Dynamic Virtual Machine Consolidation for Energy Efficient Cloud Data Centers

Dong-Ki Kang, Fawaz Alhazemi, Seong-Hwan Kim, and Chan-Hyun Youn(✉)

School of Electrical Engineering, KAIST, Daejeon, Korea
{dkkang, fawaz, s.h_kim, chyoun}@kaist.ac.kr

Abstract. As a cloud computing model have led clusters to the large-scale data centers, reducing of the energy consumption which imposes a crucial part of the whole operating expense for data centers has received a lot of attention of a wide public. At cluster-level viewpoint, the most popular method for energy efficient cloud is Dynamic Right Sizing (DRS), which turns off idle servers those do not have any of running virtual resources. To maximize the energy efficiency through DRS, one of primary adaptive resource management strategies is a Virtual Machine (VM) consolidation which integrates VM instances into as few servers as possible. In this paper, we propose Virtual machine Consolidation based Size Decision (VC-SD) approach migrates VM instances from under-utilized servers which are supposed to be turned off to sustaining ones according to their monitored resource utilizations in real time. In addition, we design a Self Adjusting Workload Prediction (SAWP) method to improve a forecasting accuracy of resource utilization even under irregular demand patterns. Through experimental results based on real cloud servers, we show various metrics such as resource utilization, energy consumption and switching overhead caused by application processing, VM migration and DRS execution to verify a necessity of our proposed methodologies.

Keywords: Cloud computing · Virtual Machine migration · Dynamic Right Sizing · Workload Prediction

1 Introduction

In modern cloud data centers, resource allocation with high energy efficiency has been a key problem as a fraction of cost caused by energy consumption has increased in recent years. According to an estimation from [1], a cost of power and cooling has increased 400 % over 10 years, and 59 % of data centers identify them as key factors limiting server deployments. Consequentially, challenges about energy consumption and cooling have led to growing push to achieve a design of energy efficient data centers. At the data center level, a promising candidate called Dynamic Right Sizing (DRS) to save energy is to dynamically adjust the number of active servers (i.e., servers those power is switched on) in proportion to the measured user demands [2]. In DRS, the energy saving can be achieved through allowing idle servers that do not have any running VM instances to go low-power mode (i.e., sleep, shut down). In order to maximize the energy efficiency via

Y. Zhang et al. (Eds.): CloudComp 2015, LNICST 167, pp. 70–80, 2016.
DOI: 10.1007/978-3-319-38904-2_8

DRS, one of primary adaptive resource management strategies is a VM consolidation in which, running VM instances can be dynamically integrated into the minimal number of cloud servers in accordance with their resource utilization collected by hypervisor monitoring module [3]. That is, running VM instances on under-utilized servers which are supposed to be turned off could be migrated to power-sustainable servers. However, it is difficult to efficiently manage cloud resources since cloud users often have heterogeneous resource demands underlying multiple service applications that experience highly variable workloads. Therefore, the inconsiderate VM consolidation using live migration might lead to undesirable performance degradation due primarily to switching overheads caused by migration and turning off servers on [4]. Running service applications with reckless VM migration and DRS execution might encounter serious execution time's delay, increased latency or failures. As a result, a careful resource management scheme considering switching overheads is necessary to reduce efficiently the energy consumption of cloud servers while ensuring acceptable Quality of Service (QoS) based on Service Level Agreements to cloud users.

In this paper, we propose Virtual machine Consolidation based Sizing Decision (VC-SD) approach to address above challenges for cloud data centers. The energy consumption model based on VC-SD approach is formulated with a performance cost (reputation loss) caused by the increased delay from downsizing active servers and an energy cost of keeping particular active servers, and a cost incurred from switching off servers on. Subsequently, we develop the design of an automated cloud resource management system called Dynamic Cloud Resource Broker (DCRB) with VC-SD approach. Moreover, we introduce our novel prediction method called Self Adjusting Workload Prediction (SAWP) to increase the prediction accuracy of users' future demands even under the unstatic and irregular workload patterns. The proposed SAWP method adaptively scales the history window size up or down according to extracted workload's autocorrelations and sample entropies which measure periodicity and burstiness of workloads [6]. To investigate performance characteristics of the proposed approaches, we conduct various experiments to evaluate a resource utilization, completion time's delay and energy consumption by live migration and DRS execution on real testbed based on Openstack which is a well known cloud platform using KVM hypervisor [5]. Through meaningful experimental results, we find that our proposed VC-SD approach and SAWP method provide significant energy saving while guaranteeing acceptable performance required by users in practice.

2 System Model Formulation

In this chapter, we introduce a system model based on reputation cost and energy cost by DRS with VM consolidation using live migration. Several notations for the system model are described as follows,

n: a total number of running VM instances

m: a total number of physical servers

$\boldsymbol{x}^t = \left(x_1^t, x_2^t, \ldots, x_n^t\right)^T$: a resource allocation vector at period t, x_i^t is an index of physical server assigned VM instance i at period t.

$\boldsymbol{pm}_j^t = \left\{ i | x_i^{t-1} = j, \forall i = 1, \ldots, n \right\}$: a set of running VM instances on physical server j at period t.

$d_{j,i}^{t,k}$: a flavor demand of resource k by VM instance i on physical server j at period t.

c_j^k: a capacity of resource k on physical server j.

C_{total}^t: a total cost at period t.

C_{repu}^t: a reputation cost (i.e., affects the future purchase of users) at period t.

C_{energy}^t: an energy cost for maintaining active servers and processing wake-up of off servers by DRS at period t.

C_{intf}: a constant value. An unit cost caused by interference of running VM instances.

C_{mig}: a constant value. An unit cost caused by live migration.

C_p: a constant value. An unit power cost for active servers.

P_{active}: a constant value. An unit power consumption for active servers.

P_{switch}: a constant value. An unit power consumption for turning off server on.

T_{mig}: an execution time for live migration of VM instances.

T_{switch}: an execution time for turning off servers on.

ω_k: a weight value for resource component k (e.g., cpu, memory, and I/O bandwidth, etc.).

$r_{j,i}^{t,k}$: a resource utilization of VM instance i on resource component k of physical server j at period t.

r_{thr}^k: a threshold value of resource utilization representing over-utilization of resource component k.

Based on above defined notations, cost models based on live VM migration and DRS can be formulated as follows,

$$C_{total}^t = C_{repu}^t + C_{energy}^t \tag{1}$$

$$\begin{aligned} C_{repu}^t = C_{intf} \sum_{j=1}^{m} \sum_{k=1}^{l} \omega_k \left| \boldsymbol{pm}_j^t \right| \cdot \left(\frac{\sum_{i=1}^{n} r_{j,i}^{t,k}}{r_{thr}^k} - 1 \right)^+ \\ + 2 C_{mig} T_{mig} \left| \left\{ x_i^{t-1} | x_i^{t-1} \neq x_i^{t-2}, \forall i = 1, \ldots, n \right\} \right| \end{aligned} \tag{2}$$

$$\begin{aligned} C_{energy}^t = C_p P_{active} \sum_{j=1}^{m} \left(\left| \boldsymbol{pm}_j^t \right| \right)^- \\ + C_p P_{switch} T_{switch} \left(\sum_{j=1}^{m} \left(\left| \boldsymbol{pm}_j^t \right| \right)^- - \sum_{j=1}^{m} \left(\left| \boldsymbol{pm}_j^{t-1} \right| \right)^- \right)^+ \end{aligned} \tag{3}$$

where $(x)^+ = \max(x, 0)$ and $(x)^- = \min(x, 1)$. The first term in the right hand of Eq. (2) represents that as the number of concurrent running VM instances on single physical server is increased, the number of users experiencing undesirable performance degradation is also increased. We multiply by the cost of live migration by 2 as shown in the second term in (2) since two physical servers obviously are required for migration.

The purpose of our algorithm is to derive an optimal consolidation plan $\boldsymbol{x}$ for resource allocation at next period iteratively. At period $t-1$, the proposed VC-SD approach aims to minimize the cost function (1) by finding a solution $\boldsymbol{x}^{t-1}$ as follows,

$$minimize_{\boldsymbol{x}^{t-1}} \qquad C^t_{total} = C^t_{repu} + C^t_{energy} \tag{4}$$

$$subject\ to. \qquad \sum_{i=1}^{n} d^{t,k}_{j,i} \leq c^k_j, \quad \forall j = 1, 2, \ldots, m \tag{5}$$

$$state\left(\boldsymbol{pm}^t_{x_i}\right) \neq off, \quad \forall i = 1, 2, \ldots, n \tag{6}$$

where $state(\cdot)$ represents the state of physical server; *on* or *off*. For simplicity, we assume that any additional VM requests are not submitted to our system during VC-SD approach execution. In general cases, this assumption is not trivial, we therefore consider this issue in future works.

To solve the objective cost function (4), we prefer a well known evolutionary metaheuristic called Genetic Algorithm (GA) in order to approximate the optimal plan x^* at each period since Eqs. (1)–(3) have non-linear characteristics. In next chapter, our proposed system with VC-SD approach and DRS method is introduced in detail.

3 Proposed System Structure

In this chapter, we introduce the architecture of an automated Dynamic Cloud Resource Brokering (DCRB) system with our proposed algorithms. In Fig. 1, two main components of DCRB are shown; VC-SD manager and SWAP manager.

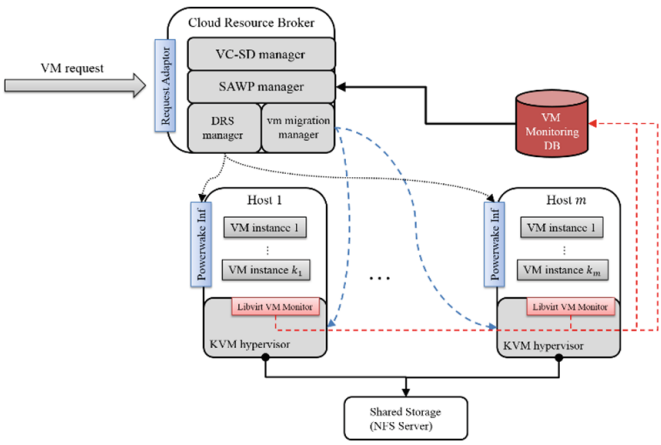


Fig. 1. Cloud Resource Broker with VC-SD manager and SAWP manager for greening the cloud data center

First, the process of VC-SD approach includes following four steps: (1) to monitor and collect resource utilization data of VM instances on each physical server through attached Libvirt based monitoring tools; (2) to go step 3 if the average utilization of whole active servers are significantly low (i.e., below the predefined threshold), otherwise go step 4; (3) to choose active servers which are supposed to be turned off, migrate all the VM instances on them to other servers and trigger DRS execution; (4) to determine the number of off servers to be turned on and send magic packets to them for wake-up if the average utilization of whole active servers are significantly high (i.e., above the predefined threshold), otherwise maintain the current number of active servers.

Figure 2 shows the procedure of the proposed VC-SD approach in DCRB. $\overline{r^k} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} r_{j,i}^{t,k}$ is an average resource utilization of whole active servers and $r^k_{thr^{low}_{green}}$ and $r^k_{thr^{high}_{green}}$ are threshold values of resource utilization to make a decision whether to turn active servers off or to turn off servers on. A term $\alpha \cdot \max_k \frac{\overline{r^k}}{r^k_{thr^{high}_{green}}}$ affects the number of off servers that supposed to be turned on, where α is a predetermined constant value. Second, the process of SAWP method also includes following four steps: (1) to analyze user demand history data based on a Workload Analysis and Classification tool (WAC) [6] and calculate fluctuation and burstiness of demands; (2) to go step 3 if derived values of fluctuation and burstiness are high (i.e., above the predefined threshold), otherwise go step 4; (3) to increase the history window size in proportion to the values of fluctuation and burstiness; (4) to decrease the history window size in proportion to values of fluctuation and burstiness if they are significantly low (i.e., below the predefined threshold), otherwise maintain the current history window size.

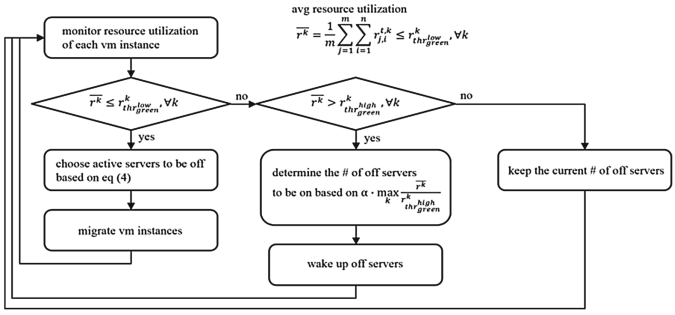


Fig. 2. Procedure of VC-SD approach

Figure 3 shows the procedure of the proposed SAWP method in DCRB. The irregularity function $g(\varepsilon, \delta)$ represents a level of unpredictability of future resource utilization where ε is its fluctuation value (i.e., levels of unstability and aperiodicity) and δ is its burstness value (i.e., levels of sudden surge and decline), both values are calculated based on [6]. According to predetermined threshold values g^{high}_{thr} and g^{low}_{thr} with $g(\varepsilon, \delta)$, the history window size σ is adaptively updated at each prediction process.

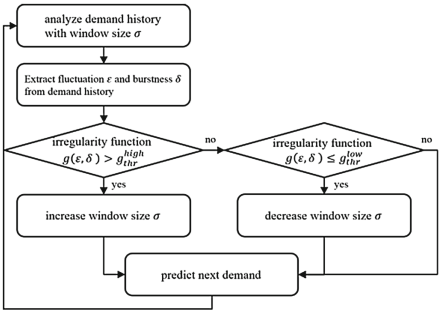


Fig. 3. Procedure of SAWP method

Algorithm 1. Genetic Algorithm (GA) for VC-SD approach

Input : $\boldsymbol{x}^{t-1} = \langle \boldsymbol{x}_1^{t-1}, \boldsymbol{x}_2^{t-1}, \dots, \boldsymbol{x}_m^{t-1} \rangle^T$, *an allocation vector at period* $t-1$
Output : $\boldsymbol{x}^{t} = \langle \boldsymbol{x}_1^{t}, \boldsymbol{x}_2^{t}, \dots, \boldsymbol{x}_m^{t} \rangle^T$, *an allocation vector at period* t

00: initialize $\boldsymbol{pop}^h = \{\boldsymbol{x}^{h,1}, \boldsymbol{x}^{h,2}, \dots, \boldsymbol{x}^{h,P}\}$, $P = size\ of\ population$, and even number
01: **while** $h \le h_{max}$
02: **for each** $\boldsymbol{x}^{h,i} \in \boldsymbol{pop}^h$
03: $fv^{h,i} = f_{C_{total}^t}(\boldsymbol{x}^{h,i}, \boldsymbol{x}^{t-1})$
04: **while** $!(\forall \boldsymbol{x}^{h,i} \in \boldsymbol{pop}^h\ are\ selected\ as\ parents)$
05: $(\boldsymbol{x}^{h,i}, \boldsymbol{x}^{h,j}) \leftarrow$ select parents randomly from $\boldsymbol{pop}^h$
06: $\boldsymbol{offspring}^h = \cup$ crossover $(\boldsymbol{x}^{h,i}, \boldsymbol{x}^{h,j})$
07: **for each** $\boldsymbol{off\_x}^{h,i} \in \boldsymbol{offspring}^h$
08: $off\_fv^{h,i} = f_{C_{total}^t}(\boldsymbol{off\_x}^{h,i}, \boldsymbol{x}^{t-1})$
09: sort $\boldsymbol{pop}^{h\sim} = \left\{\boldsymbol{x}^{h,1}, \dots, \boldsymbol{x}^{h,P}, \boldsymbol{off\_x}^{h,1}, \dots, \boldsymbol{off\_x}^{h,\frac{P}{2}}\right\}$ in ascending order of solutions' corresponding $fv^{h,i}$ *and* $off\_fv^{h,i}$
10: $\boldsymbol{pop}^{h+1} = \{\boldsymbol{pop}_1^{h\sim}, \dots, \boldsymbol{pop}_P^{h\sim}\}$
11: **if** $f_{C_{total}^t}(\boldsymbol{pop}_1^{h+1}, \boldsymbol{x}^{t-1}) \le fv_{thr}$ **then**
12: $\boldsymbol{x}^t = \boldsymbol{pop}_1^{h+1}$ and exit
13: $h++$
14: $\boldsymbol{x}^t = \mathrm{argmin}_{\boldsymbol{pop}_i^{h_{max}} \in \boldsymbol{pop}^{h_{max}}} \left(f_{C_{total}^t}(\boldsymbol{pop}_i^{h_{max}}, \boldsymbol{x}^{t-1})\right)$ s.t. (5), (6)
15: **if** $\boldsymbol{x}^t = \emptyset$ **then**
16: $\boldsymbol{x}^t = \boldsymbol{x}^{t-1}$

A relatively long history window size is not suitable to react to recent changes of workload but is tolerant of varied workload patterns in a short time while a short history window size is favorable to efficiently respond to latest workload patterns but is not good for kaleidoscope of workloads. Consequently, the SAWP method generally outperforms traditional prediction schemes at drastic utilization changes from various cloud applications since it is able to cope with temporary resource utilization

(i.e., not replect overall trends) by adjusting the history window size σ. Now, we introduce a metaheuristic called Genetic Algorithm (GA) for solving approximated optimal solution of Eq. (4) with VC-SD approach. GA is a kind of well known guided random search techniques for combinatorial optimization problems [7]. In GA for VC-SD approach, the value of gene represents an index of physical server as an integer value where the position of gene represents an index of the allocated VM instance. Consequently, each chromosome formed of multiple genes represents a possible solution (not feasible solution) x to the objective cost function (4).

Algorithm 1 describes GA for VC-SD approach in detail. $\boldsymbol{pop}^h$ is a population with size P (even number) at h^{th} generation. h_{max} is a maximum of GA iteration count. At line 03, $f(\cdot)$ is a fitness function of total cost with both of parameters; candidate solution $\boldsymbol{x}^t$ and $\boldsymbol{x}^{t-1}$ based on Eq. (1). At line 04–06, two candidate solutions are iteratively chosen randomly from $\boldsymbol{pop}$ to generate offsprings by crossover until there are no remaining unselected solutions in $\boldsymbol{pop}$. At line 07–08, the fitness function values of each offspring are calculated similar to line 03. At line 09, all the parent solutions in $\boldsymbol{pop}$ and generated offsprings are sorted in ascending order of their corresponding fitness function values. At line 10, the next population including only P solutions that achieve good performance from the union of original $\boldsymbol{pop}$ and derived offsprings is generated to improve a quality of final solution. At line $11 \sim 12$, to reduce a time complexity of GA procedures, when we encounter a first solution which has a fitness function value below the predetermined fitness threshold value fv_{thr}, it counts as a final solution for next period and the algorithm is finished. At line 14, if we cannot find a solution satisfies fv_{thr} until the iteration count reaches h_{max}, then we select a solution which has minimum fitness function value in the population satisfies conditions Eqs. (5) and (6) as a final solution for next period. If there are not any solutions satisfy conditions Eqs. (5) and (6), then we just preserve the current resource allocation vector as shown at line $15 \sim 16$. For population of GA, mutations often applied in order to include characteristics of offsprings that are not inherited trait by parents. We do not consider mutations in our GA in this paper, but it can be used to improve the quality of GA for VC-SD approach in future work.

4 Experimental Results

In our experiments, we have measured various metrics which affect the parameter decision for our proposed algorithms. To do this, we set five cluster servers for cloud platform, one server for DCRB with Mysql DB system, power measuring device of Yocto-Watt [8] and a laptop machine called VirtualHub for collecting and reporting the information of measured power consumption as shown in Fig. 4. The hardware specification of each server for cloud compute host which has Intel i7-3770 (8-cores, 3.4 GHz), 16 GB RAM memory, and two 1 Gbps NICs (Network Interface Cards). In order to measure efficiently the power consumption of cloud cluster server, we use power measuring device model called YWATTMK1 by Yocto-Watt. This model has a measurement unit 0.2 W for AC power with error ratio 3 % and 0.002 W for DC power with error ratio 1.5 %. The VirtualHub collects the information of power consumption from YWATTMK1 through Yocto-Watt Java API and reports them to power

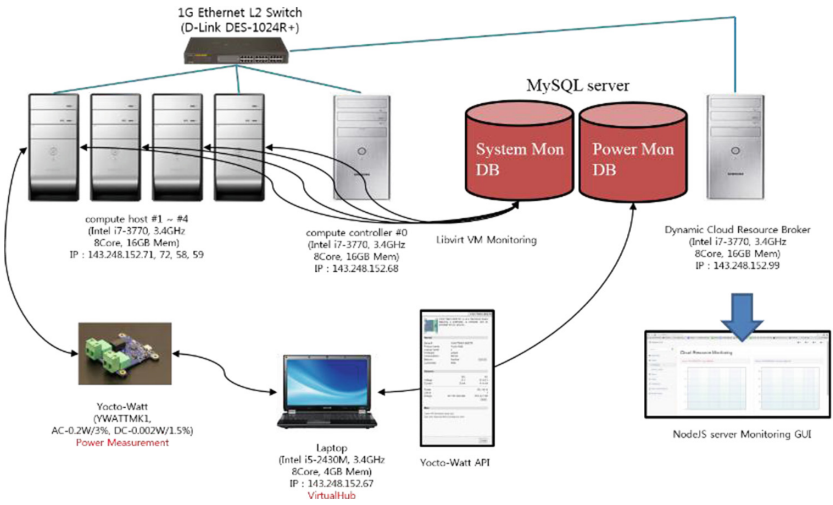


Fig. 4. Experimental environment

monitoring table of Mysql DB system in DCRB periodically. The dynamic resource utilizations by each VM instance are measured via our developed VM monitoring modules based on Libvirt API and are sent to resource monitoring table of Mysql DB system periodically. In addition, SATA 3 TB hard disk called G-drive is deployed as a NFS server in our testbed for live migration [9]. We adopt Openstack kilo version which is a well known open source solution based on KVM Hypervisor as a cloud platform to our testbed. Finally, we use Powerwake package [10] to turn remotely off servers on via Wake on Lan (WOL) technology for DRS execution.

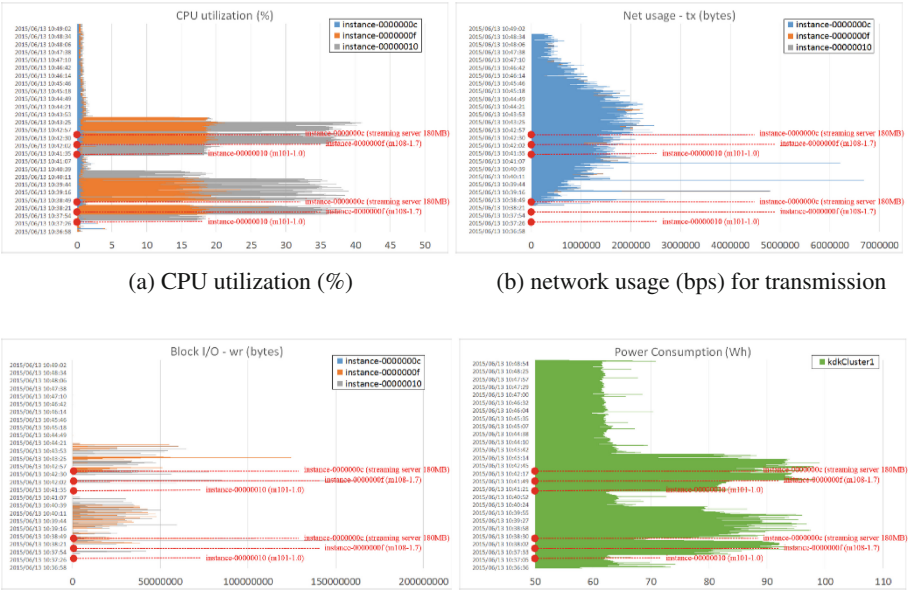
In Table 1, we show the average power consumption and resource utilization of two running applications: Montage m106-1.7 projection and ftp transfer. Montage project is an open source based scientific application and it has been invoked by NASA/IPAC Infrared Science Archive as a toolkit for assembling Flexible Image Transport System (FITS) images into custom mosaics [11]. The m106-1.7 projection in

Table 1. Average power consumption and resource utilization by Montage and ftp transfer

Host state		Power Consumption (Wh)	CPU utilization	Mem utilization	Net bandwidth
Idle		55Wh	1.5 %	3 %	20 Kbps (bytes)
Active	Montage m106-1.7 (projection)	75Wh	15 %	3.7 %	20 Kbps (bytes)
	test.avi downloading (ftp, 1.5GB)	60Wh	2 %	3.7 %	3.7 Mbps (bytes)
aSleep (to Power Off)		70–80Wh (5–7s)	1.8 % (5–7s)	3 % (5–7s)	20 Kbps (bytes)
Power Off (hibernating)		2.5Wh			
aWake (from Power Off)		78Wh (50s–1min)			

Montage is a cpu-intensive application while the ftp transfer is a network-intensive one. Therefore, m106-1.7 causes the power consumption about 75 Wh and the cpu utilization about 15 % whereas the power consumption by ftp transfer for 1.5 GB test.avi file is about 60 Wh and the network bandwidth usage is about 3.7 Mbps. That is, the cpu usage is a main part to affect the power consumption of server. In view of DRS execution, the power consumption by an off server is about 2.5 Wh (note that this value is not zero since the NIC and its some peripheral components are still powered on to keep standby mode to receive the magic packet from Powerwake controller) while aSleep and aWake procedures which cause switching overhead for DRS require the power consumption about 80 Wh to turn active servers off or to turn off servers on, respectively. The aSleep procedure is trivial since it requires a short time (i.e., 5 ∼ 7 s) to complete even though its power consumption is considerable whereas the aWake procedure requires a relatively long execution time (i.e., above 1 min) supposed to be considered carefully. The overhead for aWake procedure would be more serious problem in practice since the execution time by aWake procedure is generally far long (i.e., above 10 min) for multiple servers of rack in the data center. Therefore, it is essential to consider the switching overhead for aWake procedure to reduce efficiently the resource usage cost of the data center.

Figure 5 shows the dynamic resource utilization by several applications on running VM instances measured via Libvirt based VM monitoring modules. The instance-00000010 runs m101-1.0 mProj, instance-0000000f runs m108-1.7 mProj and

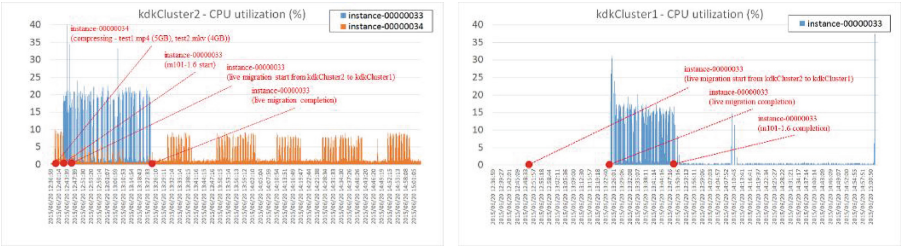


(a) CPU utilization (%) (b) network usage (bps) for transmission

c) block I/O usage (bps) for write operation (d) power consumption (Wh)

Fig. 5. Measured resource utilization and power consumption by Libvirt API based VM monitoring module and YWATTMK1 device

instance-0000000c runs a streaming server with a 180 MB movie file. Especially, Fig. 5(d) demonstrates that the Montage projection which is the cpu-intensive application increases the power consumption significantly of server whereas the streaming server which is a network-intensive application produces consistently very little effect on the power consumption by comparison with results of Table 1. Figure 6 shows cpu utilizations of a source server and a destination server for live migration. The instance-00000033 is supposed to be migrated from kdkCluster2 - source server to kdkCluster1 - destination server and the instance-00000034 is a fixed running VM instance on kdkCluster2. The instance-00000033 is migrated to kdkCluster1 during its application - m101-1.6 - execution. The completion time for live migration of instance-00000033 is longer than half an hour, therefore this overhead might cause a significant performance degradation of application and an undesirable waste of energy. All the results from experiments in this paper obviously verify that our proposed VC-SD approach and SAWP method considering switching overheads by live migration and DRS execution are necessary for real cloud environments.



(a) (b)

Fig. 6. cpu utilizations during VM migration (instance-00000033) from (a) a source server: kdkCluster2 to (b) a destination server:kdkCluster1

5 Conclusion

In this paper, we introduced a Dynamic Cloud Resource Broker (DCRB) with Virtual machine Consolidation based Size Decision (VC-SD) approach for energy efficient resource management by a real time based VM monitoring in cloud data center. Our proposed approach is able to reduce efficiently the energy consumption of servers without a significant performance degradation by live migration and Dynamic Right Sizing (DRS) execution through a considerate model considering switching overheads. Through various experimental results based on Openstack platform justify that our proposed algorithms are supposed to be deployed for prevalent cloud data centers. Moreover, the novel prediction method called Self Adjusting Workload Prediction (SAWP) is proposed in order to improve an accuracy of forecasting future demands even under drastic workload changes. In future works, we demonstrate that our proposed algorithms outperform existing approaches for energy efficient resource management through various experiments based on implemented system in practice.

Acknowledgments. This work was supported by 'Electrically phase-controlled beamforming lighting device based on 2D nano-photonic phased array for lidar' grant from Civil Military Technology Cooperation, Korea and Institute for Information (No. 14-BR-SS-02), and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0101-15-0104, The Development of Supercomputing System for the Genome Analysis).

References

1. International Data Center Corporation. http://www.idc.com
2. Lin, M., Wierman, A., Andrew, L.L.H., Thereska, E.: Dynamic right-sizing for power-proportional data centers. IEEE/ACM Trans. Networking **21**(5), 1378–1391 (2013)
3. Xiao, Z., Song, W., Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1107–1117 (2013)
4. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency Comput. Pract. Experience **24**, 1397–1420 (2012). doi:10.1002/cpe.1867
5. Openstack. http://www.openstack.org
6. A-Eldin, A., Tordsson, J., Elmroth, E., Kihl, M.: Workload Classification for Efficient Auto-Scaling of Cloud Resources. Umea University, Sweden (2013)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
8. YOCTO-WATT. http://www.yoctopuce.com/EN/products/usb-electrical-sensors/yocto-watt
9. G-Technology. http://www.g-technology.com/products/g-drive
10. PowerWake. http://manpages.ubuntu.com/manpages/utopic/man1/powerwake.1.html
11. Montage. http://montage.ipac.caltech.edu/