Asian Journal of Information Technology 15 (3): 472-480, 2016
ISSN: 1682-3915


# Priority Queue Scheduling Approach for Resource Allocation in Cloud

[1]R. Madhumathi and [2]R. Radhakrishnan
[1]Department of Computer Science and Engineering, Sri Ramakrishna Engineering College, 641022 Coimbatore, Tamil Nadu, India
[2]Faculty of Information and Communication Engineering, Vidhyamandhir Institute of Technology, 638051 Erode, Tamil Nadu, India

**Abstract:** Resource utilization in cloud is very challenging because of its dynamic nature especially in heterogeneous applications. Even though, Virtual Machine (VM) technology permits multiple workloads to be processed simultaneously, it does not ensure application performance at certain scenarios. Therefore, cloud datacenter providers either do not render any performance assurance or choose static rather than dynamic VM allocation that results in ineffective usage of resources. This study tackles the problem of resource allocation inside a datacenter which runs various kinds of application workloads, specifically non-interactive and transactional applications. It specifically focuses on how available amount of resources such as memory and Virtual machine Central Processing Unit (VCPU) of the current cloud infrastructure have been utilized according to the user requests, projects and applications assigned to the cloud environment. The available resources are allocated to each VM both within and across VCPU. In this research, Priority Queue (PQ) scheduling algorithm is proposed. Fair share policies are defined at each queue to deal with dynamic priority of the requests submitted by the user. According to the dynamic priority of user requests, they are scheduled at two levels on the basis of their resource accessibility. The proposed scheduling algorithm hosts the virtual machines on cloud nodes to utilize the resources in a well-organized manner and the performance is evaluated and compared with conventional scheduling methods.

**Key words:** Cloud computing, Virtual Machine (VM), resource allocation, scheduling, priority

## INTRODUCTION

In recent decades, the idea of virtualizing computer system's resources, including processors, memory and Input/Output (I/O) devices has become popular to improve sharing and utilization of computer systems. Cloud Computing (CC) is a set of large pool of usable and accessible virtualized resources (Rochwerger *et al*., 2010; Vaquero *et al*., 2009). It offers three service models such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). In IaaS cloud, resource is rendered to users as leases. One of the benefits of the IaaS cloud with regard to the computational virtual machines offering is end-users flexibility and efficiency. The Virtual Machine Monitor (VMM) is called a hypervisor and mediates access to the physical hardware presenting to each guest operating system which is a group of virtual platform interfaces (Mills *et al*., 2011; Gupta *et al*., 2013). In CC, where applications have got variable and dynamic necessities, capacity management, dynamic resource allocation and demand prediction becomes complicated (Zhang *et al*., 2010).

Resource allocation is one of the key features pertaining to cloud computing. Though cloud has huge resources and provides services, efficient methods to allocate resources is still a challenging task during certain demands. Hence, resource allocation has to be efficiently addressed in all the computing areas such as grid computing, datacenter management and many more. During resource allocation, the main factor to be concentrated is the maximum utilization of cloud resources in a best manner which improves the performance of the cloud in the market and it fulfills the consumers' requirement by providing seamless services (Leivadeas *et al*., 2013; Lee and Zomaya, 2012). Reduction in energy consumption and better resource management can be gained by mapping VMs to physical machines dynamically. VMM like Xen provide a mechanism for mapping VMs to physical resources (Barham *et al*., 2003). This mapping is concealed from the users of the cloud. For instance, users making use of the Amazon EC2 service do not have any knowledge about the location where their VM instances are running (Clark *et al*., 2005). It is

**Corresponding Author:** R. Madhumathi, Department of Computer Science and Engineering, Sri Ramakrishna Engineering College, 641022 Coimbatore, Tamil Nadu, India

dependent on the cloud provider to assure the Physical Machines (PMs) that are underlying to have enough resources to satisfy their requirements.VM live migration technology renders the benefits of modifying the mapping between VMs and PMs when the applications are hosted in the cloud (Nelson *et al*., 2005; Madhumathi and Radhakrishnan, 2015). The performance of the system mainly depends on the scheduling part where the best strategy has to be chosen to correlate the resources with the arrived request. The main objective of scheduling is to dispatch the demanded resources by mapping the available resources to demanded request (Tsai *et al*., 2014; Madhumathi *et al*., 2015). The other metrics such as the time at which the request has arrived, the order of processing the request, demanded resource, handling the instant requests with varying rate should also be taken into consideration. In this study, a novel approach has been proposed to handle the requests by introducing priority queue scheduling algorithm which indeed helps to achieve best resource utilization.

**Literature review:** A number of resource allocation and scheduling algorithms have been presented in the literature especially in the last two decades. However, several research works have not considered much about the memory (RAM) and CPU requirements to optimize the computing performance of VMs in CC. Therefore, proper utilization of available memory and CPU on the host machines is needed to ensure better placement of the user requests on proper VMs.

Jung and Sim (2011) proposed a model for adaptive resource allocation which assigns the consumer's job to a suitable data center. This adaptive resource allocation model to find an appropriate data center based on the location of consumer and the workload of data center in cloud computing environment. In their experiment, the adaptive resource allocation model shows higher performance than the linear and the random resource allocation model in terms of average allocation time and geographical distance. Liu *et al*. (2012) presented a novel resource scheduling model by incorporating loyalty-based trust mechanism into CC. A resource scheduling framework proposed in their research, takes customer satisfaction and the capacity of cloud platform into account. Experimentation results show that the trusted computing based on loyalty can improve the successful transaction rate and can meet the requirements of cloud computing.

Li and Li (2013) cloud resource allocation optimization algorithm is achieved through an iterative algorithm. In each iteration, the cloud users compute the unique optimal payment to SaaS provider under the deadline constraint to maximize the cloud user's satisfaction. To evaluate the performance of iterative algorithm adopt, the metrics such as resource cost, execution success ratio and resource utilization. Li *et al*. (2010) proposed two algorithms for the task scheduling namely, Adaptive List Scheduling (ALS) and Adaptive Min-Min Scheduling (AMMS). Here, the submitted tasks are partitioned in the form of DAG. These algorithms adjust the resource allocation based on the updated number of actual task executions. Adaptive procedure works even better in a homogeneous cloud system in which every task runs faster in some kind of VMs than in some other kinds.

Lee *et al*. (2014) proposed a resource allocation model based on the performance of node (Li and Li, 2013). There are two components used in this system namely node performance analyzer and VM allocation manager. The node performance analyzer helps to identify the current workload on each virtual machine and VM allocation manager allocates the incoming request to the virtual machines using the information provided by the former. The performance of each node is identified by using best fit strategy. Xiao *et al*. (2013) achieved resource multiplexing by means of virtualization. Datacenter resources are allocated dynamically based on the demand. They concentrate more on measuring the skewness, i.e., dynamic change occurred during the utilization of resources. Skewness algorithm achieves both overload avoidance and green computing for systems with multi-resource constraints. Gu *et al*. (2012) achieved optimization through the Improved Genetic Algorithm (IGA). Resources are allocated for the VM requests optimally using the dividend policy. The rate of utilization of resources is higher when compared to the traditional GA. Xiao *et al*. (2013) proposed a dynamic resource allocation system which optimizes the resources allocated to virtual machines based on application demand.

Kim *et al*. (2008) presented a single-hop ad hoc grid heterogeneous environment. Here, tasks are independent, arrive unpredictably and have priorities and deadlines. In their research, match and schedule methods are used to map tasks into devices such that the number of highest priority tasks completed by their deadlines. Ding *et al*. (2015) introduced a resource scheduling mechanism enabled with a relevance feedback network to satisfy a user's resource requirements in a better way. The feedback information that is integrated in one cycle will adjust the resource matching effectively and selection in the next subsequent cycle. Simulation results showed that this relevance feedback scheduling mechanism is very efficient in meeting users' diverse needs and it also performs better with respect to the resource utilization rate from the cloud provider's perspective. Peng *et al*. (2015)

Table 1: Mathematical notations

| Notations | Description |
|---|---|
| $R = \{r_1, r_2, ...r_n\}$ | The set of resource instances available in a cloud system where each $r_j$ $(1 \leq j \leq n)$ is an identifier of a specific resource |
| $C_m$ | Capacity of each server in terms of memory |
| $C_{CPU}$ | Capacity of each server in terms of CPU |
| $r_s$ | Total resource utilization |
| $r_{smax}$ | Maximum capacity of each server for all resources |
| $s_{Current}$ $(c_m)$ | Availability of the RAM at each server |
| $s_{current}(c_{CPU})$ | Availability of the VCPU at each server |
| $r_{ui}$ | User request |
| $r_{ui}$ (m) and $r_{ui}$(CPU) | Memory and VCPU requests of users |
| $Q^1$ | First Come First Serve (FCFS) queue |
| $Q^2$ | Memory and VCPU usage queue |
| W | Expected waiting time per users in the system |
| $W_q$ | Expected waiting time per users in the queue |
| $L_q$ | Average number of users in queue |
| $\lambda$ | Mean arrival rate |
| $\mu$ | Mean service rate |
| $Q_{FS}$ | Fair-share allocation queue |
| $b[Q_i]$ | Benefit received by each queue at different level at a regular interval |
| $B[Q_i]$ | Growing benefit received by each queue at different level at a regular interval |

proposed random task scheduling based on reinforcement learning in cloud computing to demonstrate the efficiency of the task scheduling in terms of arrival rate, number of VMs used and response time.

## Problem specification

**Motivation:** Resource allocation in cloud computing is in fact related to the actions performed by the cloud providers for allocation and utilization of resources in the limited cloud environment such as to attain the requirements of application running over the cloud. In order to fulfil the on-demand request of the user, the users need to specify the kind and the amount of resources required for their applications. In order to acquire the available resources, the time when the request has been received and the processing order of those requests has to be considered. Therefore, the cloud provider has to process the requests from the user based on a policy defined by the cloud administrator. The order of requests is also considered for allocating resources on the host nodes of the cloud. The on-demand request of the cloud consumer will be processed among the host machines in the cloud system. In doing so, the cloud providers face some challenges in making use of resources in an efficient way. Resource partition is the predominant method used to provide the resources for an on-demand resource request by the users. Allocation of required resources for the new VM creation on the host machines depends on the available resources so that the potential resources are not wasted. Since, unused resources of running VMs could not be allocated to the other request from different users, resources are allocated more than the demand and more than one user competes for the same resource. Rapid elasticity is one of the major features of the cloud computing in which the resources of cloud can be easily provisioned and de-provisioned automatically based on the demand resulting in fluctuation of resource availability and resource demand.

**Problem statement:** Choosing the best available resources and avoiding overloading are the most important factors to be considered while handling dynamic user requests. But, a policy issue remains as how to determine the mapping in an adaptive manner in order to satisfy the resource demands of VMs when the number of PMs employed is reduced. This becomes a huge challenge when the resource requirements of VMs are heterogeneous because of the diverse set of applications that they run and when the workloads increase and decrease. This has been addressed by virtualizing the available resources by using dedicated techniques. Hence, this study aims to achieve the following goals:

- The key component is availability of memory in RAM. Requests are assigned with priority as it rose with the demand based on the availability of memory in all the host nodes. Hence, RAM memory allocation is a major issue which is to be addressed
- Request has been assigned with priority as it rose with Virtual CPU (VCPU) based on the availability of total number of VCPU in all the physical nodes. Applications running on VMs may slow down its execution when there is insufficiency in VCPU on the computer servers
- The fair share policies are defined at each queue to deal with priority of the requests submitted by the user. The PM capacity should be adequate to meet the resource requirements of all VMs that are running on it. Else, the PM will be overloaded and can result in deteriorated performance of its VMs

**Problem formulation:** This section clearly discusses the important problem of dynamic on-demand request from the user and to allocate the required resources (memory and VCPU) for the VM creation on specific hosts. Table 1 represents the various notations used throughout this study.

Consider 's' as server or host and capacity of each server is defined in terms of memory ($C_m$) and VCPU ($C_{CPU}$), i.e., $C_m+C_{CUP} = r_s$ i.e. Here, the concept of skewness is introduced to calculate the irregularity in the usage of multiple resources on a server. Maximum capacity of each server is denoted as $r_{smax}$ A resource $R = (r_1,..., r_n)$ represents the set of available resources. The resources (i.e., VMs) are independent of each other. VMs may exist either in homogeneous or in heterogeneous environments. A job 'J' is a logical unit of work that is executed by a resource (R) or machine (if the job can be automated). Let $r_{ui}= r_{ui}$ (m) + $r_{ui}$ (CPU) specify the usage to 'i' users (i = 1 to m). The skewness resource of a server 's' is defined as:

$$\text{Skewness(s)} = \sqrt{\sum_{i=1}^{m}\left(\frac{r_{ui}}{r_{s\max}}-1\right)^2} \quad (1)$$

From Eq. 1, the overall usage of the server resources can be improved by minimizing the unevenness in the usage of server resources. The proposed PQ Scheduling algorithm executes periodically to evaluate the resource allocation status based on the request demands of the user. A specific VM server is defined as a hot spot in case the usage of any of its resources goes above a hot threshold. This specifies that the server is overloaded and therefore few VMs running on it has to be migrated further. The temperature of a hot spot in the server is defined as the square sum of its resource utilization, that is:

$$\text{Temperature (s)} = \sqrt{\sum_{i=1}^{m}\left(r_{ui}-r_{s\max}\right)^2} \quad (2)$$

User sends a request with required memory which will not exceed the maximum capacity of any of the host nodes in the cloud system, i.e., $r_{ui}$ (m) < ($rs_i^0$ +$rs^1_i$ +...+$rs^n_i$) and it is same for the VCPU, i.e., $r_{ui}$ (CPU) < ($rs_i^0$ +$rs^1_i$ +...+$rs^n_i$). The maximum resource utilized in all the hosts for all the requests are calculated as:

$$RU = \sum_{i=0}^{m} r_{ui} \quad (3)$$

Hence, the waiting time of the request is defined as by adding the time required for processing the accepted tasks in the queue with its service rate. That is:

$$W = W_q + \frac{1}{\mu} \quad (4)$$

$$W_q = \frac{L_q}{\lambda} \quad (5)$$

Makespan of the task is defined as the maximum time taken for the completion of all the tasks in a given application. If resources are assigned to requests of a queue with the FCFS scheduling individually regardless of fair-share policies then, the share assignment is not properly used. Hence, the fair-share policy is defined at queue level as in the form of:

$$\text{Queue\_Fairshare} = \text{USER\_SHARES}\ [[r_{ui}, \text{number\_shares}]...] \quad (6)$$

Where:

$r_{ui}$ = The request from the user
number_shares = The number to indicate the share assignment to the $r_{ui}$

For example:

$$Q_{FS} = [r_{ui},1][\text{Queue},1] \quad (7)$$

where, $Q_{FS}$ is the fair-share allocation queue. Here, two shares are assigned, one to request and another to queue, which is shared by the other requests inside the queue. Each request in queue namely $Q^1_{FS}$, $Q^2_{FS}$ has equal importance at different levels:

$$Q^1_{FS} = [r_{ui},1][Q^1,1] \quad (8)$$

$$Q^2_{FS} = [r_{ui},1][Q^2,1] \quad (9)$$

When each user request has come up with different requirement at regular intervals, they are assigned with priority 'P' for getting the host instantly for VM creation. Based on the availability of the memory ($C_m$) and VCPU ($C_{CPU}$) the priority has rated as low '$P_L$' or high '$P_H$' for all the requests. Initial placement of the VM on any host is to be done without any priority assignment since, the resource availability is more for the first request. Then, each upcoming request $r_{ui}$ is placed at first level of queue as it has come with all resource requirements such as memory, CPU and disk size, etc., when the request $r_{ui}$ ($r_{ui}$ (m) and $r_{ui}$ (CPU) is put into the second queue as it need to assign either low or high priority according to both the user demand and current availability of the resources in the cloud system. Considering the above scenario which has a couple of cases to be handled with the memory and CPU requirements are as follows.

Case 1: Consider the RAM demand from user request and availability of the same resource from the cloud will be $r_{ui}$ (m) and $s_{current}$ ($c_m$), respectively. Let, the demand of memory request be below the current capacity of the

available memory. That is $r_{ui}$ (m) <$s_{current}$ ($c_m$) then, the $r_{ui}$ (m) has assigned with high priority $P_H$ and processes them as soon as possible:

$$P_H = r_{ui}(m) < s_{current}(C_m) \quad (10)$$

Let the demand of memory request be above the current capacity of the available memory. That is, $r_{ui}$ (m) $\geq s_{current}(c_m)$ then, the $r_{ui}$ (m) is assigned with low priority $P_L$ and delegates them in the second level of queue for current CPU availability:

$$P_L = r_{ui}(m) < s_{current}(C_m) \quad (11)$$

Case 2: Consider VCPU demand from user request and availability of the same resource from the cloud will be $r_{ui}$(m) and $s_{current}$ ($c_m$), respectively. Let, the demand of CPU request be below the current capacity of the available CPU. Then:

$$P_H = r_{ui}(CPU) < s_{current}(C_{CPU}) \quad (12)$$

If the request has satisfied the above condition, the $r_{ui}$ (CPU) has assigned with high priority $P_H$ and delegates them to the second level of queue for getting demanded memory. As it satisfies Eq. 12, it has assigned with high priority and delegates them to the upper level of queue for VM creation. Then:

$$P_L = r_{ui}(CPU) \geq s_{current}(C_{CPU}) \quad (13)$$

If the request has satisfies the Eq. 13, then, the $r_{ui}$ (CPU) has assigned with low priority $P_L$ and keeps them in the same level of queue till it satisfy the high priority condition in case 2. The dynamicity has been incorporated for priority assignment which made all requests to be processed instantly for VM creation. Hence, no request has been put under starvation for long duration.

## MATERIALS AND METHODS

**System model:** The cloud framework for allocation of virtual machines to physical host nodes is shown in Fig. 1. The proposed system consists of a node performance analyzer, VM fair share scheduler and VM allocation control manager. This research considers an IaaS cloud system which delivers a basic on-demand storage and compute capacities over the internet. These computational resources are provided in the form of

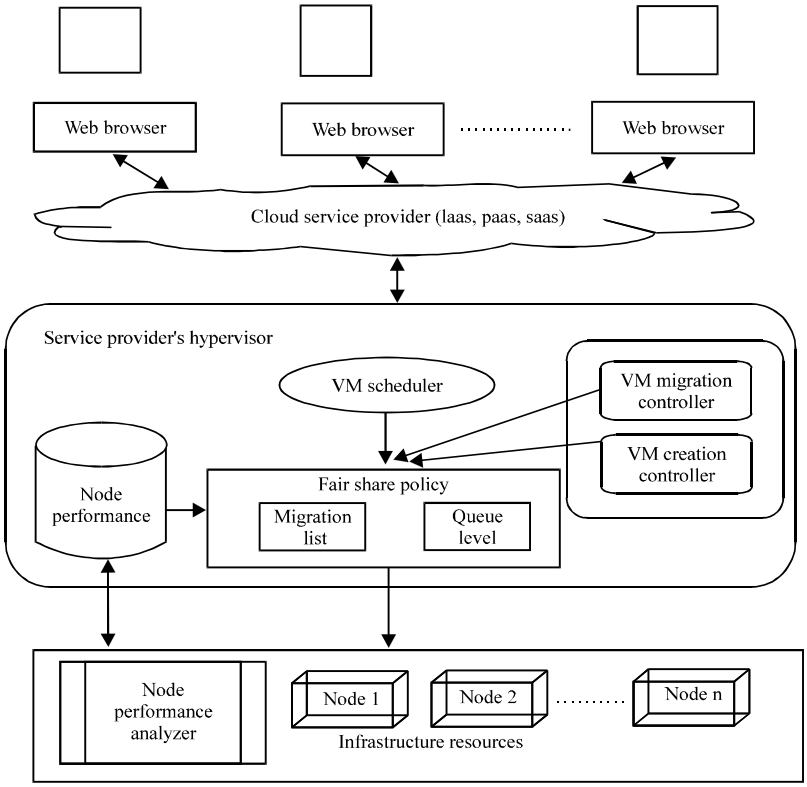


Fig. 1: System model for VM resources

Virtual Machines (VMs) which is deployed in a provider's data center. Without loss of generality, VMs from different clouds are offered in different types and each of which has different characteristics. In the proposed cloud resource allocation mechanism, every provider has VM scheduler software running in its data center. The VM scheduler's role is to direct incoming instance requests to suitable compute nodes that in turn will serve the requests by launching and making available new virtual machines sized according to the specifications of those requests by the user. Based on the incoming user requests, the scheduling policy (fair-share) is defined. All the host state information has been sent to the compute nodes to take scheduling decisions periodically. Fair-share policy defines the order of requests to be placed in the queue or a host partition used in cloud environment. This queue level concept will help the cloud infrastructure to instantly identify the high level and low level requests in the queues. Each request has been assigned a priority based on the current properties of the cloud infrastructure.

**Fair share policy:** A fair-share policy defines the order of requests to be placed in the queue or a host partition used in cloud environment. This fair share policy allocation assures to configure certain queues and leave the remaining using FCFS based scheduling. Each policy has been assigned to fixed number of shares to each requests or group of requests belonging to a project. These shares are used to represent a fraction of resource availability in the cloud system. In this architecture, two levels of queues are considered, in which the first level queue follows, first come first serve and second level queue will accommodate requests from the first level queue if it requires changing the priority of the request according to the availability of RAM and VCPU in the cloud infrastructure. With this strategy, the priorities of the requests are classified as low level and high level requests. The request which has more shares had been assigned with high priority and are put into the first queue in the form Q1FS (Fig. 1).

Each share assignment is defined with a pair of attributes enclosed in a square bracket such as user and fraction of share in number, i.e., (user, number_shares). The user attributes specifies the requests posted by the user and the number_shares attribute identifies number of shares of available cloud resources assigned to each user request, collective requests or individual request in a group. No queue has own priority, they are purely defined with fair share policy. For every incoming request $r_{ui}$ from user belonging to any one among two queues, $Q = Q(r_{ui})$ if $B[Q_i]+B(r_{ui}, Q(r_{ui}))>b[Q_i]$ is satisfied then, the user request for further processing is submitted. For every queue of requests, $Q_i$, a priority $P_H[Q_i]$or $P_L[Q_i]$ is chosen according to the availability of resources as defined in the problem formulation. These $P_H[Q_i]$ and $P_L[Q_i]$ are ordered in the same way as FCFS at each level of queue. For all the queue of requests arbitrarily assigned it with priorities $P_H$ and $P_L$ such that $P_L<P_H$. Priority is computed based on the following formulae for all the incoming requests $r_{ui}$ at any queue with priority $P_H[Q_i]$or $P_L[Q_i]$:

$$P = B[Q_i] + B(r_{ui}, Q(r_{ui}))>b[Q_i]\ P_H$$

**Algorithem 1: Queue scheduling based on requests**

Input: Requests$r_{ui}$,...$r_{ui}$ server $s$,$s_{current}$ ($c_m$) and $s_{current}$ ($c_{CPU}$)
Output: Resources allocation results in terms of task
foreach$r_{ui}$ in Queue $Q^1$ do
if not initial request then
$S_{current}$← current available state of server by last n requests
if $S_{current} > r_{ui}$ then
return priority ← $Q^1(r_{ui})$ else
return $Q^2 \leftarrow Q^1(r_{ui})$
End if // $s_{current} > r_{ui}$
else
returns$_{current}$← current available state of *server*by new requests
end if // not initial request
end

**Dynamic priority queue scheduling algorithm:** For all the requests from the same project, the priority is allocated dynamically for adopting the changes in the resource state in the cloud environment using Load Sharing Facility (LSF). The priority is dynamic because the changes in any one of its resource states keep varying at regular intervals. Here, only memory and VCPU of the host machines are considered. So, the priority assessment is done according to the changes in the memory and VCPU. Usually, the request's dynamic priority decreases once the request starts and then it increases when the request turns to be an instance. In this research, both memory and VCPU based dynamic priority adjustment are implemented as follows:

$$\text{Fairshare}_{\text{finetune}}(\text{RAM}) = (1+r_{ui})\ ((\frac{s_{current}(C_m)}{\sum r_{ui}})/(r_{ui}(m)r_{s\max})) \quad (15)$$

$$\text{Fairshare}_{\text{finetune}}(\text{VCPU}) = (1+r_{ui})\ ((\frac{s_{current}(C_{CPU})}{\sum r_{ui}})/(r_{ui}(CPU)r_{s\max})) \quad (16)$$

$$\text{Fairshare}_{\text{finetune}}(u) = \text{Fairshare}_{\text{finetune}}(\text{RAM}) + \text{Fairshare}_{\text{finetune}}(\text{VCPU}) \quad (17)$$

Current availability of each server is calculated as scurrent ($C_m$) = $C_m$-$r_{ui}(_m)$ and scurrent($C_{CPU}$) = $C_{CPU}$-$r_{ui}(_{CPU})$, respectively. The total number of request received by server is denoted as $\Sigma r_{ui}$. This algorithm automatically allocates resources as well as monitor infrastructure properties for an on-demand user request based on the fair share policies defined by the cloud infrastructure manager. Since, nature of the cloud user and cloud providers are dynamic, the capability of the cloud system is maintained at different levels of queues. Here, PQ scheduler is aware of the present status of VMs in the cloud and their communication. So, they can make schedule decision when using the information like the earliest resource available time in a certain cloud.

**Algorithm 2: priority assignment algorithm**

Input: $Q^2$ ($r_{ui}$)
Output: Low priority ($P_L$) or High priority ($P_H$)
for each $r_{ui}$ do
if $r_{ui}$ < fairshare (u)
    $r_{ui} \leftarrow P_H$
return $P_H$(Request)
else $r_{ui} \leftarrow P_L$
return $P_L$ ($r_{ui}$)
for each Priority($r_{ui}$) do
        if $P_H$ then
        put[$P_H$] in Upper Level Queue ($Q^1$)
        else
        put[$P_L$] in Same Level Queue ($Q^2$)
        end if
end
end if
end

## RESULTS AND DISCUSSION

**Experimental setup:** The proposed model is simulated using CloudSim toolkit wherein the performances of the proposed priority queue approach and the existing algorithms are evaluated. This tool provides basic classes that describe data center, virtual machine, computational resources and policies for scheduling and provisioning of resources (Komarasamy and Muthuswamy, 2015). The tasks are submitted to the VM for their execution. These tasks are considered as a poisson process because, the jobs are submitted in a specific time interval. The bandwidth of the VMs varies with respect to the system architecture. Existing scheduling algorithms such as first come first serve and Random Scheduling (RS) are considered for the evaluation with the proposed PQ technique.

Figure 2 shows the impact of resource utilization for the proposed and the existing scheduling algorithms. The resource utilization of PQ is 77% which is increased upto5-6% when compared to other scheduling algorithms. Moreover, the processing speed of VMs is fully utilized in PQ due to the deployment of VM scheduler. This

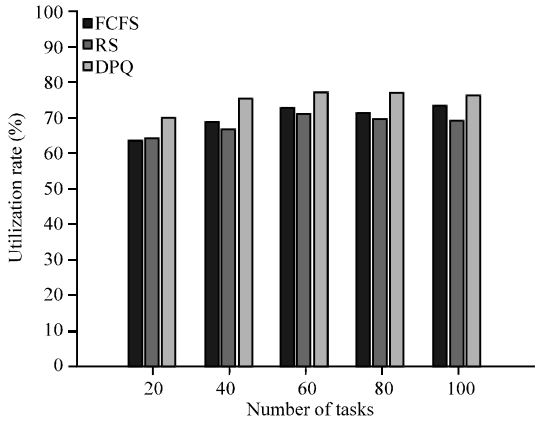


Fig. 2: Impect of resource utilization for various methods

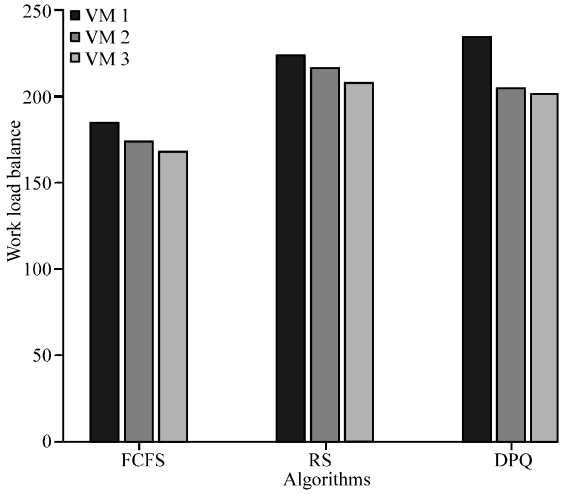


Fig. 3: Comparison of VM workload balance

proposed work tries to find the optimum scheduler instantly and more over finds the high and low level requests in the queues depending on CPU and RAM usage.

Figure 3 shows the VM workload balance results of proposed and existing scheduling algorithms. The FCFS scheduling scheme dispenses user request to running VM but it fails to consider the balanced workload of various VMs, thus taking the longest response time. Meanwhile, the RS scheme only guarantees the balanced workloads of various VMs. However, when the interval of tasks arrival is short and remaining buffer size of each VM is full, the workload balance of various VMs cannot be solved yet. But, the proposed PQ scheduling schema workload is balanced between various VMs.

Figure 4 shows the average makespan of the various scheduling. It is clearly observed that, with the increase in the number of tasks, the average makespan also gets

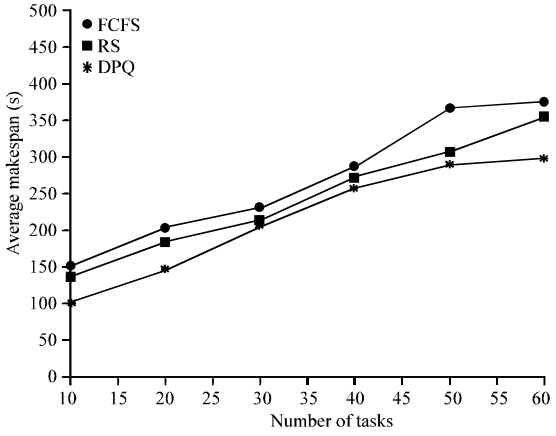


Fig. 4: Impect of average makespan for various methods

increased but the average makespan of PQ approach is observed to be lower of 50-70 sec when compared to FCFS and RS techniques.

## CONCLUSION

This study has proposed a novel design, execution and estimation approach for resource management system for cloud computing services. The proposed PQ Scheduling algorithm multiplexes virtual to physical resources relatively based on the varying needs. The main goal of this algorithm is to attain utilization of available cloud resources as much as possible. Skewness metric is used here to combine VMs with different resource characteristics appropriately so that the capacities of servers are well utilized. Both memory and virtual CPU are considered as cloud resources for experimental purpose. Finally, it is shown that both the resources are used to a maximum level successfully in cloud simulator environment. The resources are utilized significantly with lesser response time and average makespan due to the efficiency of the proposed PQ scheduling algorithm. This approach is efficient in scheduling the resources appropriately to the desired task through which the overall performance is improved. Extended PQ scheduler can be designed for processing both dependent and independent tasks. Another option to extend this work is considering other resources such as I/O performance and network bandwidth and mix of QoS to provide a more flexible approach. The key reason here is that PQ is able to manage different workload and exploits their usage patterns and QoS requirements to obtain efficient utilization of datacenter resources.

## REFERENCES

Barham, P., B. Dragovic, K. Fraser, S. Hand and T. Harris *et al.*, 2003. Xen and the art of virtualization. Proceedings of the 19th ACM Symposium on Operating Systems Principle, October 19-22, 2003, Bolton Landing, USA., pp: 164-177.

Clark, C., K. Fraser, S. Hand, J.G. Hansen and E. Jul *et al.*, 2005. Live migration of virtual machines. Proceedings of the 2nd Conferen\ce on Symposium on Networked Systems Design and Implementation, Volume 2, May 2-4, 2005, Boston, MA., USA., pp: 273-286.

Ding, D., X. Fan and S. Luo, 2015. User-oriented cloud resource scheduling with feedback integration. J. Supercomput., 1: 1-22.

Gu, J., J. Hu, T. Zhao and G. Sun, 2012. A new resource scheduling strategy based on genetic algorithm in cloud computing environment. J. Comput., 7: 42-52.

Gupta, A., L.V. Kale, D. Milojicic, P. Faraboschi and S.M. Balle, 2013. HPC-aware VM placement in infrastructure clouds. Proceedings of the 2013 IEEE International Conference on Cloud Engineering (IC2E), March 25-27, 2013, IEEE, Redwood City, CA., pp: 11-20.

Jung, G. and K.M. Sim, 2011. Agent-based adaptive resource allocation on the cloud computing environment. Proceedings of the 2011 40th International Conference on Parallel Processing Workshops (ICPPW), September 13-16, 2011, IEEE, Taipei, Taiwan, pp: 345-351.

Kim, J.K., H.J. Siegel, A.A. Maciejewski and R. Eigenmann, 2008. Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling. Parallel Distrib. Syst. IEEE. Trans., 19: 1445-1457.

Komarasamy, D. and V. Muthuswamy, 2015. Deadline constrained adaptive multilevel scheduling system in cloud environment. KSII. Trans. Internet Inf. Syst., 9: 1302-1320.

Lee, H.M., Y.S. Jeong and H.J. Jang, 2014. Performance analysis based resource allocation for green cloud computing. J. Supercomput., 69: 1013-1026.

Lee, Y.C., and A.Y. Zomaya, 2012. Energy-efficient resource utilization in cloud computing. J. Supercomput., 60: 268-280.

Leivadeas, A., C. Papagianni S. Papavassiliou, 2013. Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning. Parallel Distrib. Syst. IEEE. Trans., 24: 1077-1086.

Li, C. and L. Li, 2013. Efficient resource allocation for optimizing objectives of cloud users, IaaS provider and SaaS provider in cloud environment. J. Supercomput., 65: 866-885.

Li, J., M. Qiu, J.W. Niu, Y. Chen and Z. Ming, 2010. Adaptive resource allocation for preemptable jobs in cloud systems. Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications (ISDA), November 29-December 1, 2010, IEEE, Cairo, Egypt, ISBN: 978-1-4244-8134-7, pp: 31-36.

Liu, Y., S. Yang, Q. Lin and G.B. Kim, 2012. Loyalty-based resource allocation mechanism in cloud computing. In: Recent Advances in Computer Science and Information Engineering. Qian, Z., L. Cao, W. Su, T. Wang and H. Yang (Eds.). Springer Berlin Heidelberg, Berlin, Germany, pp: 233-238.

Madhumathi, R. and R. Radhakrishnan, 2015. A resource allocation strategy in cloud using roulette wheel selection method. Proceedings of the 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), October 8-10, 2015, IEEE, Noida, India, pp: 341-345.

Madhumathi, R., R. Radhakrishnan and A.S. Balagopalan, 2015. Dynamic resource allocation in cloud using bin-packing technique. Proceedings of the 2015 International Conference on Advanced Computing and Communication Systems, January 5-7, 2015, IEEE, Coimbatore, India, ISBN: 978-1-4799-6437-6, pp: 1-4.

Mills, K., J. Filliben and C. Dabrowski, 2011. Comparing vm-placement algorithms for on-demand clouds. Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), November 29-December 1, 2011, IEEE, Athens, Greece, ISBN: 978-1-4673-0090-2, pp: 91-98.

Nelson, M., B.H. Lim and G. Hutchins, 2005. Fast transparent migration for virtual machines. Proceedings of the USENIX Annual Technical Conference on General Track, April 13, 2005, VMware Inc., California, USA., pp: 391-394.

Peng, Z., D. Cui, J. Zuo, Q. Li, B. Xu and W. Lin, 2015. Random task scheduling scheme based on reinforcement learning in cloud computing. Cluster Comput., 18: 1595-1607.

Rochwerger, B., C. Vazquez, D. Breitgand, D. Hadas and M. Villari *et al*., 2010. An architecture for federated cloud computing. Cloud Comput., 81: 391-411.

Tsai, C.W., W.C. Huang, M.H. Chiang, M.C. Chiang and C.S. Yang, 2014. A hyper-heuristic scheduling algorithm for cloud. Cloud Comput. IEEE. Trans., 2: 236-250.

Vaquero, L.M., L.R. Merino, J. Caceres and M. Lindner, 2009. A break in the clouds: Towards a cloud definition. Comput. Commun. Rev., 39: 50-55.

Xiao, Z., W. Song and Q. Chen, 2013. Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans. Parallel Distrib. Syst., 24: 1107-1117.

Zhang, Q., L. Cheng and R. Boutaba, 2010. Cloud computing: State-of-the-art and research challenges. J. Internet Serv. Appl., 1: 7-18.