Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-01794-6

ORIGINAL RESEARCH

Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing

J. Praveenchandar[1] · **A. Tamilarasi**[2]

Received: 2 October 2019 / Accepted: 18 February 2020

Abstract
Cloud computing is one among the emerging platforms in business, IT enterprise and mobile computing applications. Resources like Software, CPU, Memory and I/O devices etc. are utilized and charged as per the usage, instead of buying it. A Proper and efficient resource allocation in this dynamic cloud environment becomes the challenging task due to drastic increment in cloud usage. Various promising technologies have been developed to improve the efficiency of resource allocation process. But still there is some incompetency in terms of task scheduling and power consumption, when the system gets overloaded. So an energy efficient task scheduling algorithm is required to improve the efficiency of resource allocation process. In this paper an improved task scheduling and an optimal power minimization approach is proposed for efficient dynamic resource allocation process. Using prediction mechanism and dynamic resource table updating algorithm, efficiency of resource allocation in terms of task completion and response time is achieved. This framework brings an efficient result in terms of power reduction since it reduces the power consumption in data centers. The proposed approach gives accurate values for updating resource table. An efficient resource allocation is achieved by an improved task scheduling technique and reduced power consumption approach. The Simulation result gives 8% better results when comparing to other existing methods.

Keywords Resource allocation · Task scheduling · DRA table · Power management

1 Introduction

Virtualization is a technology in which all the physical resources are virtualized. It is a platform implemented for resource sharing and increase the flexibility in data sharing. And it is an example for distributed and parallel data processing. Resources like memory, CPU, Input, Output devices and software's are virtualized in this environment. It is implemented through many physical and functional layers and by various levels. Virtualization monitor or Hypervisor are used to create the virtualization layer and applications are running on guest operating system with the help of virtualized resources. This Software layer virtualizes the physical machines (PM's). It involves various level, like application level virtualization, operating system level virtualization and hardware level virtualization. In application level, an application is virtualized as virtual machines (VM). In this scenario virtual layer will be sitting on the top of the operating system. Best example is Java Virtual Machine (JVM).In operating system level virtualization, idle container is created on server and that will act as real server. And it is one of the commonly used virtualization technique. Then hardware virtualization, virtual machine is created on the top of the hardware. Basic idea is to virtualize the resources like memory, I/O devices and processors. It is implemented by using the hypervisor software. This software will directly sit between operating system and physical hardware. Guest operating system and application are communicated each other through hyper calls. On the top of the virtual machines, virtual clusters are created and all virtual machines are interconnected internally. So it is very important to manage the VM's, running in huge number of physical machines (PM) and ensure an uninterrupted experience

✉ J. Praveenchandar
praveenjpc@gmail.com

A. Tamilarasi
drtamil@kongu.ac.in

[1] CSE, Vel Tech Rangarajan Dr. Sagunthala R and D Institute of Science and Technology, Chennai, India

[2] Department of Computer Applications, Kongu Engineering College, Perundurai, India

Published online: 02 March 2020

to cloud users. In data centers, a huge number of workloads will run on all servers at all times. They are heterogonous in nature and most of the datacenters servers are underutilized. So a certain amount of power, space, hardware and cost are wasted in every data center. To avoid the resource wastage and improve the performance of cloud operations, still some optimization technique is needed. Considering, some of the real-time cloud service provider (Hwang et al. 2013) like Amazon EC2 has got 1–20 EC processors, 106-169 TB of storage and 1.7–15 GB of memory, uses Xen hypervisor, Linux and windows guest operating system. And an another provider Rack space cloud is having four core CPU, 10–620 GB of storage and 0.25–16 GB of memory uses Xen hypervisor with Linux operating guest system. Python, PHP, Java C#,.NET are being used as API and access tools. A normal datacenter designed 5–10 times bigger than football ground. It is constructed by the servers having a number of processors with multi core Central Processing Unit (CPU). Local coherent DRAM attached with hard drives. Let us consider a data center constructed with 2500 servers having five 1 TB disk drives and 10 GB of DRAM. Every 50 servers are interconnected through 2Gbps link to routing switch, which has ten 1Gbps port. There is a cooling system for each data center. Cooling system consists of hiding cables, raised floor, cooling supplies and power line. Power cables are routed in floor area which is mainly used to circulate the cooling air to physical machine at racks. Cold air is blown by the computer room air conditioner. Up gradation of performance and minimization of power consumption are the major challenges and highly demanded while designing future green computing data centers.

The functional part of cloud architecture, it is having large data centers. It usually consists of broad number of cloud servers called the physical machines (PM). Each PM has different resource capabilities. Virtualization is employed to map numerous requests that are arising from different client machines to the particular physical machine. Service provider offer resources to the client based on their request and charging them as per their usage. Here is the challenge to provide the resources for all user requests dynamically and meanwhile the Service level Agreement (SLA) must be maintained effectively. Cloud user will submit the request to provider. The scheduler assigns the request possibly to resources available with different PM's or same PM's. Each PM contains multiple VM's and they kept ready to allocate. The allocation of resources to all user's request in dynamic environment is the challenging task. Because, multiple requests are received and gets allocated the virtual machines simultaneously. In such a case, some challenges are arising in cloud resource provisioning. So an effective task scheduling is needed for dynamic cloud environment with enhanced energy consumption. There are so many techniques and algorithms are available to achieve this task. But, due to the rapid development of the cloud system and growth of the cloud users day by day, still we need some improvements with the system in terms of number of jobs successfully completed, minimization of execution time, efficient utilization of cloud resources and an energy minimization in data centers. So we started research for a proficient resource allocation scheme to maintain the SLA efficiently.

Various methods of resource allocation schemes have been employed in recent years. However inefficiency in resource allocation and energy consumption are increasing day by day. Based on these constrains, an improved scheduling and resource allocation scheme is proposed in this research work. Preference based task scheduling algorithm (PBTS) is designed for scheduling process. Then, a modified prediction based (PMA) algorithm is implemented for power minimization. The information about forthcoming job is essential in resource allocation process. It decides whether all available PM's needs to be in ON state or Sleep state, so that the power consumed by unused PM's can be minimized. It helps the resource allocation process to consume the less power. In this research work we propose a new scheme for Dynamic Resource Allocation table (DRA) updating process. In that, the resource table is updated based on the analysis of predicted outputs. And there exist two processes occurring simultaneously with the input requests. First the prediction mechanism and the next is scheduling of tasks. Both these process are mainly depends on DRA table which will be updated dynamically. This table is nothing but the table containing available resources of all physical machines of the server. Then power reduction support is also provided for an energy minimization in data centers.

This research work organizes as follows, in Sect. 2, related works are analyzed. Section 3 describes about the proposed system. In this, two modules are proposed (1) PBTS based task scheduling algorithm and (2) Dynamic resource allocation by DRA table updating algorithm. It analyzed how DRA table is updated dynamically based on PBTS. Section 4 describes the future workload prediction and power minimization module. In Sect. 5 it is discussed the experimentation which tells about simulation results and analysis about efficiency improvement of proposed approach. Section 6 contains the conclusion.

2 Related work

Consumption of energy in cloud environment is discussed and proposed in (Dabbagh et al. 2015), they came up with the idea of prognostication of the VM needed from the user requests in the cloud data centers in future. It also calculates the resources required such as memory and CPU as per the customer request. This approach gives the exact count of physical machines required for the data centers to

ensure the SLA. It minimizes the energy consumption of the data centers by making the PM's off mode. In (Tseng et al. 2017) Multi objective Genetic Algorithm is implemented for dynamic forecasting of resource utilization and energy consumption. Then it describes the prediction based VM allocation technique to increase average CPU and memory utilization. In (Li et al. 2015) utilization of resources with higher efficiency is proposed. It ensures two things, first one is, rescheduling of the starting time of the job and modifying the bandwidth of the virtual machines. By ensuring the two constrains better results could be achieved in terms of resource utilization. In (Kwak et al. 2015) real network environment is considered. Some dynamic issues in the real time environment are also analyzed. First is to check the usage of CPU resource, task allocation and processing of local CPU to dispatch on networks. Then speed of the central processing unit has been also considered. By applying the method the system could minimize the energy in networks and achieve better CPU utilization. In (Xu et al. 2015) says the supporting of dynamic distribution of VM's for implementation. Then consumption of energy is closely dealt. A special algorithm is proposed for resource allocation with energy aware constrain. It provides effectiveness and efficiency to the system model. In (Xiao et al. 2012) Based on the application demand the resources are allocated dynamically. It is done by predicting the future resource needs. Then by combining the different types of workloads, server utilization is improved. Energy saving is also dealt to support Green computing (AlShahwan et al. 2016).

In (Chou et al. 2016) Based on a particle swarm optimization algorithm, dynamic power saving in resource allocation is discussed. Then efficiency ratio of air conditioner is also considered. In (Pahlevan et al. 2017) Machine learning based and heuristic based VM allocation methods are compared in terms of energy, QoS, migrations and network traffic. In (Nejad et al. 2014) designing of enhanced mechanism for VM provisioning and resource allocation is presented. It also provides efficient and higher utilization of available resources to get good profit. Then virtual machine provisioning model is proposed. Resource allocation is implemented by auction based on, the bids value is also discussed. This model is used to formulate the virtual machine provisioning dynamically with considering variety of resources. It assured better profit. In (Metwally et al. 2015) resource (like CPU, memory and storage) provision is proposed in the type of VM instances. Then that will be allocated for user request. To implement the virtual machine provisioning, Pricing and allocation, a special approach is proposed. This is, online mechanism based on auction. Various types of resources are considered. When customer gives a request or a fetched resource is released, immediately this approach will be enabled. And it proved as incentive compatible. From all the above discussed mechanism, we could know that how to boost the efficiency of the system model. In (Shi et al. 2015) a new mechanism for allocating the resources is proposed. Resource pricing in cloud environments is also discussed. It optimizes the efficiency of the system in the temporal domain. This work mainly deals with three scenarios. To decay the long term optimization problem into versatile serious one shot problem. To improve the performance, binary search algorithm is used. Finally the improved, cost sensitive auction mechanism is achieved. (Wang et al. 2015; Mashayekhy et al. 2015; Zaman and Grosu 2013; Peng et al. 2014) are discussed about versatile dynamic resource allocation techniques with respect to pricing and power minimization.

In (Teymoori et al. 2015) multi-server polling system is designed. It improves the performance measures such as delay variance and mitigating the short-term unfairness. Then (Landa et al. 2016; Qu et al. 2016) describes two different approaches for resource provisioning and resource scheduling respectively. In (Dai et al. 2016) Resource allocation processes for few Big Data applications are discussed. In that paper, they adopted a multi-objective optimization algorithm for trading off the performance, cost and availability of Big Data application running on Cloud environment and it is proved that their approach can run about 20% faster than other optimization methods. In (Du et al. 2018) they dealt the computation offloading problem in cloud or fog environment. This is done by optimizing the offloading decisions and the allocation of computation resources with guaranteeing user fairness and maximum tolerable delay. This optimization problem is implemented for minimizing the maximal weighted cost of delay and energy consumption. They designed low-complexity suboptimal algorithm to solve this NP-Hard problem. So the offloading decisions are derived through semi-definite relaxation and randomization, and the resource allocation is achieved by using fractional programming theory.

In (Nguyen et al. 2017) a Resource Allocation process for Heterogeneous large scale IoT Cloud is discussed. In this method, they designed an approach to predict the least completion time distribution that is applicable to make an advanced trade off decisions in rescheduling and resource allocation. In (Gawali and Shinde 2018) a heuristic method for resource allocation and task scheduling is proposed. In this approach modules like task scheduling and resource allocation using divide and conquer method, modified analytic hierarchy process, longest expected processing time preemption and bandwidth aware divisible scheduling are implemented. Before the task's actual allocation, the tasks are processed. By considering the load and bandwidth of the cloud resources, the resource allocation is done using BAR optimization and combined BATS algorithms. Response time and turnaround time are taken as performance metrics.

3 Proposed system

Our research work has three major modules: Preference Based Task Scheduling (PBTS), Dynamic Resource Table (DRA) Updating module and Power Minimization module. This is illustrated in Fig. 1.This section, describes the system model preliminaries followed by PBTS algorithm and DRA Table updating process.

Consider the clients request as N and it is represented as

$$N = \{n_1, n_2, n_3 \ldots n_i\} \tag{1}$$

where, $1 < i < \infty$.

Since it is the dynamic process, all requests are considered for a unit time. The task scheduler makes the users request in terms of task T, where $T \leq N$. The task T is represented as

$$T = \{t_1, t_2, t_3, \ldots t_n\} \tag{2}$$

where $1 < n < \infty$.

Each t_n is a task made by the scheduler based on the user's request. Every task is scheduled by a scheduling algorithm in such a way that each task is fitted to specific agents.

$$A = \{a_1, a_2, a_3, \ldots a_j\} \tag{3}$$

where $1 < j < \infty$

Agents establish the connectivity between client and server based on the resource availability. The connection is made when

$$C_{n=}\begin{cases} 0 \text{if} w_n \geq 0 \\ 1 \text{otherwise} \end{cases} \tag{4}$$

where C_n refers to connectivity between request and server. This connection will be made during the runtime. W_n refers the workload of the input request.

3.1 Preference based task scheduling (PBTS)

In this proposed research work PBTS algorithm for task scheduling is designed and adopted for the dynamic resource allocation process. This task scheduling consists of following steps (1) The process done by PBTS algorithm (2) Updating the resource table based on the connectivity made by an agents. All the inputs from clients are given to task scheduler. All the inputs are changed as various tasks. These tasks are scheduled and given to agents based on PBTS Algorithm. As per Eq. (2) consider for all request N, the tasks are given by T = {set of tasks},$T = \{t_1, t_2, t_3, \ldots t_n\}$ where $1 < n < \infty$ Each t_n is the task of a given input request. The PBTS algorithm uses two metrics for scheduling purpose. One is the task size s_x and inter arrival time I_{at}. The scheduling algorithm checks for the task size first, As per the Eq. (3), consider the agents A is a set of all agents, $A = \{a_1, a_2, a_3, \ldots a_m\}$ where $1 < m < \infty$. The agents are act as an interface between the client and the server. The function of the agent is to collect the scheduled tasks from the job controller and finds an available resource using the resource table. If there exist a resource a connection is made and the service is provided. Let Q_i be the priority queue for the task scheduling. The study state distribution can be calculated as follows,

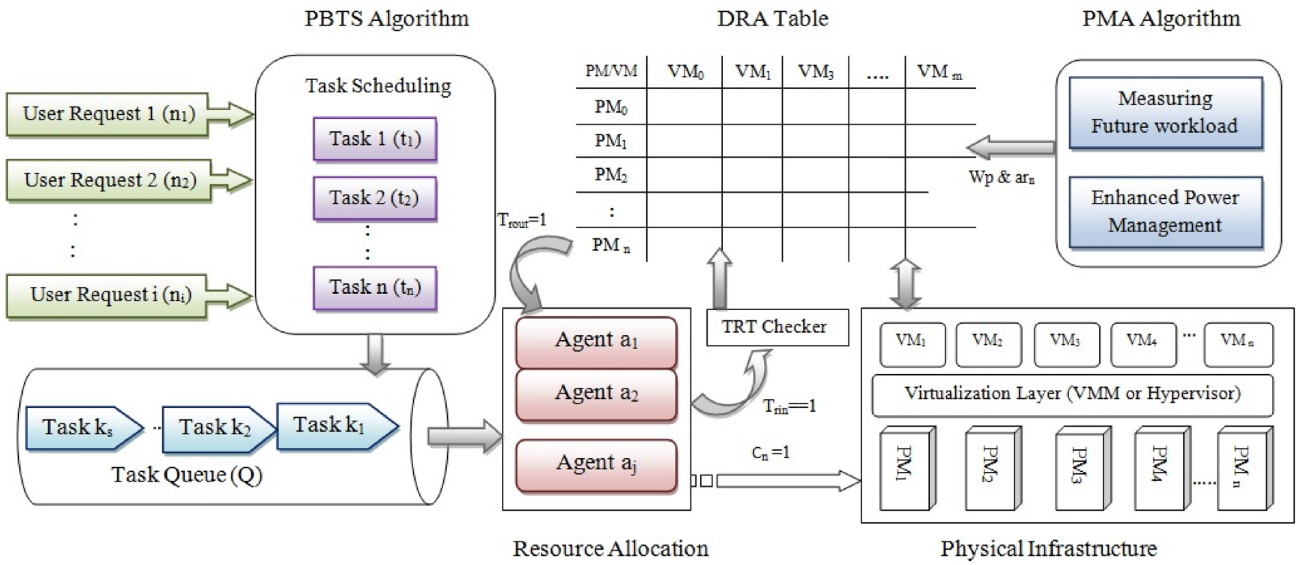

Fig. 1 Proposed architecture

$$Q_s=\begin{cases}\frac{\lambda^s}{s!}\mu Q_0 & \text{for } 0 \le s \le c\\ \frac{\lambda^s}{C^{s-c}C!\mu^s}Q_0 & \text{for } c \le s \le X\end{cases} \tag{5}$$

The task scheduler uses proposed Preference Based Task Scheduling (PBTS) algorithm to assign the tasks to agents. The priorities to each tasks are assigned based on the job size and inter arrival time. Then the tasks are converted into priority based tasks represented by $K=\{k_1,k_2,k_3.....k_s\}$ where $1<s<\infty$ then they will search for priority queue Q_i. And the normalized condition of Q_0 can be

$$Q_0=\left(\sum_{s=0}^{c-1}\frac{\lambda^s}{s!\mu^s}+\sum_{s=c}^{x}\frac{\lambda^s}{c^{s-c}c!\mu^s}\right)^{-1} \text{ and, let } \rho=\frac{\lambda}{\mu} \text{ and } a=\frac{\rho}{c} \tag{6}$$

Let us simplify the expression

$$\sum_{s=c}^{x}\frac{\rho^s}{c^{s-c}c!}=\frac{\rho^c}{c!}\sum_{s-c}^{x}a^{s-c}=\begin{cases}\frac{\rho^c}{c!}\frac{1-a^{X-c+1}}{1-a} & \text{if } a\neq 1\\ \frac{\rho^c}{c!}(X-c+1) & \text{if } a=1\end{cases}. \tag{7}$$

It gives

$$Q_0=\begin{cases}\left[\frac{\rho^c}{c!}\frac{1-a^{x-c+1}}{1-a}+\sum_{s=0}^{c-1}\frac{\rho^s}{s!}\right]^{-1} & \text{if } a\neq 1\\ \left[\frac{\rho^C}{C!}(X-C+1)+\sum_{S=0}^{C-1}\frac{\rho^S}{S!}\right]^{-1} & \text{if } a=1\end{cases} \tag{8}$$

The algorithm 3.1 is designed to implement the task scheduling for 'i' user requests per unit time. The PBTS gives the protocol of allocation process of the proposed approach. In this algorithm 'i' tasks will be taken as input and efficiency improved resource table will be an output.

Then mean number of 's' tasks in Q_i could be obtained as,

$$\overline{\lambda}=\lambda(1-Q_x)=\mu\overline{c} \tag{9}$$

$$\overline{S}=\overline{L}+\overline{c} \tag{10}$$

$$\overline{S}=\overline{L}+\rho(1-Q_x) \tag{11}$$

And mean response time and waiting time of a tasks can be obtained as

$$\overline{R}=\frac{\overline{S}}{\lambda(1-Q_x)} \tag{12}$$

$$\overline{W}=\frac{\overline{L}}{\lambda(1-Q_x)} \tag{13}$$

Distribution at the arrival instance

$\Pi_s=Q$ (Tasks available in the Queuel A Task is about to enter into Queue)

$$\pi_s=\Delta\lim_{c\to 0}\left\{\frac{[\lambda\Delta e+o(\Delta e)]Q_s}{\sum_{s=0}^{x-1}[\lambda\Delta e+o(\Delta e)]Q_s}\right\} \tag{14}$$

$$\pi_s=\Delta\lim_{c\to 0}\left\{\frac{[\lambda+o(\Delta e)/\Delta e]Q_s}{\sum_{s=0}^{x-1}[\lambda+o(\Delta e/\Delta e)]Q_s}\right\} \tag{15}$$

$$=\frac{\lambda Q_s}{\lambda\sum_{s=0}^{x-1}Q_s}=\frac{Q_s}{1-Q_s}(s\le x-1) \tag{16}$$

Number of tasks in the queue can be found as

$$\overline{\lambda}=\lambda(1-Q_x)=\mu\overline{c} \tag{17}$$

$$\overline{S}=\overline{L}+\overline{c}=\overline{L}+\rho(1-Q_x) \tag{18}$$

Mean length of the queue can be calculated as

$$\overline{L}=\sum_{s=c+1}^{x}(s-c)Q_s=\sum_{s=c+1}^{x}(s-c)\frac{\lambda^s}{c^sc!\mu^s}Q_o=\frac{Q_0\rho^c a}{c!}\frac{d}{da}\left(\frac{1-a^{x-c+1}}{1-a}\right) \tag{19}$$

It gives

$$\overline{L}=\frac{Q_0\rho^c a}{c!(1-a)}\left[1-a^{x-c-1}-(1-a)(x-c+1)a^{x-c}\right] \tag{20}$$

Distribution of response time and waiting time

$$D_w(e)=D_w(0)+\sum_{s=c}^{x-1}\Pi_s-\sum_{s-c}^{x-1}\Pi_s\sum_{j=0}^{s-c}\frac{(c\mu e^j x^{-c\mu e})}{j!} \tag{21}$$

$$=1-\sum_{s=c}^{x-1}\Pi_s\sum_{j=0}^{s-c}\frac{(c\mu e)^j x^{-c\mu e}}{j!} \tag{22}$$

All tasks for t_1 to t_n will undergone the following process. Check whether 'm' tasks will have a same size (S_x), if so the next consideration is an arrival time of each'm' jobs. As per this scenario, preference (k_s) will be assigned to each task based on the size of the task and an inter arrival time, then EnQueued to priority queue Q_i with all k_s. From the priority queue all K_s task are fetched and an agent identification process is initiated and searching for available agents. Let the Q_i is the variation of a multi server system and maximum 'X' tasks are allowed in the queue.

This process is repeated until the Q_i gets empty. Since it is a dynamic process PBTS Algorithm is applied for every unit time. Then agent's searches for the resources, requested by

the tasks in DRA table and the resource allocation process is initiated. Figure 2 describes the control flow of the task scheduling scheme. Algorithm 3.1 is designed to implement PBTS approach.

Algorithm 3.1

Preference based task Scheduling Algorithm

1. **Input:** The tasks T = $\{t_1, t_2, t_3, \ldots t_n\}$
2. **Output:** Connection by the agent A=$\{a_1, a_2, a_3 \ldots a_m\}$
3. **do** for all $t_1, t_2 \ldots t_n$ in Task scheduler
4. **Consider** for all task size S_x
5. **if** 'm' tasks having same size s_x then
6. **Check** for Arrival time Ia_t
7. **Assign** the $k_1, k_2 \ldots k_m$ based on Ia_t
8. EnQueue $Q_i \leftarrow k_m$ *where* $1 < i,m < \infty$
9. **else** assign $k_1, k_2 \ldots k_s$ based on size S_i
10. EnQueue $Q_i \leftarrow k_s$ *where* $1 < i,s < \infty$
11. **While** Q_i! = Null (for all $k_1, k_2 \ldots k_s$)
12. **fetch** k_n from Q_i
13. **Search** for a_m in A
14. **If** a_m is identified then
15. $a_m \leftarrow k_s$
16. **else**
17. **Check** for next Agent a_{m+1}
18. **End**

3.2 Dynamic resource allocation (DRA)

Resource table is a virtual table used to determine availability of resources for allocation. The table consists of resources provided by all PM's by means of its Virtual Machines (VM).This table is updated periodically for each request given by the user and its responses for the request. In this paper the resource table is used in such a way that agents check for availability of resources. Here three different notations are used for the process TRT, Tr_{in}, Tr_{out}.

TRT - Tag of Table,
Tr_{in} - Task request in
Tr_{out} - Task request out (23)

As from Eq. (3), Let Agents A = $\{a_1, a_2, a_3, \ldots . a_j\}$ and resources in table is referred as ar_n.The simple process involved here is whenever, $a_j = tr_{in}$, TRT is initiated to check for ar_n.If ar_n available, Tr_{out} ==1. Therefore a connection $C_n = 1$ (here C_n is the connectivity) is made by the agent between client and the server.

3.3 Resource allocation algorithm

The resource allocation algorithm makes use of DRA table to allocate the resources for a client request. Each PM will have multiple numbers of VM's and all will be ready for an allocation based on the user requests. The resource table is updated periodically for each prioritized task taken from Q_i. And this updating process happens dynamically during the run time. The resource table consists of a tag TRT (Task Request Tag) (Xu et al. 2015). This is enabled always by two variable tr_{in} and tr_{out}, As per the Eq. (23) tr_{in} is the task request in and tr_{out} is the task request out. Updating the resource table is necessary so as to allocate the resources dynamically. Resource table is updated whenever the TRT is initiated based on tr_{out}. And also it checks for predicted inputs.

Algorithm 3.2 and 3.3 designed to implement the resource allocation and for updating the DRA table. In this algorithm, all agents with jobs with TRT, tr_{in} and tr_{out} are taken as inputs and updated resource table is considered as an output. Initially check for ar_n when the situation arises a_m same as tr_{in}, then TRT will be initiated. Check whether ar_n is available, if so tr_{out} will be assigned as 1 and the resource

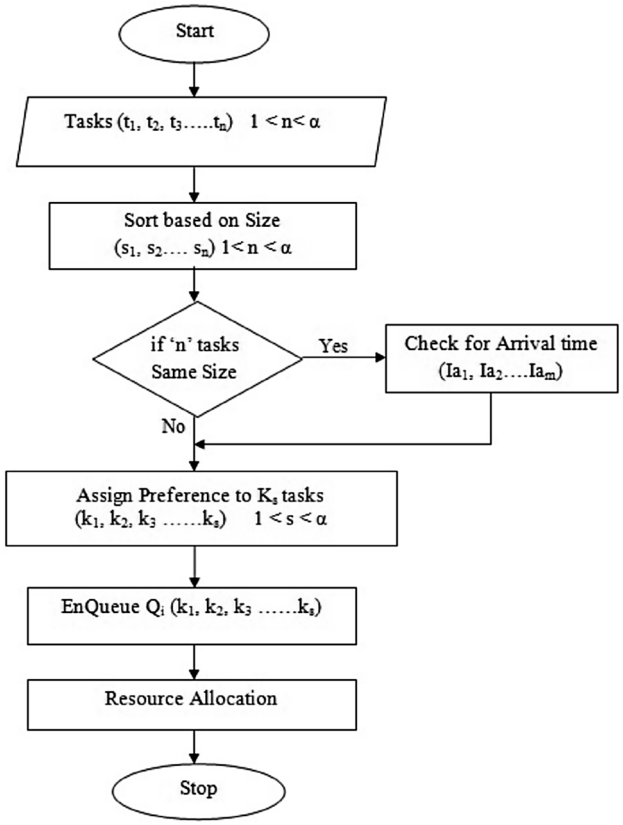

Fig. 2 Flow chart of task scheduling scheme

table is updated with all available resources. Then priority of the each job will be checked and using Power Minimization Algorithm (PMA) the resource table is updated simultaneously. In this proposed PBTS algorithm task size (S_x) is considered as a primary factor. So the Smaller tasks are given the maximum priority than larger tasks. Then all smaller tasks are assigned to available resources identified and gets executed. So that the average waiting time of smaller tasks are minimized. Mean waiting time and Mean response time of the queue can be calculated using Eqs. (12) and (13). Average waiting time and Average response time of the queue could be obtained using Eqs. (21) and (22). It depends on the mean length of the queue, which is obtained using Eqs. (19) and (20). Number of tasks available in the queue can be found using the Eqs. (17) and (18). The results shows that response time and waiting time of the scheduling process are minimized, because of average waiting time of all small tasks are eliminated. Such a way both PBTS algorithm and DRA table updating algorithm contributes the efficiency improvement in dynamic resource allocation process.

Algorithm: 3.2

Resource table Updating Algorithm

1. **Input:** Agents A= {a_1, a_2 ,a_3,…a_m}, TRT, tr_{in} and tr_{out}.
2. **Output**: Updated resource table with resources ar_n.
3. **Check** for ar_n.
4. Check **If** a_m= tr_{in}, then TRT is initiated.
5. **If** ar_n is available, tr_{out} ←1.
6. **update** with available resource
7. **Check** for K and use PMA algorithm.
8. **end**

Algorithm 3.3

Resource Allocation based on PBTS Algorithm

1. **Input:** The Client request N= $\{n_1,n_2,n_3 \ldots n_i\}$.
2. **Output:** A Complete Resource allocation Scheme with PM's status.
3. **Get** the inputs N.
4. **Assign** each $n_i \rightarrow t_k$
5. **for** all task T, do algorithm 3.1.
6. **if** a_m=ar_n **then**
7. C_n==1 else C_n=0.
8. **Update** the DRA table using algorithm 3.1& 3.2.
9. **Continue** step 3 to 5 for all incoming jobs for each unit time't'.
10. **End**

4 Power management scheme

Proposed PBTS algorithm improves the efficiency of the resource allocation process in terms of response time and task completion ratio. Power consumption in datacenter became one of the factor affects the performance of the resource allocation process. The energy consumption in each data center is directly proportional to the Resource exploitation. In order to increase availability of resource and minimize the energy consumption in the cloud environment, it is mandatory to reduce the power being used by each physical machine (PM). To achieve the goal, measuring the future workload and based on that resource allocation is implemented in this research work. Simulation results shows that it minimizes the Power consumption in the data centers.

4.1 Measuring future workloads

Requests are raised from the user side dynamically and frequently. So, we estimate the number of VM's needed for future. Based on past VM's utilization data observed, we predict the future VM requirements. Work load per unit time 'w' and each second workload is observed. Based on the observation next second workload is calculated. From the simulation values, it is observed that the proposed prediction technique gives improved results. We have also explored other prediction techniques adopted by other works like (Nejad et al. 2014) and Linear Auto Regression method (Metwally et al. 2015) is embraced broadly in other works. In that predicted results are modeled as linear function from the previous results. In this proposed Prediction mechanism, the future job size, inter-arrival time, execution time can be determined earlier. Based on the predicted values idle physical machines are put either in sleep or ON mode. So energy consumption by Idle VM is conserved. Let us consider for N incoming jobs, the predicted jobs are given by,

$$P = \{ p_1,p_2,p_3 \ldots p_q \} \text{ Where } 1 < q < \infty \tag{24}$$

The parameters of each predicted jobs are the inter arrival time 'u', job size 's' and an execution time 'e'. In this algorithm,double exponential smoothing method is utilized and to improve the accuracy of predection results multivarient regression method is accessed. By using Brown's double exponential smoothing a series of predicted parameters can be obtained. This DES method gives better output. Also only job size is predicted by using ES method proposed in one of the referred work. All the three parameters are obtained by the concept below. First Considering the job size to be predicted, the forecast value is determined as.

$$F_{t+m} = \{ a_t + m \times b_t \} \tag{25}$$

By DES method [2], the initial step is $s_0' = x_0$ and $s_0'' = x_0$ and a_t and b_t is obtained by,

$$a_t = 2 \times s_t' - s_t'' and\ b_t = \frac{\alpha}{(1-\alpha)\left(s_t' - s_t''\right)}$$
$$s_t' = \alpha xt + (1-\alpha)s_t' - 1 \text{ and } s_t'' = \alpha xt' + (1-\alpha)s_t'' - 1 \quad (26)$$

where α is a smoothing factor.

By using this method, the predicted job size can be determined. Similarly predicted jobs inter arrival time and its execution time are obtained.

4.2 Power minimization algorithm (PMA)

The power management algorithm decides, which PM needs to go to sleep mode and which PM needs to be ON. Then it monitors all PM's available in the network and stores the current states and utilization chart of PM's to cloud cluster. The proposed power minimization algorithm is an effective, simple and easy to adopt. Though various energy saving approaches are available, our proposed approach provides better energy savings in datacenters. Because all other methods are adopted to save the energy by tuning the operating frequency of CPU and to reduce the energy at a compiler level, hardware and application level. After the time out period only, it will switch to lower state. So, there is no possibility of an energy savings in this period. To solve this issue, we propose an advanced machine learning technique for workload prediction and the energy reduction at OS level by shifting the PM state to sleep mode completely. Server level and data center level energy saving are adopted with VM cluster. This is about to change idle PM's state to sleep mode. Considering various types of energy consumed in the cloud data centers. PM baseline energy consumption, active VM's energy consumption for Application Execution and Energy generated by Internal and external Communications are some of the major energy consumption types. In this proposed work, Energy consumption by Idle VM's is taken for consideration. From the predicted values observed from the previous module, we try to reduce an energy consumed by the idle machines when they are in use. This will be achieved by keeping the PM's in ON mode or PM's in sleep mode. The Number of PM's state is necessary to determine the usage of resources. This is necessary to keep the resource table updated for an allocation of resources. Let the total capacity of the CPU utilization of all PM_i is given by:

$$\text{Cap T} = \sum_{i=1}^{n} \text{cap i} \quad (27)$$

This approach reduces the energy consumption of by free PM's by using the concepts based on modified best fit algorithm, all PM_i's are arranged in the order based on (1) PM's status (sleep mode or ON mode) and (2) PM's Capacity and memory. Algorithm 4.1 is designed to minimize the power consumed by the idle machines using the proposed approach. Let ar_n and p_n are the available resource and predicted jobs respectively. All the PM's are arranged in descending order and threshold value is decided. The sorting process in descending order is done as currently running PM's as first and Idle PM's as last and an incoming predicted workload W_p is calculated. Let an average workload of all PM's are given by W_{pmi}. If $W_p \leq W_{pmi}$, use available PM in ON mode. If $W_p \geq W_{pmi}$, make idle PM in ON mode based on the requirement of W_p given by the predicted results. This process will be repeated until all w_p assigned to all W_{pmi}. Then remaining Idle PM's are put into sleep mode. Such a way the state of the PM is determined. Then DRA table is updated simultaneously based on the PM's state. So the energy consumed by the idle PM's are conserved. Both PBTS algorithm and PMA module access the DRA table dynamically based on the outputs calculated.

Algorithm 4.1

Power Minimization Algorithm (PMA)

1. **Algorithm** Power management (ar_1, ar_2, ar_3…ar_n & p_1, p_2, p3….p_n)
2. **Input:** ar_1, ar_2… ar_n and p_1, p_2…p_n
3. **Output:** Table with PM ON and PM Sleep.
4. **Check** for PM's Usage & Cap T
5. **Sort** all PMs based on Usage & Cap T
6. **do** for all p_1,p_2,p_3…p_n
7. **If** (W_p <= W_{pmi}) then
8. Search ar_n for p_n in current PM
9. **If** found then
10. ar_n ← p_n
11. **else** search for next PM
12. **else** Idle PM← ON Mode
13. Repeat from 8 to 13 until (P= =null)
14. Set Idle PM's ← Sleep Mode
15. **Update** DRA table for W_p and ar_n
16. **End**

PBTS algorithm minimizes the response time, time delay and improves the successful task completion ratio. Then PMA module minimizes the power consumption in cloud data centers and supports green computing (Han et al. 2019). In Such a way the proposed approach improves an efficiency of resource allocation process in cloud. Simulation results analyzed an efficiency improvement.

5 Experimental setup and results

The simulations and experimental results are observed to evaluate an efficiency of the proposed system. From the results, some factors are analyzed such as framework Evaluations, Resource Utilization, Acceptance ratio, Execution time and Power Management.

The most commonly used cloud sim software is adopted for this purpose. The scalable cloud data center is created by using a series of PM's. Datacenters along with resource agents are initiated. Each datacenter with multiple numbers of hosts and their corresponding VM's are initiated. The client tasks are referred to cloudlets and cloudlet scheduler schedules the incoming tasks.

Table 1 provides the Hardware requirements and Table 2 provides the cloud sim specification of the proposed system. The simulation results are analyzed with different metrics like make span, resource utilization, execution time, and completion ratio and power consumption. Proposed (ITSEPM approach) improves system efficiency and is compared with some existing methods.

We evaluate the performance of our proposed system with different metrics such as energy consumption, data center resource utilization, acceptance ratio and execution time.

The performance of this approach is shown by means of its values and graphical representation of the same. Resource Utilization is calculated as the ratio of the used resource such as CPU, memory, bandwidth to their total capacity in data center. The tasks are given to this model are arrived

Table 1 Hardware Requirements

Required	Component specification
Hard Disk	1 TB
Processor	Intel® Pentium® CPU G2030 @ 3.00 GHZ
RAM	4 GB
Operating System	Windows (X86 ultimate) 64-bit OS
System	64 Bit OS System

Table 2 Simulation Specification

Component	Specification	Values
Virtual Machine	Host	4
Cloudlets	No of tasks	30–300
	Length of task	1600–3400
Physical Machine	Memory	540
	Bandwidth	25,00,00
	Storage	500 GB
	MIPS/PE	500

as a batch. We use batch processing concept for job arrival technique. A set of jobs are arrived simultaneously at t = 0.

Then we use our proposed method of allocation for allocation purpose. The jobs are prioritized by task scheduling concept employed in our framework.

The prioritized tasks are given to the agents which allocates the available resources from the resource table. During this process acceptance ratio, Bandwidth, execution time are calculated and analyzed with standard scheduling algorithms FCFS and Round Robin methods.

5.1 Performance evaluation

Performance could be evaluated using some metrics like make span, resource utilization, execution time, task completion ratio and power consumption.

5.1.1 Make span (MK_{Span})

Make span is a time taken for an execution of the set of tasks from the beginning to till completion of last task.

Let T_C is the maximum time to complete all tasks then make span is represented as follows

$$MK_{Span} = MAX(T_C)$$

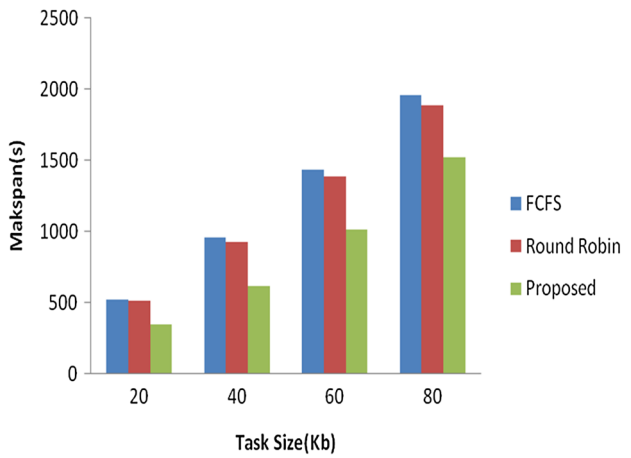

Fig. 3 Comparison of makespan

The simulation result gives shorter Mk $_{span}$ when compared with other two taken algorithms, A comparative analysis graph Fig. 3, provides decreased Make span and it increases efficiency about 30% with other two algorithms.

5.1.2 Task response time (TS_{Res})

Response time of a task is defined as the time taken from the task's arrival to complete execution. Let TS_{cmpt} be the completion time of the task and TS_{art} be the arrival time of the task, response time TS_{Res} can be represented as follows,

$$TS_{Res} = TS_{cmpt} - TS_{art}$$

The response time analysis at different levels of task execution is observed during the simulation and the values are tabulated below (Table 3).

The timing shows an increased performance of our proposed algorithm with other algorithms.

A comparative graph shown in Fig. 4 shows the lesser response time when compared with existing methods.

Table 3 Response Time Analysis

Offline	Execution time
(i) Task monitoring and scheduling	20 min
(ii) Workload prediction	10 min
Online	
(iii) Connection to agents	0.050 s
(iv) Response to users	0.010 s
(v) Power management	2.015 s

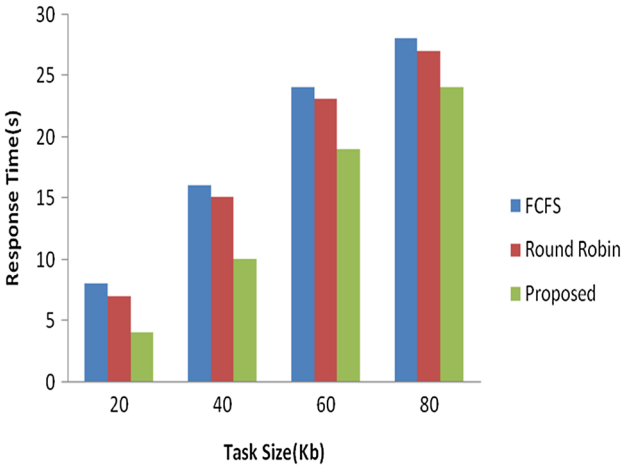

Fig. 4 Comparison of response time

5.1.3 Task completion ratio (TC_R)

It is a ratio between number of tasks successfully completed and a number of tasks submitted to the system at a particular time. Let N_{tcmpt} be the number of tasks successfully completed and N_{sbmt} be the number of tasks submitted, the Task completion ratio can be represented as follows,

$$TC_R = N_{tcmpt}/N_{sbmt}$$

A set of tasks completed for a certain period of time are analyzed during simulation. The simulation result gives a better performance ratio. A comparative study with an existing algorithms is made to show our proposed method performance. The graph shown in Fig. 5 proves the superiority of our proposed approach with other algorithms.

5.1.4 Resource utilization (R_U)

It is denoted as amount of available resources utilized by the tasks to get executed. Let R_{avl} is denoted as available resources and R_{nu} be the unused resource, Resource Utilization (R_U) can be represented as

$$R_U = R_{avl} - R_{nu}$$

The resource utilization involves both CPU and Memory utilization of our proposed work. The Utilization percentage of the proposed algorithm is always higher when it is compared with other two algorithms. The graph shown in Fig. 6 gives the maximum resource utilization. This is mainly due to the PMA algorithm proposed in our work which involves the idle PM's to be in OFF state. It takes part in resource utilization mechanism.

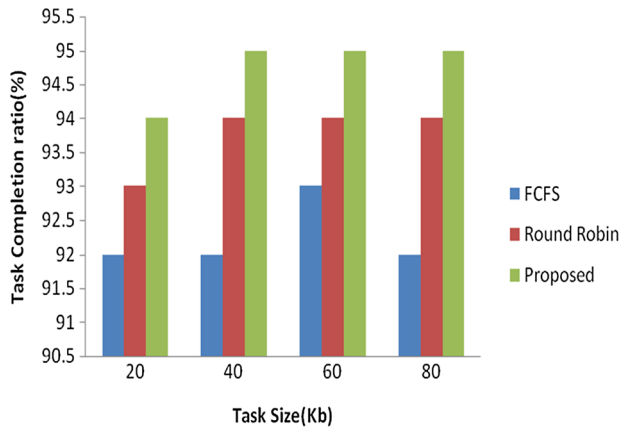

Fig. 5 Comparison task completion ratio

5.1.5 Power consumption

It can be defined as a unit of power consumed by the all PMs during the resource allocation process. In this proposed work, power management module is implemented to minimize the energy consumption. Though there are so many power consumption technologies available in real time data centers such as Dynamic voltage and Frequency scaling, Dynamic Voltage Scaling and resource hibernation, they do not consider the virtualized environment.

When compared to the existing methods (Xu et al. 2015), our proposed approach for power minimization gives the optimal results. The graph shown in Fig. 7 proves that. The power management model presented in this paper reduces the energy consumed by Idle VM, energy consumption for

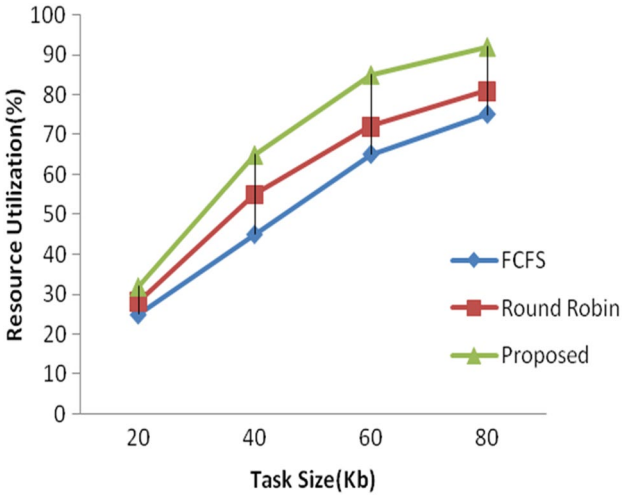

Fig. 6 Comparison of resource utilization

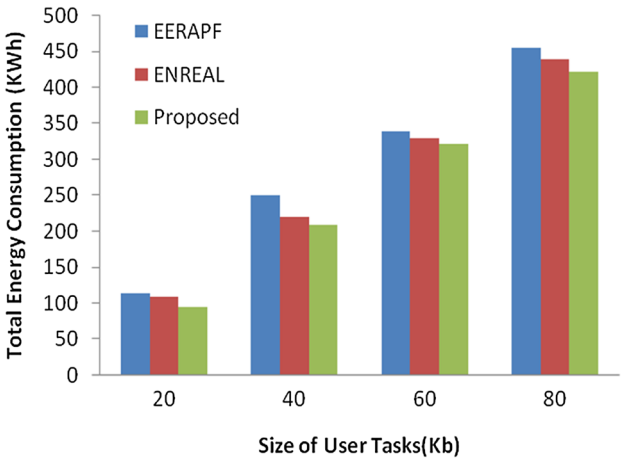

Fig. 7 Comparison power consumption

an external and an internal communication and baseline energy consumption by PM.

6 Conclusion and future work

In this research work, we proposed a Dynamic Resource Allocation approach with Improved task scheduling and an enhanced power management modules to improve the efficiency of resource allocation process. This is the novel approach for energy efficient task scheduling process for dynamic resource allocation in cloud environments. In the proposed task scheduling mechanism, we considered the task size and an inter arrival time. But there may be a possibility that some of the tasks have same size and same inter arrival time. It will be addressed in future works. Then an improved PMA module minimizes energy consumption by changing the mode of Idle VM's. The preference based task allocation algorithm, resource table updating algorithm and power management module are adopted with dynamic workload prediction. Power minimization is achieved in datacenters by switching off the unused servers by identifying them using prediction algorithm. It helps to improve the efficiency of resource allocation process in terms of response time, resource utilization, task completion ration and make span. It improves the efficiency of dynamic resource allocation process. Simulation results and comparative analysis shows the efficiency improvement in all aspects.

References

AlShahwan F, Faisal M, Ansa G (2016) Security framework for RESTful mobile cloud computing Web services. J Ambient Intell Hum Comput 7(5):649–659

Chou LD, Chen HF, Tseng FH, Chang HC, Chang YJ (2016) DPRA: dynamic power-saving resource allocation for cloud data center using particle swarm optimization. IEEE Syst J 12(2):1554–1565

Dabbagh M, Hamdaoui B, Guizani M, Rayes A (2015) Energy-efficient resource allocation and provisioning framework for cloud data centers. IEEE Trans Netw Serv Manag 12(3):377–391

Dai W, Qiu L, Wu A, Qiu M (2016) Cloud infrastructure resource allocation for big data applications. IEEE Trans Big Data 4(3):313–324

Du J, Zhao L, Feng J, Chu X (2018) Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. IEEE Trans Commun 66(4):1594–1608

Gawali MB, Shinde SK (2018) Task scheduling and resource allocation in cloud computing using a heuristic approach. J Cloud Comput 7(1):1–16

Han S, Min S, Lee H (2019) Energy efficient VM scheduling for big data processing in cloud computing environments. J Ambient Intell Hum Comput. https://doi.org/10.1007/s12652-019-01361-8

Hwang K, Dongarra J, Fox GC (2013) Distributed and cloud computing: from parallel processing to the internet of things. Morgan Kaufmann, Burlington, pp 1–57

Kwak J, Kim Y, Lee J, Chong S (2015) DREAM: dynamic resource and task allocation for energy minimization in mobile cloud systems. IEEE J Sel Areas Commun 33(12):2510–2523

Landa R, Charalambides M, Clegg RG, Griffin D, Rio M (2016) Self-tuning service provisioning for decentralized cloud applications. IEEE Trans Netw Serv Manag 13(2):197–211

Li D, Chen C, Guan J, Zhang Y, Zhu J, Yu R (2015) DCloud: deadline-aware resource allocation for cloud computing jobs. IEEE Trans Parallel Distrib Syst 27(8):2248–2260

Mashayekhy L, Nejad MM, Grosu D, Vasilakos AV (2015) An online mechanism for resource allocation and pricing in clouds. IEEE Trans Comput 65(4):1172–1184

Metwally K, Jarray A, Karmouch A (2015) A cost-efficient QoS-aware model for cloud IaaS resource allocation in large datacenters. IEEE Int Conf Cloud Netw. https://doi.org/10.1109/CloudNet.2015.7335277

Nejad MM, Mashayekhy L, Grosu D (2014) Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. IEEE Trans Parallel Distrib Syst 26(2):594–603

Nguyen HHC, Solanki VK, Van Thang D, Nguyen TT (2017) Resource allocation for heterogeneous cloud computing. Resource 9(1–2):1–15

Pahlevan A, Qu X, Zapater M, Atienza D (2017) Integrating heuristic and machine-learning methods for efficient virtual machine allocation in data centers. IEEE Trans Comput Aided Des Integr Circuits Syst 37(8):1667–1680

Peng M, Zhang K, Jiang J, Wang J, Wang W (2014) Energy-efficient resource assignment and power allocation in heterogeneous cloud radio access networks. IEEE Trans Veh Technol 64(11):5275–5287

Qu L, Assi C, Shaban K (2016) Delay-aware scheduling and resource optimization with network function virtualization. IEEE Trans Commun 64(9):3746–3758

Shi W, Zhang L, Wu C, Li Z, Lau FC (2015) An online auction framework for dynamic resource provisioning in cloud computing. IEEE/ACM Trans Netw 24(4):2060–2073

Teymoori P, Sohraby K, Kim K (2015) A fair and efficient resource allocation scheme for multi-server distributed systems and networks. IEEE Trans Mob Comput 15(9):2137–2150

Tseng FH, Wang X, Chou LD, Chao HC, Leung VC (2017) Dynamic resource prediction and allocation for cloud data center using the multi objective genetic algorithm. IEEE Syst J 12(2):1688–1699

Wang X, Wang X, Che H, Li K, Huang M, Gao C (2015) An intelligent economic approach for dynamic resource allocation in cloud services. IEEE Trans Cloud Comput 3(3):275–289

Xiao Z, Song W, Chen Q (2012) Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans Parallel Distrib Syst 24(6):1107–1117

Xu X, Dou W, Zhang X, Chen J (2015) EnReal: an energy-aware resource allocation method for scientific workflow executions in cloud environment. IEEE Trans Cloud Comput 4(2):166–179

Zaman S, Grosu D (2013) A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds. IEEE Trans Cloud Comput 1(2):129–141