs to a host based on its service reputation. We evaluated our work through simulations and results show that our work outperforms non-power aware methods that don't support migration as well as those based on static thresholds and random selection policy.

☞ keyword : Cloud Computing, Resource Provisioning, Virtualization, Green Computing

1. Introduction

As a new computing paradigm, cloud computing allowed the provisioning of application services and computational resources through the Internet which provides an efficient and economical way to share resources within organizations as well as promote business strategy. With this approach, clients are charged only for the actual resource usage. It provides cloud consumers with immediate and cost-effective way to scale down or scale up their system without having to worry about the constraints imposed by physical resources such as hardware. Currently, the research and development of various cloud-based services are rapidly increasing [1] while at the same time, improvements in the cloud system infrastructure continue to move forward [2].

However, the provisioning of services over the cloud brings many challenges. To efficiently serve the customers, the virtualized resources have to be carefully configured and allocated according to the incoming workloads. As such, resource demand in the cloud is considered more unpredictable as compared with traditional IT environments.

1 Institute of ICT, West Visayas State University, Philippines
2 Dept. of Information and Communication Engineering, Kunsan National University, Korea
* Corresponding author (jwlee@kunsan.ac.kr)
☆ This paper was supported by research funds of Kunsan National University
[Received 2 June 2013, Reviewed 13 June 2013, Accepted 10 September 2013]

It is therefore necessary for the provider to accurately estimate and predict resource utilization in order to maintain the quality of service (QoS) as well as to optimize the utilization of physical resources. By realizing this, the Service Level Agreement (SLA) can be properly met and idle resources can be minimized leading to mutual benefits for the cloud consumer and the cloud service provider.

However, the increasing energy consumption of computing systems has started to limit further performance growth due to overwhelming electricity bills and carbon dioxide footprints. Therefore, the goal of the computing system design has been shifted to power and energy efficiency [3].

With the goal of maintaining QoS as well as setting the balance between performance and energy efficiency, we present a resource provisioning approach for cloud systems which considers adaptive monitoring and allocation of resources. Specifically, the optimal consolidation of virtualized resources is achieved by dynamically adapting the system's utilization threshold to the resource consumption data. As such, it enables the cloud infrastructure to easily scale up or scale down the provisioned resources based on the truly occurring usage scenario. To verify the performance of our proposed approach, we compare it with other previously proposed techniques and also with non-optimized conventional approaches.

2. Related Work

2.1 Load prediction in Cloud Data Centers

In order to efficiently provision computing resources in the cloud, system administrators need the capabilities of characterizing and predicting server workload. In [4], they use data center traces to search for repeatable workload patterns on different servers. The experimental results conclude that the method could help system administrators better understand group-level workload characteristics in a cloud and make more accurate predictions on workload changes over time. Consolidation of virtual machines using correlation or peak cluster-based placement is proposed in [5]. A trace-based workload forecasting method was used in [6] for capacity management. These approaches rely on the statistical measurements of individual workload time series to predict future capacity demand. Another approach for the management of virtualized data centers was proposed in [7]. The problem of scheduling tasks in a data center and their allocation to physical machines is expressed as a multi-objective optimization problem. An improved strategy based on the tendency with several steps backward was introduced in [8], using polynomial fitting method to produce the prediction values. Presumably, these models generally perform well; however, they have an obvious source of error whenever the time series changes its direction.

2.2 Resource Provisioning in Virtualized Environments

The authors of [9] proposed an algorithm for virtual machine allocation based on the ant colony optimization meta-heuristic. The core of their approach is on the generation of solutions that minimizes the number of physical machines used in order to decrease the energy consumed by the data centers. A study in [10] presents a statistical approach for the adaptive allocation of virtual machines to physical servers. The authors looked into the possibility of live migration of virtual machines both to avoid the occurrence of servers with small workloads and to avoid the possible overloading of the servers which may result in a violation of the service agreements. Alternatively, overbooking [11] is used to improve the overall resource utilization. By regulating resource consumption, performance isolation among co-located applications is achieved by guaranteeing that no application can consume more resources than those allocated to it. In contrary, dedicated hosting is a dynamic provisioning approach characterized by physical machines that run at a single application wherein workload increases are handled by spawning a new replica of the application on idle servers.

2.3 Energy Efficiency in Cloud Data Centers

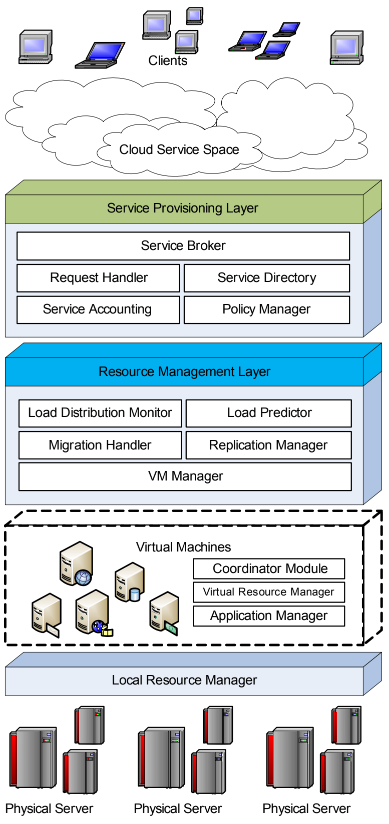
The work in [12] is considered as one of the first works which were able to apply power management in a virtualized data center setting. The authors proposed a system architecture for the management of data center resources which is divided into local and global policies. At the local level the system leverages the guest OS's power management strategies. The global manager gets the information on the current resource allocation from the local managers and applies its policy to decide whether the VM placement needs to be adapted. Reducing power consumption in server clusters has been the aim of [13] and [14]. To achieve this, they utilize CPU clock throttling while switching entire servers on/off as needed, depending on the incoming workload. However, considering switching costs, a study in [15] pointed out two crucial issues that must be addressed: First, turning servers off in a dynamic environment could be risky in terms of QoS. If servers were just powered off expecting a lighter workload and the workload suddenly increases, this would seriously affect the QoS received by users. Similarly, excessive power cycling of a server could reduce its reliability. These two scenarios, if not properly addressed would result to an SLA violation which would require the provider to pay the customer for every failed delivery of service.

3. Performance Aware and Energy Efficient Resource Provisioning

3.1 Cloud System Architecture

As shown in Figure 1, the proposed cloud system is presented as a three-layer structure composed of the Service Provisioning, Resource Management, and Virtual Machine layers. Each layer includes various components which contribute to the functionalities of the system.

The core of the provisioning mechanism resides within the Resource Management Layer, which is the focus of the host utilization monitoring and optimization approach that we propose. Specifically, the load predictor and VM manager are responsible for the prediction and efficient management of the resources utilized by the services provided to the users. Aside from keeping track of the host's load, this layer also concerns the creation of the virtual platform, followed by an initial installation of the necessary software components, along with the configuration and subsequent deployment of the software service. Put together, our VM selection and VM



(Figure 1) The architecture of the proposed cloud system.

assignment strategies handle the distribution, replication, and migration of services within the virtualized environment through the load distribution monitor, replication manager, and migration handler.

3.2 Host Utilization Monitoring and Optimization

Physical machines can turn into *hot spots* in which available resources are not sufficient to satisfy the provisioning requirements; while *cold spots* are over-provisioned hosts which lead to underutilization of resources. From a cloud provider's point of view, handling *hot spots* is extremely important in order to meet the quality of service agreed upon with the clients. Moreover, eliminating *cold spots* would also leverage the optimal utilization of physical resources and eliminate resource wastage thereby taking advantage of virtualization to its full potential.

```
Algorithm: Optimize Utilization
While request <> 0
  uThreshold= GetUpperThreshold()
  lThreshold= GetLowerThreshold()
  For each host in ActiveHosts{
     if hostOverloaded(host, uThreshold)
        HMigList.Add(host)
     if hostUnderloaded(host, lThreshold)
        HMigList.Add(host)
  }
  For each host in HMigList{
   SelectVM(host) //select vm from host
     VMList.Add(vm)
  }
  clear HMigList
  For each vm in VMList{
     AllocateVM(vm,host)
  }
  Clear VMList
End while
```

Algorithm 1. The utilization optimization approach.

To attain this, a number of considerations have to be met. First, we need to know whether a host is overloaded which would require migrating one or more VMs to a less loaded host. Similarly, an underloaded host also needs to migrate its VMs to another host so it can be put to a low-power mode. Another consideration is the policy for the selection of VMs that need to be migrated. Finally, the VMs chosen for migration need to be re-deployed to new hosts; this process is not straightforward and also needs an efficient technique. Below, we show the overall algorithm for the optimization approach.

As shown in algorithm 1, the process starts by setting the upper and lower thresholds for the utilization level of the hosts. A naïve solution to this problem is to simply set fixed values for the thresholds. However, in an environment where heterogeneous services share the same physical resources, workloads are highly dynamic. This makes fixed utilization thresholds unsuitable. For this reason, we devise an adaptive method which extracts time series data of the host's most recent utilization history. Using the derived accumulated data, we determine its standard deviation:

$$S_N = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{1}$$

Where $\{x_1,x_2,.,x_N\}$ are the observed values of the host's utilization history, while is the average utilization level for the given N samples. Put in the case of CPU utilization, a higher deviation among data points results to a lower value of the upper utilization threshold. Furthermore, a higher deviation would likely lead to a CPU utilization close to 100% resulting to an SLA violation. Upon deriving the standard deviation, we obtain the upper threshold as:

$$T_U = 1 - (S_N * s) \tag{2}$$

where parameter s is a value which influences the tradeoff between QoS and energy efficiency. Once the upper threshold has been determined, the next step is to derive the lower threshold. To obtain the lower threshold, we use the following equation:

$$T_L = T_U - (T_U * p) \tag{3}$$

The variable p is used to determine the gap between the upper and lower threshold. From our initial experiment, we observed that the distance between the thresholds also has a significant impact on the performance of the cloud system's provisioning mechanism.

3.3 Virtual Machine Selection Strategy-Minimum Mean Volume

Following the previous phase, comes the mitigation and load balancing processes. 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. 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 realize the goal of minimizing 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:

$$v = \sum (w * \frac{resource\_consumed}{resource\_allocated}) \qquad (4)$$

In equation 4, the volume is derived by summing up the fractions of resources (CPU, RAM, Network Bandwidth) actually consumed by the VMs which is 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:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} v_i \qquad (5)$$

These 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: Reputation-based Best Fit
Input: VMList, ActiveHosts
Output: NewHosts
   //Updated active hosts   list
Sort(ActiveHosts, utilization)
   //sort   hosts, decreasing utilization
Sort(VMList, volume)
   //sort VMs,   decreasing volume
For each vm in   VMList{
   BestReputation = Max
   AssignedHost = null
   For each host in ActiveHosts {
      if host.canSupport(vm){
         curReputation =         GetSLAHistory(host)
         if curReputation   <    BestReputation {
            BestReputation = curReputation
            NewHosts.update(vm, host)
            Break
         }
      }
   }
}
Return NewHosts
```

(Algorithm 2) The VM Selection Strategy.

3.4 Virtual Machine Assignment Strategy - Reputation-based Best Fit

Once the VMs that need to be migrated have been chosen, the next step is to assign them to their new hosts. The strategy for choosing the hosts for the migrating VMs is concerned not only about finding hosts that can support them but also to maintain the desired system throughput by keeping the disruption as little as possible. We believe that simply choosing a capable host based on the principles of bin packing is not enough; the problem of VM consolidation is more than just provisioning resources. It should also consider the past performance of the hosts in order to anticipate their future behavior.

To do this, we propose a scheme for VM assignment based on the host's reputation with regards to SLA violations. 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:

$$SLA_V = \sum_{i=1}^{N} (T_e - T_s)_i * \sum_{i=1}^{N} (\frac{v_{allocated}}{v_{requested}})_i \qquad (6)$$

where is the difference between the end time and start time of an SLA violation, and is the actual fraction of the requested resource that has been allocated to VMs i to N respectively. Finally, equation (6) is subjected to equation (7) in order to derive the mean SLA violation of a host for a given data set *N*:

$$\mu SLA_V = \frac{1}{N}\sum_{i=1}^{N} SLA_{V_i} \tag{7}$$

Aside from the capability to support a given VM, another criterion is to look at the host's previous performance based on how well it upheld the SLA. The idea is to migrate high-volume VMs to hosts with outstanding reputation based on the lightness of SLA violations they got involved with. The lower the SLA violation caused by a host, the more likely it will be chosen to handle a VM of higher volume. That way, light VMs can just be migrated to hosts with fair reputation. The proposed approach is shown in Algorithm 3.

Initially, the hosts capable of supporting the VMs to be migrated are listed in decreasing order according to their current utilization level. This is done in support of the Best Fit approach for Bin Packing which aims to find a host that can provision the required volume of the incoming VM while at the same time leaving the least unallocated resource. Similarly, the VMs are also sorted according to their volume size, giving priority to those with high volumes in acquiring the better hosts. Once the host and VM lists have been established, the algorithm starts to traverse the *VMList* and initializes the variables for keeping track of the current best reputation 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 better reputation than the previous one. For each VM, the process keeps on until a suitable host is found. The entire procedure is repeated until all the VMs have been successfully migrated.

```
Algorithm: Minimum Mean Volume
Input: HMigList,    //host migration list
Output: VMList //VM    migration list
Sort(HMigList, utilization)
   //sort   hosts, decreasing utilization
For each host in   HMigList {
   curVolume    = Max
   For each vm in host{
vmVolume =              GetMeanVolume(vm)
      if vmVolume < curVolume {
         curVolume = vmVolume
         minVM=vm
      }
      VMList.Add(minVM)
   }
}
Return VMList
```

(Algorithm 3) The VM allocation strategy.

4. Simulation and Evaluation Results

4.1. Simulation Setup

To ensure the repeatability, scalability, and dependability of experiments, we use simulation to evaluate the performance of the proposed methods. For the simulation platform, we used CloudSim toolkit [16]; a simulation framework made in Java aimed at cloud computing environments. As opposed to other simulation toolkits, CloudSim enables the modeling of virtualized environments, as well as supporting on-demand provisioning of resources, and their management. By modifying and extending parts of the simulator, we implemented our proposed algorithms.

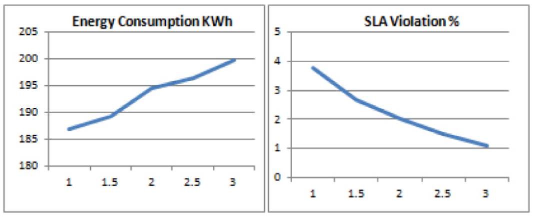
We built the setup of our simulated data center by using realistic models of VM instances and host machines. For the 400 VM instances, we used 4 types of VM instances with characteristics similar to the Amazon EC2 instance types in Table 1 [17]. As with the hosts, we considered 100 physical machines which were equally distributed among the two types of servers with specifications and power consumptions derived from [18] and [19]. The first variant is HP ProLiant DL380 G7 (6 cores, Intel Xeon X5675 3.07 GHz processor, 12GB RAM) with 2 CPUs enabled. The other is IBM System X3550 M3 (6 cores, Intel Xeon X5670 2.9 GHz processor, 12GB RAM) with 2 CPUs enabled. Both servers were configured with 1000 Mb network bandwidth.

(Table 1) VM instances specification.

Instance Type	CPU (1 compute unit = 1.0 Ghz)	RAM (GiB)
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

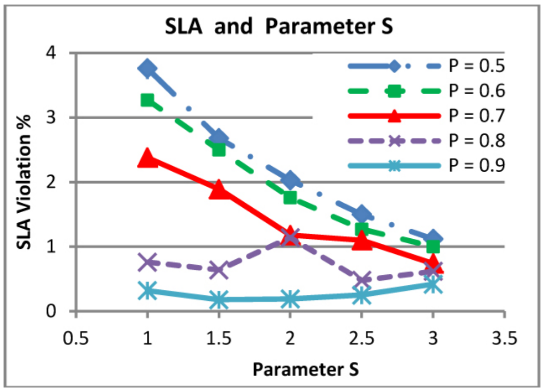
4.2. Evaluation Results

To determine the optimum configuration, we first performed experiments on different values for the parameters *s* and *p*. As shown in Figure 2, setting the value higher would result to a lower SLA violation rate although at the expense of a higher energy consumption:



(Figure 2) The effect of tweaking parameter s on the energy consumption and SLA violation.

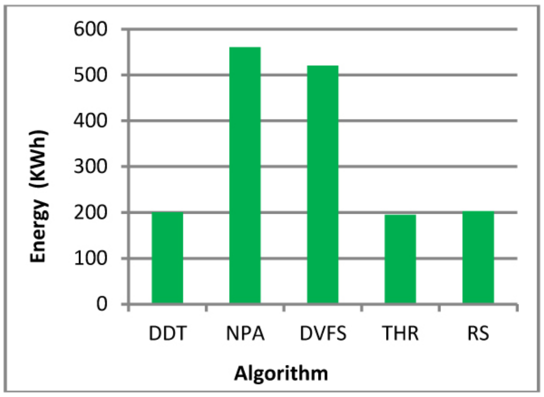
In Figure 3, we show the respective levels of SLA violation for a given value of parameter *p*. The result confirms that setting the value over 0.7 would result to unpredictable SLA behavior and higher energy consumption which brings us to a conclusion that 0.8 and .0.9, although have initially shown lower SLA violations are not safe for use. The basis for this conclusion is that, setting parameter p to a value higher than 0.7 would increase the gap between the upper and lower utilization thresholds. Furthermore, this would also result to a significantly reduced lower threshold which affects the algorithm's judgment towards underutilized hosts. With such configuration, the optimization routine becomes overly unbalanced due to its impractical bias towards low SLA violations. This results to fewer VM migrations, fewer host shutdowns, and disproportionately high energy consumption. From this, we decide that the best configuration for the parameters *s* and *p* that would produce the best *Energy X SLA* combination is 3.0 and 0.7 respectively.



(Figure 3) The influence of parameter p on SLA

After we derive the best configuration for our proposed scheme, we evaluate its performance and compare it with other methods. The methods chosen for comparison are: *a)* The 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. *b)* Dynamic Voltage and Frequency Scaling (DVFS), which uses dynamic voltage scaling to reduce the energy consumption of hosts. c) Threshold-Based (THR) approach, which requires setting the upper limit for host utilization and keeping the total CPU utilization below such threshold *d)* 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. As for the evaluation metrics, we compare our work with the aforementioned methods in terms of energy consumption, SLA violation rate, number of VM migrations, and the *Energy X SLA* combination.

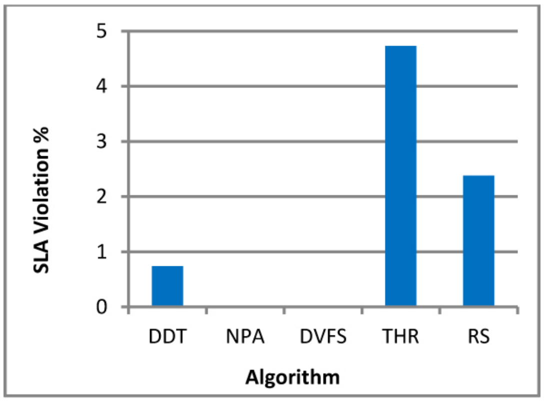
In Figure 4, we present the result of evaluating the respective energy consumption of the given resource provisioning techniques. As expected, the NPA approach has the highest energy consumption at around 560 KWh



(Figure 4) Total energy consumption.

followed by DVFS at around 520, then by Dynamic Double Threshold (DDT) and RS both close to 200, and THR at around 195. This initial result indeed confirms that energy efficiency is really important in large scale computing infrastructures such as cloud data centers.

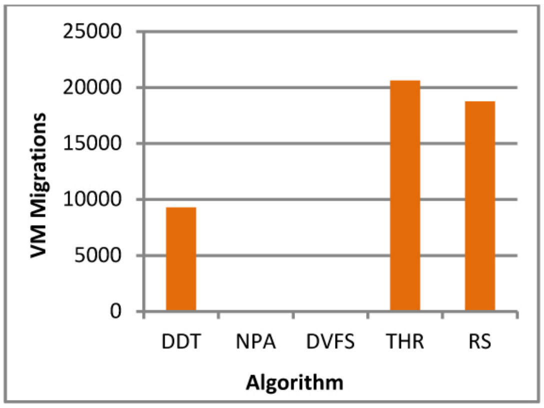
Before we go further, we would like to point out that in Figures 5, 6, and 7, the metrics presented does not apply to NPA and DVFS. This is for the reason that both approaches have no capabilities to dynamically optimize resource allocation, as well as monitoring SLA violations and energy consumption.



(Figure 5) Overall SLA violation rate.

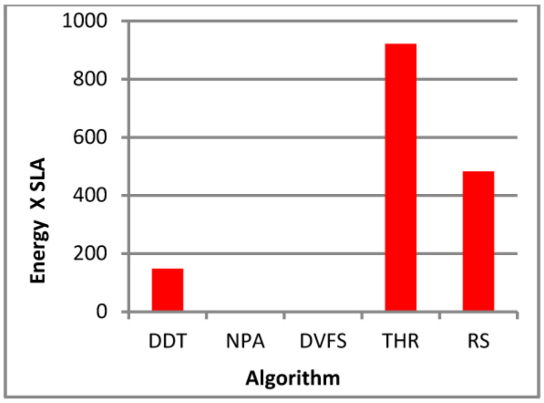
The results presented in Figure 5 show that the DDT approach has the lowest overall SLA violation rate at 0.75%; followed by Random Selection approach at around 2.40%, while the Threshold-Based approach has the highest SLA violation rate at about 4.75%. The result implies that in the entire operation of the data center, the DDT approach performed best and was able to deliver the agreed SLA level at 99.25%, while the Threshold-Based provided the agreed performance at the lowest rate of 95.25%. This tells us that fixed utilization thresholds are not suited for environments such as cloud data centers which are designed to handle highly dynamic workloads involving unpredictable usage patterns.

Shown in Figure 6 is the number of VMs migrated during the entire operation of the simulated data center. In the following metric, DDT has the smallest number of migrated VMs at around 9000, followed by the RS approach at about 19000, and THR at a little more than 20000. This supports the result from the previous metric in the sense that lesser migrations would minimize the disruption of a VM's operation. This is due to the fact that each time a VM is migrated, certain overheads would cause the VM to suffer temporary performance degradation thereby resulting to an SLA violation. The cost of migrating a VM depends on its actual size as well as its current utilization level at the time that the migration process was invoked.



(Figure 6) Number of VMs migrated.

Finally, we decide on the performance of the three power-aware optimization schemes using the metric which combines energy efficiency and performance quality derived from the product of their respective energy consumption and SLA violation rate. As shown in Figure 7, DDT produced the best result at around 150, followed by RS at about 490, and the worst is that of THR at more than 900. Despite the similarity in the total energy consumption of DDT, THR, and RS in Figure 4, the low *Energy X SLA* value of DDT is heavily influenced by its very minimal SLA violation rate which is less than 1%. This brings us to a conclusion that the technique of how virtualized resources in a cloud data center are provisioned and utilized leads to effective VM consolidation. Thus, achieving such efficiency sets the desired balance between service quality and energy efficiency.



(Figure 7) SLA violation and energy consumption combined.

5. Conclusion and Future Work

In this paper, we presented an efficient resource provisioning approach which considers service quality and energy efficiency in a cloud data center. By dynamically adjusting the utilization thresholds, it can easily adopt to the dynamic workload behavior of the system. Furthermore, a VM selection technique based on the actual resource consumption history is also presented. Finally, a reputation-based VM assignment strategy is employed to maintain system throughput and minimize overheads brought about by VM migration. Evaluation results indicate that our approach surpassed non-power aware methods that don’t support migration as well as those based on static thresholds and random selection policy. The significant performance of our work is verified by its outstanding combination of very minimal SLA violation and low energy consumption.

In our future work, we intend to incorporate autonomous hosts and see how it can improve resource provisioning in cloud data centers. We believe that incorporating decentralized resource management, allocation, and monitoring can greatly improve the throughput and energy efficiency of a cloud system.

References

[1] K. Gai, S. Li, Towards Cloud Computing: A Literature Review on Cloud Computing and Its Development Trends, Multimedia Information Networking and Security (MINES), 2012 Fourth International Conference on, (2012) pp.142-146.

[2] A. Tantar, Alexandru-Adrian; A. Q. Nguyen; P. Bouvry, B. Dorronsoro, E.G. Talbi: Computational intelligence for cloud management current trends and opportunities, Evolutionary Computation (CEC), 2013 IEEE Congress on, (2013) pp.1286-1293.

[3] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya: A taxonomy and survey of energy-efficient data centers and cloud computing systems, Univ. of Melbourne, Tech. Rep. CLOUDS-TR-2010-3 (2010).

[4] Khan, X. Yan, T. Shu, N. Anerousis: Workload characterization and prediction in the cloud: A multiple time series approach, Network Operations and Management Symposium, (2012) pp.1287-1294.

[5] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari: Server workload analysis for power minimization using consolidation, in Proceedings of USENIX Annual Technical Conference (2009).

[6] J. Rolia, L. Cherkasova, M. Arlitt, and A. Andrzejak: A capacity management service for resource pools, In Proceedings of ACM Workshop on Software and Performance (2005).

[7] X. Kong, C. Lin, Y. Jiang, W. Yan, X. Chu: Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction, Journal of Network and Computer Applications, Vol. 34, (2011) pp. 1068-1077.

[8] Y. Zhang, W. Sun, and Y. Inoguchi: CPU Load Predictions on the Computational Grid, in Proc. of IEEE International Symposium on Cluster

Computing and the Grid, (2006) pp. 321-326.

[9] E. Feller, L. Rilling, C. Morin: Energy-Aware Ant Colony Based Workload Placement in Clouds, Technical Report, INRIA (2011).

[10] M. Mastroianni, M. Meo, G. Papuzzo: Self-economy in cloud data centers: statistical assignment and migration of virtual machines, In Proc. of the 17th International Conference on Parallel Processing, Vol. 1 (2011).

[11] B. Urgaonkar, P. Shenoy, and et al.: Resource overbooking and application profiling in shared hosting platforms, In Proc. OSDI (2002).

[12] R. Nathuji and K. Schwan, Virtualpower: Coordinated power management in virtualized enterprise systems. ACM SIGOPS Operating Systems Review (2007) pp.265-278.

[13] P. Ranganathan, P. Leech, D. E. Irwin, and J. S. Chase: Ensemble-level power management for dense blade servers, in Proc. of the 33th Annual Intl. Symposium on Computer Architecture (2006).

[14] C. Lefurgy, X. Wang, and M. Ware: Server-level power control, in Proc. of the Intl. Conference on Autonomic Computing (2007).

[15] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang: Power and performance management of virtualized computing environments via lookahead control, Cluster Computing, vol. 12, no. 1, (2009) pp. 1-15.

[16] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, Buyya R: CloudSim: a toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience (2011) pp. 23-50.

[17] “Amazon EC2 Instance Types”, http://aws.amazon.com/ec2/instance-types

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

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

◐ 저 자 소 개 ◑

프랭크 엘리호데 (Frank I. Elijorde)

2003년 Western Visayas College of Science and Technology, Philippines BS in Information Technology
2007년 Western Visayas College of Science and Technology, Philippines MS in Computer Science
2011년～현재 Kunsan National University, South Korea, Graduate Student in Ph. D. Course
관심분야 : Distributed systems, cloud computing, data mining, ubiquitous sensor networks, RFID
E-mail : frank@kunsan.ac.kr

이 재 완(Jaewan Lee)

1984년 중앙대학교 이학사-전자계산학
1987년 중앙대학교 이학석사-전자계산학
1992년 중앙대학교 공학박사-전자계산학
1996년 3월~1998년 1월 한국학술진흥재단 전문위원
1992년~현재 군산대학교 교수
관심분야 : 분산 시스템, 운영체제, 유비쿼터스 시스템, 클라우드 컴퓨팅 등
E-mail: jwlee@kunsan.ac.kr