Mobile Netw Appl
DOI 10.1007/s11036-016-0690-z

Adaptive VM Management with Two Phase Power Consumption Cost Models in Cloud Datacenter

Dong-Ki Kang[1] · **Fawaz Al-Hazemi**[1] · **Seong-Hwan Kim**[1] · **Min Chen**[2] · **Limei Peng**[3] · **Chan-Hyun Youn**[1]

Abstract As cloud computing models have evolved from clusters to large-scale data centers, reducing the energy consumption, which is a large part of the overall operating expense of data centers, has received much attention lately. From a cluster-level viewpoint, the most popular method for an energy efficient cloud is Dynamic Right Sizing (DRS), which turns off idle servers that do not have any virtual resources running. To maximize the energy efficiency with DRS, one of the primary adaptive resource management strategies is a Virtual Machine (VM) migration which consolidates VM instances into as few servers as possible. In this paper, we propose a Two Phase based Adaptive Resource Management (TP-ARM) scheme that migrates VM instances from under-utilized servers that are supposed to be turned off to sustainable ones based on their monitored resource utilizations in real time. In addition, we designed a Self-Adjusting Workload Prediction (SAWP) method to improve the forecasting accuracy of resource utilization even under irregular demand patterns. From the experimental results using real cloud servers, we show that our proposed schemes provide the superior performance of energy consumption, resource utilization and job completion time over existing resource allocation schemes.

✉ Chan-Hyun Youn
chyoun@kaist.ac.kr

Dong-Ki Kang
dkkang@kaist.ac.kr

Fawaz Al-Hazemi
fawaz@kaist.ac.kr

Seong-Hwan Kim
s.h_kim@kaist.ac.kr

Min Chen
Minchen2012@hust.edu.cn

Limei Peng
auroraplm@ajou.ac.kr

[1] School of Electrical Engineering, KAIST, Daejeon, South Korea

[2] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

[3] Department of Industrial Engineering, Ajou University, Suwon, South Korea

Keywords Cloud computing · Virtual machine migration · Dynamic right sizing · Energy saving

1 Introduction

In modern cloud data centers, resource allocation with high energy efficiency has been a key problem because the operating cost from energy consumption has increased in recent years. According to an estimation from [1], the 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 from energy consumption and cooling have led to a growing push to achieve an energy efficient design for data centers. At the data center level, a promising technology called Dynamic Right Sizing (DRS) to save energy dynamically adjusts the number of active servers (i.e., servers whose power is switched on) in proportion to the measured user demands [2]. In DRS, energy saving can be achieved by enabling idle servers that do not have any running VM instances to go into low-power mode (i.e., sleep or shut down). In order to maximize the energy efficiency via DRS, one of the primary adaptive resource management strategies is VM consolidation in which running VM instances can be dynamically integrated into the minimal

Published online: 31 March 2016

number of cloud servers based on their resource utilization collected by a 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 because cloud users often have heterogeneous resource demands underlying multiple service applications which experience highly variable workloads. Therefore, inconsiderate VM consolidation using live migration might lead to undesirable performance degradation due primarily to switching overheads caused by the migration and by turning off servers on [4]. Running service applications with reckless VM migration and DRS execution could encounter serious execution time delays, increased latency or failures [5]. As a result, a careful resource management scheme that considers 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 [6, 7].

In this paper, we propose a Two Phase based Adaptive Resource Management (TP-ARM) scheme to address the above challenges for cloud datacenters. The energy consumption model based on TP-ARM was formulated with a performance cost (reputation loss) caused by an increased delay from downsizing active servers, by an energy costfrom keeping particular servers active, and by a cost incurred from switching off servers on. Subsequently, we designed an automated cloud resource management system called the Adaptive Cloud Resource Broker (ACRB) system with the TP-ARM scheme. Moreover, we introduced our novel prediction method called Self-Adjusting Workload Prediction (SAWP) to increase the prediction accuracy of users' future demands even under unstatic and irregular workload patterns. The proposed SAWP method adaptively scales the history window size up or down according to the extracted workload's autocorrelations and sample entropies which measure the periodicity and burstiness of the workloads [8]. To investigate the performance characteristics of the proposed approaches, we conducted various experiments to evaluate energy consumption, resource utilization and completion time delay by live migration and DRS execution on a real testbed based on Openstack which is a well-known cloud platform using KVM hypervisor [9]. Through meaningful experimental results, we found that our proposed TP-ARM scheme and SAWP method provide significant energy savings while guaranteeing acceptable performance required by users in practice.

2 Proposed adaptive cloud resource brokering system

In this section, we discuss the architecture of the automated Adaptive Cloud Resource Brokering (ACRB) system. Our considered cloud environment including ACRB, which supports deploying a resource management scheme in order to migrate VM requests and adjusts the number of active servers according to the workload level, is depicted in Fig. 1. There are m physical hosts and n VM requests in the cloud datacenter. In the cloud datacenter, the information on resource utilization is collected through KVM hypervisor based monitoring modules into the Resource Monitoring DB module, and reported to the ACRB which is responsible for solving the migration of allocated VM requests and sizing the datacenter. The ACRB has two modules: the TP-ARM module and the SAWP Manager. The TP-ARM module includes the VM Migration Manager and the DRS Manager. In the first phase, the VM Migration Manager is responsible for selecting the appropriate VM requests to be migrated based on the measured resource utilization level. We describe the metrics for determining the VM request migration in detail in section 4. In the second phase, the DRS Manager is responsible for finding the optimal number of active servers in the cloud datacenter. The DRS plan derived by the DRS Manager based on the amount of submitted VM requests and the measured resource utilization from each VM request is delivered to the Datacenter Manager, and the determined percentage of idle servers are powered off. The SAWP Manager is responsible for adjusting the window size of historical data adaptively in order to predict the future demand of VM requests. The SAWP Manager is able to achieve the exact prediction of future demands even under varied workload levels by considering the periodicity and the fluctuation of the historical data. The owner of the cloud datacenter has to minimize the costs for resource operations while boosting the benefits which can increase based on the good reputation of the observed QoS of the cloud services.

In this paper, our ACRB tries to find a resource management solution to minimize the total cost of resource operations including two sub cost models: the energy consumption cost and the performance reputation cost. From the perspective of the energy consumption cost, the VM Migration Manager tries to maximize the resource utilization of each physical host by consolidating running VM requests based on the whole offered load measured through the Libvirt VM Monitor module attached to each host. The Libvirt VM Monitor module gauges the utilization of resources such as CPU, memory, and I/O bandwidth in order to check whether whole hosts are overloaded [10, 11]. VM requests on overloaded hosts are preferably migrated to other hosts which have enough extra resource capacity to accommodate newly allocated VM requests. The DRS Manager sends the shutdown messages to servers which have to be powered off, while it sends magic packets to the ones which have to be powered on again. Obviously, the transition of servers from sleeping mode to active mode (aWake transition) requires additional energy consumption (note that the transition overhead from active to sleep (aSleep transition)

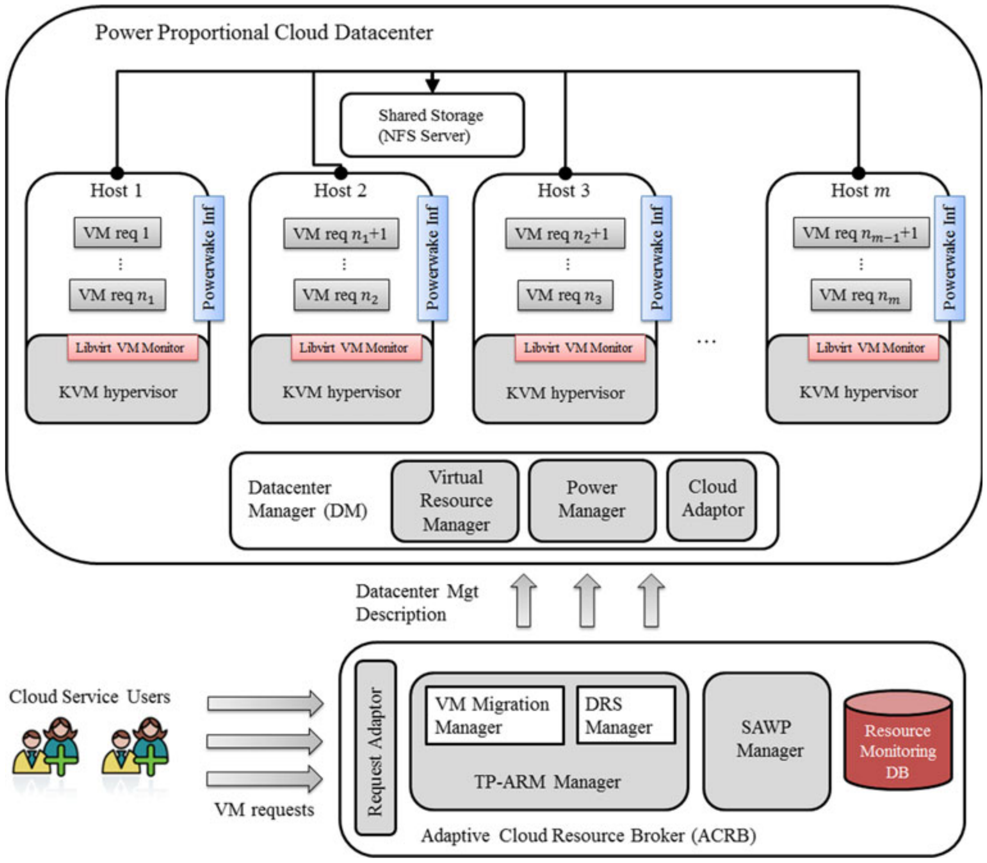


Fig. 1 Adaptive Cloud Resource Brokering (ACRB) system including TP-ARM module and SAWP manager for green cloud datacenters

can be negligible because it requires only a short time to be carried out compared to the aWake transition). Therefore, it is clear that frequent aWake transitions have to be discouraged in order to minimize unnecessary energy consumption. To simplify our model, we assume that the aSleep transition causes no additional energy consumption (in fact, it requires non-zero energy consumption). In terms of the performance reputation cost, the VM Migration Manager tries to decentralize running VM requests over multiple physical hosts in order to avoid QoS deterioration caused by VM interference. In cloud datacenters, the VM co-location interference is the key factor that makes servers undergo severe performance degradation [12, 13]. VM co-location interference is caused by resource contention which is reflected mainly by the number of co-located VM instances and the resource utilization of them. Briefly, VM co-location interference becomes larger as more VM instances are co-located on a common server, and subsequently, higher resource utilization occurs. Therefore, VM requests have to be scattered in order to avoid performance degradation by VM co-location interference as best as possible. Because of the complexity of the optimization for resource management in large-scale cloud datacenters, our TP-ARM scheme adopts a metaheuristic based on GA to obtain a near optimal solution for VM migration to achieve energy savings and QoS assurance in a cloud datacenter.

3 Proposed model for cost optimization of TP-ARM in the cloud

In this section, we introduce an energy cost model and performance reputation cost model based on our proposed TP-ARM scheme including VM live migration and DRS execution in the ACRM.

3.1 Workload cost model for phase 1: VM migration

In general, VM requests have heterogeneous workloads in cloud datacenters. There are three types of VM request workloads: CPU, block I/O, and network I/O intensive workloads. The CPU intensive workloads such as scientific applications, high-definition video compression or big data analysis should be processed within an acceptable completion time determined by the cloud service users. The block I/O intensive workloads such as huge engineering simulations for critical areas include astrophysics, climate, and high energy physics which are highly data intensive [14]. These applications contain a large number of I/O accesses where large amounts of data are stored to and retrieved from disks. Network I/O intensive workloads such as Internet web services and multimedia streaming services have to be processed within a desirable response time according to their application type [10, 15]. Each

VM request requires different resource utilizations according to their running applications. We categorized the resource utilization of running VM requests on a physical host in a cloud datacenter into two parts in this paper, the Flavor Utilization (FU) and the Virtual Utilization (VU). FU represents the ratio of the resource flavor (i.e., the specification of the resource requirement) of a VM request to the resource capacity of the physical host. VU represents the resource utilization of an assigned virtual resource. The Flavor Utilization $FU_{j,\ i}^{k}$ of a VM request i on a physical host j in the resource component k and the Virtual Utilization $VU_{j,\ i}^{k}$ of the VM request i on the physical host j in the resource component k are given by

$$FU_{j,\ i}^{k}=\frac{flv_{i}^{k}}{rcp_{j}^{k}} \tag{1}$$

$$RU_{j,\ i}^{k}=FU_{j,\ i}^{k}\cdot VU_{j,\ i}^{k} \tag{2}$$

where flv_{i}^{k} is the flavor of the VM request i in the resource component k; rcp_{j}^{k} is the resource capacity of the physical host j in the resource component k, and $RU_{j,\ i}^{k}$ is the actual resource utilization of the VM request i on the physical host j in the resource component k; therefore, it describes the practical workload in the resource component k. Note that the FU of the VM request can be determined through its attached service description to the ACRM in advance, while the VU can only be measured by the internal monitoring module in the cloud server during the running duration of the VM request. In the period t, the VM migration plan is designed according to the FU of the submitted VM requests at period t and the measured RU of each host in the past history $t-1$. The VM migration plan should satisfy the following constraints:

$$flv_{i}^{k}\leq S_{k}^{t}(i),\ \forall i,\ k,\ t \tag{3}$$

$$flv_{i}^{k}\geq 0,\ \forall i,\ k \tag{4}$$

$$S_{state}^{t}(i)=1,\ \forall i,\ t \tag{5}$$

where $S_{k}^{t}(i)$ denotes the remain capacity of resource k in the physical server S which is assigned to the VM request i at time period t, and $S_{state}^{t}(i)$ represents the state of the physical server S which is assigned to the VM request i at time period t. If the server S is in the sleep mode, then $S_{state}^{t}(i)=0$, otherwise, $S_{state}^{t}(i)=1$ (i.e., it is in the active mode). Constraint (3) represents that the total requirements of the resource capacity of the newly allocated VM requests and migrated VM requests at time period t cannot exceed the resource capacity provided by their assigned physical server S. Next, we consider a VM performance reputation which is determined based on the RU of each physical server. The VM co-location interference implies that the virtualization of the cloud supports resource isolation explicitly when multiple VM requests are running simultaneously on a common PM however, it does not mean the assurance of performance isolation between VM requests internally. There is a strong relationship between VM co-location interference and both of the number of co-located VMs and their resource utilization in the PM. As the number of co-located VM requests increase and the RU of the VM requests becomes larger, VM co-location interference becomes more severe. The VM Migration Manager selects server candidates to be migrated based on the amount of RU for each physical server. To achieve this, the RU size of each server is measured through an internal monitoring module, and the VM Migration Manager checks whether they exceed the predetermined RU threshold value RU_{thr}. In servers which have an RU size over the RU_{thr}, they are considered as candidates for migration servers during the next period. After the migration servers are chosen, the VM requests to be migrated and their destination servers are determined based on the performance reputation cost model. We propose the objective function of the performance reputation cost C_{repu}^{t} at time C_{repu}^{t} given by

$$C_{repu}^{t}=\rho_{intf}\sum_{\forall j}\sum_{\forall k}\omega_{k}\left|\mathcal{PM}_{j}^{t}\right|\cdot\left(\frac{\sum_{i=1}^{\left|\mathcal{PM}_{j}^{t}\right|}RU_{idx_{j,\ i}^{t}}^{k}}{RU_{thr}^{k}}-1\right)^{+}+\rho_{mig}T_{mig}\sum_{\forall j}\left|\left|\mathcal{PM}_{j}^{t-1}\right|-\left|\mathcal{PM}_{j}^{t}\right|\right| \tag{6}$$

where $(x)^{+}=\max(x,\ 0)$, and ρ_{intf} and ρ_{mig} are the price constants of the VM interference cost and VM migration cost, respectively. We use T_{mig} to denote the processing time for the VM migration. The first term in the right hand of Eq. (6) represents the following: as the number of concurrent running VM requests on a physical server increases, the number of users experiencing undesirable performance degradation also increases. The second term represents the following: as the number of migrated VM requests increases, the migration overhead is also increases. Therefore, a migration plan needs to be found that satisfies both avoiding unnecessary migration overhead and minimizing performance reputation degradation.

3.2 Energy consumption cost model for phase 2: DRS procedure

To achieve a power-proportional cloud datacenter which consumes power only in proportion to the workload, we considered a DRS procedure which adjusts the number of active servers by turning them on or off dynamically [2]. Obviously, there is no need to turn all the servers in a cloud datacenter on when the total workload is low. In the DRS procedure, the state of servers which have no running applications can be transited to the power saving mode (e.g., sleep or hibernation) in order to avoid wasting energy. At each time t, the number of active servers in the cloud datacenter is determined by a DRS procedure plan according to the workload of the allocated VM requests. In order to successfully deploy the DRS procedure onto our system, we considered the switching overhead for adjusting the number of active servers (i.e., for turning servers in sleep mode on again). The switching overhead includes the following: 1) additional energy consumption from the transition from a sleep to active state (i.e., awaken transition); 2) wear-and-tear cost of the server; 3) fault occurrence by turning servers in sleep mode [2]. We considered the energy consumption as the overhead from DRS execution. Therefore, we define the constant P_{aWake} to denote the amount of energy consumption for the aWake transition of the servers. Then, the total energy consumption C^t_{energy} of the cloud datacenter at time t is given by

$$C^t_{energy} = \rho_p P_{active} \sum\nolimits_{\forall j} \left(\left| \mathcal{PM}^t_j \right| \right)^- + \rho_p P_{aWake} T_{switch} \left(\sum\nolimits_{\forall j} \left(\left| \mathcal{PM}^t_j \right| \right)^- - \sum\nolimits_{\forall j} \left(\left| \mathcal{PM}^t_j \right| \right)^- \right)^+ \quad (7)$$

where $(x)^- = \min(x, 1)$. We use ρ_p to denote the constant of the power consumption price, and P_{active} and P_{aWake} are the amount of power consumption for the active mode and the aWake transition of the server, respectively. T_{switch} is the time requirement for the aWake transition. The first term on the right side of Eq. (7) represents the energy consumption for using servers to serve VM requests allocated to all the physical servers in the cloud datacenter at time t, and the second term represents the energy consumption for the awaken transition of sleeping servers. Especially, the second term implies that a frequent change in the number of active servers could increase an undesirable waste of energy. Note that the overhead by the transition from the active to the sleep state (i.e., asleep transition) is ignored in our model because the time required for the asleep transition is relatively short compared to the one for the awaken transition. The purpose of our algorithm is to achieve a desirable VM migration and DRS procedure plan that are energy efficient as well as a QoS aware resource management at each period iteratively. At period $t-1$, the proposed TP-ARM approach aims to minimize the cost function in Eq. (8) by finding the solution $\mathcal{PM}^t$ as follows,

$$\begin{aligned} &minimize_{\mathcal{PM}^t} && C^t_{total} = C^t_{repu} + C^t_{energy} \\ &subject\ to. && \text{Eq } (3),\ (4)\ and\ (5) \end{aligned} \quad (8)$$

To solve the objective cost function in Eq. (8), we prefer a well-known evolutionary metaheuristic called the Genetic Algorithm (GA) in order to approximate the optimal plan $\mathcal{PM}^t$ at each period because Eqs. (6) and (7) have non-linear characteristics. In next section, our proposed TP-ARM with the VM migration and DRS procedure is introduced in detail.

4 Heuristic algorithms for the proposed TP-ARM scheme

In this section, we describe heuristic algorithms for the proposed TP-ARM scheme discussed in Algorithms 1, 2, and 3 shown in Figs. 2, 3, and 4. First, the process of VM migration in the TP-ARM approach includes the following four steps: 1) monitor and collect the resource utilization data on VM requests for each physical server through the attached Libvirt based monitoring tools; 2) go step 3 if the average utilization of all the active servers are significantly low (i.e., below the predefined threshold), otherwise go to step 4; 3) 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) determine the number of sleeping servers which are supposed to be turned on and send magic packets to them for wake-up if the average utilization of all the active servers are significantly high (i.e., above the predefined threshold), otherwise maintain the current number of active servers. Algorithm 1 shows the procedure for Phase 1:VM migration in the TP-ARM scheme. From line 00 to 06, the resource utilization RU of all the VM requests on the physical servers in the cloud datacenter is measured by the Libvirt API monitoring module. The average resource utilization $\overline{\mathrm{RU}^k}$ at all resource components k is calculated in line 07. If the $\overline{\mathrm{RU}^k}$ is below the predetermined $\mathrm{RU}^k_{\mathrm{thr}^{\mathrm{low}}}$ at all the resource components, then the optimal VM migration plan $\mathcal{PM}^t$ is derived, and the algorithm is finished. Otherwise, if the $\overline{\mathrm{RU}^k}$ exceeds the predetermined $\mathrm{RU}^k_{\mathrm{thr}^{\mathrm{high}}}$, then Algorithm 2 for the DRS procedure is triggered.

Algorithm 2 shows the procedure for Phase 2: DRS in the TP-ARM scheme. In line 00, the history window size ϖ' is determined by the ACRM through Algorithm 4: the SAWP scheme that is described in next section. In line 02, we calculate the sign φ and the angle ξ of the slope of the historical resource utilization curve from the current time $t-1$ to the time $t-\varpi'$. If $\varphi \geq 0$ (i.e., the resource utilization is increased), and then, we get the coefficient α based on $\alpha_{\mathrm{large}} \cdot \xi$, ($\alpha_{\mathrm{large}} > \alpha_{\mathrm{small}}$) to adaptively react to the large workload level in the cloud datacenter. Otherwise, if $\varphi < 0$ (i.e., the resource utilization is decreased), then we get α based on $\alpha_{\mathrm{small}} \cdot \xi$, to maximize the energy saving of the cloud datacenter. In line 08, we determine the number of servers to be in active mode in the next period.

Algorithm 3 describes the GA for the TP-ARM scheme in detail. $\mathrm{pop}^{\mathrm{h}}$ is a population with size P (even number) at the h^{th} generation. $\mathrm{h}_{\max}$ is the maximum of the GA iteration count. In line 03, f(·) is a fitness function of the total cost with two parameters, candidate solution $\mathcal{PM}^{\mathrm{h},\ \mathrm{i}}$ and the previous solution $\mathcal{PM}^{\mathrm{t}-1}$ at time $\mathrm{t}-1$. From line 04 to 06, the two candidate solutions are iteratively chosen randomly from pop to generate offsprings by crossover until there are no remaining unselected solutions in pop. From line 07 to 08, the fitness function values of each offspring are calculated similar to line 03. In line 09, all

Algorithm 1. Phase 1: VM migration in TP-ARM

```
    Input : the VM requests allocation of each server $\mathcal{PM}^{t-1}, \forall j$
    Output : the VM migration plan $\mathcal{PM}^{t}, \forall j$
00: for each $\mathcal{PM}_j^{t-1} \in \mathcal{PM}^{t-1}, \forall j$
01:   for each $VM_{idx_{j,i}^{t-1}}, \forall i$
02:     for each resource component $k$
03:       measure resource utilization $RU_{idx_{j,i}^{t-1}}^{k}$ of the VM request with $idx_{j,i}^{t-1}$
04:     end for
05:   end for
06: end for
07: calculate the average resource utilization $\overline{RU^k}$ at all resource components $k$
08: if $\overline{RU^k} < RU_{thr^{low}}^{k}, \forall k$ then
09:   derive the VM migration plan $\mathcal{PM}^{t}$ based on Eq.(8) through Algorithm 3.
10: else if $\overline{RU^k} > RU_{thr^{high}}^{k}$ then
11:   go to Algorithm 2.
12: end if
```

Fig. 2 Phase 1: VM migration procedure consolidates running VM instances from the low-utilized servers to others in order to maximize the energy saving of the cloud datacenter

the parent solutions in pop and generated offsprings are sorted in ascending order for their corresponding fitness function values. In line 10, the next population including only P solutions that achieve good performance from the union of the original pop and derived offsprings is generated to improve the quality of the final solution. From line 11 to 12, to reduce the time complexity of the GA procedures, when we encounter the first solution that has a fitness function value below the predetermined fitness threshold value fv_{thr}, it counts as a final solution for the next period, and the algorithm is finished. In line 14, if we cannot find a solution that satisfies fv_{thr} when the iteration count reaches h_{max}, then we select a solution that has a minimum fitness function value in the population which satisfies the conditions in Eqs. (3), (4) and (5) as a final solution for the next period. If there are no solutions to satisfy all the constraints, then we just preserve the current resource allocation vector shown from line 15 to 16. In the population of a GA, mutations are often applied in order to include new characteristics from the offsprings that are not inherited traits from the parents [16]. We did not consider mutations in our GA in this paper; however, it can be used to improve the quality of the GA for the TP-DRM in future work.

5 Self-adjusting workload prediction scheme

Figure 5 describes the procedure of the proposed SAWP algorithm in ACRB. The irregularity function $g(\varepsilon, \delta)$ represents a level of unpredictability for future resource

Algorithm 2. Phase 2: DRS procedure in TP-ARM

```
    Input : the average resource utilization $\overline{RU^k}, \forall k$
            the VM requests allocation of each server $\mathcal{PM}^{t-1}, \forall j$
            the historical data of resource utilization $\overline{RU^k}$ in $t-1, t-2, \ldots, t-\varpi$
    Output : the set of powered-off servers to be turned on at time $t$.
00: determine the boundary size of the history window, $\varpi'(\le \varpi)$ through Algorithm 4.
01: estimate the sign $\varphi$ and angle $\xi$ of an slope of the historical resource utilization
    curve from $t-1$ to $t-\varpi'$.
02: if $\varphi \ge 0$ then
03:   $\alpha = \alpha_{large} \cdot \xi$
04: end if
05: if $\varphi < 0$ then
06:   $\alpha = \alpha_{small} \cdot \xi$
07: end if
08: determine the number of sleeping servers supposed to be turned on according
      to $\alpha \cdot max_k \frac{\overline{RU^k}}{RU_{thr^{high}}^{k}}$
09: derive the set of powered-off servers to be turned on at time $t$
```

Fig. 3 Phase 2: DRS procedure turns sleeping servers on in order to react the increased workload level

Algorithm 3. GA for searching VM migration plan in TP-ARM

```
Input : the set of VM requests allocation of whole servers $\mathcal{PM}^{t-1}$ at time $t-1$
Output : the set of VM requests migration plan, $\mathcal{PM}^{t}$ at time $t$
00: initialize $\boldsymbol{pop}^h = \{\mathcal{PM}^{h,1}, \mathcal{PM}^{h,2}, \ldots, \mathcal{PM}^{h,P}\}$, $P = size\ of\ population$,
    and even number
01: while $h \le h_{max}$
02:     for each $\mathcal{PM}^{h,i} \in \boldsymbol{pop}^h$
03:         $fv^{h,i} = f_{C^t_{total}}(\mathcal{PM}^{h,i}, \mathcal{PM}^{t-1})$
04:     while $!(\forall \mathcal{PM}^{h,i} \in \boldsymbol{pop}^h\ are\ selected\ as\ parents)$
05:         $(\mathcal{PM}^{h,i}, \mathcal{PM}^{h,j}) \leftarrow$ select parents randomly from $\boldsymbol{pop}^h$
06:         $\boldsymbol{offspring}^h =\cup$ crossover $(\mathcal{PM}^{h,i}, \mathcal{PM}^{h,j})$
07:     for each $off\_\mathcal{PM}^{h,i} \in \boldsymbol{offspring}^h$
08:         $off\_fv^{h,i} = f_{C^t_{total}}(off\_\mathcal{PM}^{h,i}, \mathcal{PM}^{t-1})$
09:     sort $\boldsymbol{pop}^{h\sim} = \left\{\mathcal{PM}^{h,i}, \ldots, \mathcal{PM}^{h,P}, off\_\mathcal{PM}^{h,1}, \ldots, off\_\mathcal{PM}^{h,\frac{P}{2}},\right\}$ in
        ascending order of solutions' corresponding $fv^{h,i}\ and\ off\_fv^{h,i}$
10:     $\boldsymbol{pop}^{h+1} = \{\boldsymbol{pop}^{h\sim}_1, \ldots, \boldsymbol{pop}^{h\sim}_P\}$
11:     if $f_{C^t_{total}}(\boldsymbol{pop}^{h+1}_1, \mathcal{PM}^{t-1}) \le fv_{thr}$ then
12:         $\mathcal{PM}^t = \boldsymbol{pop}^{h+1}_1$ and exit
13:     $h++$
14:     $\mathcal{PM}^t = \mathrm{argmin}_{\boldsymbol{pop}^{h_{max}}_i \in \boldsymbol{pop}^{h_{max}}} \left(f_{C^t_{total}}(\boldsymbol{pop}^{h_{max}}_i, \mathcal{PM}^{t-1})\right)$ s.t. Eq (3), (4), (5)
15: if $\mathcal{PM}^t = \emptyset$ then
16:     $\mathcal{PM}^t = \mathcal{PM}^{t-1}$
```

Fig. 4 GA for VM migration determines the appropriate VM requests supposed to be migrated in order to maximize the energy saving of the cloud datacenter

utilization where ε is its fluctuation value (i.e., levels of instability and aperiodicity), and δ is its burstiness value (i.e., levels of sudden surge and decline), and both values are calculated based on [8].

According to the predetermined threshold values g^{high}_{thr} and g^{low}_{thr} with $g(\varepsilon, \delta)$, the history window size ϖ' is adaptively updated at each prediction process. A relatively long history window size is not suitable to react to recent changes in the workload but is tolerant to varied workload patterns over a short time while a short history window size is favorable to efficiently respond to the latest workload patterns but is not good for widely varying workloads. Consequently, the SAWP algorithm generally outperforms traditional prediction schemes during drastic utilization changes from various cloud applications because it is able to cope with temporary resource utilizations (i.e., does not reflect overall trends) by adjusting the history window size ϖ'.

Algorithm 4. Self Adjusting Workload Prediction scheme

```
Input : the historical data of resource utilization $\overline{RU^k}$ in $t-1, t-2, \ldots, t-\varpi$
Output : the boundary size of history window $\varpi'$
00: $\varpi' = \varpi$
01: analyze the historical data of resource utilization $\overline{RU^k}$ from $t-1$ to $t-\varpi$
02: extract the fluctuation size $\varepsilon$ and the burstness value $\delta$ based on [6]
03: if $g(\varepsilon,\delta) > g^{high}_{thr}$ then
04:     increase $\varpi'$ based on $\frac{g(\varepsilon,\delta)}{g^{high}_{thr}}$
05: end if
06: if $g(\varepsilon,\delta) < g^{low}_{thr}$ then
07:     decrease $\varpi'$ based on $\frac{g^{low}_{thr}}{g(\varepsilon,\delta)}$
08: end if
09: predict the future demand based on $\overline{RU^k}$ from $t-1$ to $t-\varpi'$
```

Fig. 5 SAWP scheme is able to increase the accuracy of the prediction even under the dynamic workload

6 Experimental results and discussion

In our experiments, we measured various metrics which affect the parameter decision for our proposed algorithms. To this end, we established five cluster servers as a cloud platform, one server for ACRB with a Mysql DB system, a power measuring device from Yocto-Watt [17] and a laptop machine called the VirtualHub for collecting and reporting information on the measured power consumption shown in Fig. 6. The hardware specification of each server for the cloud compute host is as follows: an Intel i7-3770 (8-cores, 3.4Ghz), 16 GB RAM memory, and two 1 Gbps NICs (Network Interface Cards). In order to measure efficiently the power consumption of the cloud cluster server, we used a power measuring device model called YWATTMK1 by Yocto-Watt. This model has a measurement unit of 0.2 W for AC power with an error ratio of 3 % and 0.002 W for the DC power with an error ratio of 1.5 %. The VirtualHub collects information on power consumption from YWATTMK1 through the Yocto-Watt Java API and reports it to the power monitoring table of the Mysql DB system in the ACRB periodically. The dynamic resource utilizations by each VM instance are measured via our developed VM monitoring modules based on the Libvirt API and are sent to the resource monitoring table of the Mysql DB system periodically. In addition, a SATA 3 TB hard disk called the G-drive was deployed as a NFS server in our testbed for live migration [18]. We adopted Openstack kilo version which is a well-known open source solution based on KVM Hypervisor as a cloud platform in our testbed. Finally, we used Powerwake package [19] 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 [20, 21]. The m106-1.7 projection in Montage is a cpu-intensive application while the ftp transfer is a network-intensive one. Therefore, the running of m106-1.7 causes a power consumption of about 75 Wh and a cpu utilization of about 15 % whereas the power consumption by ftp transfer for a 1.5 GB test.avi file is about 60 Wh, and the network bandwidth usage is about 3.7Mbps. That is, the cpu usage is the main part that affects the power consumption of server. In terms of DRS execution, the power consumption by an off server is about 2.5 Wh (note that this value is not zero because the NIC and some its peripheral components are still powered on to maintain the standby mode to receive the magic packets from the Powerwake controller) while the asleep and awaken transition procedures, which cause the switching overhead for DRS, require a power consumption of about 80 Wh to turn the active servers off or to turn off servers on, respectively.

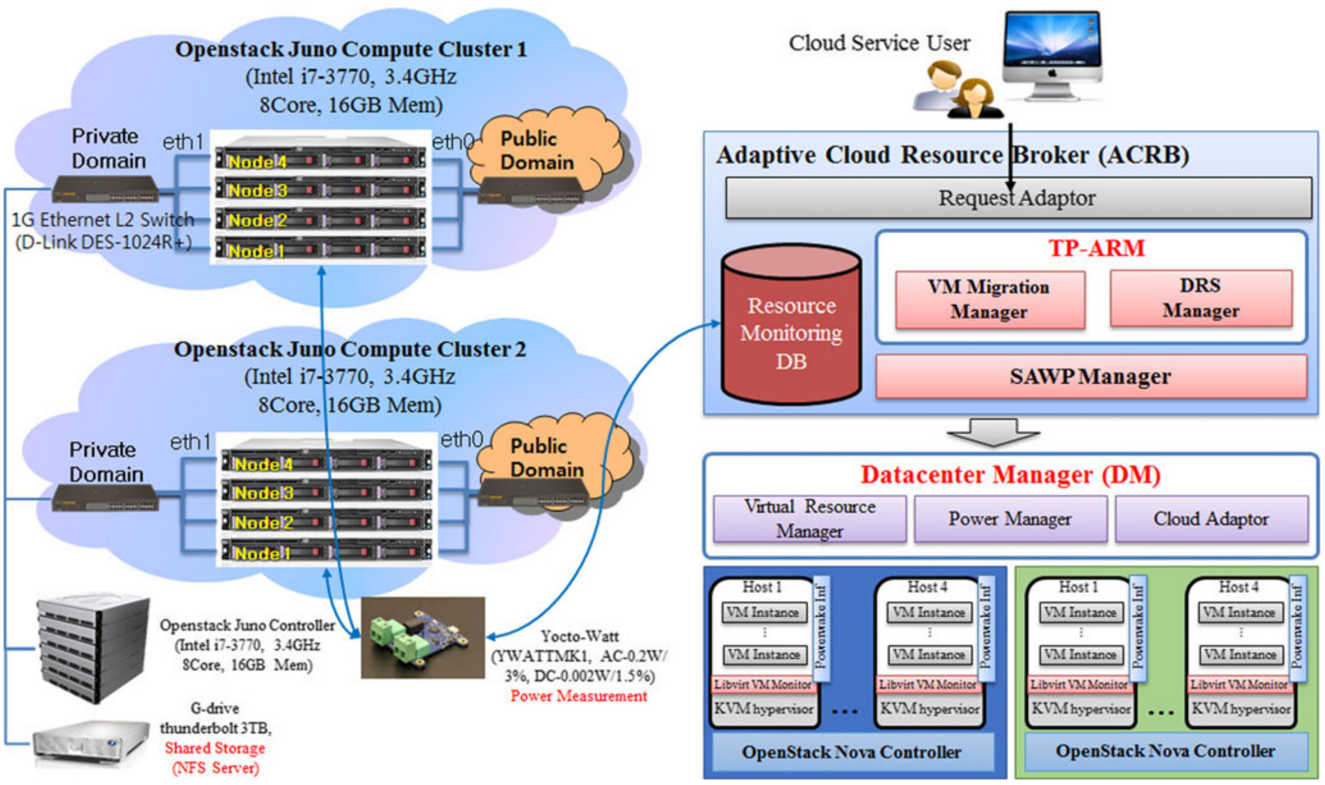


Fig. 6 Experimental Environment

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

The asleep transition procedure is trivial because it requires a short time (i.e., 5 ~ 7 s) to complete even though its power consumption is considerable whereas the awaken transition procedure requires a relatively long execution time (i.e., more than 1 min) and should be considered carefully. The overhead for the awaken transition would be a more serious problem in practice because its required execution time is generally far longer (i.e., above 10 min) for multiple servers of racks in a datacenter. Therefore, it is essential to consider the switching overhead for awaken transition to reduce efficiently the resource usage cost of the datacenter. Figure 7 shows the additional energy consumption overhead of a single physical server from the DRS execution with respect to the asleep transition (i.e., from the active mode to the sleep mode) and the awaken transition (i.e., from the sleep mode to the active mode). Note that the asleep transition requires a relatively short processing time of about 7 or 8 s and causes an additional energy consumption of about 20 % compared to the idle state of the server, while the awaken transition needs 60 or 90 s to be carried out. Although not included in this paper, we also verified that some of the HP physical servers in our laboratory required more than 10 min to be turned on. We expect that physical servers in a real cloud datacenter require tens of minutes or hundreds of minutes to get back from the sleep mode to the active mode. Moreover, the aWake transition causes an additional energy consumption of about 40 % on average compared to the idle state of a server. These results imply that the frequent DRS execution might cause the degradation of the energy saving performance in a cloud datacenter.

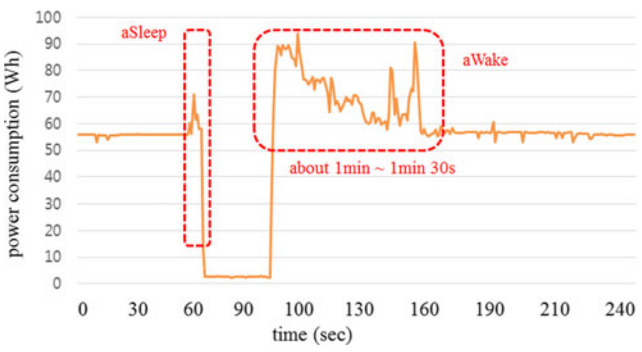


Fig. 7 Power Consumption Overhead from DRS execution of the test server with respect to the asleep and awaken transition

Our proposed TP-ARM scheme is able to avoid the unnecessary energy consumption overhead by the awaken transition through the cost model by considering the DRS transition overhead shown in Eq. (7). Figure 8 shows the resource utilization and the power consumption measured by the Libvirt API monitoring module and Yocto-Watt power measuring device. VM request instance-10 runs m101-1.0 mProj, instance-0f runs m108-1.7, and instance-0c runs the streaming server with the 180 MB movie file. Figure 8 (a) ~ (c) shows that their resource utilization is consistent with the resource intensive characteristic of their running workload.

Figure 8d shows the different energy consumptions according to each workload type. The energy consumption of the physical server is 60 Wh in the idle state; it is about 82 Wh when running m101-1.0 and 95 Wh when running both m101-1.0 and m108-1.7, while the streaming server just causes an additional energy consumption of about 2 Wh. As mentioned in section 3, the main part of the resource components affecting the power consumption in a physical server is the CPU resource, and the effects of other components such as the memory, storage, and network interface cards are negligible in general. These results are consistent with the ones in Table 1.

Therefore, these results imply that the energy consumption can be different according to which resource component has high utilization. Our proposed workload cost model of the TP-ARM scheme considers different weight values for each resource component in order to derive the practical cost of the resource management. Figure 9 shows the performance of the CPU utilization for the VM live migrationbetween the source machine (kdkCluster2) and the destination machine (kdkCluster1). There are VM request instances, such as running instance-33 and 34, which execute the compression of video files 5 GB and 4 GB in size, respectively. Instance-33 executed the process of m101-1.6mProj at 12:40:14 local time and started migration to the kdkCluster1 at 12:44:00 local time. The migration of instance-33 was completed by 13:24:30 local time, and its execution of m101-1.6 mProj ended at 13:47:16. Namely, the completion time of m101-1.6 mProj at instance-33 was 67 min in total, and its migration times were over 30 min. If we consider the time to complete m101-1.6 mProj in the case of no VM live migration, it is forecasted to be around 10 min. We can see that

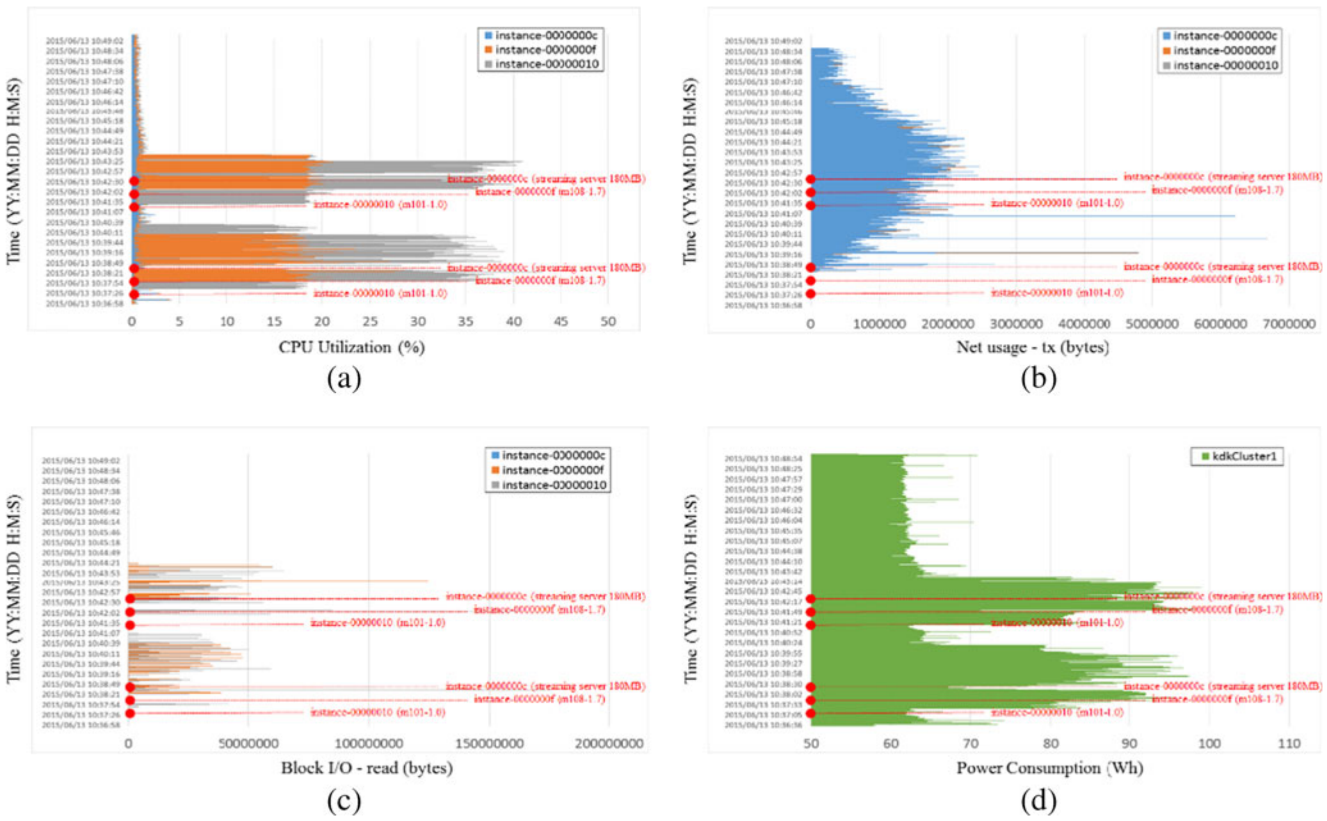


Fig. 8 Libirt API monitoring module shows the results of resource utilization of running VM requests in the kdkCluster1 in (**a**), (**b**) and (**c**). Yocto-Watt module [17] was used to measure the power consumption of the kdkCluster1 as shown in (**d**)

there exist quite big overheads in VM live migration. In addition, VM migration shows some problems in power consumption. We considered cost models to identify an efficient migration scheme which was defined in Eq. (6). Figure 10 shows the test environments and the operation of the VM migration when the proposed TP-ARM schemes are applied to the test cloud systems with heterogeneous applications and test programsincluding m108-1.7, pbzip2, and netperf. From the experiments, we obtained interesting results in the comparison of the utilization performance, which examined the CPU utilization monitoring results and the measured power consumption, respectively, shown in Fig. 11. From the results, we found rapidly changing instants of utilization when the cloud brokering system considered the power consumptions for each running application under cloud data center environments. Additionally, the test set 'netperf', a network intensive workload of a test cloud system (e.g., kdkCluster4 with instance-35) in Fig. 11d, showed very low performance in CPU utilization. It was enough to satisfy the threshold value to run the VM migration efficiently. And then, the proposed TP-ARM algorithm adjusts the VM migration procedure to reduce the power consumption sufficiently. Basically, the TP-ARM is triggered to migrate instance-35 to another system kdkCluster2; thereafter, it changes the kdkCluster's sleeping mode in advance. Next, because instance-34 of the kdkCluster

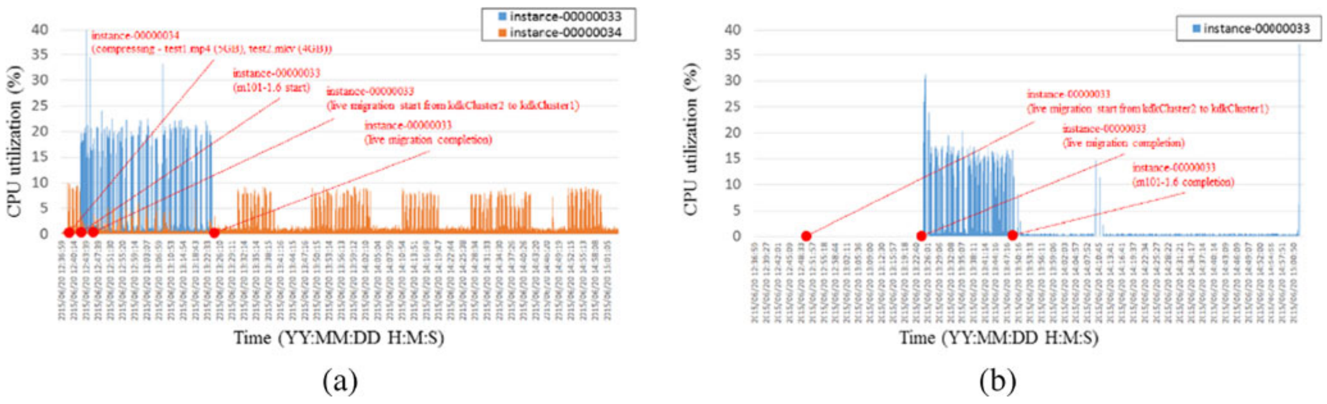
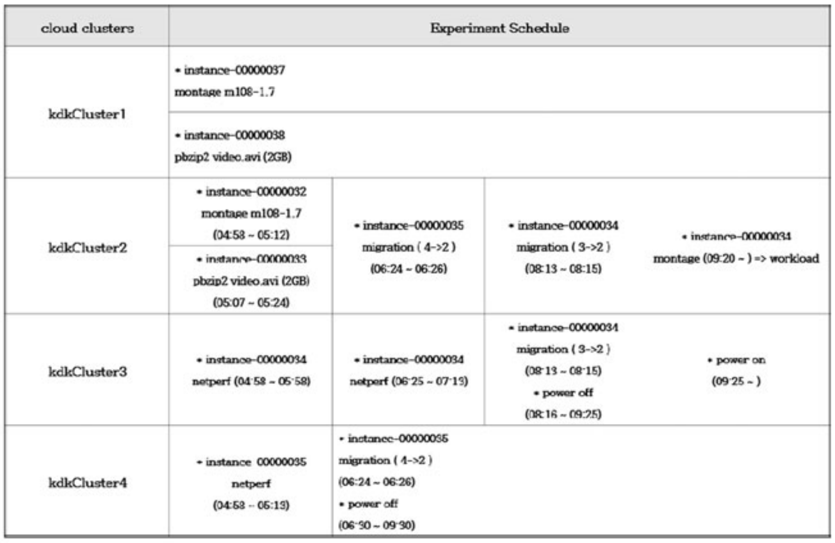
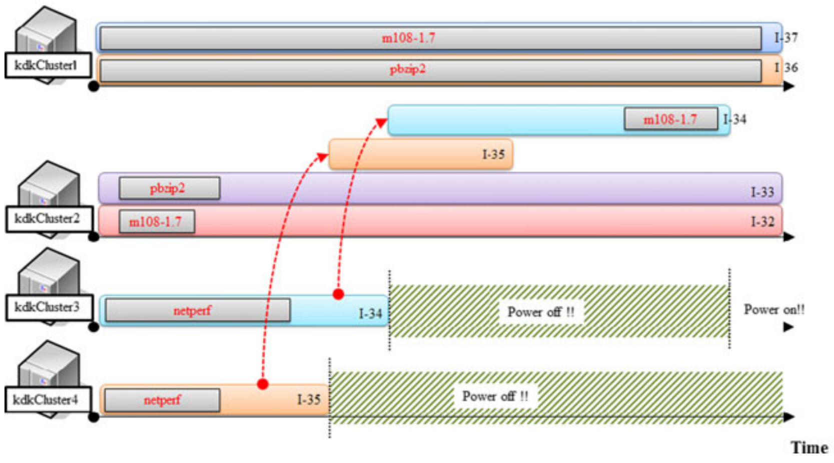


Fig. 9 Results of CPU utilization of the migrated VM request from (**a**) a source server:kdkCluster2 to (**b**) a destination server:kdkCluster1



cloud clusters	Experiment Schedule			
kdkCluster1	• instance-00000037 montage m108-1.7			
	• instance-00000038 pbzip2 video.avi (2GB)			
kdkCluster2	• instance-00000032 montage m108-1.7 (04:58 ~ 05:12) • instance-00000033 pbzip2 video.avi (2GB) (05:07 ~ 05:24)	• instance-00000035 migration (4->2) (06:24 ~ 06:26)	• instance-00000034 migration (3->2) (08:13 ~ 08:15)	• instance-00000034 montage (09:20 ~) => workload
kdkCluster3	• instance-00000034 netperf (04:58 ~ 05:58)	• instance-00000034 netperf (06:25 ~ 07:15)	• instance-00000034 migration (3->2) (08:13 ~ 08:15) • power off (08:16 ~ 09:25)	• power on (09:25 ~)
kdkCluster4	• instance 00000035 netperf (04:58 ~ 05:13)	• instance-00000035 migration (4->2) (06:24 ~ 06:26) • power off (06:30 ~ 09:30)		

(a) The schedule of the use case



(b) The illustration of the use case

Fig. 10 Test environments and operation of the VM migration when the proposed TP-ARM schemes are applied for test cloud servers (e.g., kdkCluster1 ~ 5 in lab test environments) with heterogeneous applications and test programs, such as m108-1.7, pbzip2, and netperf. (**a**) An example of test schedule under cloud testbed environments (**b**) Illustration of the VM migration using two-phase power consumption cost models

system was running netperf, which generated a very low performance of utilization, the TP-ARM also did a migration of the instance-34 to the kdkCluster2 and transited to the sleeping mode as well. Therefore, we could expect efficient power consumption through the change in the sleeping mode of kdkCluster3 and kdkCluster4, respectively, as well as keeping active modes for both kdkCluster1 and kdkCluster2. Figure 11 shows the performance comparison of the power consumption for the VM migration and DRS process based on the TP-ARM algorithm, e.g., seen in Fig. 10. We can see a performance improvement of 60 Wh each in power consumption after the change in the sleeping mode in the case of the VM migration requests. Through the experiments, we verified that the proposed TP-ARM scheme, which carries out adaptive migration and asleep transition through real-time monitoring of resource utilization, has good performance in reducing the power consumption effectively. This study provides good results for achieving an energy efficient cloud service for users by not increasing QoS degradation as much.

7 Conclusion

In this paper, we introduced an Adaptive Cloud Resource Broker (ACRB) with Two Phases based the Adaptive Resource Management (TP-ARM) scheme for energy efficient resource management by real time based VM monitoring in a cloud data center. Our proposed approach is able to reduce efficiently the energy consumption of the servers without a significant performance degradation by live migration and Dynamic Right Sizing (DRS) execution through a considerate model that considers switching overheads. The various experimental results based on the Openstack platform suggest that our proposed algorithms can be deployed to prevalent cloud data centers. The novel prediction

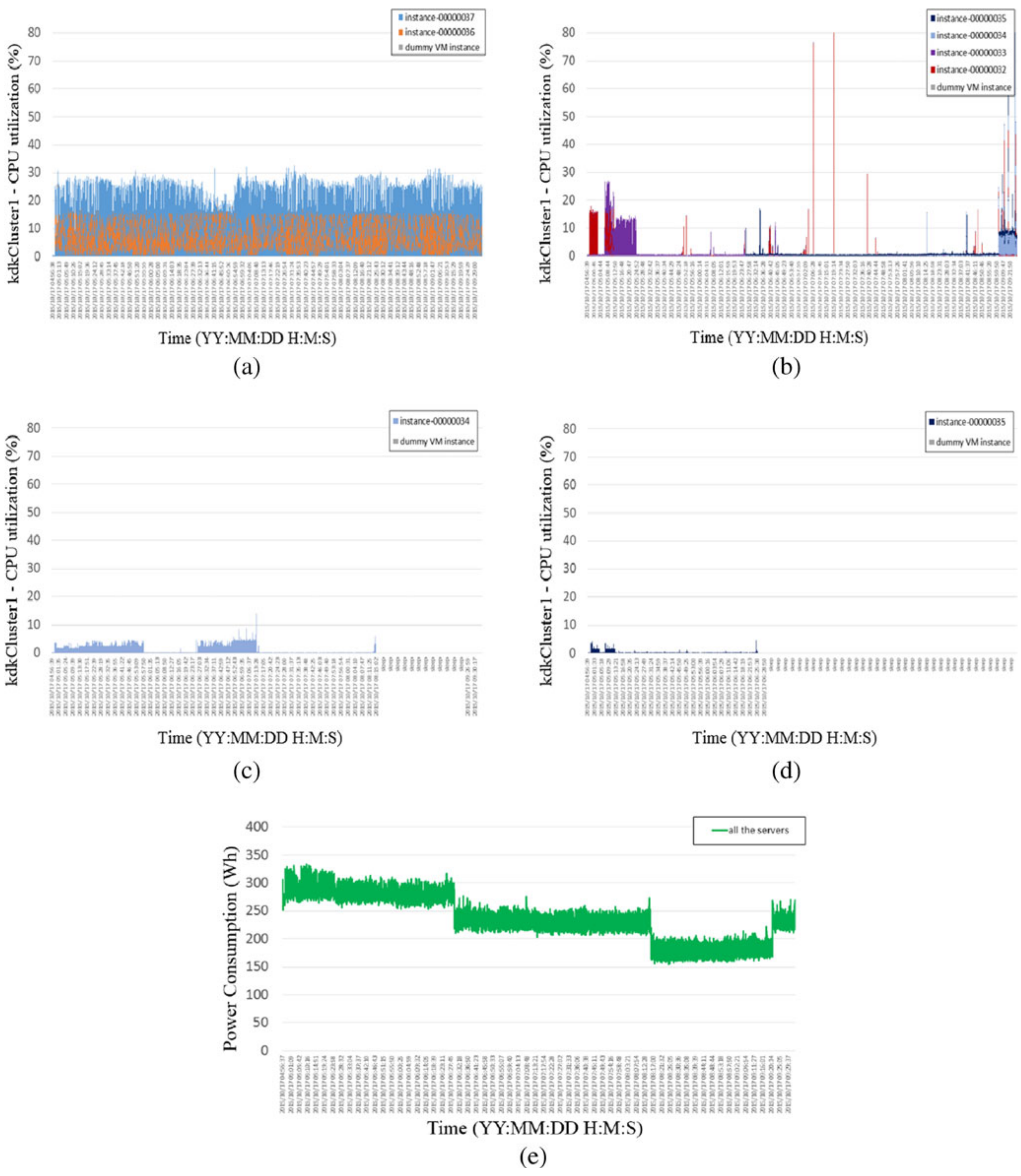


Fig. 11 Performance comparison of CPU utilization using 4 test systems in lab environments. (**a**), (**b**), and (**c**) show the results of the measured utilization during the VM live migration in test use case as shown in Fig. 9. (**d**) shows utilization of the proposed TP-ARM scheme that was used in test cloud environments. (**e**) Measured power consumption of test cloud systems (e.g., kdkCluster1 ~ 4) in lab cloud test environments

method called Self Adjusting Workload Prediction (SAWP) is proposed in order to improve the accuracy of forecasting future demands even under drastic workload changes.

Especially, we evaluated the performance of our proposed TP-ARM scheme through various applications such as Montage, pbzip2, netperf, and the streaming server which have heterogeneous workload demands. Our TP-ARM scheme could maximize the energy saving performance of the DRS procedure by Phase 1: VM migration achieves consolidated resource allocation under a low workload level. Moreover, it could ensure the QoS of cloud service users by Phase 2: the DRS procedure increases adaptively the number of active servers under a high workload level. Through experiments based on a practical use case, our proposed scheme is not only feasible from a theoretical point of view but also practical in a real cloud environment. In future work, we will demonstrate that our proposed algorithm outperforms existing approaches for energy efficient resource management through various experiments based on the implemented system in practice.

Acknowledgments This work was supported by "The Cross-Ministry Giga KOREA Project" of the Ministry of Science, ICT and Future Planning, Korea [GK13P0100, Development of Tele-Experience Service SW Platform based on Giga Media].

References

1. International Data center Corporation, http://www.idc.com
2. Lin M, Wierman A, Andrew LLH, Thereska E (2013) Dynamic right-sizing for power-proportional data centers. IEEE/ACM Trans Networking 21(5):1378–1391
3. Xiao Z, Song W, Chen Q (2013) Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans Parallel Distrib Syst 24(6):1107–1117
4. Beloglazov A, Buyya R (2012) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr Comput: Pract Experience 24:1397–1420. doi:10.1002/cpe.1867
5. Kang DK, Hazemi FA, Kim SH, Youn CH (2015) Dynamic virtual machine consolidation for energy efficient cloud data centers. In: Proc. EAI Int. Conf on Cloud Computing, Oct
6. Kim SH, Kang DK, Kim WJ, Chen M, Youn CH () A science gateway cloud with cost adaptive VM management for computational science and applications. to be appeared in IEEE Syst J, 2016
7. Chen M, Hao Y, Li Y, Lai CF, Wu D (2015) On the computation offloading Ad Hoc Cloudlet: architecture and service models. IEEE Commun Mag 53(6):18–24
8. A-Eldin A, Tordsson J, Elmroth E, Kihl M (2013) Workload classfication for efficient auto-scaling of cloud resources. Umea University, Sweden
9. Openstack, http://www.openstack.org
10. Chen M, Zhana Y, Hu L, Taleb T, Shena Z (2015) Cloud-based wireless network: virtualized, reconfigurable, smart wireless network to enable 5G technologies. ACM/Springer Mob Netw Appl 20(6):704–712
11. Chen M, Wen Y, Jin H, Leuna V (2013) Enaling technologies for future data center networking: a primer. IEEE Netw 27(4): 8–15
12. Xu F, Liu F, Liu L, Jin H, Li B, Li B (2014) iAware: making live migration of virtual machines interference-aware in the cloud. IEEE Trans Comput 63(12):3012–3025
13. Gupta D, Cherkasove L, Gardner R, Vahdata A (2006) Enforcing performance isolation across virtual machines in Xen. In: Proc. ACM/IFIP/USENIX 2006 Int. Conf. Middleware, Nov
14. Nisar A, Liao WK, Choudhary A (2008) Scaling Parallel I/O Peformance through I/O delegate and caching system. In: Proc. ACM/IEEE conf on Supercomputing, Nov
15. Chen M, Zhang Y, Li Y, Mao S, Leung VCM (2015) EMC: emotion-aware mobile cloud computing in 5G. IEEE Netw 29(2): 32–38
16. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm:NSGA-II. IEEE Trans Evol Comput 6(2):182–197
17. YOCTO-WATT, http://www.yoctopuce.com/EN/products/usb-electrical-sensors/yocto-watt
18. G-Technology, http://www.g-technology.com/products/g-drive
19. PowerWake, http://manpages.ubuntu.com/manpages/utopic/man1/powerwake.1.html
20. Montage, http://montage.ipac.caltech.edu/
21. Kim WJ, Kang DK, Kim SH, Youn CH (2015) Cost adaptive VM management for scientific workflow application in mobile cloud. J Mob Netw Appl, Springer 20(3):328–336