Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers

Anton Beloglazov and Rajkumar Buyya
CLOUDS Lab, Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia
{abe, raj}@csse.unimelb.edu.au

ABSTRACT

The rapid growth in demand for computational power driven by modern service applications combined with the shift to the Cloud computing model have led to the establishment of large-scale virtualized data centers. Such data centers consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. Dynamic consolidation of virtual machines (VMs) and switching idle nodes off allow Cloud providers to optimize resource usage and reduce energy consumption. However, the obligation of providing high quality of service to customers leads to the necessity in dealing with the energy-performance trade-off. We propose a novel technique for dynamic consolidation of VMs based on adaptive utilization thresholds, which ensures a high level of meeting the Service Level Agreements (SLA). We validate the high efficiency of the proposed technique across different kinds of workloads using workload traces from more than a thousand PlanetLab servers.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems

General Terms: Algorithms, Experimentation

Keywords: Green IT, Cloud computing, VM placement

1. INTRODUCTION

The Cloud computing model leverages virtualization of computing resources allowing customers to provision resources on-demand on a pay-as-you-go basis [4]. Instead of incurring high upfront costs in purchasing IT infrastructure and dealing with the maintenance and upgrades of both software and hardware, organizations can outsource their computational needs to the Cloud. The proliferation of Cloud computing has resulted in the establishment of large-scale data centers containing thousands of computing nodes and consuming enormous amounts of electrical energy. It has been estimated that in 2006 the energy consumed by IT infrastructure in the US was about 61 billion kWh, leading to 4.5 billion dollars in electricity costs [3]. Under current efficiency trends this is likely to double by 2011. The reason

MGC '2010, 29 November - 3 December 2010, Bangalore, India.

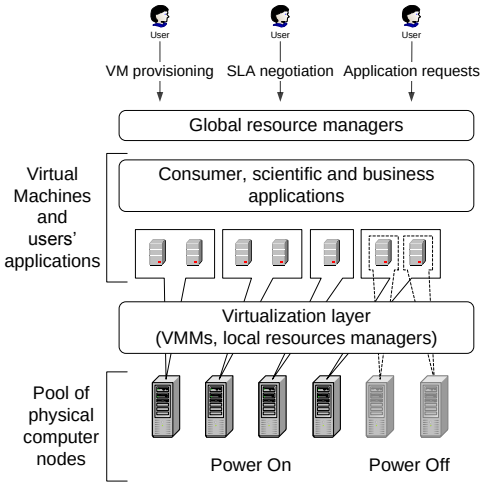


Figure 1: The system view

for this extremely high energy consumption is not just in the amount of computing resources used and the power inefficiency of hardware, but rather lies in the inefficient usage of these resources. Data collected from more than 5000 production servers over a six-month period showed that servers operate only at 10-50% of their full capacity most of the time, leading to expenses on over-provisioning, and thus extra Total Cost of Acquisition (TCA) [1]. Another problem is the narrow dynamic power range of servers: even completely idle servers still consume about 70% of their peak power [6].

Virtualization technology allows Cloud providers to address the energy inefficiency by creating multiple Virtual Machine (VMs) instances on a physical server, thus improving the utilization of resources and increasing the Return On Investment (ROI). The reduction in energy consumption can be achieved by switching idle nodes off, thus eliminating the idle power consumption (Fig. 1). Moreover, by using live migration the VMs can be dynamically consolidated on the minimal number of physical nodes according to their current resource requirements. However, efficient resource management in Clouds is not trivial, as modern service applications often experience highly variable workloads causing dynamic resource usage patterns. Therefore, aggressive consolidation of VMs can lead to performance degradation when an application encounters an increasing demand resulting in a rise in the resource usage. Ensuring reliable Quality of Service (QoS) defined via Service Level Agreements (SLA) is essential for Cloud computing environments; therefore, Cloud providers have to deal with the energy-performance trade-

off. The focus of this work is on energy and performance efficient resource management strategies that can be applied in a virtualized data center by a Cloud provider. We propose a novel approach for dynamic consolidation of VMs, which is able to reduce energy consumption and maintain the level of SLA violation in the system at as low as 1%.

The remainder of the paper is organized as follows. In Section 2 we discuss the related work followed by the system model in Section 3. We present a thorough description of the proposed approach in Section 4, continuing with an evaluation in Section 5 and analysis of the obtained experimental results in Section 5.2. We make a conclusion and discuss possible directions for future research in Section 6.

2. RELATED WORK

Nathuji and Schwan [9] have proposed an architecture of an energy management system for virtualized data centers, where resource management is divided into local and global policies. Consolidation of VMs is handled by global policies applying live migration to reallocate the VMs. Kusic et al. [8] have stated the problem of continuous consolidation as a sequential optimization and addressed it using Limited Lookahead Control (LLC). The proposed model requires simulation-based learning, and the execution time reaches 30 minutes even 15 nodes. On the contrary, our approach is heuristic-based allowing a reasonable performance even for large-scale. Song et al. [10] have proposed resource allocation to applications according to their priorities in multi-application virtualized clusters. Unlike our work, it does not apply migration of VMs to optimize the allocation.

Verma et al. [12] have applied a heuristic for the bin packing problem to tackle the dynamic placement of applications in virtualized heterogeneous systems. On the contrary to our approach, the proposed algorithms do not ensure QoS fulfillment: SLA can be violated due to the workload variability. Gmach et al. [7] have investigated a threshold-based reactive approach to dynamic workload consolidation. The proposed approach is in line with our preliminary work [2]. However, this approach is not suitable for an IaaS environment serving different kinds of applications, as the threshold values have to be tuned for each workload type to allow the consolidation controller to perform efficiently. VMware Distributed Power Management [13] operates based on the same idea with the lower and upper utilization thresholds set to 45% and 81% respectively. However, as it was justified before, fixed values of the thresholds are not suitable for systems with dynamic and unpredictable workloads. In our current work, we propose an approach to set the threshold values dynamically, depending on a current set of instantiated VMs and historical data of the resource usage by each VM.

3. SYSTEM MODEL

In this paper, the target system is an IaaS environment, represented by a large-scale data center consisting of N heterogeneous physical nodes. Each node i is characterized by the CPU performance defined in Millions Instructions Per Second (MIPS), amount of RAM, network bandwidth and disk storage. The type of the environment implies no knowledge of application workloads and time for which VMs are provisioned. Users submit requests for provisioning of M heterogeneous VMs characterized by requirements to CPU performance, RAM, network bandwidth and disk storage.

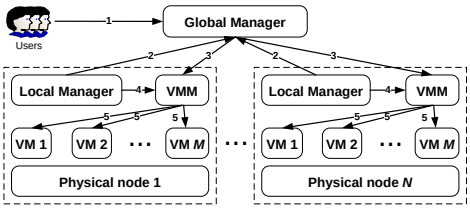


Figure 2: The system model

The software layer of the system is tiered comprising local and global managers (Fig. 2). The local managers reside on each node as a module of the VM monitor (VMM). Their objective is continuous monitoring of a node's CPU utilization, resizing the VMs according to their resource needs, and deciding when and which VMs have to be migrated from the host node (4). The global manager resides on a master node and collects information from the local managers to maintain the overall view of the utilization of resources (2). The global manager issues commands for the optimization of the VM placement (3). VMMs perform actual resizing and migration of VMs as well as changes in power states of nodes (5). Most of the idle nodes are kept switched off, whereas some temporary nodes are kept in sleep / hibernate mode (with less transition time) to allow the system to rapidly respond to load peaks.

3.1 Power Model

Recent studies [6, 8] show that the power consumption by servers can be accurately described by a linear relationship between the power consumption and CPU utilization, even when Dynamic Voltage and Frequency Scaling (DVFS) is applied. The reason lies in the limited number of states that can be set to the frequency and voltage of CPU and the fact that voltage and performance scaling are not applied to other system components, such as memory and network interface. Moreover, these studies show that on average an idle server consumes approximately 70% of the power consumed when it is fully utilized. Therefore, for our experimental studies we define the power consumption as a function of the CPU utilization ($P(u)$) as shown in (1).

$$P(u) = k \cdot P_{max} + (1-k) \cdot P_{max} \cdot u = P_{max} \cdot (0.7 + 0.3 \cdot u), \quad (1)$$

where P_{max} is set to 250 W, which is a usual value for modern computing servers; k is the fraction of power consumed by an idle server; and u is the CPU utilization. As the utilization of CPU may change over time due to the workload variability, it is a function of the time: $u(t)$. Therefore, to define the total energy consumption by a server we use the model defined in (2).

$$E = \int_t P(u(t))\, dt \quad (2)$$

According to this model, the energy consumption by a server is determined by the CPU utilization. Therefore, to reduce the energy consumption, our approach is to improve the CPU utilization of physical nodes in a data center.

3.2 Cost of VM Live Migration

Live migration of VMs allows transferring VMs between physical nodes without suspension and with a short downtime. However, live migration has a negative impact on the performance of applications running in a VM during a migration. Voorsluys et al. have performed an experimental

study to investigate the value of this impact and find a way to model it [14]. They have found that performance degradation and downtime depend on the application behavior, i.e. how many memory pages the application updates during its execution. However, for the class of web-applications the average performance degradation including the downtime can be estimated as approximately 10% of the CPU utilization. This means that each migration may cause some SLA violation; therefore, it is crucial to minimize the number of VM migrations. The length of a live migration depends on the total amount of memory used by the VM and available network bandwidth. Therefore, for our experiments we define the performance degradation experienced by VM j as in (3).

$$U_{d_j} = 0.1 \cdot \int_{t_0}^{t_0+T_{m_j}} u_j(t)\,dt, \qquad T_{m_j} = \frac{M_j}{B_j}, \tag{3}$$

where U_{d_j} is the total performance degradation by VM j, t_0 is the time when the migration starts, T_{m_j} is the time taken to complete the migration, $u_j(t)$ is the CPU utilization by VM j, M_j is the amount of memory used by VM j, and B_j is the available network bandwidth.

3.3 SLA Violation Metric

Meeting QoS requirements is extremely important for Cloud computing environments. QoS requirements are commonly formalized in the form of SLA, which can be determined in terms of such characteristics as minimum throughput or maximum response time delivered by the deployed system. As these characteristics can vary for different applications, it is necessary to define a generic metric that can be used in our simulation experiments to estimate the level to which the SLA are delivered by the infrastructure. By delivering the SLA we mean providing the performance requested by applications inside a VM at any time bounded only by the parameters of the VM. For our experiments we define the overall level of SLA violation caused by the system (SLA) as a fraction of the difference between the requested MIPS by all the VMs ($U_{r_j}(t)$) and the actually allocated MIPS ($U_{a_j}(t)$) relatively to the total requested MIPS over the lifetime of the VMs (4), where M is the number of VMs.

$$SLA = \frac{\sum_{j=1}^{M} \int_t U_{r_j}(t) - U_{a_j}(t)\,dt}{\sum_{j=1}^{M} \int_t U_{r_j}(t)\,dt} \tag{4}$$

This metric represents the percentage of the CPU performance that has not been allocated when demanded by applications relatively to the total demand.

4. ALLOCATION POLICIES

The system operation can be divided in two parts: (1) selection of VMs that have to be migrated to optimize the allocation; and (2) placement of the VMs selected for migration and new VMs requested by the users on physical nodes. We discuss these parts in the following sections.

4.1 VM Selection

4.1.1 Fixed Utilization Thresholds

In our previous work we have proposed four heuristics for choosing VMs to migrate [2]. The first heuristic, Single Threshold (ST), is based on the idea of setting an upper utilization threshold for hosts and placing VMs while keeping the total utilization of the CPU below this threshold. The aim is to preserve free resources to prevent SLA violation due to consolidation in cases when the resource demand by VMs increases. At each time frame all the VMs are reallocated using the Modified Best Fit Decreasing (MBFD) algorithm (Section 4.2) with an additional condition of keeping the upper utilization threshold not violated. New placement is achieved by live migration of VMs.

The other three heuristics are based on the idea of setting upper and lower utilization thresholds for hosts and keeping the total utilization of the CPU by all the VMs between these thresholds. If the CPU utilization of a host falls below the lower threshold, all VMs have to be migrated from this host and the host has to be switched off in order to eliminate the idle power consumption. If the utilization exceeds the upper threshold, some VMs have to be migrated from the host to reduce the utilization in order to prevent potential SLA violation. We have proposed three policies for choosing VMs that have to be migrated from an over-utilized host.

1. **Minimization of Migrations (MM)** – migrate the least number of VMs to minimize migration overhead
2. **Highest Potential Growth (HPG)** – migrate VMs that have the lowest usage of CPU relatively to requested in order to minimize total potential increase of the utilization and SLA violation
3. **Random Choice (RC)** – choose the necessary number of VMs randomly

4.1.2 Dynamic Utilization Thresholds

As mentioned earlier, fixed values for the thresholds are unsuitable for an environment with dynamic and unpredictable workloads, in which different types of applications can share a physical resource. The system should be able to automatically adjust its behavior depending on the workload patterns exhibited by the applications. Therefore, we propose **a novel technique** for auto-adjustment of the utilization thresholds based on a statistical analysis of the historical data collected during the lifetime of VMs.

First of all, we assume that the CPU utilization created by each VM can be described by a random variable (u_j) with a particular distribution, which persists at least over some recent period of time. In this case, the CPU utilization of a host can be represented by a random variable (U_i), which is a sum of utilizations by m VMs allocated to this host. We assume that as the distributions created by different VMs are different, the distribution of the host's utilization is approximately normal and can be modeled by the t-distribution. We cannot predict the CPU utilization of a physical node in the future; however, we can calculate characteristics of the distribution over some recent period of time, such as the sample mean ($\overline{U_i}$) and standard deviation (s_{U_i}) as in (5).

$$\overline{U_i} = \sum_{j=1}^{m} \overline{u_j}, \qquad s_{U_i} = \sqrt{\sum_{j=1}^{m} s_{u_j}^2} \tag{5}$$

The advantage of collecting the data for each VM separately and then using the summation is that a VM is migrated together with the data of its resource usage and the data will be actual even after a VM migration. Using this information and the inverse cumulative probability function for the t-distribution ($t_{inv_n}(P)$), it is possible to find out an interval of the CPU utilization, which will be reached with

a low probability (e.g. 5%). We can set the upper utilization threshold (T_{u_i}) for each host i preserving this amount of spare CPU capacity defined by the lower (P_{ul}) and upper (P_{uu}) limits of the probability interval as shown in (6), where n is the number of data points collected, and $n-1$ represents the degrees of freedom for the t-distribution.

$$T_{u_i} = 1 - ((t_{inv_{n-1}}(P_{uu}) \cdot s_{U_i} + \overline{U_i}) - (t_{inv_{n-1}}(P_{ul}) \cdot s_{U_i} + \overline{U_i})) \quad (6)$$

The lower threshold is calculated in a similar way; however, the difference is that a single value is obtained for all the hosts in the system. The idea is to determine the hosts that have lower utilizations relatively to the average value across all the nodes. To tackle the case when all the hosts have low CPU utilizations, we introduce a limit (U_l) to cap the decrease of the lower utilization threshold. In our previous work [2] we have found the value $U_l = 30\%$ to be effective for the lower threshold. We calculate the lower threshold (T_l) as shown in (7).

$$\overline{U} = \frac{1}{N}\sum_{i=1}^{N}\overline{U_i}, \qquad s_U = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(\overline{U_i} - \overline{U})^2},$$
$$T_l = \begin{cases} \overline{U} - t_{inv_{n-1}}(P_l) \cdot s_U, \text{ if } < U_l, \\ U_l, \text{ otherwise.} \end{cases} \quad (7)$$

The reallocation algorithm using the dynamic thresholds (DT) is presented in Alg. 1. For the DT algorithm we apply the MM policy for VM selection, as in our previous work it has shown the superiority over the alternatives [2]. The complexity of the algorithm is proportional to the sum of the number of non over-utilized host plus the product of the number of over-utilized hosts and the number of VMs allocated to these over-utilized hosts.

Algorithm 1: Dynamic Thresholds (DT)

```
1  Input: hostList, vmList   Output: migrationList
2  vmList.sortDecreasingUtilization()
3  foreach h in hostList do
4  |  hUtil ← h.util()
5  |  bestFitUtil ← MAX
6  |  while hUtil > h.upThresh() do
7  |  |  foreach vm in vmList do
8  |  |  |  if vm.util() > hUtil − h.upThresh() then
9  |  |  |  |  t ← vm.util() − hUtil + h.upThresh()
10 |  |  |  |  if t < bestFitUtil then
11 |  |  |  |  |  bestFitUtil ← t
12 |  |  |  |  |  bestFitVm ← vm
13 |  |  |  else
14 |  |  |  |  if bestFitUtil = MAX then
15 |  |  |  |  |  bestFitVm ← vm
16 |  |  |  |  break
17 |  |  hUtil ← hUtil − bestFitVm.util()
18 |  |  migrationList.add(bestFitVm)
19 |  |  vmList.remove(vm)
20 |  if hUtil < lowThresh() then
21 |  |  migrationList.add(h.getVmList())
22 |  |  vmList.remove(h.getVmList())
23 return migrationList
```

4.2 VM Placement

The VM placement can be seen as a bin packing problem with variable bin sizes and prices, where bins represent the physical nodes; items are the VMs that have to be allocated; bin sizes are the available CPU capacities of the nodes; and prices correspond to the power consumption by the nodes. As the bin packing problem is NP-hard, to solve it we apply a modification of the Best Fit Decreasing (BFD) algorithm that is shown to use no more than $11/9 \cdot OPT + 1$ bins (where OPT is the number of bins provided by the optimal solution) [15]. In our modification (MBFD) we sort all the VMs in the decreasing order of current CPU utilizations and allocate each VM to a host that provides the least increase of the power consumption caused by the allocation. This allows the leveraging the nodes heterogeneity by choosing the most power-efficient ones first. The pseudo-code for the algorithm is presented in Alg. 2. The complexity of the algorithm is $n \cdot m$, where n is the number of nodes and m is the number of VMs that have to be allocated.

Algorithm 2: Modified Best Fit Decreasing (MBFD)

```
1  Input: hostList, vmList   Output: allocation of VMs
2  vmList.sortDecreasingUtilization()
3  foreach vm in vmList do
4  |  minPower ← MAX
5  |  allocatedHost ← NULL
6  |  foreach host in hostList do
7  |  |  if host has enough resource for vm then
8  |  |  |  power ← estimatePower(host, vm)
9  |  |  |  if power < minPower then
10 |  |  |  |  allocatedHost ← host
11 |  |  |  |  minPower ← power
12 |  if allocatedHost ≠ NULL then
13 |  |  allocate vm to allocatedHost
14 return allocation
```

5. EVALUATION

As the target system is a generic Cloud computing environment, it is essential to evaluate it on a large-scale virtualized data center infrastructure. However, it is difficult to conduct large-scale experiments on a real infrastructure, especially when it is necessary to reproduce the experiment with the same conditions to compare different algorithms. Therefore, a simulation has been chosen as a way to evaluate the proposed algorithms.

The CloudSim toolkit 2.0 [5] has been chosen as a simulation platform, as it is a modern simulation framework aimed at Cloud computing environments. In contrast to alternative simulation toolkits (e.g. SimGrid, GangSim), it allows the modeling of virtualized environments, supporting on-demand resource provisioning, and their management. It has been extended in order to enable power-aware simulations and dynamic workloads, as the core framework does not provide these capabilities. The implemented extensions have been included in the 2.0 version of the CloudSim toolkit.

5.1 Workload Data

To make a simulation-based evaluation applicable, it is important to conduct experiments using workload traces from a real system. For our experiments we have used data provided as a part of the CoMon project, a monitoring infrastructure for PlanetLab (http://comon.cs.princeton.edu). We have used the data of the CPU utilization by more than a thousand servers located at more than 500 places around the world. The data have been collected each five minutes during the period from the 10^{th} to 19^{th} of May 2010. Ad-

Table 1: DT-99-98-99.9 against DT-90-90-95

Parameters	Energy	SLA viol.	VM migr.
99-98-99.9	1204 kWh	**0.96%**	**20577**
90-90-95	**1154 kWh**	1.47%	37758
Difference	50 kWh	-0.51%	-17180

ditional values for each second have been generated using linear interpolation. The data confirm the statement made in the beginning: the average CPU utilization is below 50%. The mean CPU utilization is 36.44% with 95% Confidence Interval (CI): (36.40%, 36.47%).

One of the requirements to the system is the independence of the workload type, which means that the system should operate effectively under different and mixed workloads. Therefore, we have split the data into 10 categories by the mean value of the CPU utilization each containing single-day workload data for 500 randomly chosen servers.

5.2 Experimental Results

We have simulated a data center that comprises 1500 heterogeneous physical nodes. Each node is modeled to have one CPU core with performance equivalent to 2000, 2500, 3000 or 3500 Million Instructions Per Second (MIPS), 16 GB of RAM, 10 GB/s network bandwidth and 1 TB of storage. Power consumption by the hosts is defined by the model described in Section 3.1. According to this model, a host consumes from 175 W with 0% CPU utilization up to 250 W with 100% CPU utilization. Each VM requires one CPU core with maximum of 1000, 2000, 2500 or 3250 MIPS, 1 GB of RAM, 100 Mb/s network bandwidth and 1 GB of storage. However, during the lifetime VMs may use less resources creating the opportunity for dynamic consolidation. The CPU MIPS ratings are equivalent to Amazon EC2 instance types. 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 described in Section 5.1. Initially, VMs are allocated according to their parameters assuming 100% utilization.

The simulations have been run on 10 hours of each workload category to determine the algorithm that delivers the best energy consumption, SLA violation and number of VM migrations over different workload types. The optimization algorithms have been run each minute of the simulation time. We have run the experiments varying three parameters of the dynamic thresholds: P_l – the probability for the lower threshold from 90% to 99%; P_{ul} – the lower bound of the probability interval for the upper threshold from 90% to 98%; and P_{uu} – the upper bound of the probability interval for the upper threshold from 95% to 99.9%.

First, we need to determine the best parameters for the DT algorithm and then compare it with the alternatives. The results for the energy consumption and SLA violation have not passed the normality test with the P-value < 0.01. Therefore, we have used the non-parametric Friedman test. For both the energy consumption and SLA violation the P-value < 0.001, meaning that there is a statistically significant difference between the results produced by the DT algorithm with different parameters. The distributions of the differences are approximately normal; therefore, we have used the paired T-test to compare parameters that provides the least energy consumption (DT-90-90-95) and the least SLA violation and number of VM migration (DT-99-98-99.9). The

Table 2: DT-99-98-99.9 against the other algorithms

Algorithm	Energy	SLA viol.	VM migr.
DT-99-98-99.9	1204 kWh	**0.96%**	**20,577**
MM-20-60	1218 kWh	3.15%	100,339
MM-30-70	1158 kWh	3.73%	101,661
MM-45-81	1046 kWh	5.88%	125,199
ST-50	1180 kWh	3.60%	270,766
ST-60	**866 kWh**	5.8%	319,991
DVFS	1847 kWh	–	–
NPA	8975 kWh	–	–

P-values for the energy consumption, SLA violation and VM migrations are 0.047, 0.005 and 0.001 respectively, which means that there is a statistically significant difference for all the characteristics. The detailed results are presented in Table 1. We have chosen DT-99-98-99.9 as it provides significantly lower SLA violation level and fewer VM migrations with the comparable with DT-90-90-95 energy consumption.

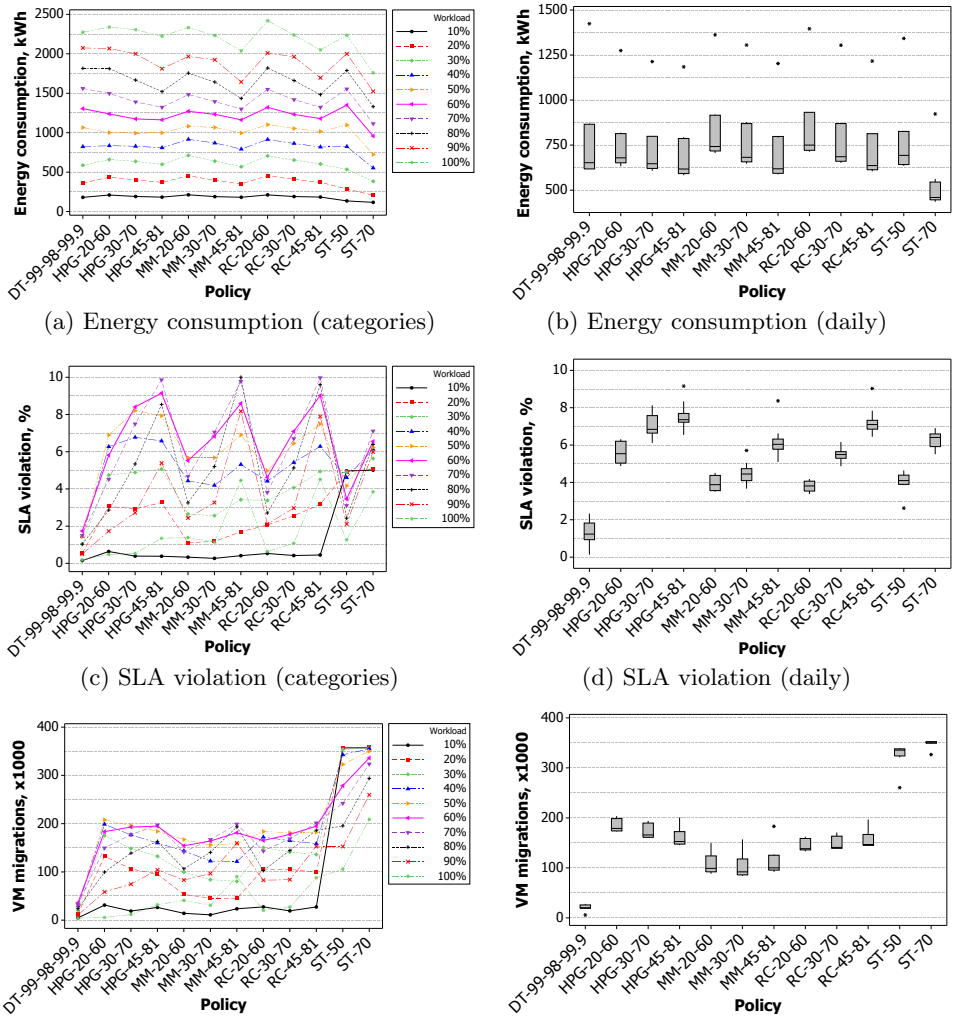
The energy consumption, SLA violation and migrations data for DT-99-98-99.9 and the other algorithms, including DVFS and the Non-Power Aware policy (NPA), meet the assumptions for the ANOVA test: normal distribution of the standardized residuals and equal variances. The P-values for all three comparisons using the two-way ANOVA test are < 0.001, which means that there is a statistically significant difference between the results produced by the algorithms. The results are presented in Table 2 and Fig. 3 (a, c, e).

According to the results, DT-99-98-99.9 provides a similar level of the energy consumption with substantially reduced SLA violation and VM migrations in comparison to other migration-aware algorithms, including the fixed threshold values introduced by VMware (45%, 81%). Over the 10 categories of the workload DT-99-98-99.9 ensures less than 1% of the SLA violation. We have also run the experiments using the daily data and mixed workloads. The results for all the migration-aware algorithms averaged over 10 days are presented in Fig. 3 (b, d, f). These results resemble the superiority of the DT-99-98-99.9 algorithm over the alternatives in terms of the SLA violation level and number VM migrations with approximately the same energy consumption.

6. CONCLUSION AND FUTURE WORK

To maximize their ROI Cloud providers have to apply energy-efficient resource management strategies, such as dynamic consolidation of VMs and switching idle servers off. However, such consolidation is not trivial, as it can result in violation of the SLA negotiated with customers. In this paper we have proposed a novel technique for the energy-efficient threshold-based dynamic consolidation of VMs with auto-adjustment of the threshold values. We have evaluated the proposed algorithms through extensive simulations on a large-scale experimental setup using workload traces from more than a thousand PlanetLab nodes. The experimental results show that the proposed technique outperforms other migration-aware policies in terms of the level of SLA violation ($< 1\%$) and number of VM migrations, while providing a similar level of energy consumption. Moreover, the behavior of the proposed DT algorithm can be adjusted by changing its parameters: SLA can be relaxed leading to further reduction in energy consumption.

For the future work, we propose to investigate the focusing on multi-core CPU architectures, as well as consideration of multiple system resources, such as memory and network



(a) Energy consumption (categories)

(b) Energy consumption (daily)

(c) SLA violation (categories)

(d) SLA violation (daily)

(e) VM migrations (categories)

(f) VM migrations (daily)

Figure 3: Experimental results

interface, as these resources also significantly contribute to the overall energy consumption. In order to evaluate the proposed system in a real Cloud infrastructure, we plan to implement it by extending a real-world Cloud platform, such as Aneka [11]. Besides the reduction in infrastructure and on-going operating costs, this work also has social significance as it decreases carbon dioxide footprints and energy consumption by modern IT infrastructures.

7. REFERENCES

[1] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, pages 33–37, 2007.

[2] A. Beloglazov and R. Buyya. Energy effcient allocation of virtual machines in cloud data centers. In *Proc. of the 10th IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Computing (CCGrid 2010)*, 2010.

[3] R. Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431. *Lawrence Berkeley National Laboratory*, 2008.

[4] R. Buyya et al. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *Proc. of the 10th IEEE Intl. Conf. on High Performance Computing and Communications (HPCC'08)*, 2008.

[5] R. N. Calheiros et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience, Wiley Press, NY, USA*, 2010.

[6] X. Fan et al. Power provisioning for a warehouse-sized computer. In *Proc. of the 34th Annual Intl. Symp. on Computer Architecture*, pages 13–23, 2007.

[7] D. Gmach et al. Resource pool management: Reactive versus proactive or letŠs be friends. *Computer Networks*, 2009.

[8] D. Kusic et al. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15, 2009.

[9] R. Nathuji and K. Schwan. Virtualpower: Coordinated power management in virtualized enterprise systems. *ACM SIGOPS Operating Systems Review*, 41(6):265–278, 2007.

[10] Y. Song et al. Multi-Tiered On-Demand resource scheduling for VM-Based data center. In *Proc. of the 2009 9th IEEE/ACM Intl. Symp. on Cluster Computing*, pages 148–155, 2009.

[11] C. Vecchiola et al. Aneka: a software platform for .NET-based cloud computing. *High Performance & Large Scale Comp., Advances in Parallel Computing*, pages 267–295, 2009.

[12] A. Verma et al. pMapper: power and migration cost aware application placement in virtualized systems. In *Proc. of the 9th ACM/IFIP/USENIX Intl. Conf. on Middleware*, pages 243–264, 2008.

[13] VMware Inc. VMware distributed power management concepts and use, 2010.

[14] W. Voorsluys et al. Cost of virtual machine live migration in clouds: A performance evaluation. In *Proc. of the 1st Intl. Conf. on Cloud Computing*, pages 254–265, 2009.

[15] M. Yue. A simple proof of the inequality FFD (L)< 11/9 OPT (L)+ 1,for all l for the FFD bin-packing algorithm. *Acta Mathematicae Applicatae Sinica*, 7(4):321–331, 1991.