Computer Communications 197 (2023) 255–266

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

An Energy-optimized Embedded load balancing using DVFS computing in Cloud Data centers

Amir Javadpour [a,b], Arun Kumar Sangaiah [d,*], Pedro Pinto [b,c], Forough Ja'fari [e], Weizhe Zhang [a,*], Ali Majed Hossein Abadi [f], HamidReza Ahmadi [g]

[a] Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China
[b] Department of Electrotechnics and Telecommunications, Instituto Politécnico de Viana do Castelo, Viana do Castelo, Portugal
[c] Universidade da Maia and INESC TEC, Maia and Porto, Portugal
[d] International Graduate School of Artificial Intelligence, National Yunlin University of Science and Technology, Yunlin, Taiwan
[e] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[f] Department of Computer Science, University of Isfahan, Isfahan, Iran
[g] Department of Network Science and Technology, Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran

ARTICLE INFO

Keywords:
Cloud computing
Load balancing scheduling
DVFS
Big datacenter
SLA
Power consumption

ABSTRACT

Task scheduling is a significant challenge in the cloud environment as it affects the network's performance regarding the workload of the cloud machines. It also directly impacts the consumed energy, therefore the profit of the cloud provider. This paper proposed an algorithm that prioritizes the tasks regarding their execution deadline. We also categorize the physical machines considering their configuration status. Henceforth, the proposed method assigns the jobs to the physical machines with the same priority class close to the user. Furthermore, we reduce the consumed energy of the machines processing the low-priority tasks using the DVFS method. The proposed method migrates the jobs to maintain the workload balance, or if the machines' class changed according to their scores. We have evaluated and validated the proposed method in the CloudSim library. The simulation results demonstrate that the proposed method optimized energy consumption by 12% and power consumption by 20%.

1. Introduction

Cloud computing, an emergent technology, assists grid computing in further development and absorbs a large portion of individual or company users. The cloud refers to a separate and distinct environment designed for remote monitoring and access to scalable and measurable resources in information technology. In this definition, the source could be software such as applications or hardware infrastructures such as storage devices and networks [1,2]. Cloud computing is a model that virtualizes computing resources such as network infrastructure, storage hardware, processing units, and applications in a secure environment, using a network such as the Internet for the users. Virtualization allows multiple users to use the resources shared on the network simultaneously by creating virtual machines for the users on the servers. Virtualization also may separate the VMs to automatically configure each VM according to a user's requirements across multiple hosts. The users pay for the resources they use, called the pay-as-you-go model [3,4]. In return, they expect the desired performance while using the cloud resources. The cloud provider and the user agree upon the quality of the services in the cloud, written on a contract called service-level agreement (SLA). The cloud provider has to pay a fine if the quality of the provided services violates the SLA [5,6]. Hence, proposing the algorithms to improve the performance of the cloud environment and the quality of the provided services is essential.

Balancing the workload among the servers is a technique that improves the performance of the cloud. Some servers may be under high utilization due to heavy processing tasks or countless processes. In contrast, some other servers could be idle at the same time. In this situation, the efficiency of the network downgrades while the idle servers waste the energy of the network. Moreover, the requests sent to an overloaded server must wait until the busy server finishes the current in-progress tasks. This delay in responding to the jobs is because a server can simultaneously handle a limited number of requests. Hence, the service availability downgrades, leading to dissatisfaction [7,8]. The service availability means that the user can access the desired service whenever needed [1]. In this way, balancing the load can optimize energy consumption and increase performance. Transferring the VMs from an overloaded host to an idle one without interruption is a technique that balances the workload among the servers leading

* Corresponding authors.
E-mail addresses: a_javadpour@e.gzhu.edu.cn (A. Javadpour), aksangaiah@ieee.org (A.K. Sangaiah), azadeh.mth@gmail.com (F. Ja'fari), wzzhang@hit.edu.cn (W. Zhang).

https://doi.org/10.1016/j.comcom.2022.10.019
Received 1 August 2022; Received in revised form 6 October 2022; Accepted 22 October 2022
Available online 2 November 2022

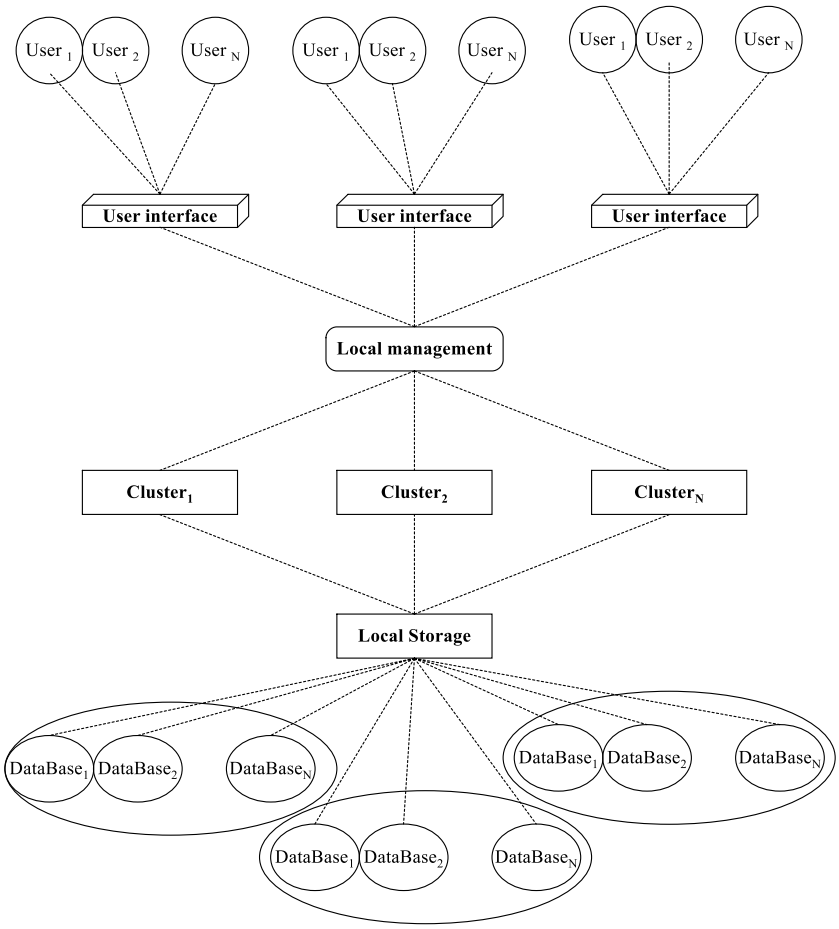

Fig. 1. The general architecture of workload balancing techniques.

to improved network efficiency [9–11]. The general architecture of the workload balancing techniques is shown in Fig. 1.

Along the side of the network performance that is crucial for a cloud environment, the profit of the cloud provider matters. A high-performance but unprofitable network for the cloud provider has no reason to exist. A feature of each network that directly affects the expenditures is energy consumption. A cloud environment consists of data centers. A data center includes secure and protected buildings to maintain equipment and infrastructure for the cloud network, uninterruptible power supplies, robust cooling systems, and a secure communication network [3,9]. Providing energy is a crucial feature to keep a data center up and running. Besides the costs and expenses, the side effects of the consumption of fossil fuels to supply such energy are an environmental concern. We can dynamically control the energy consumption using Dynamic Power Management (DPM) techniques, which we divide into two main methods, Dynamic Component Deactivation (DCD) and Dynamic Performance Scaling (DPS) [12].

The usage of the DCD method is mainly in a system with components that do not support the performance scaling, so we can only bring the elements into active or inactive modes. This transition results in a delay, making this method improper for real-time applications. It also imposes the power overhead to the network, so if the idle period is not long enough, the DCD method negatively impacts the performance [13]. In contrast, we use the DPS method when the system's performance can vary. The total consumed power in an integrated circuit (IC) is the sum of the static and dynamic power consumption as presented in Eq. (1).

$$P = P_{static} + P_{dynamic} \tag{1}$$

The static power consumption relies on the area of the IC. Eq. (2) demonstrates the dynamic power consumption [14].

$$P_{dynamic} = \beta.C_L.V_{cc}^2.f \tag{2}$$

In Eq. (2) the β is the activity factor, C_L represents the load capacitance, the V_{cc} denotes the supply voltage, and the f is the clock frequency at which the device operates. We infer from Eq. (2) that we can reduce the dynamic power consumption by decreasing the device's supply voltage or clock frequency. This input parameter variation introduces the Dynamic Voltage and Frequency Scaling (DVFS) method, a sub-method of the DPS technique. Reducing the supply voltage is expected as it has a quadratic relationship with power consumption (i.e. $P_{dynamic} \propto V_{cc}$). Moreover, reducing the clock frequency increases the execution time and downgrades the network's performance.

Eq. (3) demonstrates the consumed energy in the system where the E_{static} is the static energy consumption for keeping a server up and running without processing tasks [12].

$$E = E_{static} + E_{dynamic} \tag{3}$$

$E_{dynamic}$ is the dynamic energy consumption, and we calculate it in a period t_p using Eq. (4).

$$E_{dynamic} = \sum_{t_p} P_{dynamic}.t_p \tag{4}$$

Considering Eqs. (3) and (4), suppose we have to assign two jobs, $task_1$ and $task_2$ having two Physical Machines (PMs), namely PM_1 and PM_2. Now assume related parameters as following:

- The dynamic power of the processor for both PMs is P_{cpu}.
- The static energy consumption of PM_1 is P_{cpu}, so $E_s(PM_1) = P_{cpu}$.

- The static energy consumption of PM_2 is $0.7P_{cpu}$, so $E_s(PM_2) = 0.7P_{cpu}$.
- The execution time of the tasks, $task_1$ and $task_2$, is equal to t_1 and t_2, respectively.

There are four situations for task assignment and related consumed energy:

(1) $task_1$ to PM_1 and $task_2$ to PM_2:

$$\begin{aligned} &E = E(PM_1) + E(PM_2) = \\ &[E_s(PM_1) + E_d(PM_1)] + [E_s(PM_2) + E_d(PM_1)] = \\ &[(t_1.P_{cpu}) + (t_1.P_{cpu})] + [(t_2.P_{cpu}) + (0.7t_2.P_{cpu})] \\ &= 2t_1.P_{cpu} + 1.7t_2.P_{cpu} \end{aligned} \tag{5}$$

(2) $task_1$ to PM_2 and $task_2$ to PM_1:

$$E = 2t_2.P_{cpu} + 1.7t_1.P_{cpu} \tag{6}$$

(3) $task_1$ and $task_2$ to PM_1 and bring PM_2 into standby mode:

$$E = 2t_2.P_{cpu} + 2t_1.P_{cpu} = 2(t_1 + t_2).P_{cpu} \tag{7}$$

(4) $task_1$ and $task_2$ to PM_2 and bring PM_1 into standby mode:

$$E = 1.7t_2.P_{cpu} + 1.7t_1.P_{cpu} = 1.7(t_1 + t_2).P_{cpu} \tag{8}$$

In the simplest scenario, if we assume $t_1 = t_2$, the lowest energy consumption occurs at situation four, where we assign both tasks to PM_2 and bring PM_1 into standby mode. This reduction in energy consumption is due to powering down a server leading to eliminating static energy consumption of that server.

This paper proposes a task scheduling algorithm for cloud environment networks that dynamically controls the consumed energy using the DVFS method. In the second part of this paper, we have briefly represented the most recent studies related to our proposed method. The third section consists of the simulation and the evaluation results, where we have compared our algorithm to the most efficient ones to figure out the performance of our method. In the 4th part of this study, we have summarized an overall view of this paper. And the last portion of this paper introduces future works to optimize the performance of the proposed method.

2. Literature review

Nowadays, modern applications, which have commercial or scientific applications, need a high-performance computing infrastructure. This leads to high-scale computing data centers that have high electric power consumption. Despite improvements and reformations in hardware energy efficiency, the overall energy consumption of the data center is increasing due to the increasing demand for computing resources day by day. Apart from the staggering applied costs, building a data center that is designed to respond to the peak load or workload of the request, while low average utilization is higher than other power consumption in data centers.

In addition, the shortage of cooling systems or their failure can cause severe heating of the data center resources to lower the reliability of the system and reduce the lifespan of the devices and equipment inside the center. Moreover, the high-power consumption infrastructure causes a significant emission of carbon dioxide gas to the greenhouse effects to improve the energy consumption efficiency of several methods or decomposition so far. These methods are as follows: By utilizing virtualization technology on a physical server, the application algorithms can be improved by creating multiple virtual machines that have efficient energy consumption (low and improved). This reduces resource utilization costs and the value of hardware. The cloud offers customers resources based on demand due to the use of virtualization technology. The following reasons explain why cloud computing reduces energy consumption:

- Improving the efficiency of resources
- Lack of local-virtual machines dependency can be handled and implemented where energy is less expensive
- Less or more usage of resources — use of resources can be either low or higher on the basis of the current requirements (the number of resources needed at present).

Among the studies, Beloglazov et al. [13] looked into reducing each data center's power consumption by virtualizing the data. Their proposal provides a quality of service while reducing operational costs in a cloud data center. By continuously controlling virtual machines, we can reduce power consumption. In this project, the primary tool was live transmission, which allowed virtual machines to be transferred between physical hosts over a network. VMS providers can therefore utilize dynamic resource allocation in cloud environments. Resources can be dynamically allocated and on a priority basis, resource allocation policies, and the required resources to re-allocate (their resources are low or high). Aujla and Kumar [15] presented an architecture for energy management in virtual data centers where resource management takes place both local and global cases. At the local level, the system uses guest operating systems to manage the power management strategies, in other words, the same strategy that would be used if the operating system was implemented on a physical machine. The global level is used to allocate virtual machines, general policies (globally) that benefit from living transfer technology.

Barik et al. [16] have explored how to allocate resources efficiently in virtual environments by identifying the asymmetrical properties. The Ghosh method uses maximum and minimum parameters and shares value to identify the maximum and minimum resources that can be shared, as well as the relative number of resources each virtual machine can consume. This method is appropriate only in environments (enterprise clouds) in which service level agreements are unnecessary, no service level agreements are needed, and no application priorities are known. The dynamic ant colony optimization concentration, proposed by Sangeetha et al. [17], focuses on dynamic environments where virtual machines are required to be inserted or moved or removed. In laboratory and simulation, they have carried out research in this field which has been done in different environments using different controllers and uploads have been performed to clarify the facts which determine which control strategies will lead to the optimal energy consumption mode. The result was that, with the combination of controllers and periodic mobility, virtual machines can be used to obtain the best state of energy consumption according to the utilization. The proposed approach does not take heat distribution into account. The network structure and cooling system of a physical machine are examined. In this framework, we provide a probabilistic and novel method to optimization problems: mapping virtual machines to a collection of physical machines under this restriction, which necessitates the utilization of numerous of our resources. This approach claims to reduce basic energy usage by up to 69.2%. This has limited number of nodes, as well as the hardware component, for example, it did not considered the processor frequency scale . In the presented method by Gupta and Namasudra [18] first, the principle used for cloud computing live migration is stated with suitable energy utilization and a framework for architecture is defined considering this insight structure presented in this paper, research challenges have been proposed, and at the end the provision of required resources and the cloud computing environment placement has been developed to optimize energy consumption. It also helps the data center facility to provide the lowest energy consumption and, in another way, match the customers ' goals. In this paper, an assessment of optimal energy consumption takes place, and the following is presented:

- Principles and procedures for optimizing energy consumption in cloud computing architecture
- Algorithms of scheduling and resource placement for optimal energy consumption, plus sharing policies

- By using simulation in Cloudsim, a set of challenges is investigated resulting in finding the most beneficial solution for each resource and customer.

Khan [19] have presented an dynamic VM consolidation on cloud platforms that is configurable based on the specifications of the target data center and makes it optimal for energy consumption. This algorithm performs virtual machines placing function determining the state of servers based on many inputs that are automatically extracted from the service level contract. This algorithm's flexibility derives from the fact that it permits multiple restrictions to be formulated independently of one another and from the energy consumption model. If we are allowed to change the hardware to the workload, the algorithm will be able to reduce energy consumption, this does not focus on renewable energies, which requires the existence of powers management over time. In the work proposed by Lakzaei et al. [20] a two-stage method is presented for placing virtual machines on brokers. The method of this algorithm is that it adjusts the virtual machines in descending order with respect to their processor productivity, and then allocates them to brokers so that the minimum amount of increase in power consumption can be created. This method defines the processor efficiency of each minimum and maximum broker and tries to place virtual machines in such a way that the CPU productivity of brokers does not exceed less or more than the specified value. If brokerage utilization is less than determined amount, it should be switched off to prevent energy loss, as well as if brokerage utilization exceeds limit, it may be difficult to provide the required resources by the virtual machines that reside on it and cannot meet the service level agreement. Each of the cases that occurs is the need for the migration of virtual machines, which is, in fact, the second stage of the proposed method. Three different policies have been proposed to choose the virtual machines that need to migrate: Choosing the virtual machines in a way that the least number of migrations is made. When it wants to do immigration, it selects those virtual machines that are currently minimal compared to their utilization to the maximum utilization they can have, because these virtual machines have the capacity to increase their utilization, so the broker's utilization exceeds the threshold. It randomly selects a number of brokers.

Dynamic aggregation along with migration control is the method proposed by Tan et al. [21] in this study. In addition to the integration of the brokers, the migration of virtual machines is also controlled to prevent the operation of machines as much as possible or if it happened, it would have the least effect. For this purpose, virtual machines are divided into two categories:

- Machines whose capacity is stable
- Machines that are changing their capacity and their needs change over time

They try to get an immigration done in necessary time, among the existing virtual machines, choose a machine that has variable capacity and has a migration to it. The argument proposed for this work is that if a machine has a fixed capacity in a condition that the migration on it affects the performance of the machine, the change is negligible if the machine's capacity is variable. Load balancing algorithms are grouped into three categories:

- In order to balance loading, senders must initiate the process.
- A receiver initiates the load balancing algorithm.
- Combined modes of symmetry

The Round Robin (RR) algorithm [22] is one of the methods of load balancing with the optimal energy consumption approach. This algorithm will distribute tasks to all the secondary processors. All tasks are distributed to secondary processors in RR sequence. This indicates that the selected processor will run in the same sequence, and if the last processor comes, it will return to the first processor. RR algorithms have the advantage that there is no need to link processes between them. The use of RR is typically found in Web servers that have similar functions and, therefore, are equally distributed. However, this approach does not adhere to SLA principles.

Random algorithms are also used to select secondary processors as a means of load balancing. Statistically distributed random numbers are used to select secondary processors. Among all load balancing algorithms, it may be the best-performing algorithm for an application that has a specific goal that does not comply with the concept of service level agreements.

As Shaw and Singh-2015 studied that an algorithm has been proposed to reduce network energy consumption using migration. The proposed algorithm of [23] is used by scheduling to reduce energy consumption by considering DVFS calculations to be used to measure the frequency voltage of the hardware. The migration decision taking place in terms of the recorded history of the network work process, the MIPS (1 million instructions per second) computing, in terms of threshold, host utilization with the flag changes. The correct migration in the network is to reduce the total energy consumption of the network.

Next, the central processor will select a second processor for each step, balancing load and reducing energy consumption. Secondary processors with minimum loads are selected. This algorithm can be used to select secondary processors based on the load information gathered by the central processor. When establishing the process, the load management considers the load information when balancing the process. In the case of dynamic activities generated by different hosts, this algorithm should perform better than parallel applications.

The threshold algorithm comes next. As soon as the processes are formed in this procedure, they are allocated to the hosts. In the case of load conditions exceeding the load level limit, the processor notifies all distant processors from the new load status. The system should be regularly updated with the load status.

Load balance algorithms involving stochastic, RR, central manager model, dynamic load balance algorithm (distributed or concentrated), bee exploration, OLB+, threshold and timing are surveyed. Regarding the existing methods in load balancing, algorithms due to scheduling and stochastic patterns such as RR are of great importance. In this chapter, we reviewed all methods in terms of load balance and load-applying patterns in datacenters, metaheuristic patterns, scheduling patterns.

A scheduler, presented by Shaw and Singh [24] in 2015, can reduce network energy consumption by utilizing migration. As part of the DVFS calculation, the energy consumption is reduced. We have investigated the methods related to the dynamic scheduling pattern and the rest, the DVFS is not regarded, so we start our study based on the frequency scale pattern in the next section.

The summary of the load balancing algorithms is presented in Table 1.

3. The proposed method

In our proposed method, we split up the cloud environment into data centers containing several PMs and a provisioning system to manage the data center. A PM, also known as a bare-metal machine, refers to a hardware server with the motherboard, CPU, memory, and IO controllers. One user can use a PM simultaneously, and that is why we call them single-tenant computers. We have Virtual Machines (VMs) running on each physical computer. A VM is a software computer used to emulate an actual PM. A VM operates in a multi-tenant environment, meaning that multiple VMs run on the same PM [23]. In this case, the computing resources of a physical server are virtualized and shared among all VMs running on it. The cloud prevision should assign any VM for processing when a task arrives. This task assignment impacts the cloud performance, the overall workload, and energy consumption; hence, dispatching a job and selecting the most suitable VM to process it is a challenge in cloud computing.

Table 1
Load balancing algorithms considering advantages and disadvantages.

Algorithm	Explanation	Advantages	Disadvantages
RR Algorithm [25]	Assigned tasks are distributed proportionally to the RR order. When the selected processor reaches the last one, it will return to the first processor and continue in this way. Each processor is independent of others.	Communication between processes is not required.	Each Client processor distributes action to the others.
Randomized Algorithm [26]	A statistical distribution of random numbers is used to randomly select the slave processors.	It can be best applied to run a specific purpose application.	For simultaneous implementation of programs, it is hard.
Central Manager Algorithm [27]	The slave processor with the minimal load will be chosen to handle the work. Based on the system load, Load Manager chooses the best load balancing method throughout the process.	It is proper for parallel applications.	In case of non-allocation of work, load balance management is disrupted and difficulty in implementing and deciding dynamic activities.
Threshold Algorithm [28]	It assigns thresholds to the hosts. Whenever a loaded processor exceeds its loading limit, it updates all remote processors of the new load state.	By initial assumption that the performances are low, it goes well.	It is difficult to maintain load balance when two high-load processors are compared, which increases running time.
Dynamic Load Power Algorithm [29]	The workload is distributed between processors at runtime. Master assigns a new process to the slave processor. Dynamic load balance can be done in two ways: distributed and not distributed.	If one or more of the machines fail, the overall load balancing process will not fail, it can only affect the efficiency of the system.	If one or more of the machines fail, the overall load balancing process fails and the efficiency of system decreases.
Local Line Algorithm [30]	Static allocation of all new processes with processes that have begun migration is done by a host.	By migrating properly, network performance increases.	During low load, remote hosts are taken over causing a slower network.
Bee Search Algorithm [31]	It is based on the behavior of honeybees to find and collect food. It is an algorithm for decentralized bee load balancing. There is a virtual service queue for every virtual server.	Depending on the profit, the server may return to browse, increasing performance, or remain on the virtual server.	Increasing the size of a system does not increase the throughput.
OLB + LBMM Algorithm [32]	Combining OLB scheduling with LBMM (Min–Min load balance) for a better performance.	It helps to effectively apply resources and increases work efficiency.	By combining the patterns, high space complexities is required.
Scheduling Algorithm [24]	An algorithm for reducing network energy consumption applying migration.	Decreasing energy consumption by DVFS calculations	Lack of synchronization in submitting virtual machine requests.

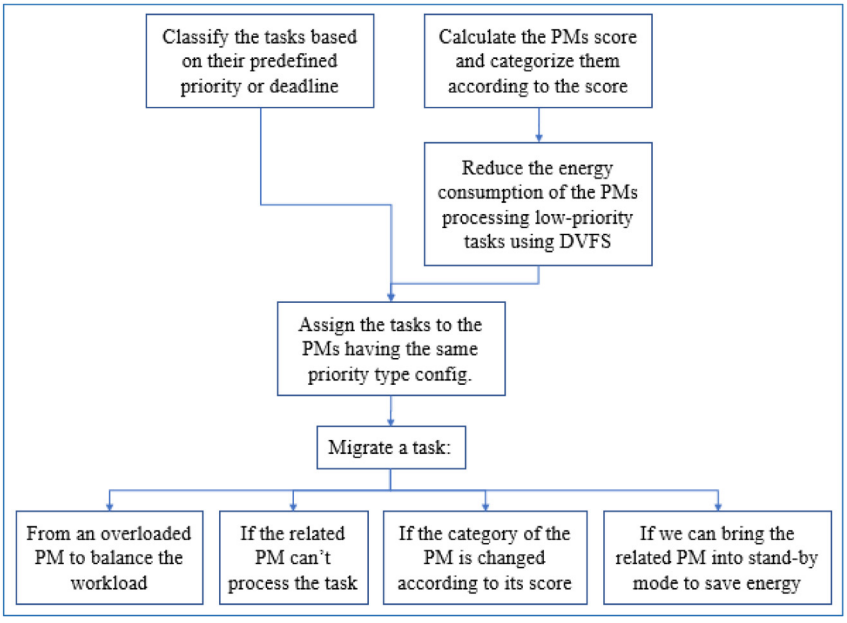

Fig. 2. The workflow diagram of the proposed method.

The proposed method first classifies the tasks into three priority groups. It then calculates the score of PMs to assign the jobs to the PM with the same priority class. The proposed method also utilized the DVFS to reduce energy consumption to maximize the profit of the cloud provider and protect nature. Furthermore, we migrate the tasks under certain circumstances to improve the network's performance or save energy. Fig. 2 shows the workflow diagram of the proposed method and Fig. 3 illustrates its flowchart.

3.1. Task prioritization

Each task, also known as a job, has some properties, including the execution priority, the arrival time, the deadline, and the minimum required configuration essential for processing the task. The proposed method then adds an estimated execution time to the task based on the number of instructions composing the job and the mean MIPS property of the PMs. We have three queues for storing jobs with three different types of priority, i.e., low, medium, and high. The user will determine the precedence of a job, or we calculate that considering the remaining

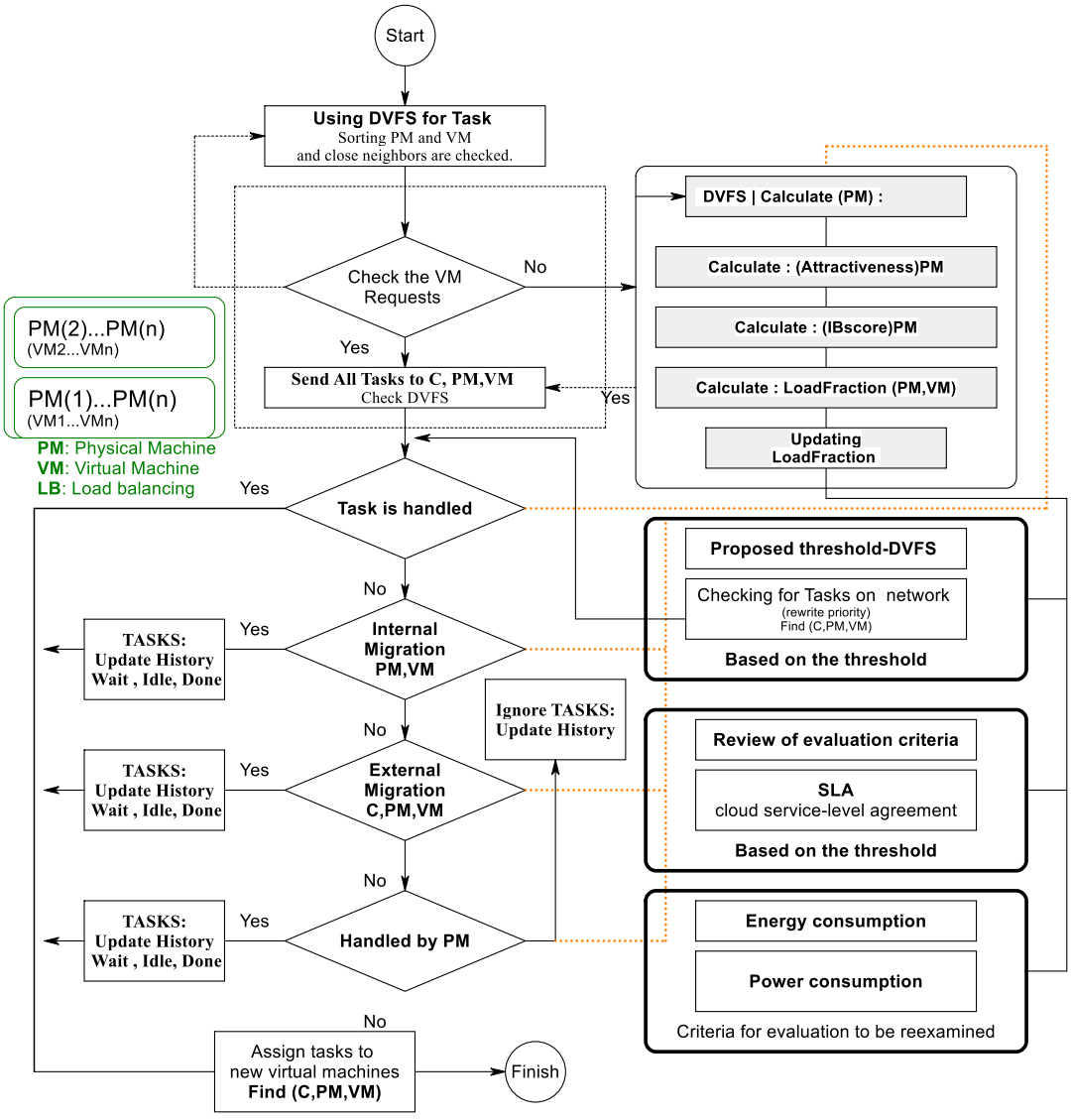

Fig. 3. The flowchart of the proposed method.

time before the deadline of the task execution arrives using Eq. (9).

$$remTime = deadLine - [execTime + currTime] \tag{9}$$

The notations used in Eq. (9) are:

- $remTime$: the remaining time we have to start the job before.
- $deadLine$: the time we have to execute the job before.
- $execTime$: the estimated job execution time according to the number of instructions the job has and the mean processing power of the PMs in the network in terms of MIPS.
- $currTime$: the current time.

Then we categorize the task based on the calculated priority through Eq. (10).

$$priority = \begin{cases} high, & 3t_p < remTime \\ medium, & 2t_p < remTime < 3t_p \\ low, & t_p < remTime < 2t_p \end{cases} \tag{10}$$

t_p is the period in which we recalculate the priority of the tasks, and after each time interval, a job may move to a higher-order priority queue. By prioritizing the tasks, we ensure that the cloud will not process any lower-order priority job if a higher-order one exists. In this way, if there is a lack of resources, we process nearly all jobs before their deadline reaches. 1 shows the task prioritization algorithm.

Algorithm 1 Task Prioritization Algorithm

```
for each period T do
    get the newly arrived task
    if the priority of the task is predefined then
        add the task to the related queue
    else
        calculate the priority of the task Eq. (10)
        add the task to the related queue
```

Fig. 4 demonstrates the flowchart of the task prioritization algorithm.

3.2. Task assignment

The proposed method nominates the VMs to assign the jobs based on four parameters regarding the current task, i.e., the distance between

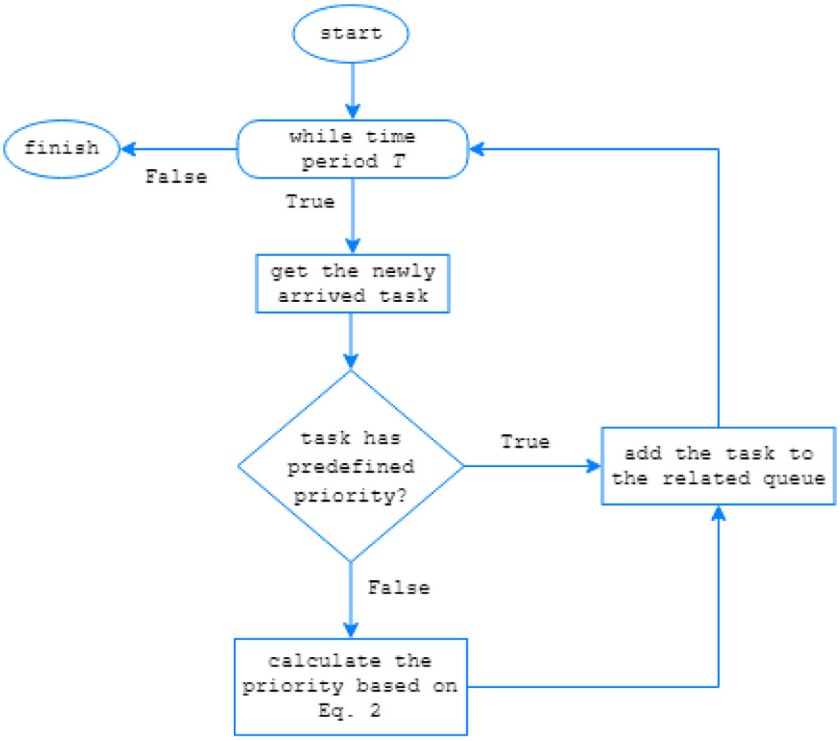

Fig. 4. The flowchart diagram of the task prioritization algorithm.

the user who originated the job, the score of the PMs, the energy consumption, and the workload of the network.

3.2.1. The distance parameter

The distance between the user and the PM processing the job highly affects the round trip time, and of course, the response time. By assigning the job to a PM located close to the user, we can reduce the network's response time, leading to better performance. For this reason, the proposed method first selects the closest data center to the user to assign the current task to the related PMs in that data center.

3.2.2. The score of the PMs

Another parameter that affects the PM selection for task scheduling is the configuration of the PMs.

$$Score(PM_i) = \frac{P(PM_i) \times S(PM_i)}{Fp(PM_i)} \times \frac{T_s - T_f}{T_s} \tag{11}$$

The notations used in Eq. (11) are:

- $Fp(PM_i)$: the failure probability of the ith PM according to Eq. (12), where AFR_i and $MTTF_i$ are the annual failure rate and mean time to failure, respectively, that the device's manufacturer determines. Also, 8760 indicates the number of hours per year [1].

$$Fp(PM_i) = AFR_i = 1 - exp\left(\frac{-8760}{MTTF_i}\right) \tag{12}$$

- $P(PM_i)$: the ith PM processing power regarding the frequency of the clock speed, in GHz, and the number of cores.
- $S(PM_i)$: the ith PM storage capacity in GB.
- T_s: the number of tasks that the ith PM successfully processed.
- T_f: the number of tasks that the ith PM failed to process; we migrate these tasks to another PM explained later in this chapter in more detail.

By calculating the PMs score in each data center, we categorize the PMs into three sorts to handle jobs with different priorities. We determine the top 40 percent PMs for processing the high-priority tasks, the next 30 percent PMs for the medium-priority, and the last 30 percent for processing the low-priority jobs. In this way, the proposed method utilizes the heavy-duty PMs' maximum power and processing speed for high-priority tasks. However, we reduce the servers' supply voltage using DVFS to save energy for medium and low-priority jobs on less powerful PMs.

Before assigning the tasks to the VMs in each PM, we examine if the PM can host new jobs using Eq. (13).

$$\beta(PM_i) = VM_{cond} \wedge Wl_{cond} \tag{13}$$

$\beta(PM_i)$ is 0 if any of these conditions are false. We define the VM_{cond} condition as in Eq. (14).

$$VM_{cond} = \begin{cases} 1, & \text{if the PM has available VM} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

If all the VMs of the PM are busy and have tasks assigned, the PM score is equal to 0 since we cannot allocate more jobs to the PM. The second condition is Wl_{cond} that checks if the PM workload is greater than the average workload of the data center plus a fraction of the mean workload that we define practically in simulations. In this situation, the Wl_{cond} is equal to 0. Eq. (15) demonstrates this condition.

$$Wl_{cond} = \begin{cases} 0, & Wl(PM_i) > \left(Wl_{mean} + \frac{Wl_{mean}}{\alpha}\right) \\ 1, & \text{otherwise} \end{cases} \tag{15}$$

The simulation determines the parameter α. Considering the workload condition when assigning the tasks to the PMs, we ensure that the proposed method will not impose an extra workload on the servers. If the $\beta(PM_i)$ is 0, the PM cannot host any more tasks until finished processing the current jobs.

3.2.3. Assigning the tasks

Fig. 5 shows the task assignment process in the proposed method. The proposed method prioritizes the jobs into three queues, i.e., low-priority (lp), medium-priority (mp), and high-priority (hp). It also

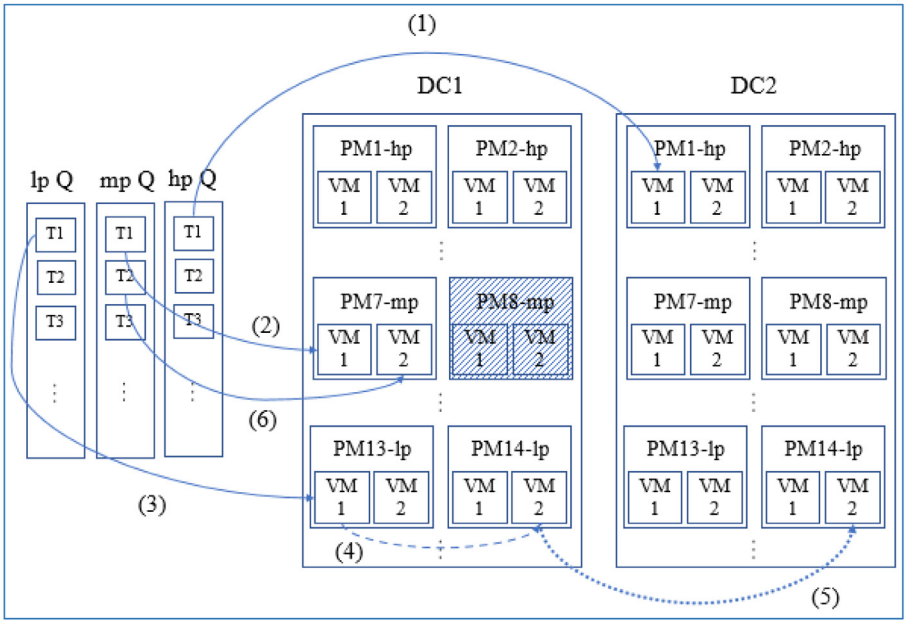

Fig. 5. The task assignment schema.

categorizes the PMs on each data center regarding the PM score for processing the tasks based on their priorities. If any condition in Eq. (13) occurs, we skip the related PM for task assignment. We explain the six task assignment in Fig. 5 in the following:

(1) The task dispatcher assigns a high-priority job to a VM registers in the PM having perfect configuration properties located in a data center close to the user.
(2) The task dispatcher assigns a medium-priority job to a VM registers in the PM having suitable configuration properties located in a data center close to the user. This PM has a lower score than the PMs processing the high-priority task. Also, we reduced the supply voltage of these machines using DVFS to save energy.
(3) The task dispatcher assigns a high-priority job to a VM registers in the PM having good configuration properties located in a data center close to the user. This PM has a lower score than the PMs processing the medium-priority task. Also, we reduced the supply voltage of these machines using DVFS to save more energy as we have pretty much time before their deadline reaches.
(4) A task migrates in some circumstances, such as having an overloaded PM. In this situation, we first select a PM from the same category inside the same data center called internal migration. For example, we first migrate the unprocessed task to the VM registered in PM14 having a low-priority configuration in DC1. The proposed method migrates the jobs to up and running PMs to balance the workload among servers. It keeps standby PMs to prevent energy loss.
(5) If this PM refuses the task again, we then migrate the task to a PM with the same category inside another data center called external migration.
(6) The task dispatcher assigns the job T2 in the medium-priority queue to the same PM where the job T1 resides. In this way, we can bring the PM8 in DC1 into standby mode to save energy by eliminating the static energy consumption of the PM8.

Fig. 6 demonstrates the flowchart of the task assignment algorithm.

Algorithm 2 Task Assignment Algorithm

```
for each period T do
    calculate the score of the PMs Eq. (11)
    categorize the PMs according to the scores
    for each PM p in the data center do
        if the category of p has changed then
            migrate the tasks on p to a PM on d with the same priority
            update the supply voltage using DFVS
    determine the candidate PMs for task assignments Eq. (13)
    while there is a task in the higher-order queue do
        if there is an available VM in the PM at the same class then
            assign the task to the closest active PM
        else if there is any PM in standby mode then
            activate PM p1
            assign the task to p1
            migrate tasks in the PMs with a high workload to p1
        else
            upgrade a lower level PM, p2, into a higher level PM
            increase the supply voltage to raise the throughput
            assign the task to P2
        if the current VM cannot process the task then
            migrate the task to a PM at the same priority level
```

4. Simulation results

Checking the load balance and availability of cloud applications is one of the things which still needs a lot of scrutiny and study. Nowadays, the requirement and application for reliable cloud applications are increasingly high. In the old methods of designing reliable systems, there are four effective methods, fault prevention, fault removal, fault tolerance, and fault forecasting.

Making complex large-scale applications for the cloud by applying fault prevention and removal techniques is challenging. Another approach in creating reliable systems is fault tolerance, which masks them instead of removing faults. In the most sensitive systems, such as air traffic control and nuclear power plants, we use fault-tolerance just because it is costly to develop and maintain. Cloud-based applications can be produced in a more productive and cloud-enhancing environment than in older systems because the cloud provides additional resources. Cloud applications include many components, so providing replacements for all would be very time-consuming and expensive. We must duplicate some sensitive and critical elements for building reliable and low-cost applications in the cloud environment. Creating a substantial section of a cloud application can make it more reliable. We will examine this method in this section.

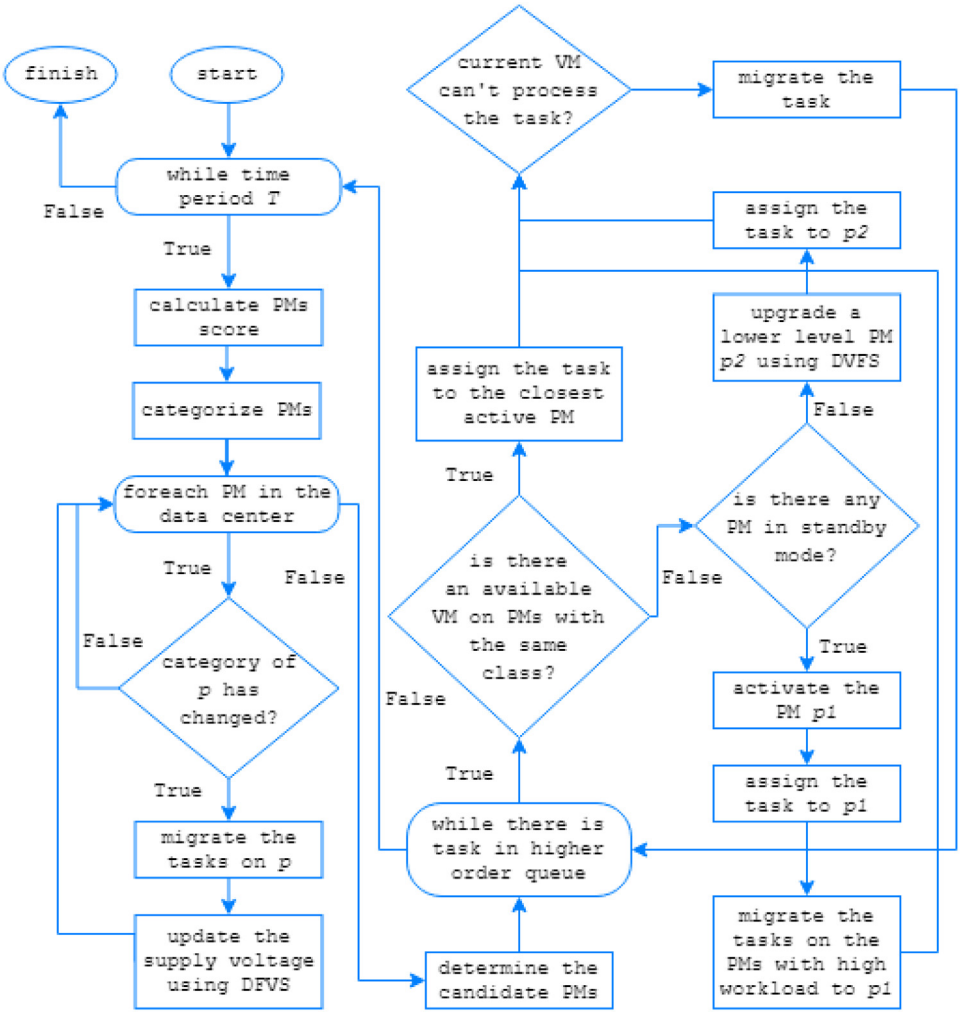

Fig. 6. The flowchart diagram of the task assignment algorithm.

For designing a large-scale cloud data center system, we require a multiple-algorithm assessment of existing cloud infrastructure. Thus, such experiments are difficult to perform on existing infrastructure, especially if we need to replicate them more than once. Simulating multiple suggested policies and algorithms can be a more helpful method of evaluating them. We have utilized the CloudSim simulator to evaluate and validate the proposed method. Generally, this toolkit works by making several entities or entities, each of which handles its work and can exchange messages and information together if required. This toolkit captured its original idea of the SimJava, a library for the java environment developed at Edinburg University. This implementation uses Cloudsim to calculate results and then Matlab simulator software to obtain the necessary consequences from the results provided by the simulator software.

We have to reach a minimum value before achieving a maximum value. VM parameters increase from their minimum values to their maximum values. The proposed method dynamically creates a virtual machine and continues this VM creation until the total resources of that machine are 1.5 times that of the virtual machine. We need 200 units of time to complete the test.

After submitting requests in the form of tasks and user interface requests, the local management unit selects the appropriate host. It directs the user request to it after examining the number of available resources of the hosts in the data center clusters and their productivity at the time of the request. However, if the user has multiple requests, they are listed and addressed one by one. The management unit of a specific VM does not need to respond to those requests or assign a specific cluster in this case. Still, as previously stated, the management unit examines the productivity of the machines and, as the second parameter in the ranking, examines their distance from the requested location.

As noted above, CloudSim and Matlab emulators were applied to assess the results. In reality, the development of this simulator's core was to model the load balance algorithm. The impact of DVFS technology on power consumption and overhead calculation added to that expansion in the direction of the proposed algorithm. We compute the SLA value of the data center at each time interval to show the overhead value. The cloud computing data center simulated in this study comprises 100 virtualization hosts; in fact, it is supposed that virtualizations such as Xen are installed on them that can share resources. We modeled every host as a single core capable of 1000, 2000, or 3000 mips, i.e., they can perform 1000, 2000, or 3000 million commands per second, as well as each host, has 8 GB of RAM and 1 TB of storage memory.

On this structure, we have made 220 VMs, each one executing an application. Each application is composed of 1500,000 instructions. As we have noted before, the objective of the load modulation algorithm is to balance the workload distribution among hosts to inhibit the accumulation of the load on a host. Thus, the implementation of this algorithm also causes overhead or SLA violations. Fig. 7 demonstrates the amount of SLA violations with an interest threshold of 80% on every host during service time or the implementation of applications. This part compares the suggested method with DVFS and without DVFS and RR mode [20,23].

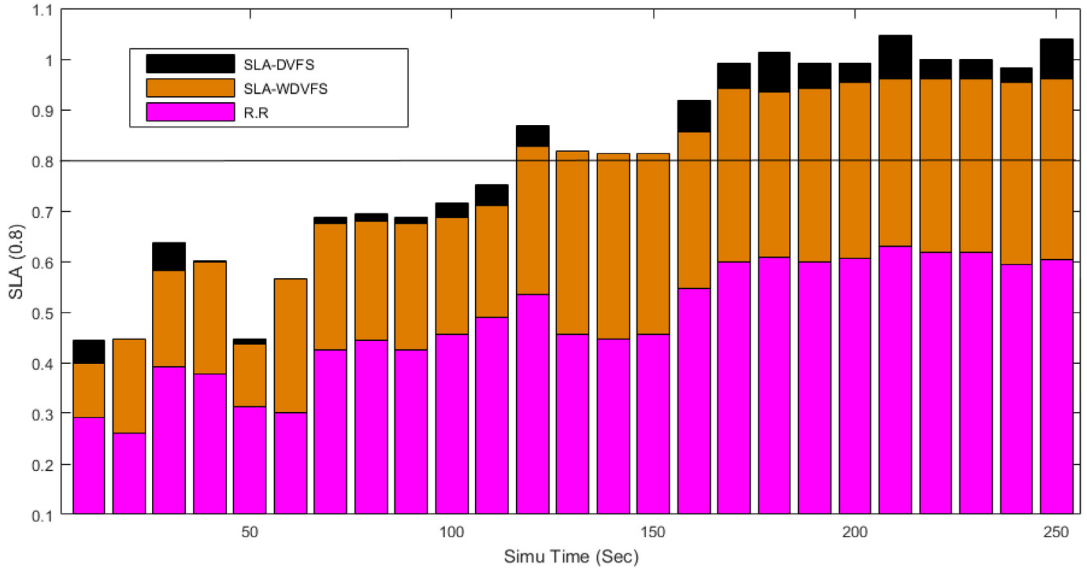

Fig. 7. The amount of SLA violations in the suggested load balancing algorithm, DVFS, and RR.

2,417 VMs transfers or migrations have taken place in the total running time, and the average SLA violation was 19.68%. If you do not use this algorithm, we will not violate SLA and VM transmission. Since power consumption imposes a considerable cost on cloud service suppliers, DVFS technology and a standby mode of winning idle hosts were applied. The performed clustering here can also be very efficient in productivity because a closer machine can process the users' requests by clustering based on VM location. As a result, the operational power increases due to reducing the time of sending and receiving information.

Regarding every CPU core works at an optimal frequency, in the first phase, allocation of every VM to PM and especially its proper PM core is performed at any time interval, then by measuring the dynamic voltage and frequency of PM's (DVFS calculations) and adjusting the optimal frequency for each active PM core, the information of VMs and PM's is updated. At last, to decrease energy consumption and integrate computational resources, the cloud is reconstituted. In this method, three models which are the basis of the scheduling algorithm are described. These models are power models, physical machines, and virtual machines, in each of which the amount of power consumed and the total amount of energy is computed.

The reason to select the appropriate PM is to regard the higher efficiency-power ratio. By sorting the downside of PM due to higher OPER value (Optimal Power-Efficiency Ratio), the proposed method assigns every VM to a good PM. If active PMs are not suitable, sleep PMs can be activated, and this schedule will take place until all VMs are completed or defeated. At last, regardless of the effectiveness overhead, less energy consumption and the timing of the number of virtual machines are achieved.

Fig. 8 compares the mode of using the suggested load balancing method with DVFS technology and applying DVFS by the scheduling method, specifically the algorithm in [20,23].

In this part, the methods of RR and DVFS without it were reviewed that the time of applying DVFS is more effective. The total amount of power consumption during service when utilizing DVFS in the load balancing method is from 33 KWh to 60 KWh, while in the scheduling method, it is from 40 KWh to 63 KWh. Thus, using this technology can help decrease power. It can be observed due to clustering, selecting the right machine, and sleeping idle machines, the suggested solution works much better than [20,23] and as a result, consumes less power than the algorithm and is able to continue working for longer periods of time with less power consumption and this can be realized that it is better than The other method works. If VMs are not transferable in the live mode and stop to transfer machines and then a transfer is performed, the power consumption will be highly decreased.

The load balance and availability of cloud computing services without constant glitches are key issues in the design of cloud computing data centers. In this chapter, the simulation of cloud computing architecture was surveyed, which performs system resource management to increase optimal access in the data center. Any program that is requested can be run in the cloud computing data center that runs inside a virtual machine. There are multiple virtual machines created on separate physical machines. The virtual machines come in different software classes. Depending on their processing power, some of them may be classified into different classes (high, medium, and low) . The local manager unit in multiple time intervals reviews and compares the status and mode of virtual machines due to a request and, based on the policies and purposes of that request and the tolerance management algorithm, decides on the reliability of the result of implementing that application in that timeframe, and in case of failure of the host or their results, it can make another virtual machine on another host and continue running the application from that machine. Using this algorithm imposes a slag on the system, which we observed as an SLA violation . The suggested algorithm significantly increases the power of the system. Overall, 2417 virtual machine migrations were accomplished, with an average SLA violation of 19.68%. We will not have SLA violations or VM transfers if this algorithm is not used.

5. Conclusion

Cloud service providers confront several issues, including infrastructure management and energy reduction. Cloud computing, virtualization, DVFS computing, load balancing, and service level agreements are all explored in detail in this research. This article aims to examine different methods for balancing load and reducing power consumption. Dynamic load adjustment is a method to distribute the workload among present hosts by monitoring workload performance at multiple time intervals and performing virtual machine migration.

DVFS calculations applied in simulation reduce power consumption by balancing the load among existing hosts. The purpose of this is to prevent cloud service providers from incurring significant costs related to power consumption by placing unemployed hosts in standby mode. SLA violations are resulted from using this algorithm, which causes the system to lag. Load balancing to avoid overworking hosts and mitigate

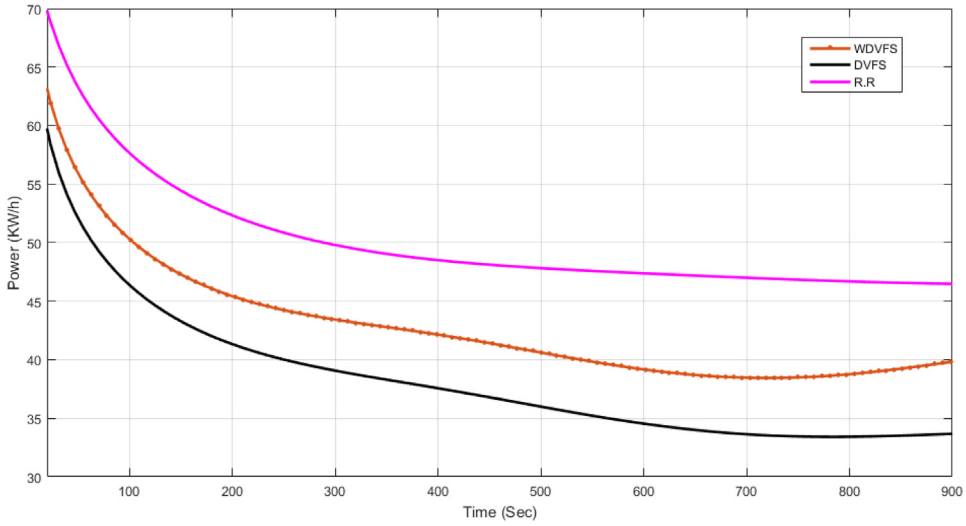

Fig. 8. The comparison of the suggested load balancing algorithm with DVFS and RR.

the situation when one hosting has a low load, and another has a high workload will ensure resource efficiencies, improve response times, and reduce energy consumption.

Also, we investigate whether distributed cloud computing from multiple geographically located data centers can be a reliable kind of cloud computing. In the future, our research will examine whether cloud computing data centers are available and how their concept of availability is related to reliability.

CRediT authorship contribution statement

Amir Javadpour: Conceptualization, Methodology, Software. **Arun Kumar Sangaiah:** Data curation, Writing – original draft. **Pedro Pinto:** Visualization, Investigation. **Forough Ja'fari:** Supervision. **Weizhe Zhang:** Software, Validation. **Ali Majed Hossein Abadi:** Writing – review & editing. **HamidReza Ahmadi:** Visualization, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant No. HIT.OCEF.2021007, the Shenzhen Science and Technology Research and Development Fundation under Grant No. JCYJ20190806143418198, the National Key Research and Development Program of China under Grant No. 2020YFB1406902, the Key-Area Research and Development Program of Guangdong Province under Grant No. 2020B0101360001, the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies under Grant No. 2022B1212010005. Professor Weizhe Zhang is the corresponding author.

References

[1] T. Erl, R. Puttini, Z. Mahmood, Cloud Computing: Concepts, Technology, & Architecture, Pearson Education, 2013.
[2] A. Javadpour, G. Wang, S. Rezaei, Resource management in a peer to peer cloud network for IoT, Wirel. Pers. Commun. 115 (3) (2020) 2471–2488.
[3] N.K. Sehgal, P.C.P. Bhatt, J.M. Acken, Cloud computing with security, in: Concepts and Practices, second ed., Springer, Switzerland, 2020.
[4] S. Mahmoodi Khaniabadi, A. Javadpour, M. Gheisari, W. Zhang, Y. Liu, A.K. Sangaiah, An intelligent sustainable efficient transmission internet protocol to switch between user datagram protocol and transmission control protocol in IoT computing, Expert Syst. (2022) e13129.
[5] W. Voorsluys, J. Broberg, R. Buyya, et al., Introduction to cloud computing, Cloud Comput.: Princ. Paradigms (2011) 1–44.
[6] A. Javadpour, A.M.H. Abadi, S. Rezaei, M. Zomorodian, A.S. Rostami, Improving load balancing for data-duplication in big data cloud computing networks, Cluster Comput. 25 (4) (2022) 2613–2631.
[7] D. DeJonghe, Load Balancing in the Cloud: Practical Solutions with NGINX and AWS, O'Reilly Media, 2018.
[8] M. Mesbahi, A.M. Rahmani, Load balancing in cloud computing: A state of the art survey, Int. J. Mod. Educ. Comput. Sci. 8 (3) (2016) 64.
[9] A. Javadpour, P. Pinto, F. Ja'fari, W. Zhang, DMAIDPS: A distributed multi-agent intrusion detection and prevention system for cloud IoT environments, Cluster Comput. (2022) 1–18.
[10] A. Javadpour, Improving resources management in network virtualization by utilizing a software-based network, Wirel. Pers. Commun. 106 (2) (2019) 505–519.
[11] A. Javadpour, Providing a way to create balance between reliability and delays in SDN networks by using the appropriate placement of controllers, Wirel. Pers. Commun. 110 (2) (2020) 1057–1071.
[12] S.K. Mishra, P.P. Parida, S. Sahoo, B. Sahoo, S.K. Jena, Improving energy usage in cloud computing using DVFS, in: Progress in Advanced Computing and Intelligent Engineering, Springer, 2018, pp. 623–632.
[13] A. Beloglazov, R. Buyya, Y.C. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, in: Advances in Computers, vol. 82, Elsevier, 2011, pp. 47–111.
[14] J. Cardoso, J.G.F. Coutinho, P.C. Diniz, High-performance embedded computing, Embed. Comput. High Perform. (2017) 17–56.
[15] G.S. Aujla, N. Kumar, MEnSuS: An efficient scheme for energy management with sustainability of cloud data centers in edge–cloud environment, Future Gener. Comput. Syst. 86 (2018) 1279–1300.
[16] R.K. Barik, R.K. Lenka, K.R. Rao, D. Ghose, Performance analysis of virtual machines and containers in cloud computing, in: 2016 International Conference on Computing, Communication and Automation, ICCCA, 2016, pp. 1204–1210.
[17] V. Sangeetha, R. Krishankumar, K.S. Ravichandran, F. Cavallaro, S. Kar, D. Pamucar, A. Mardani, A fuzzy gain-based dynamic ant colony optimization for path planning in dynamic environments, Symmetry 13 (2) (2021) 280.
[18] A. Gupta, S. Namasudra, A novel technique for accelerating live migration in cloud computing, Autom. Softw. Eng. 29 (1) (2022) 1–21.

[19] M.A. Khan, An efficient energy-aware approach for dynamic VM consolidation on cloud platforms, Cluster Comput. 24 (4) (2021) 3293–3310.
[20] M. Lakzaei, V. Sattari-Naeini, A. Sabbagh Molahosseini, A. Javadpour, A joint computational and resource allocation model for fast parallel data processing in fog computing, J. Supercomput. (2022) 1–24.
[21] C. Tan, J. Zhang, X. Liu, Z. Zhou, X. Shen, Survey of virtual machine deployment algorithms in cloud data center, in: International Conference on Frontiers of Electronics, Information and Computation Technologies, 2021, pp. 1–7.
[22] F. Alhaidari, T.Z. Balharith, Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling, Computers 10 (5) (2021) 63.
[23] A. Javadpour, G. Wang, S. Rezaei, S. Chend, Power curtailment in cloud environment utilising load balancing machine allocation, in: 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI, 2018, pp. 1364–1370.
[24] S.B. Shaw, A.K. Singh, Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center, Comput. Electr. Eng. 47 (2015) 241–254.
[25] R.K. Mishra, S. Kumar, B. Sreenu Naik, Priority based round-robin service broker algorithm for cloud-analyst, in: 2014 IEEE International Advance Computing Conference, IACC, 2014, pp. 878–881.
[26] Z. Tang, L. Qi, Z. Cheng, K. Li, S.U. Khan, K. Li, An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment, J. Grid Comput. 14 (1) (2016) 55–74.
[27] A. Pourghaffari, M. Barari, S. Sedighian Kashi, An efficient method for allocating resources in a cloud computing environment with a load balancing approach, Concurr. Comput.: Pract. Exper. 31 (17) (2019) e5285.
[28] J. Adhikari, S. Patil, Double threshold energy aware load balancing in cloud computing, in: 2013 Fourth International Conference on Computing, Communications and Networking Technologies, ICCCNT, IEEE, 2013, pp. 1–6.
[29] M. Tarahomi, M. Izadi, New approach for reducing energy consumption and load balancing in data centers of cloud computing, J. Intell. Fuzzy Systems 37 (5) (2019) 6443–6455.
[30] L. Zeng, B. Veeravalli, A.Y. Zomaya, An integrated task computation and data management scheduling strategy for workflow applications in cloud environments, J. Netw. Comput. Appl. 50 (2015) 39–48.
[31] J. Yao, J.-h. He, Load balancing strategy of cloud computing based on artificial bee algorithm, in: 2012 8th International Conference on Computing Technology and Information Management, Vol. 1, NCM and ICNIT, IEEE, 2012, pp. 185–189.
[32] S. Aslam, M.A. Shah, Load balancing algorithms in cloud computing: A survey of modern techniques, in: 2015 National Software Engineering Conference, NSEC, IEEE, 2015, pp. 30–35.