Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Min-max exclusive virtual machine placement in cloud computing for scientific data environment

Moon-Hyun Kim[1], Jun-Yeong Lee[1], Syed Asif Raza Shah[2], Tae-Hyung Kim[3] and Seo-Young Noh[1*]

Abstract

In cloud computing, there is a trade-off between SLAV (Service Level Agreement Violation) and system operating cost. Violation rates can be decreased when using more hosts, which increases system operating costs. Therefore, to manage the resources of those hosts efficiently, finding an optimal balancing point between SLAV and system operating cost is critical. In addition, a cost-effective load balancing approach based on the proper migration of VMs (Virtual Machines) in the hosts is needed. For this purpose, some indicators are also necessary to identify the abnormal hosts that violate SLA. One of the primary indicators, CPU usage, is closely related to energy consumption and can be used to reduce SLAV and energy consumption effectively. Our approach focuses on the special environment such as the cloud environment for the scientific data. Here, most of the jobs are data-intensive and a large amount of disk operations is required. Owing to disk operations are likely to affect the performance degradation of the host, disk bandwidth usage as well as CPU usage should be also considered. In this study, we propose the Min-Max Exclusive VM Placement (MMEVMP) strategy to minimize both SLAV and energy consumption. The current working system called KIAF (KISTI Analysis Facility), the CERN ALICE experimental cloud environment for scientific data analysis, is used to analyze the characteristics of data-intensive jobs within it. In this experiment, a lightweight version of the CloudSim simulator was developed and the results were compared to the other methods of different policies. Our evaluation showed that our proposed strategy can reduce SLA violation reasonably as well as the system operating cost-effectively.

Keywords: Cloud computing, Scientific data, Data-intensive, VM placement, Disk load balancing

Introduction

In recent years, data plays a crucial role in many fields, including industries and scientific communities. According to the advancement of science and technology, tremendous amounts of data are being produced. Nowadays, it is possible to store and analyze huge amounts of data due to ever-increasing computing power and cost-effective storage capacity, which results in scientific discoveries that could not be otherwise made. The amount of data being gradually increased requires more computing resources. Such trends in research require a cost-effective computing environment for data analysis, naturally leading to cloud computing environment.

A cloud service provider should manage the underlying physical system, so we only care for services running on them. To prevent the degradation of the host's performance, workloads for those services should be properly distributed across the entire hosts. In particular, the excessive workload on a distinct host can result in performance degradation and violate SLA(Service Level Agreement) that is the level of service we expect a vendor to provide. In general, this agreement uses the metrics that identify whether the user's expectation is satisfactory and includes

*Correspondence: rsyoung@cbnu.ac.kr
[1]Department of Computer Science, Chungbuk National University, Chungdae-ro, Cheongju, Republic of Korea
Full list of author information is available at the end of the article

the remedies or penalties provided when agreement violation occurs. SLA is a critical component of any technological vendor contract [1]. Therefore, if SLAV(Service Level Agreement Violation) occurs, the provider has to pay penalties to clients in proportion to the rate of SLAV.

A simple way to reduce the SLAV rate is increasing the number of the hosts for services. However, it should be noted that there is a trade-off between SLAV rate and system operating cost. SLAV rate could be reduced by means of increasing the number of hosts, but energy consumption is expected to be increased. Accordingly, the system operating cost becomes higher. In this respect, finding an optimal balancing point between SLAV rate and operating cost is important in order to effectively manage computing resources. Since VMs in cloud environment can be migrated among the hosts. The first step of moving VMs is to find an overloaded or underloaded host, and the next is to choose another host on which they will be placed. In this migration process, we need indicators to check the host's status and obtain reasonable decision-making information. The indicator may vary depending on the perspective of the host's performance or the perspective of the operating overall system infrastructure. From the perspective of the host's performance, indicator can be determined according to their impact on the host's performance. The effect of each resource on the host performance depends on the cloud environment and architecture to which the host belongs. For example, CPU usage can be an important indicator in the environment with a lot of computational work, and disk I/O and data latency can be more important than other indicators in such environments with a lot of data read/write. Also, depending on the cloud architecture, network bandwidth can be an important indicator in an environment that uses storage connected by a network. From the perspective of the system infrastructure operation, indicators that can be helpful to reduce overall operating costs are needed. In this case, it is necessary to understand the impact of each computing resource on total energy consumption. The indicators will be determined according to each level of impact.

Previous studies [2, 3] show that there is a close correlation between CPU usage and energy consumption. In addition, since most tasks in the cloud use a lot of CPU, its usage can be an important indicator from the perspective of the host's performance. For these reasons, CPU usage is commonly considered as a main indicator and used by many VM placement policies.

However, in a special cloud environment for scientific data where most of jobs are data-intensive, CPU usage alone is not enough to determine the status of the host. Since the jobs for data-intensive science inevitably incur a large amount of disk operations, they significantly increase the disk bandwidth usage of the host. Higher disk bandwidth usage will likely result in the performance degradation for the host [4]. Therefore, disk bandwidth usage as well as CPU usage should be considered carefully, especially in cloud for data-intensive jobs.

In this study, we focused on the cloud for data-intensive jobs and studied how to prevent host performance degradation due to disk bandwidth usage. For this purpose, we analyzed the KIAF(KISTI Analysis Facility), a CERN ALICE experimental cloud environment used by Korean researchers. When a job comes in KIAF cluster, it is assigned to a VM with one core CPU which was configured in advance, and the job is processed. In order to properly distribute the workload in this environment, a research on how to place VMs is needed. Therefore, we studied a VM placement strategy and for this, we extracted and analyzed the VM usage data of KIAF. Our analysis consists of three steps as follows: Firstly, we analyzed the resource usage of scientific jobs from the extracted data. Secondly, we developed the Min-Max Exclusive VM placement algorithm considering the cloud for scientific research. Lastly, we evaluated our strategy using a cloud simulator that is a lightweight version of CloudSim.

The main contributions of this paper can be summarized as follows:

1 Unlike previous VM placement policies, we considered disk bandwidth usage as a critical factor focusing on heavy scientific data processing environment. In our experiment, we considered other factors that may cause performance degradation. Therefore, we believe that our experimental results contribute to improve the overall performance of many hosts in the cloud environment.
2 In the cloud environment for scientific research, our proposed model can reduce SLAV rate as well as system operating cost. It is also possible to find an optimal balancing point between those two indicators in other similar cloud environments.
3 We have built our placement model based on the actual observation data. Therefore, the reference models presented in this paper can be applied to the cloud environments for other fields.

The rest of the paper is organized as follows: The previous studies related with VM placement are described in the "Related works" section. "Background" section presents fundamental knowledge that helps to understand our work. "Problem statements" section describes problems with existing techniques. Virtual machine placement algorithm proposed in this paper is explained in "Proposed methods" section. The experimental results of our algorithm are discussed in "Experiment" section including overall performance. Finally, this paper ends with the "Conclusion & future work" section.

Related works

In cloud computing, it is important to distribute workloads properly to computing hosts to avoid biased load assignments. Finding such a workload balancing point can be achieved through virtual machine migrations. This migration consists of several steps. Firstly, it detects which host is in an abnormal status. In this step, workloads are usually considered as the basic factor for judgment and used to measure the host's resource usage. There are two methods to manage computing resources: reactive resource management and proactive resource management [5].

Reactive resource management is based on monitored workloads at a certain time, which can lead to SLA violations owing to time-lagging. The proactive resource management also uses monitored workload, but it predicts future workloads using their monitoring data. Predicting the future usage of the host can prevent the host from becoming overloaded in advance. Therefore, many recent studies [6–10] have developed host detection techniques based on the proactive resource management.

The proactive resource management determines host's status based on its predicted resource usage. However, there is a problem that it consumes a non-negligible level of computing resources when measuring their usage. Predicting usage precisely, in general, is a difficult task because of the large unit of measurements across many hosts. In order to alleviate this challenge, Chen et al. suggested a way to predict the future usage of a host-based on the sum of the future usage of the VMs, not the usage of the host itself [11].

After an abnormal host is detected, a VM selection policy is applied. In this step, one or more VMs are moved to another host to mitigate the workload on the detected host. The final step is the VM placement process which chooses the host on which the VMs will be placed. During this migration process, the policy of placing the VM is required to find the appropriate host for the VMs to be moved.

There are many previous studies about the VM placement method. Fabio and Baran introduced the overall VM placement policies [12]. They classified the purpose of VM placement into several categories, each of which has an approach to optimize a particular goal. Typical optimization goals include minimizing energy consumption, minimizing operating costs and maximizing resource utilization.

Beloglazov et al. surveyed the cost model of the cloud system and suggested that there is a linear relationship between CPU usage and host's energy consumption [13, 14]. They claim that distributing loads based on CPU usage can prevent system overload and reduce overall energy consumption in data centers. For this reason, early studies focused on optimizing cloud resources utilization based on CPU usage. Hence, many VM placement policies have been introduced to maximize resource usage with a greedy approach.

Hui Zhao et al. proposed the VM placement method based on the VM performance model [15]. They claimed that most previous studies [16–18] are focused on the operational efficiency of data centers. In contrast to the previous work, they emphasized boosting VM performance which is also an important part of the user's perspective. In that sense, they introduced the performance model of the VM and the algorithm that can optimize the host's performance by assigning some VMs.

Many big data research and analysis have been performed in the cloud environment. Jian Guo et al. suggested a VM placement policy to consider disk usage in addition to CPU usage [19]. They applied Ford Fullkerson algorithm to VM placement policy and evaluated their method using benchmark tools. When deploying VMs, they applied a scheduling policy to ensure that disk-intensive job is not concentrated in a specific host.

Pathan et al. also proposed a VM placement policy considering disk I/O [4]. They suggested a live migration algorithm with a static threshold. Their policy selected an appropriate host to which VMs are deployed based on the corresponding disk threshold. Unlike the previous studies, they set a threshold for the disk usage on the host so that VMs could be placed on the host as long as it does not exceed the disk threshold. When the disk usage exceeds at a certain threshold, migration is performed in order to balance workload. Their method is evaluated using the CloudSim simulator.

The two methods above are focused on the purpose of preventing host performance degradation. In addition to how to place VMs to avoid host performance degradation, VM placement methods of increasing disk utilization have also been performed in some studies.

Shabeera et al. proposed VM and data placement method for data-intensive application in cloud [20]. They focused on reducing data latency when data transfer between VMs is active. They claimed that the distance between the VM and the PM(Physical Machine) where the data is stored affects the data latency, and used the meta-heuristic algorithm to minimize the distance between the two. They proved that the disk utilization can be effectively increased through this VM placement.

Background

In this section, we described the background knowledge related to our work.

CERN

CERN [21], the European Organization for Nuclear Research, is a research organization that operates the largest particle physics laboratory in the world. Currently, there are four underway major large-scale experiments such as ATLAS, ALICE, CMS, and LHCb, named after each detector. Physicists and engineers at CERN use LHC, the world's largest and most complex scientific instruments, to study the basic constituents of matter - fundamental particles. They analyze data from the LHC to explore the basic structure of particles that make up everything in the world. LHC experiment creates 600 million crashes per second and generates massive data of 1 million GB per second. To store and analyze these data, CERN has one of the largest data centers consisting of 230,000 cores, 15,000 servers, and over 200PB of storage.

KIAF

CERN generates 20PB of experimental data every year. All of these data should be stored permanently and distributed to physicists around the world. To increase data accessibility of distant researchers, CERN sends raw data to Tier-1 data centers around the world. In case of CERN's ALICE experiment, KISTI(Korea Institute of Science and Technology Information) serves as a Tier-1 data center for Korean researchers who are involved in the ALICE experiment. They also provide KIAF (KISTI Analysis Facility) cloud computing environment for researchers. Domestic users utilize KIAF cloud infrastructure to analyze data created from LHC particle collisions. Figure 1 shows the KIAF system architecture. KIAF consist of UI(User Interface) host, network switch, NAS(Network Attached Storage), scheduling node and 30 compute nodes each with 72 CPUs and 400GB RAM. To increase the stability of system, network redundancy has been implemented using two network switches. The compute node to switch links are 10GigE and the links between compute node and NAS are 40GigE. The raw data for analysis are stored on NAS. Researchers can access the KIAF UI host using SSH protocol. Once logged onto the system, users can handle raw data in NAS through the XRootD server and also, they can submit jobs in UI hosts. Submitted jobs are transferred to the scheduling node and they are distributed to the compute node following scheduling policy. After that, available computing resources are allocated to each of the jobs. The number of computing resources allocated to users are depending on the number of active users and job priorities. When jobs have been successfully completed, the results of the analysis will be sent back to the users who submit the jobs.

Host detection & VM selection policy

To balance workloads of the hosts, first we have to check their current status. After the state of a host is detected, the next step is to select the VMs to be moved from that host to another one. Since the migration of the selected VM may result in performance degradation, an appropriate VM selection policy is required. There are several ways to gather information about the host's status and VM selection policies. We used the previously studied Threshold(Thr) host detection policy [14], and VM selection algorithms Random Choice(RC) [22], Maximum Correlation(MC), Minimal Migration Time(MMT) [23] to compare with our policy.

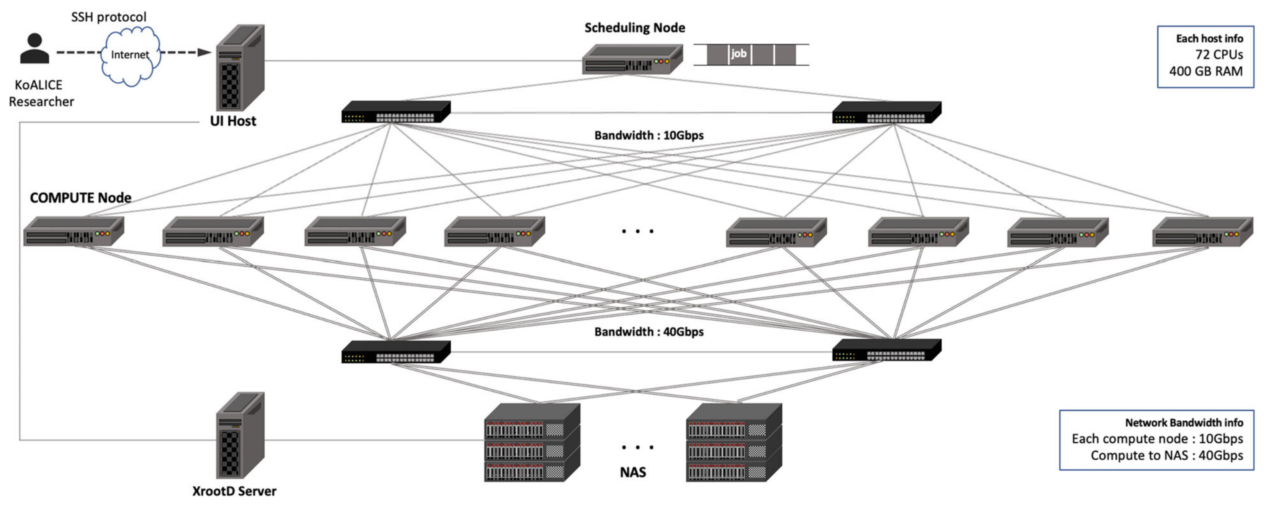

Fig. 1 KIAF System Architecture

Problem statements

As Stated in "Related works" section, most of the existing VM placement policies were based on CPU usage. However, depending on the nature of the cloud environment, the resources that were mainly used in the virtual machine were different. We focused on the data-intensive cloud environment and investigated the effect of disk usage on host performance.

A few studies have considered disk usage and among these studies, Jian Guo et al revealed that through their research, VMs with high disk usage caused host's performance degradation [19]. Firstly, they used CPI in OpenMPI as CPU-consuming benchmark and bonnie++ as disk I/O benchmark in their study. Using these tools, they placed two instances in one node and used a combination of the two benchmarks as a test set to measure the changing run-time as the benchmark was running.

According to the experimental results, the run-time increased when running the disk benchmark on two instances simultaneously. Based on this, they concluded that when disk-intensive jobs are operated in one host, they can increase the total amount of running time by up to 30% while CPU-intensive jobs in the host do not have a significant impact on the running time or other VM's performance. From this previous study, we have found a problem that high disk usage in the host can degrade the performance of the VMs in the host. Hence, we first checked the disk usage of the host to see if the identified problem also occurs in a real cloud environment.

To extract the host data of CPU, RAM, and Disk bandwidth usage from KIAF that has been actively being used for data analysis, we used *iostat*, which is a computer monitoring tool included in *sysstat* package in Linux. The disk bandwidth usage is different from the disk usage, however, we will be referring to the disk usage in the rest of this paper for convenience.

Figure 2 shows the host usage variation measured every minute over 12 hours. For 12 hours, many physicists submitted jobs to KIAF to analyze huge amounts of raw data. As shown in Fig. 2, we can see a much computing resource being used when the analysis was in progress. The blue, green, and red lines in the figure represent CPU usage, RAM usage, and disk usage respectively, while the yellow bar represents the number of jobs. Among them, we focused on disk usage, the red line in the figure.

We found that the CPU usage was relatively stable, but the variation in disk usage was significantly fluctuating for all periods of time. As noted in Fig. 2, disk usage was increased up to almost 100% irregularly.

In this case, the quality of service may deteriorate, and SLA violations are very likely to happen. Therefore, we have studied how to properly distribute the disk load as well as the CPU load to prevent host performance degradation.

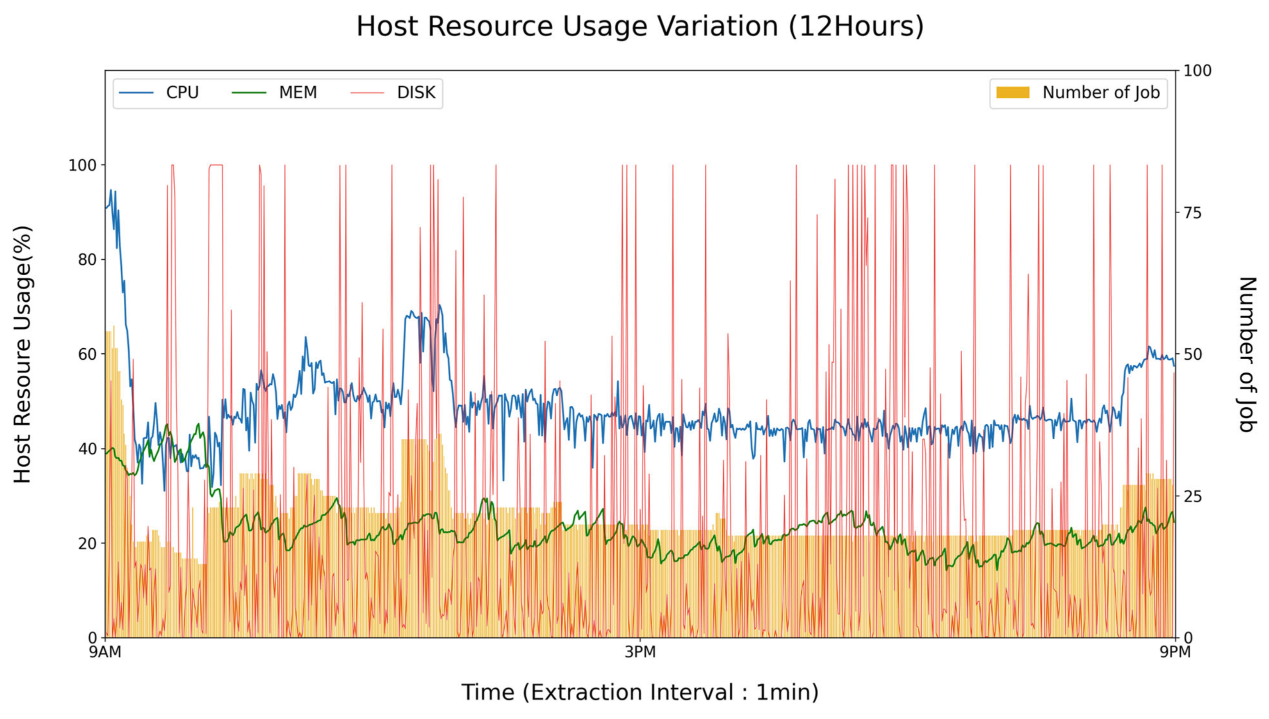

Fig. 2 Host Usage Variation over 12 hours

Proposed methods

Data-intensive job's characteristic

There were only limited studies that considers disk usage with real used cases in their policies. Moreover, resources in literature that focuses on data intensive jobs are inadequate. In this paper, we considered developing a more practical model that can be applied to the real cloud environment for data-intensive computing. However, it is important to note that finding a causal relationship between increased disk usage and other factors is a complex task. In order to simplify such a task without avoiding the missing underlying concept, we changed the host-based large-scale problem into a VM-based microscope level problem by measuring usage at 3 s interval for 10 min.

Figure 3 shows the host usage variation measured every 3 s over 10 min but more detailed than Fig. 2. We found that CPU usage and disk usage increased at the same time when jobs were submitted to the host. At this time, the changes in disk usage were greater than the changes in CPU usage. When jobs were finished, we were able to notice that the disk usage increased sharply. According to resource usage measuring shown in Fig. 3, one of the key characteristics of data-intensive jobs in KIAF is that increase in disk usage can be identified in the beginning and at the end of the job.

The data analysis process is as follows: A submitted job will import data that is necessary for the job and will record events that occur during data analysis. When all analysis is completed, the intermediate analysis results will be made into a single final result.

We were able to find a definite variation in resource usage on the host according to these sequences of the analysis process. The reason for the significant increase in disk usage at the beginning and the end of the job was the result of increased data read and write respectively. Based on the two graphs shown in Figs. 2 and 3, we were able to identify the resource usage pattern of the data-intensive jobs.

If a VM will be assigned to a host at the request of a new analysis work, disk usage in that host will increase sharply. For this reason, when new VMs will be assigned, they should not be placed into a single host in order to avoid overload problems. Therefore, VMs should be evenly or reasonably allocated to hosts in the cloud computing farm. For this purpose, such an important key property should be counted as part of the VM allocations or placements. The host performance degradation caused by an increase in disk usage can be prevented. In addition, since we can prevent the overloading host, it is possible to reduce the violation rate of SLA. Based on our observation of data-intensive job's characteristics, we designed a new VM placement strategy. We simulated our model using the

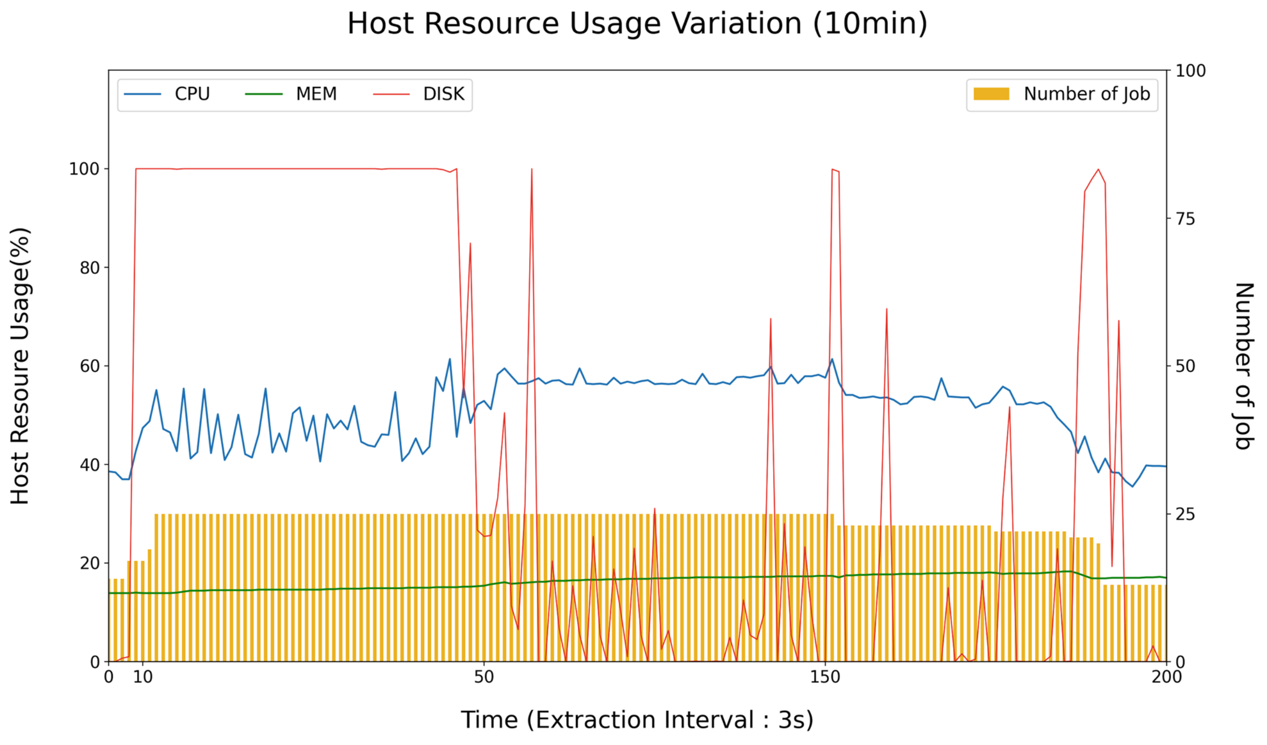

Fig. 3 Host Usage Variation over 10 min

lightweight version of CloudSim and compared the results to other VM placement policies.

Energy consumption model

As described earlier in this paper, there is a trade-off relationship between system operating costs and SLAV. Therefore, the cost comparison is necessary before discussing the effectiveness of our work. The energy cost needs to be calculated first because it accounts for most of the system operating costs. There was a basic power consumption model proposed by the previous works [24].

$$P_{host} = P_{cpu} + P_{mem} + P_{hdd} + P_{board} + P_{fan} \tag{1}$$

Equation 1 simply represents host's power as the sum of the power consumption of the components that consume power within the host. The number of components above may vary from host to host. Therefore, overall power consumption also depends on the number of components. We can organize Eq. 1 into more specific expression as shown in Eq. 2.

$$\begin{aligned} P^t_{host} &= \alpha \sum_{i=1}^{m} \left(P_{cpu_i} \times U^t_{cpu_i}\right) + \beta \sum_{j=1}^{n} \left(P_{mem_j} \times U^t_{mem_j}\right) \\ &\quad + \gamma \sum_{k=1}^{o} \left(P_{hdd_k} \times U^t_{hdd_k}\right) + \sum_{l=1}^{p} P_{fan_l} + P_{board} \end{aligned} \tag{2}$$

where:

t	= time
m, n, o, p	= Number of each parts
P_{part}	= Power consumption unit of the part
U^t_{part}	= Usage of the part at a point in a time t
α, β, γ	= Impact value on host's power consumption

Equation 2 is a power model that reflects the number of components in the host. P^t_{host} represents the electrical power consumption based on *host*'s resource usage measured at a specific point in time t. In the symbol P_{part}, the word *part* symbolizes the components of host such as CPU, Ram, Disk usage so on. For example, P_{cpu_i} means the power consumption unit of the i-th CPU. And $U^t_{cpu_i}$ represents the usage of the i-th CPU at a point in a specific time t. In the same way, the usage and the power unit of other components are also described in Eq. 2. For each component, the amount of power consumption used at that time was calculated and added together. At this time, the CPU, disk and memory usage have a great influence on power consumption. However, each computing resource has a different impact on the total power consumption of the host. Therefore, the magnitude of the impact of each resource on power consumption is described as α, β and γ values. These values are close to 1 if it significantly affects the overall power consumption of the host. On the contrary, when these values approach zero, it means that they have less impact on total power consumption. The total power consumption can be calculated by multiplying these values to the power consumption of each resource.

In addition to the power consumption model above, the power consumption should be additionally taken into account when the host reboots to calculate the total energy cost. The hosts with no jobs could be turned off in order to reduce power consumption while it should be powered on if none of the current hosts are in operation that is suitable to accept a VM. When the host reboots, it consumes more power than in the idle state. Therefore, the power consumption for the rebooting host needs to be measured differently from the one with the idle state.

Figure 4 shows the power utilization variation when the host reboots. As shown in the figure above, the power utilization increases irregularly until the host is in an idle state which is ready to receive jobs. We need to check the amount of energy consumed during the reboot, by definition the energy can be represented as an integration of power over time. Therefore, with the graph in Fig. 4, we can calculate the approximate amount of energy when the host reboots. The calculated result shows that the host consumes an average of 1.15 times more energy when it reboots. Therefore, the energy consumption for rebooting of the host is expressed as 1.15 times of the energy consumption for the host on the idle state as $E^{reboot}_{host} = 1.15 \times E^{idle}_{host}$. Based on this, the total energy cost spent on operating the system can be calculated as shown in Eq. 3:

$$\begin{aligned} C_{total} = C_{energy} \times \sum_{i=0}^{host} \Bigg(&\int_{t=0}^{OT} P^t_{host_i}\, dt \\ &+ E^{reboot}_{host_i} \times N^{reboot}_{host_i} \Bigg) \end{aligned} \tag{3}$$

where:

C_{energy}	= Energy cost
$N^{reboot}_{host_i}$	= Number of the $host_i$ reboots
$E^{reboot}_{host_i}$	= Energy consumption when the $host_i$ reboots

In Eq. 3, the operating time is expressed in OT and t means time. C_{total}, C_{energy} and $N^{reboot}_{host_i}$ refer to the total system operating cost, energy cost and number of host reboots respectively. Using the power consumption model obtained in Eq. 3 above, the energy consumption used by the host can be obtained by integrating the power consumption used every seconds. The amount of energy consumed by the operation can be obtained by summing up all the energy consumption of the active host. In addition, in order to calculate the amount of the energy generated by the rebooting of the host, the number of host reboots is multiplied by $E^{reboot}_{host_i}$. By multiplying the total amount of energy consumption by the energy cost, we can obtain

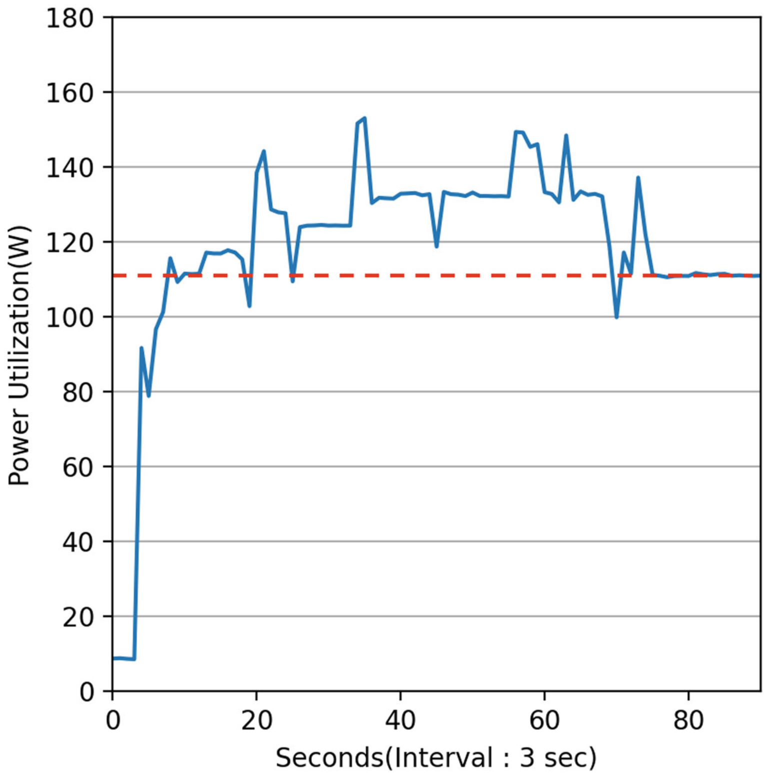

Fig. 4 Power Utilization Variation When the Host Reboots

the total cost of the operating system. The energy cost model shown above allows us to compare the calculated operating cost with the SLAV costs. This process makes it possible to judge the effectiveness of the VM placement policy.

MMEVMP VM placement algorithm

In this section, we will propose our VM placement strategy called **MMEVMP**(Min-Max Exclusive VM Placement). The basic concept of this algorithm comes from the characteristics of the data-intensive job that we described in Data-intensive Job's Characteristic.

Algorithm 1 represents the pseudo-code of the MMEVMP algorithm. What we have found from the previous analysis was that data-intensive jobs show a rapid increment in disk usage early and late in the job. If a new job is assigned to the cluster, it must be assigned to the appropriate host. At this time, the host to which the VM is recently assigned should be avoided if possible because all the jobs in KIAF is handled by the host's VM. Furthermore, the hosts with VMs that have been working for a long time should also be excluded from the selection due to its high possibility of increasing the disk usage. The host's `time` value represents the amount of time that has elapsed after the last job is assigned to the host. Therefore, the hosts with maximum or minimum `time` values were excluded from the selection of hosts.

In Algorithm 1, the MMEVMP function receives the newly assigned VM, allocation denied time value, and host list as input and returns the host where the VM will be placed. The first step is adding only those currently active hosts of which `flag` value is *Not_Allocated* to the `temp_list`. If all of the host's `flag` value is *Allocated*, our algorithm will try to find the hosts with VMs less than `Limitation`. The variable, `Limitation`, represents the limit on the number of VMs that can be received simultaneously. For example, if `Limitation` is 3, it means that even though the host has recently received a VM, if the current number of VMs does not exceed 3, it can receive more jobs. The value of `Limitation` can be determined according to the specification of the host in the cluster.

The next step is comparing the `time` value of the hosts in `temp_list` in order to find the average value, and adding hosts with similar with `time` value and `avg_time` value to the `available_group`. At this step, it is necessary to define the tolerance for the difference, which can be tuned according to the cloud environment. The tolerance is needed to widen the range of host selection reasonably. In this step, hosts whose `time` value is within a predetermined tolerance range are selected when compared to the `avg_time` value. For example, if the tolerance is ±10%, hosts with a `time` value that is within the difference between `avg_time` and ±10% are added to the `available_group`. Finally, it finds the hosts in the `available_group` to which the VM can be assigned and returns it as an output. If no suitable host is found, it will select and return one of the powered off hosts.

$$FLAG = \begin{cases} 1(Allocated), \text{ if Host's } \texttt{time} \leq ADT \\ 0(Not\_Allocated), \text{ if Host's } \texttt{time} > ADT \end{cases} \tag{4}$$

The `flag` value in the host indicates that the host has recently received a job. As defined in Eq. 4, the `flag` value is determined by comparing the time value with the `ADT` value, where `ADT` represents the allocation denied time. When an algorithm is called, the `ADT` value passed in will determine the time at which the job allocation is denied. The host's `time` value is initialized to zero when a new job is assigned, and the `flag` value is also set to zero. The larger the `ADT` value, the lower the host's job assignment, which may make fewer SLAV. However, by requiring more hosts for the operation, the operational cost of the host increases. Therefore, the optimal `ADT` values should be set by comparing the cost of SLAV with the system operating cost.

Experiment

Experiment environment

Previous studies [4, 11, 25, 26] used CloudSim [27] simulator to evaluate their works. It supports the modeling of virtualized environments, on-demand resource provisioning, and their management. CloudSim provides CPU usage data from planetlab, allowing users to compare various algorithms or policies in a virtual cloud environment. Since CloudSim is performed only based on CPU usage, it is inappropriate to apply disk usage data. There is a CloudSimEx simulator which is an extended version of CloudSim [28]. CloudSimEx is designed to take into account disk usage in the data center simulation process. However, since the disk usage data used by CloudSimEx is MIOPs and our disk usage data is percentages, the disk usage data that we extracted from KIAF are not compatible with CloudSimEx. Therefore, by referring to the structure of CloudSim, we have developed a lightweight simulator that can cover our CPU and DISK usage data in order to test the performance of our algorithm.

Experiment workflow

Figure 5 shows the flow chart of a lightweight version of CloudSim we developed. As shown in Fig. 5, the workflow can be roughly divided into an initialization phase and a simulation phase. In the simulation phase, input parameters are needed such as VM usage data, job scheduling queue, VM selection policy and VM placement policy. When initialization phase is executed, the system creates hosts and VMs. When a VM is created, usage data extracted from KIAF will be mapped to each VM and those VMs are assigned to hosts. At this time, only one VM will be assigned to each host, and then the status of the host will be checked in order to transfer the under loaded host's VM to another host. In the host optimization phase, VMs are concentrated on a few hosts as long as the host usage does not exceed the upper threshold through a greedy approach. Lastly a host without any VMs will be shut down to reduce power consumption. This ends the

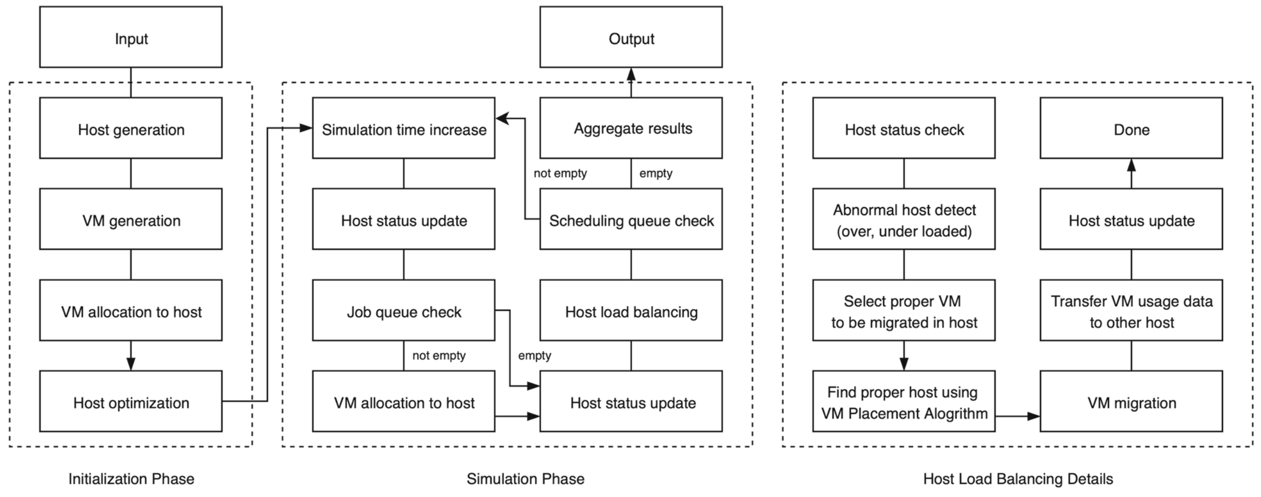

Fig. 5 Experiment Workflow Chart

Algorithm 1 `Min-Max Exclusive VM Placement`

```
Require: New VM V, Host List H, Allocation Denied Time ADT
Ensure: Select Host in H
  Host h in H → [ flag,                      ▷ Boolean type
                  status,                    ▷ String type
                  VMs,                       ▷ List type
                  time,                      ▷ Integer type
                  CPU_usage,                 ▷ Float type
                  Disk_usage]                ▷ Float type

  procedure MMEVMP
  for h in H : do
      if h.status ≠ Shutdown and h.flag ≠ Allocated then
          Append(temp_list, h)
      else if h.flag = Allocated and len(h.VMs) ≤ Limitation then
          Append(temp_list, h)
      end if
  end for
  if temp_list is Empty then
      return rebootHost()
  else
      avg_time ← findMedianTime(temp_list)
      for h in temp_list : do
          if h.time ≈ avg_time then          ▷ Within tolerance
              Append(Available_Host_list, h)
          end if
      end for
      if Available_Host_list : then
          for h in Available_Host_list do
              if h.CPU_usage + V.CPU_usage ≤ upper_threshold
                  and h.time ≤ ADT then
                  setHostValue(h)
                  return h
              end if
          end for
          return rebootHost()                ▷ No host is suitable in list
      else
          return rebootHost()                ▷ When list is empty
      end if
  end if
  procedure end


  sub procedure1 rebootHost()
    host ← Shutdown_host_list.pop()          ▷ Reboot Host
    setHostValue(host)
    return host
  sub procedure1 end


  sub procedure2 setHostValue(host)
    reset(host.time)                         ▷ Set time value to 0
    set(host.flag, Allocated)                ▷ Set flag value to 1
    set(host.status, Activated)
  sub procedure2 end
```

initialization phase, and the simulation is ready to be started.

When the simulation starts, the simulation time value is increased, and current host usage information can be updated by reflecting the corresponding time value then the simulator checks the job schedule queue. If there are jobs in the queue, it will allocate VMs as many as the number of jobs. After the VMs are assigned, the current status of the hosts will be checked. If any of them is underloaded or overloaded, the host load balancing will be activated to improve utilization or reduce overload. To alleviate host's status, a suitable VM will be selected according to the given VM selection policy in the host load balancing phase and the selected VM is moved to the designated host according to the VM placement policy. After the host load balancing, the simulator will check the job scheduling queue. The process will be repeated until there are no more jobs in the scheduling queue. At the end of all jobs, the simulation results such as total energy consumption, SLAV rate, etc. calculated according to the reference model will be printed.

Experiment scenario

In the experiment, scenarios are needed to prove whether the VM placement strategy we proposed is suitable for various situations in cloud. Depending on cloud system, there may be several kinds of job submission scenarios. Since we focused on the specific-purpose cloud called KIAF, we analyzed the patterns of job submitting in that system.

Figure 6 shows the monthly submitted and queued jobs in KIAF [29]. As shown in the figure above, the KIAF system is congested in a certain time but it is idle most of the time. The main reason why the jobs are concentrated in a certain time period is highly related with researcher's schedule. The total number of jobs are increased before the workshop or conference schedule due to the activities of the researchers become more active during this period. At this time, submitted jobs cannot be processed immediately because of the limit on computing resource and it enters the queued state. Since it is difficult to fundamentally solve the queued problem through scheduling method, a method such as dynamic allocation of external cloud resources is required. Therefore, we considered a way to increase the efficient resource utilization of the cloud in a more general situation, and accordingly, we applied a random job submission scenario to the simulation.

A total of 1,100 data extracted from KIAF will be used in the simulation. Each data includes CPU usage and disk usage and has up to 400 usage data records. By random submission scenario, 0 to 10 jobs will be assigned at each point in time.

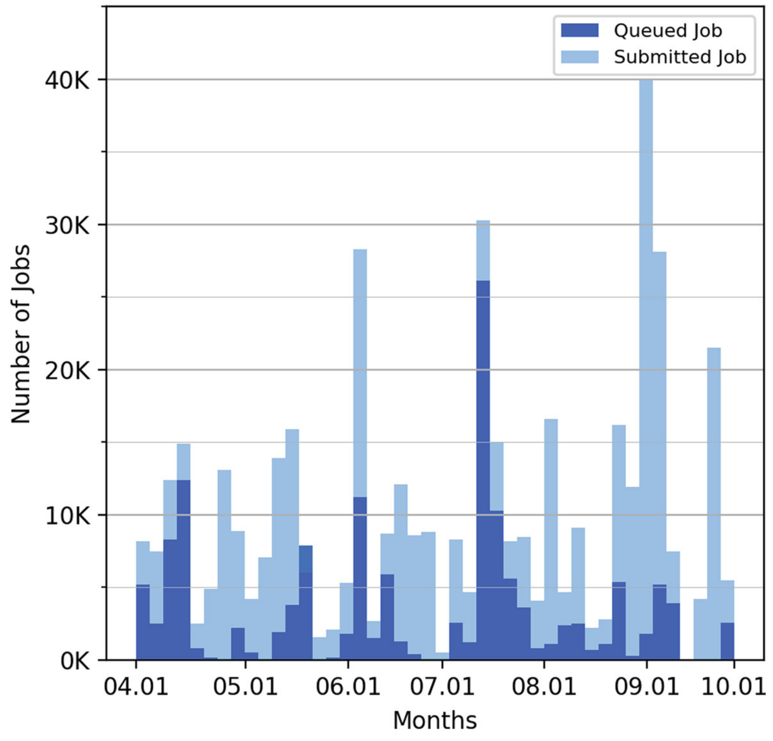

Fig. 6 Submitted and Queued Jobs in KIAF

Performance evaluation indicator

Among several indicators to evaluate the algorithm's performance, the following indicators are used to measure the performance of our proposed model [10, 30].

- CPU SLAV
- Disk SLAV
- Number of migrations
- Number of host reboots
- Maximum number of hosts
- Host activate time
- Energy consumption
- ESV (Energy Consumption × SLAV rate)

In order to show the effect of disk usage on the Host's performance, it was divided into two, the CPU and disk SLAV. Although SLAV is generally not distinguished by resource types like the above indicators, it was assumed that a SLAV occurred when the usage of these two resources on the host exceeded the threshold set by the host detection policy. The SLAV rate was calculated as in the following equation.

$$SLAV = \frac{\sum_{i=1}^{host}\left(\sum_{j=1}^{OT} R_{host_i}^{j} \ mod \ U_{thr}\right)}{Total\ Host\ Active\ Time} \tag{5}$$

Equation 5 is the process of finding the SLAV rate. $R_{host_i}^{j}$ refers to the total amount of resource required by $host_i$ at time j. It can be calculated by the sum of VM's resource requests in each host. And U_{thr} represents the upper threshold of the resource usage within the host. A SLAV occurred at the time when the amount of a required resource exceeds this threshold. In this case, SLAV is represented by one value. The average SLAV rate can be obtained by dividing the number of SLAV over the time when the entire host is activated.

The rest of the indicators are explained in the "Results and discussion" section.

Experimental method

To run the cloud computing infrastructure in a load balance, methods are used to measure the host's workload and select an appropriate VM selection policy. The threshold policy *Thr* which was introduced in the background of this study was used as a host detection policy. Random choice, maximum correlation and minimal migration time that were referred to as *Rc, Mu* and *Mmt* respectively were used as a VM selection policy.

Three VM placement policies were used to compare our strategy which are random choice, low CPU, and low CPU-Disk. The random choice is a method of randomly selecting a host to which a new VM is assigned. The low CPU method selects the host with the lowest CPU usage

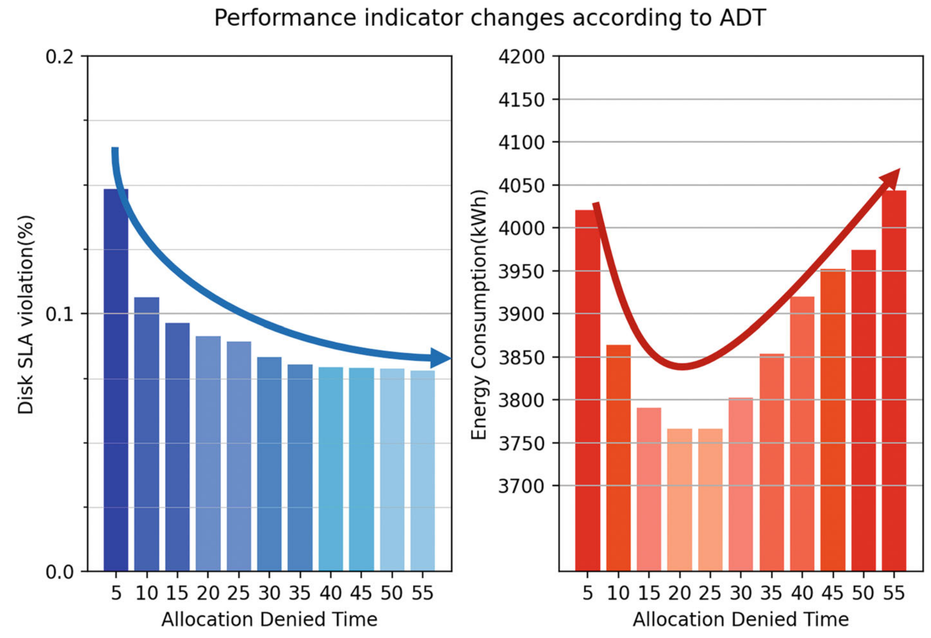

Fig. 7 Disk SLAV(left) and Energy consumption(right)

while the VM is allocated to the available hosts. The low CPU-Disk method opts for the host with the lowest disk usage from a set of hosts selected by the low CPU method. There is a difference in our algorithm, this algorithm does not take into account other factors and simply selects host with the least CPU and disk usage when the algorithm is called. On the other hand, our algorithm considers the characteristic of the job and selects a host that is less likely to cause disk SLAV in the future from a set of hosts with low CPU usage. We will evaluate the performance of our proposed strategy based on a total of 12 combined policies including MMEVMP presented in this paper. Each policy was evaluated 50 times and it gave average results.

Before proceeding with the simulation, it is necessary to set the ADT value and tolerance for the difference described in the Algorithm 1. In order to find the optimal ADT value, pre-simulation was conducted by increasing the ADT value. Figure 7 showed the energy consumption and disk SLAV rate. As ADT increases, the energy consumption increases as well while the disk SLAV rate decreases along with the ADT values. Therefore, we pondered that a trade-off relationship existed between the two indicators. In that sense, we also checked the ESV [14, 31] value to find an optimal ADT value.

As mentioned, the disk SLAV rate decreases as ADT increases. However, the amount of its reduction decreases gradually. On the other hand, as the energy consumption decreases, it will start an abrupt increase rapidly at a particular point. Therefore, to comprehensively evaluate the effects of the two indicators, an ESV alterations graph will be obtained. As shown in Fig. 8, the ESV value kept decreasing from the start of the simulation, but it started to increase when the ADT value is 35. Since ADT is greater than 35, the increase in energy consumption is greater than the decrease in the disk SLAV rate. In the case of KIAF data, the optimal ADT value was 35.

In the case of the tolerance for the difference, it was set to ±5%, and the process of obtaining it was obtained through pre-simulation similar to the above process. Even in environments different from KIAF, these optimal values can be obtained for each environment through the above process. Finally, simulation was performed to measure the performance of our proposed algorithm using the obtained optimal ADT value and tolerance for the difference.

Results and discussion

The test results will be discussed and presented in this section. Figure 9 shows the CPU and Disk SLAV rate according to the VM placement policies.

As shown in Fig. 9 above, these graphs indicated the maximum and minimum values range of the SLAV rate. The small square on top of each bar represents the average value. Figure 9a showed the CPU SLAV rate was effectively reduced when our proposed strategy was applied. Moreover, the results pointed out that the gap between the minimum and maximum values of CPU SLAV is evident that our proposed algorithm is narrow. This can be seen that our algorithm shows high reliability compared to other algorithms. In Fig. 9b, Low CPU usage

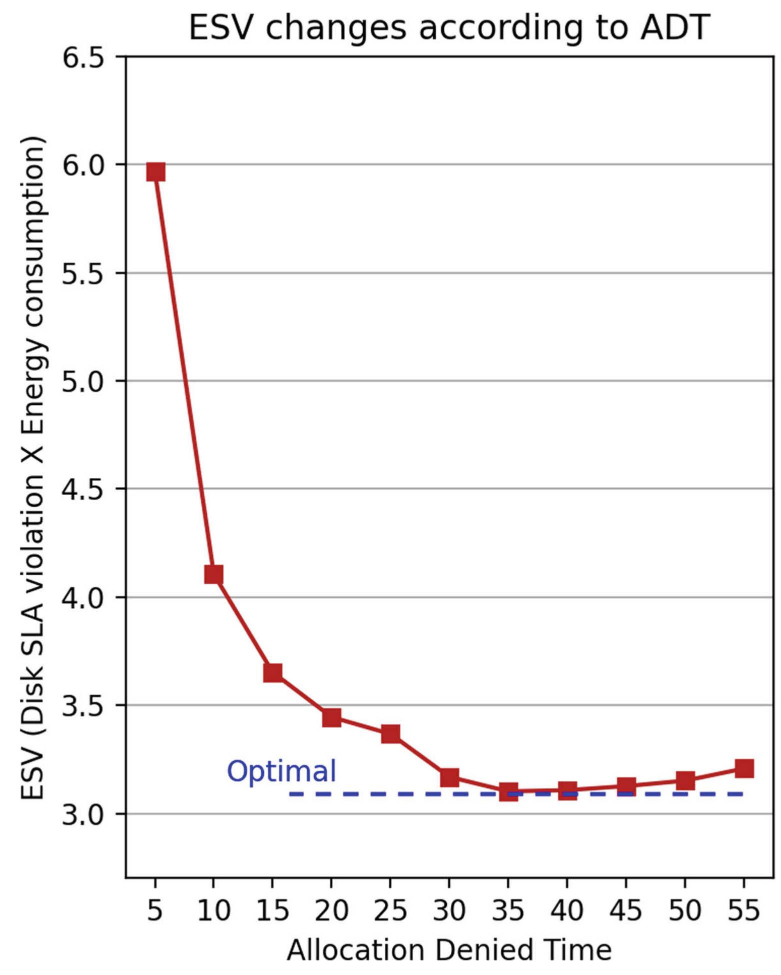

Fig. 8 ESV Alterations According to ADT

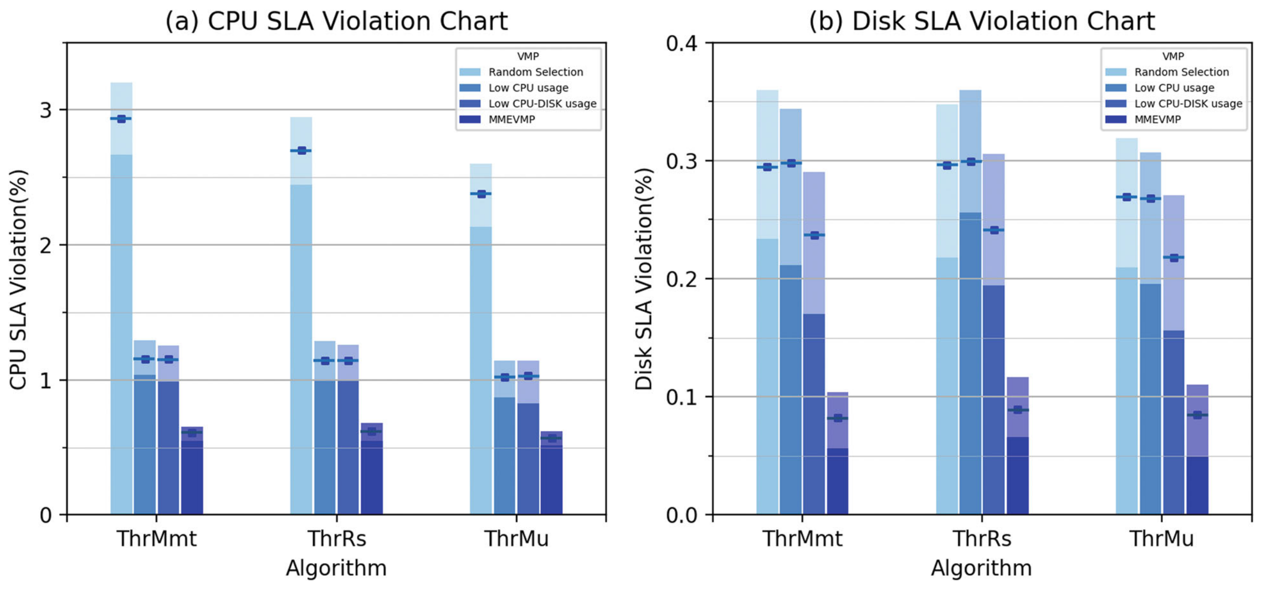

Fig. 9 SLAV Chart CPU SLAV(Left), Disk SLAV(Right)

algorithm shows a similar disk SLAV rate with the random choice. On the other hand, Low CPU-Disk usage algorithm showed a lower disk SLAV rate than Random Selection and Low-CPU algorithm. When our algorithm was applied, it showed a drastic reduction in the disk SLAV rate.

Figure 10 shows a visualized graph of performance evaluation indicators other than SLAV for each strategy. In Figure 10a, it showed that when our strategy was used, the number of migrations were also reduced. The small number of migrations is a good indicator of overall system performance. For the reason that if the number of migrations escalates, performance degradation is likely to occur [17]. However, as shown in Fig. 10b, it is important to note that the number of migrations diminished while the number of host reboots increased significantly for the fact that our algorithm requires more hosts than the others as seen on Fig. 10d. Figure 10c showed the total host activation time was also longer than other algorithms. From the results shown above, it was confirmed that our proposed algorithm has not only advantages but also disadvantages. Therefore, in order to evaluate the overall performance of the algorithm, we compared energy consumption for each strategy, which has the greatest effect on overall operating cost.

Figure 11 shows the simulation results for performance indicators other than SLAV. As a result of the increase in the host activation time and the number of host reboots, the energy consumption of our strategy is higher than that of other policies. To determine whether our strategy was

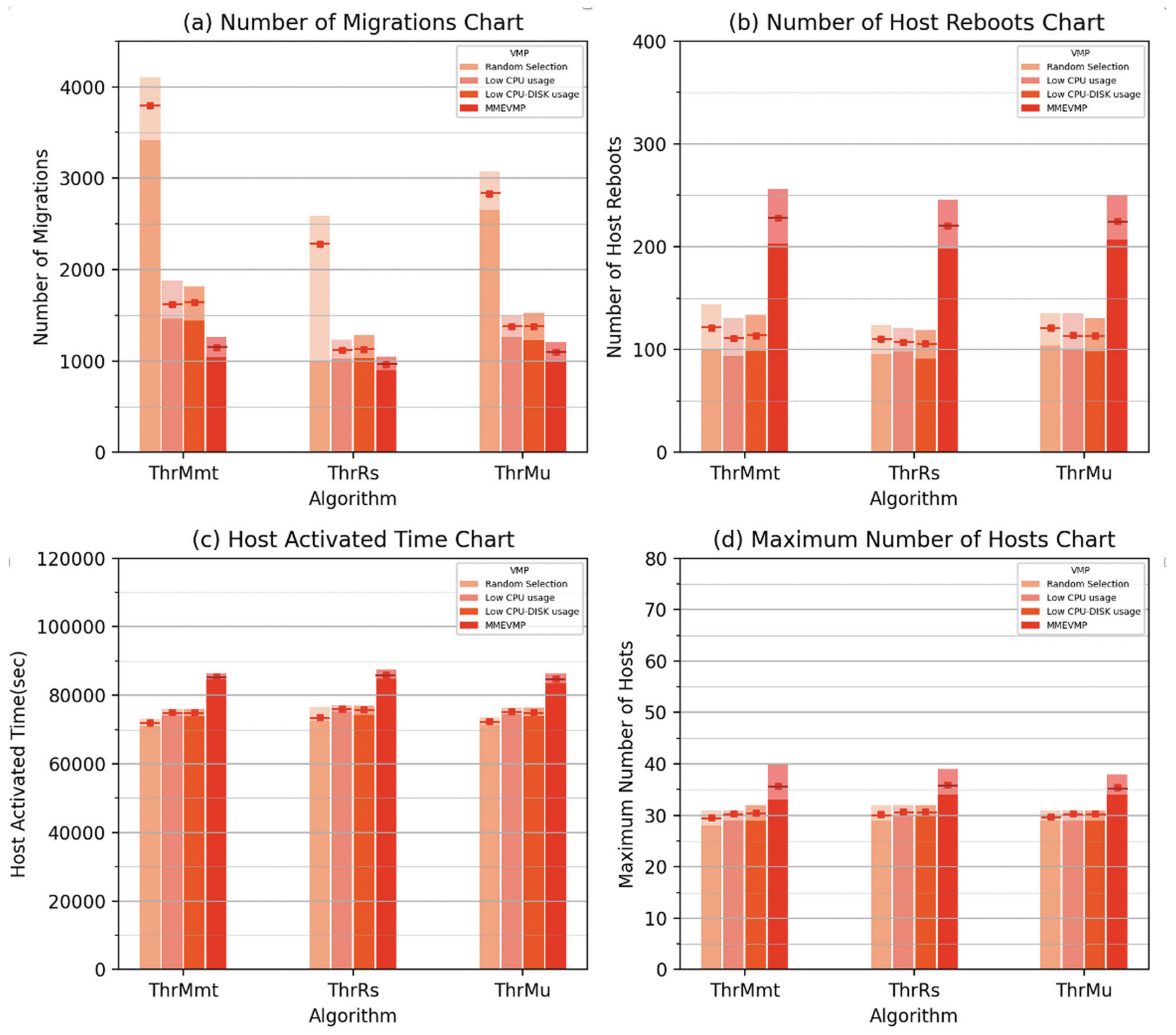

Fig. 10 Performance Indicator Chart

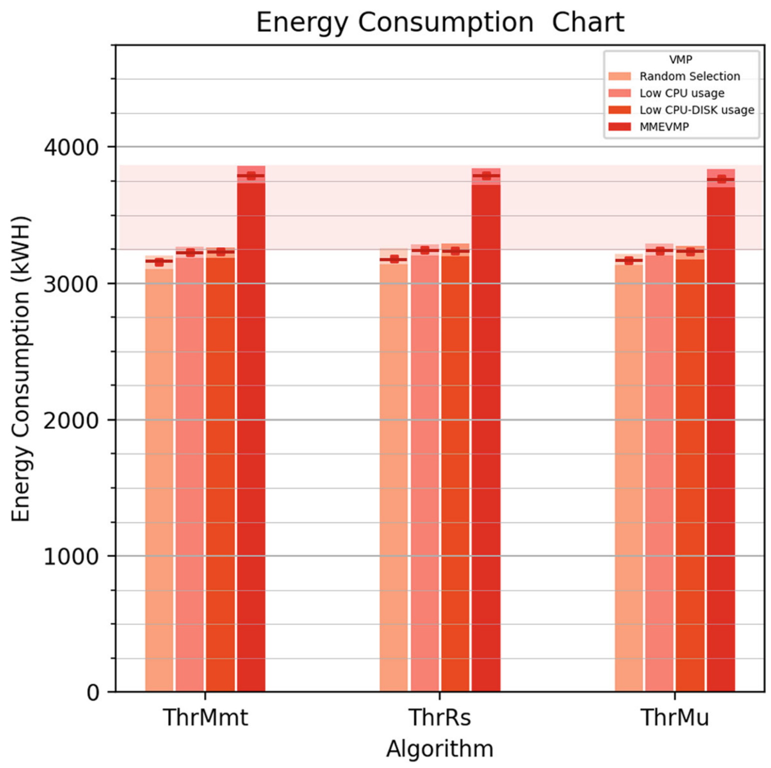

Fig. 11 Energy Consumption Chart. Y-axis refers power utilization per hour

efficient, we compared the energy consumption with the key indicators of SLAV.

For this purpose, we derived an ESV graph for the comparison of total SLAV and energy consumption. As shown in Figure 12, it appears that our strategy has a lower ESV than the other policies. It can be seen that our algorithm's SLAV reduction effect is greater than the increase in energy consumption. Therefore, it was confirmed that our proposed strategy is still efficient despite its shortcomings.

However, estimating the SLAV cost evenly is a difficult task because the cost of violations varies according to the level of the SLA. Energy costs may also vary depending on the actual system operating environment. Therefore, in each situation, the efficiency of the strategy in each situation can be judged by comparing the cost savings to energy consumption savings in terms of the reduction of SLAV. If there is an overall cost savings effect when using our algorithm, we can say that our strategy is an effective model rather than other existing policies.

Conclusion & future work

Cloud computing is emerging and widely adopted in data-intensive research fields. Nowadays, the big data analysis environment is moved to the cloud computing infrastructure. In such an infrastructure, the VM placement policy is important in order to utilize the entire computing resource as well as to avoid unexpected SLA violations.

Most of the existing VM policies for selecting a host to place into the VM only considers the CPU usage of the host not the disk usage of each job. In this paper, we proposed a VM placement strategy considering disk usage as well as CPU usage in operating the cloud system. After actively analyzing the characteristics of data intensive jobs used in a real working system called KIAF, we introduced the algorithm for placing VMs reflecting those characteristics identified. In order to evaluate our model, we have developed a lightweight version of CloudSim and derived 8 performance indicators.

Our proposed strategy showed a considerably lower SLAV rate, although energy consumption is relatively higher than the other strategies. In this case, it is more effective if the reduced cost savings of the SLAV rate is greater than the overall operation cost due to the increase in energy consumption. When we checked the ESV considering the two indicators together, we found that our strategy is more effective than other existing strategies. Therefore, our strategy can support data-intensive analysis in the cloud to effectively use computing resources by preventing SLA violations directly related to operating costs.

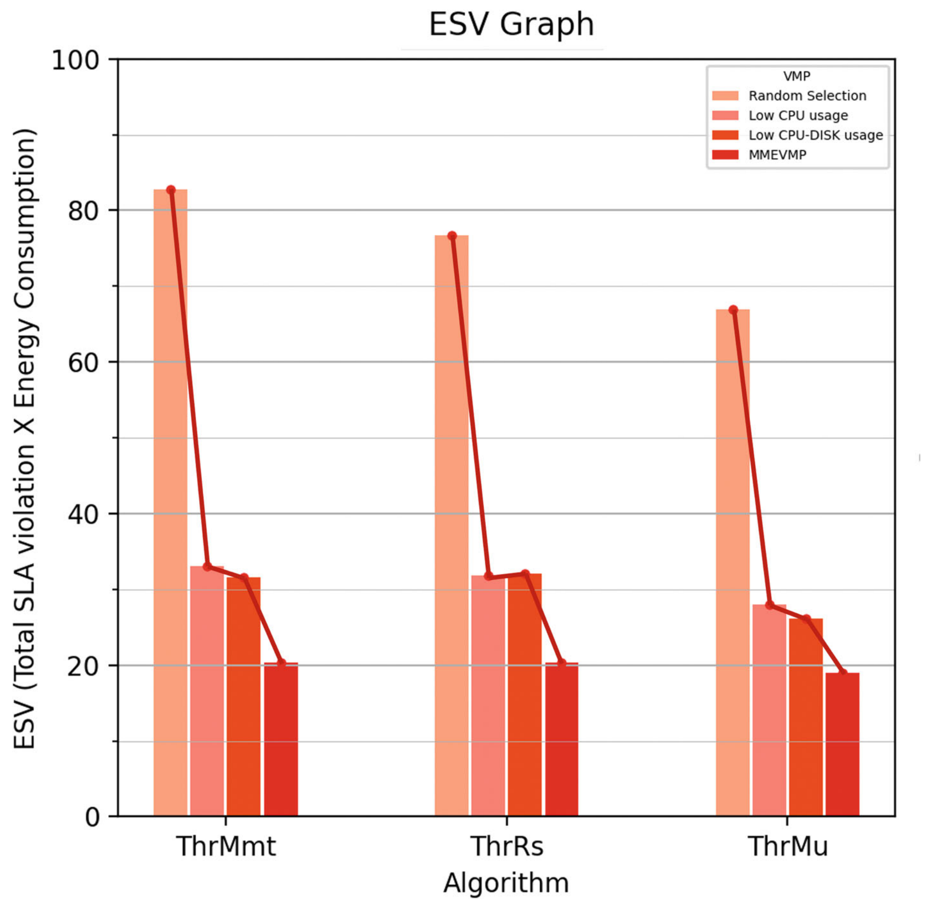

Fig. 12 ESV Chart

However, since our strategy was designed to focus on a specific environment, there may be limitations when calculating ADT values or tolerances and it may not be suitable for all cloud environments. Nevertheless, the amount of data analysis job is on the rise, and in an environment where such data-intensive job is dominating, our proposed strategy will be sufficiently effective.

Our future work is to compare our algorithm with other modern algorithms. We focused on preventing host performance degradation, but some of the previous studies focused on efficiently improving disk usage. Therefore, we will develop an extended version of the current algorithm to increase disk usage while avoiding host performance degradation. We will analyze the advantages of the algorithm proposed in this paper compared to other latest algorithms.

Acknowledgements
The authors would like to extend our sincere thanks to Global Science experimental Data hub Center at Korea Institute of Science Technology Information for their supporting our research and providing the experimental environment.

Authors' contributions
MK and *SN* designed the model. *MK* wrote the first draft of the manuscript and developed the algorithm. *MK* and *JL* carried out the experiment. *JL*, *TK* and *RA* contributed to mathematical deployment. *SN* supervised this study. All authors contributed to the final version of the manuscript and data analysis. The author(s) read and approved the final manuscript.

Authors' information
Moon-Hyun Kim and Jun-Young Lee are master students in Department of Computer Science, Chungbuk National University, Cheongju, Korea, under the advising of Seo-Young Noh.
Syed Asif Raza Shah is an assistant professor of Computer Science Department at Sukkur IBA University, Pakistan. He has earned his PhD in Computer Science from University of Science and Technology (UST), South Korea.
Tae-Hyung Kim received the ME in Computer Science from New York University and the PhD degree in Computer Science from Iowa State University. He is currently a principal software engineer at the Samsung Research, Samsung Electronics. His research interests include software architecture, software engineering and next-generation computing.
Seo-Young Noh received the BE and ME degrees in Computer Engineering from Chungbuk National University in Korea, and the MS and PhD degrees in Computer Science from Iowa State University, respectively. He is an Assistant Professor with the Department of Computer Science at Chungbuk National University, Cheongju, Korea.

Funding
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.NRF-2008 -00458).

Availability of data and materials
There is no supporting data available.

Competing interests
There are no competing interests involving the author in the execution of this research work.

Author details
[1]Department of Computer Science, Chungbuk National University, Chungdae-ro, Cheongju, Republic of Korea. [2]Department of Computer Science & CRAIB, Sukkur Institute of Business Administration University (SIBAU), Airport Road, Sukkur, Pakistan. [3]Samsung Electronics, Seoul, Republic of Korea.

Received: 14 August 2020 Accepted: 14 December 2020
Published online: 11 January 2021

References

1. Stephanie Overby LG, Paul LG What is an SLA? Best practices for service-level agreements – CIO. https://www.cio.com/article/2438284/outsourcing-sla-definitions-and-solutions.html. Accessed 07 June 2020
2. Fan X, Weber W-D, Barroso LA (2007) Power provisioning for a warehouse-sized computer. ACM SIGARCH Comput Archit News 35(2):13–23
3. Kusic D, Kephart JO, Hanson JE, Kandasamy N, Jiang G (2009) Power and performance management of virtualized computing environments via lookahead control. Clust Comput 12(1):1–15
4. Sayeedkhan PN, Balaji S (2014) Virtual machine placement based on disk i/o load in cloud. Int J Comput Sci Inf Technol 5(4):5477–5479
5. Moreno IS, Garraghan P, Townend P, Xu J (2014) Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. IEEE Trans Cloud Comput 2(2):208–221
6. Qiu F, Zhang B, Guo J (2016) A deep learning approach for VM workload prediction in the cloud. In: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). IEEE, Shanghai. pp 319–324. https://doi.org/10.1109/SNPD.2016.7515919
7. Gahlawat M, Sharma P (2016) Support vector machine-based model forhost overload detection in clouds. In: Satapathy SC, Joshi A, Modi N, Pathak N (eds). Proceedings of International Conference on ICTfor Sustainable Development. Springer, Singapore. pp 369–376
8. Gulenko A, Wallschläger M, Schmidt F, Kao O, Liu F (2016) Evaluating machine learning algorithms for anomaly detection in clouds. In: 2016 IEEE International Conference on Big Data (Big Data). IEEE, Washington, DC. pp 2716–2721. https://doi.org/10.1109/BigData.2016.7840917
9. Lu S-L, Chen J-H (2018) Host overloading detection based on EWMA algorithm in cloud computing environment. In: 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE). IEEE, Xi'an. pp 274–279. https://doi.org/10.1109/ICEBE.2018.00052
10. Melhem SB, Agarwal A, Goel N, Zaman M (2017) Markov prediction model for host load detection and VM placement in live migration. IEEE Access 6:7190–7205
11. Chen C, He K, Deng D (2016) Optimization of the overload detection algorithm for virtual machine consolidation. In: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE, Beijing. pp 207–210. https://doi.org/10.1109/ICSESS.2016.7883050
12. Pires FL, Barìan B (2015) Virtual machine placement literature review. arXiv preprint arXiv:1506.01509 CoRRabs/1506.01509. 1506.01509
13. Dhanoa IS, Khurmi SS (2015) Analyzing energy consumption during VM live migration. In: International Conference on Computing, Communication & Automation. IEEE, Noida. pp 584–588. https://doi.org/10.1109/CCAA.2015.7148475
14. 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(13):1397–1420
15. Zhao H, Zheng Q, Zhang W, Chen Y, Huang Y (2015) Virtual machine placement based on the VM performance models in cloud. In: 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC). IEEE, Nanjing. pp 1–8. https://doi.org/10.1109/PCCC.2015.7410296
16. Chaurasia N, Tapaswi S, Dhar J (2016) An over-utilization avoidance host selection scheme for affording workload of migrated VM. In: 2016 SAI Computing Conference (SAI). IEEE, London. pp 553–556. https://doi.org/10.1109/SAI.2016.7556034
17. Qaiser H, Shu G (2018) Efficient VM selection heuristics for dynamic VM consolidation in cloud datacenters. In: 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). IEEE, Melbourne. pp 832–839. https://doi.org/10.1109/BDCloud.2018.00124
18. Dayarathna M, Wen Y, Fan R (2015) Data center energy consumption modeling: A survey. IEEE Commun Surv Tutorials 18(1):732–794
19. Guo J, Zhu Z-M, Zhou X-M, Zhang G-X (2012) An instances placement algorithm based on disk i/o load for big data in private cloud. In: 2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP). IEEE, Chengdu. pp 287–290. https://doi.org/10.1109/ICWAMTIP.2012.6413495
20. Shabeera T, Kumar SM, Salam SM, Krishnan KM (2017) Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm. Eng Sci Technol Int J 20(2):616–628
21. Our Mission –CERN. https://home.cern/about/who-we-are/our-mission. Accessed 7 June 2020
22. Beloglazov A, Buyya R (2010) Energy efficient allocation of virtual machines in cloud data centers. In: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. IEEE, Melbourne. pp 577–578. https://doi.org/10.1109/CCGRID.2010.45
23. Beloglazov A (2013) Energy-efficient management of virtual machines in data centers for cloud computing. PhD thesis
24. Warkozek G, Drayer E, Debusschere V, Bacha S (2012) A new approach to model energy consumption of servers in data centers. In: 2012 IEEE International Conference on Industrial Technology. IEEE, Athens. pp 211–216. https://doi.org/10.1109/ICIT.2012.6209940
25. Chowdhury MR, Mahmud MR, Rahman RM (2015) Implementation and performance analysis of various VM placement strategies in cloudsim. J Cloud Comput 4(1):20
26. Benali R, Teyeb H, Balma A, Tata S, Hadj-Alouane NB (2016) Evaluation of traffic-aware VM placement policies in distributed cloud using cloudsim. In: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). IEEE, Paris. pp 95–100. https://doi.org/10.1109/WETICE.2016.29
27. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Experience 41(1):23–50
28. Louis B, Mitra K, Saguna S, Åhlund C (2015) Cloudsimdisk: Energy-aware storage simulation in cloudsim. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC). IEEE, Limassol. pp 11–15. https://doi.org/10.1109/UCC.2015.15
29. Kong B, Ryu G, Bae S, Noh S-Y, Yoon H (2020) An efficient approach to consolidating job schedulers in traditional independent scientific workflows. Appl Sci 10(4):1455
30. Daraghmeh M, Melhem SB, Agarwal A, Goel N, Zaman M (2018) Linear and logistic regression based monitoring for resource management in cloud networks. In: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, Barcelona. pp 259–266. https://doi.org/10.1109/FiCloud.2018.00045
31. Ibrahim A, Noshy M, Ali HA, Badawy M (2020) Papso: A power-aware VM placement technique based on particle swarm optimization. IEEE Access 8:81747–81764

Publisher's Note