International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 1, February 2021, pp. 671~685
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i1.pp671-685

A latency-aware max-min algorithm for resource allocation in cloud

Kashish Ara Shakil[1], Mansaf Alam[2], Samiya Khan[3]
[1]Department of Computer Science, Princess Nourah Bint Abdulrahman University, Saudi Arabia
[1,2,3]Department of Computer Science, Jamia Millia Islamia, India

Article Info

Article history:

Received Feb 26, 2020
Revised Jul 10, 2020
Accepted Aug 5, 2020

Keywords:

AHP
Big data
Cloud computing
Priority scheduling
Resource allocation

ABSTRACT

Cloud computing is an emerging distributed computing paradigm. However, it requires certain initiatives that need to be tailored for the cloud environment such as the provision of an on-the-fly mechanism for providing resource availability based on the rapidly changing demands of the customers. Although, resource allocation is an important problem and has been widely studied, there are certain criteria that need to be considered. These criteria include meeting user's quality of service (QoS) requirements. High QoS can be guaranteed only if resources are allocated in an optimal manner. This paper proposes a latency-aware max-min algorithm (LAM) for allocation of resources in cloud infrastructures. The proposed algorithm was designed to address challenges associated with resource allocation such as variations in user demands and on-demand access to unlimited resources. It is capable of allocating resources in a cloud-based environment with the target of enhancing infrastructure-level performance and maximization of profits with the optimum allocation of resources. A priority value is also associated with each user, which is calculated by analytic hierarchy process (AHP). The results validate the superiority for LAM due to better performance in comparison to other state-of-the-art algorithms with flexibility in resource allocation for fluctuating resource demand patterns.

Corresponding Author:

Kashish Ara Shakil,
Department of Computer Science,
Princess Nourah Bint Abdulrahman University,
Airport Road, Riyadh 11671, Kingdom of Saudi Arabia.
Email: kashakil@pnu.edu.sa

1. INTRODUCTION

Cloud computing is the next generation technology and exploits the vision of utility computing where computing resources are provided as a service [1, 2] via an internet connection. According to NIST [3], it can be thought of as a model for enabling a ubiquitous, convenient and on-demand way of accessing a pool of resources. It has led to the growth of virtual and physical devices, which in turn has led to an escalation of data generated on a continuous basis. Thus, big data is generally characterized by high volume, variety, and velocity of data [4, 5]. Virtualization techniques act as the backbone of cloud technology [6, 7]. They involve constant allocation and reallocation of resources, which is done at the service provider level [8] and provides the clients with a view of an infinite computing and storage capacity. The service providers [9] are constantly flooded with demands for high-performance computing and newer resources. This constant and varying demand for resources by the clients is what leads to the development of novel scheduling techniques across a distributed system, such as cloud.

Until recently, meeting the demands of the consumers has been the sole concern of service providers, without paying any attention to the optimum allocation of resources for maximization of profits. However, even though the number of cloud providers and consumers is increasing, there is also an escalating amount of resource wastage. Moreover, there is a need to maintain high service-level performance along with optimization of the resource usage.

The existing scheduling algorithms for real-time tasks are based on aspects like infrastructure, provisioning of resources for power management [10-17] and cost efficient selection [18]. Unfortunately, none of these scheduling algorithms takes into account the fluctuating demands of clients and user priorities. In a cloud environment, scheduling algorithms have to address heterogeneity [19] at all levels, which in itself is a big challenge. Besides this, network latency is also an important factor as the presence of latency degrades the crispness of system response and has an adverse effect on factors like energy consumption [20]. This work proposes LAM, which is a latency aware algorithm for resource allocation in the cloud. It also takes into account fluctuating demands and user priorities. The user priorities are decided upon using AHP technique. Furthermore, unlike the previous works in literature, LAM shows a better resource consumption and thereby optimum allocation of resources.

SMI cloud [21] provides a framework for ranking and comparing different cloud services to the cloud users. Apart from these, there are other approaches such as multi-criteria decision making (MCDM) [22], which have been explored for ranking and selection of different cloud services. These initiatives are meant to facilitate decision making for the cloud users but no concrete efforts have been made for decision making at the service provider level. Decision making at the service providers side usually involves allocation of resources, distribution of workload, catering to the dynamic resource requirements by the consumers and consumer prioritization. The work done in this paper helps the service providers make important decisions regarding allocation of resources through LAM.

The problem of resource allocation in cloud environment involves twin steps, which are identification of user requirements and mapping of user requests to the actual resources at the service provider's side. This paper focuses on the latter that is optimum utilization of resources. Therefore, measures need to be adopted by the cloud service providers to ensure high availability of resources along with optimum utilization. The optimum utilization in this context means reduced total cost of ownership (TCO) and increased return on investment (ROI). This makes optimum allocation of resources, a challenging and complex task. Moreover, the computational requirements of users are growing so fast that increasingly large number of servers and resources are required to handle them. In particular, the cloud service providers are required to allocate resources for satisfying quality of service (QoS) requirements via a service level agreement (SLAs) and ensure optimum allocation of resources thereby leading to efficient computing resource utilization.

The main objectives of this study are to present our vision, discuss resource allocation and develop an algorithm for resource allocation in cloud environment [23, 24] such that cloud computing can be adopted as a more sustainable technology leading to technological advancements and better utilization of resources for the coming generations. Thus, the major contributions of this work are:

a. Modeling of the proposed system based on parameters such as machines available, user priority, memory, bandwidth and CPU usage (fraction of CPU usage), in addition to a few others. The key objectives are listed below.
b. Formulation of resource allocation problem.
c. Development of an algorithm for efficient allocation of resources in the cloud, which not only helps in optimum allocation of resources but can also, be implemented readily without any special requirements.
d. Evaluation and validation of the proposed algorithm by comparing it with five existing algorithms Greedy-R, Greedy-P, FCFS, dynamic resource allocation [25] and preference-based resource allocation schemes [18] in the cloud through extensive simulations. The results show that the proposed algorithm shows better resource consumption with increasing number of tasks and arrival interval and hence helps in improved infrastructure performance.

The remainder of this paper is organized as follows. Section 2 gives an insight into work that is already done in the literature relating to the proposed algorithm. In section 3, models that are used in this paper are described and resource allocation problem is stated. Section 4 describes our proposed algorithm i.e. LAM. In section 5, performance evaluation of LAM is done based on the number of tasks and task arrival interval. Finally, the paper ends with the conclusion and future works in section 6.

2. RESEARCH METHOD

2.1. Literature survey

Resource allocation has always received a great deal of importance from the research community. Thus, algorithms and methods related to it have been rigorously evaluated. Resource allocation in cloud

computing is a non-cooperative problem as the consumers who wish to access the same resources are competitors and thus, they are reluctant to cooperate with each other [26]. A vast number of the solutions proposed, generally target the execution time as the solution objective but the optimum allocation of resources is rarely a concern as the resources available are considered to be infinite. Thus, none of the approaches consider allocation from the service provider's perspective.

Condor [27] and Load Sharing facility [28] are traditional resource management approaches based on system-centric resource allocation. These resource allocation strategies implicitly assume that all the job users have equal priorities and do not take into consideration the levels of usage of services by its users. Thus, they fail to fulfill the key requirements of cloud computing and utility computing characterized by fluctuating resource needs. Fluctuating resource demands refer to the change in resource usage demands by the cloud users. Sometimes, the resource demands may be high in case of peak computing requirements and low at other times. The algorithm proposed in this paper has been designed for handling such kinds of computing needs and takes into account the amount of usage of resources and the priority of each user.

Scheduling of tasks, proper assignment and mapping of resources is an integral part of cloud computing. Monitoring the activities and performance of cloud is equally important in order to ensure proper resource provisioning. This monitoring of activities in the cloud involves twin perspectives:

Cloud service providers perspective and Cloud user's perspective. Most of the studies in the literature are based on the client's perspective, but in this study, we focus on the cloud service provider's perspective. The advantage of using this perspective is that it helps in optimum allocation of resources based on the dynamically changing user demands and priorities.

Cloud service providers monitor activities like allocation of resources and meeting the end users demand. The end users, on the other hand, monitor the quality of service, which is being provided to them apart from data security and safeguarding data against potential threats. There are two categories under which performance monitoring can be classified according to Vineetha [29, 30] which are Infrastructure performance and Application performance. Infrastructure performance involves measuring the performance of cloud resources that are provided as a service to the cloud users, which may include network, storage, and servers. This study takes into consideration infrastructure level performance by monitoring resource usage pattern of infrastructure level resources. Application performance management involves monitoring of databases and applications that provide support for application program performances.

Resource management and allocation are possible in the cloud with the help of virtualization technology. Virtualization offers its users complete transparency, but this transparency has now resulted in further complications in terms of distribution of resources and flexibility [31, 32]. The characteristics for task scheduling and resource allocation in the cloud as enlisted by Sun *et al.* [33] as a) Cater to the distribution of resources of a unified platform. This platform may involve all the different types of PCs, workstations and servers; b) Task scheduling in the cloud must be globally centralized; c) Independent scheduling of every node in the cloud; d) Task scheduling must cope up to the scalability feature of cloud computing; e) Support for dynamic scheduling depending upon the increase or decrease in demand for the number of resources; f) scheduling strategies must proceed in sets, which involves scheduling of cloud applications and scheduling of port resources. The framework proposed in this paper makes use of LAM, which is a scheduling algorithm for allocation of resources in the cloud. The algorithm proposed is flexible in nature catering to dynamically changing user and resource priorities. It also caters to the scalability and independent scheduling of nodes.

Moreno *et al.* [16] have performed analysis, modeling, and simulation of workload patterns in utility clouds of very large scale. They have carried out their study using cloud data centers having approximately 900 users who submit 25 million tasks in one month. They have modeled the scenario by extending the capabilities of cloudSim [34] framework. Their work provides a platform for researchers to simulate resource consumption patterns in the production environments. They also made several conclusions about the dependency of workload on user behavior along with tasks. According to them, a higher degree of diversity exists in user's pattern as compared to task patterns.

Tsai *et al.* [9] have proposed a Hyper-Heuristic scheduling algorithm for providing scheduling solutions in a cloud-based environment. This algorithm has been implemented on cloudSim and Hadoop. Rodriguez and Buyya [28] have put forward a particle swarm optimization-based algorithm, which meets the deadline constraint and minimizes the execution cost of workflows. They have evaluated their approach using cloudSim. We have also carried out our experiments by using cloudSim [34] rigorously on the Google cluster [35] data set for performance evaluation.

Table 1 further shows a comparative study of existing approaches of resource allocation in literature [12, 20, 22, 24, 36-38] and LAM algorithm. LAM has features such as assisting service providers with decision-making based on factors such as user priorities, network latency, duration and time of resource use. On the other hand, the aforementioned approaches assume equal priorities of the users and do not take into account network latency between different entities in the cloud environment.

Table 1. Study of existing approaches on resource management in cloud computing

Resource Management Approaches	References	Purpose	Assumptions	Drawbacks
Condor	[39]	System centric, Traditional approach	Assumes equal priorities of users and fixed user requirements	Does not consider fluctuating user requirements
Load Sharing	[28]	System centric, Traditional approach	Assumes equal priorities of users and static user requirements	Fails to serve requirements of utility and cloud computing
Polymorphic ant colony optimization	[22]	Improved resource utilization	Assumes equal priorities of users	Does not take into account user priorities
Many-Objective Virtual Machine Placement (MaVMP) problems.	[40]	Virtual Machine Placement	Assume equal priorities of users	Does not take into account user priorities and energy efficient approaches
Semi-Markov decision process	[12]	Adaptive, multi resource allocation for resource and latency-sensitive mobile applications	Assumes equal priorities of users	Greedy admission approach
Proposed Approach		Assumes cloud service providers perspective, optimum allocation of resources, better QoS adhering to SLA's	Takes into account user priorities and dynamically changing user requirements	Does not consider energy efficiency

2.2. System model

2.2.1. System overview

Figure 1 shows a broad outline of the proposed system model. The actors involved in the given system include cloud service providers and cloud users. The usage of cloud resources by a consumer is not fixed and varies depending on the consumer requirements. The system monitors this variation in usage pattern and depending upon this pattern and various other parameters such as user priorities and availability of resources, decisions regarding assignment of resources are made by the service providers. The key elements of the system, therefore, include the following:

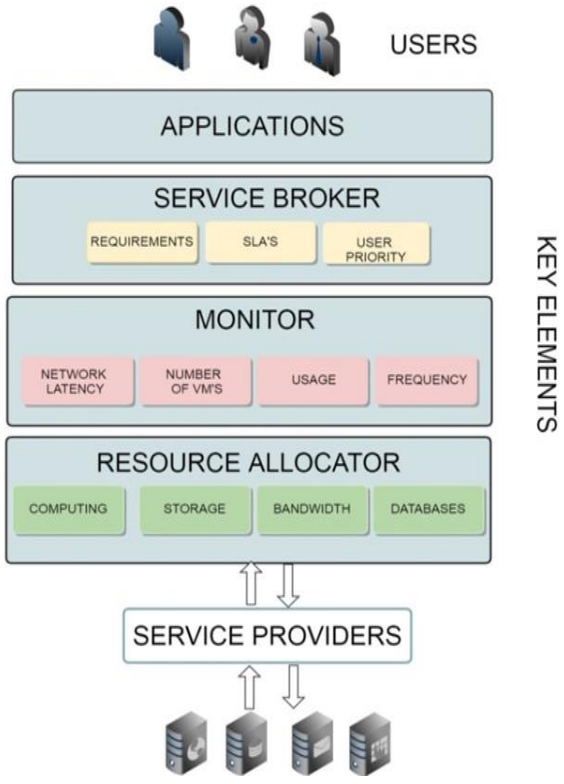


Figure 1. Proposed system model

1) Application

The application component consists of service requests that are submitted by end users for processing. An application can furthermore be defined, as an entity to which resources are allocated, be it an operating system process, a complete application or a data warehouse job.

2) Service broker

This component has the responsibility of coordinating with cloud users, interacting with them and understanding their requirements and needs. Customer SLAs with the service providers are maintained at this component. It also keeps track of whether a customer has certain special privileges or requirements that can be helpful in future for prioritizing these customers. The priority of different users submitting requests has been requests has been calculated using AHP [41]. It is one of the most popular approaches for decision-making and helps in making decisions by arranging the factors in a hierarchical manner. AHP is preferred over other decision-making approaches like multiple attribute utility theory (MAUT) [21] and outranking [42] because it is based on pairwise comparison of utility and weighing functions allowing scope for flexibility and ability to check inconsistencies. It reduces biasness in decision making by providing powerful consistency evaluation mechanism. Figure 2 shows the AHP hierarchy for cloud users. Here, the different users can be pairwise compared to one another based on factors like type of tasks being submitted, SLAs and computational requirements. It is assumed that the temporal demands of the tasks such as task deadlines are mentioned in the SLA requirements and they are considered while assigning user priorities. The comparison matrix can be built based on judgments carried through Saaty Rating Scale as shown in Table 2 [43]. It can be used to determine the relative importance of each of the users and thereby assign them with a priority value.

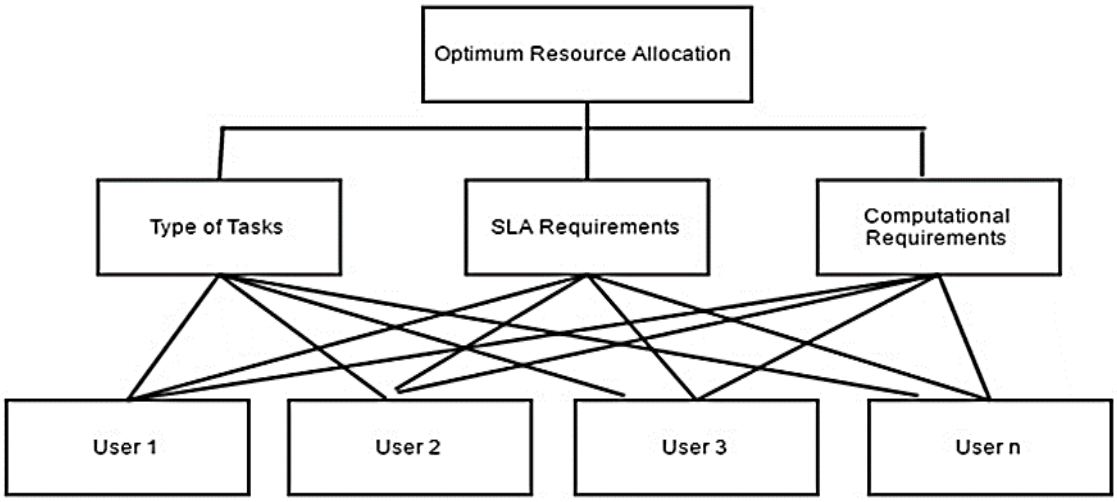


Figure 2. AHP hierarchy of cloud users

Table 2. Saaty rating scale

Intensity of Importance	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
3	Weak importance of one activity over another	Experience and judgment slightly favor one over another
5	Essential or strong importance	Experience and judgment strongly favor one activity over another
7	Demonstrated importance	An activity is strongly favored and its dominance demonstrated in practice
9	Absolute importance	The evidence favoring one activity over another is of the highest possible order of affirmation
2, 4, 6, 8	Intermediate values	When compromise is needed between the two adjacent judgments
Reciprocals of above nonzero	If activity i has one of the above nonzero numbers assigned to it when compared with activity j , then j has the reciprocal value when compared with i	

Consider 'n' cloud users (denoted by C) that are to be compared, (C_1, C_2, … C_n) and aij denotes the relative priority of user Ci with respect to Cj. It forms a square matrix called reciprocal matrix A = (a_{ij}) of order n with the constraints that $a_{ij} = 1/a_{ji}$, for $i \neq j$, and $a_{ii} = 1$, $\forall$ i. The weights are consistent if they are transitive, that is $a_{ik} = a_{ij}.a_jk \forall i$, $\forall j$, and $\forall k$. This step is followed by calculation of a vector ω of order n such that $A\omega = \lambda_\omega$. Where ω is an eigenvector and λ is an eigenvalue. For a consistent matrix, λ = n. The difference between λmax and n shows the inconsistency of the judgments taken at service provider's side. If $\lambda_{max} = n$ then the judgments are considered to be consistent. After this a Consistency Index is calculated from (λ_{max}-n)/ (n-1), that needs to be assessed against judgments made completely at random. Saaty has calculated large samples of random matrices of increasing order and the Consistency Indices of those matrices. The Consistency Index for the set of judgments is divided by the Index for the corresponding random matrix to yield Consistency Ratio (CR). According to Saaty [44], if CR exceeds 0.1 the judgments are considered too inconsistent to be reliable. Judgments are perfectly consistent if CR equals 0. In order to detect inconsistencies in elements and improve CR, when CR is more than 0.1, an induced bias matrix technique [45] can be used. It can be used to identify inconsistent elements not only in case of CR greater than 0.1 but also when CR is less than 0.1.

Example:

Let w, x, y, and z be four users submitting requests to service providers, the eigenvector of w, x, y, and z are (0.058, 0.262, 0.454, 0.226). Thus, y user can be considered to be that of highest priority, followed by x and z and w is the task of lowest priority. Aω is obtained as (0.240, 1.16, 1.916 and 0.928) and λmax is 4.18. The consistency index is 0.060. The consistency ratio is then obtained as 0.060/0.90=0.0677. Since, CR <0.1, it indicates the consistency of judgment.

3) Monitor

This component is responsible for monitoring the usage pattern of resources by the cloud users. This usage pattern includes frequency of usage of machines, the number of VMs required bandwidth usage and scalability requirements. The usage pattern has been predicted in our system using [35]. This data set consists of traces of production workloads that were acquired after running on google cluster for about 29 days. It comprises of machines that are connected by a very high bandwidth cluster network. The total number of machines used is about 12,000. The workload is divided into a set of jobs; each job consists of one or many tasks. Each of these tasks consists of linux programs with multiple processes. The data set is divided and comprises of 6 tables namely machine events table, machine attributes table, job events table, task events table, task constraints table and task resource usage table. The tasks in the data sets differ based on their resource requirements.

To observe how the number of tasks differs with respect to time, we created a plot between number of tasks and time as shown in Figure 3. This plot shows that the number of tasks arriving fluctuates with time. Thus, when a large number of tasks arrive, the resource requirement is at its peak and it decreases as the number of tasks reduces. Based on this observation, it can be concluded that proper allocation of resources is important for a cloud system.

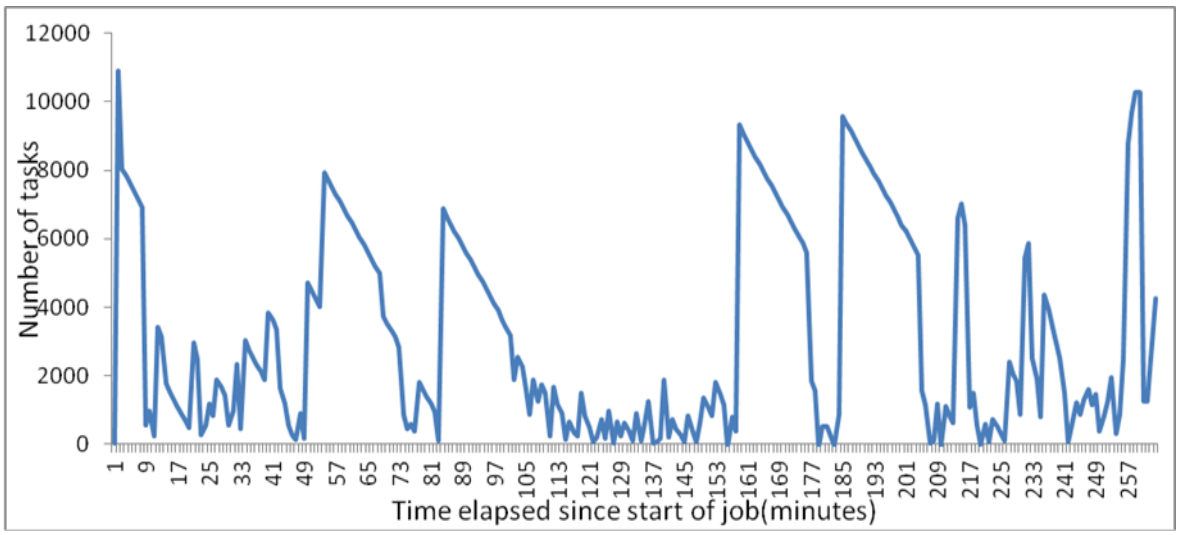


Figure 3. Number of tasks v/s time google trace

Figure 4 further shows resource usage pattern of jobs in Google cluster trace with respect to time. From this, it can be deduced that out of all the jobs that were submitted for execution only half of them were actually completed. After evaluation of all the job tables, we were able to identify the timestamp at which maximum demand for resources was made and the timestamp at which demand was least. It was also observed that resource allocation in Google trace followed a zip-f like distribution [46]. This pattern was later on used for provisioning of resources. This information can be very vital for the service providers and will aid them in making important decisions.

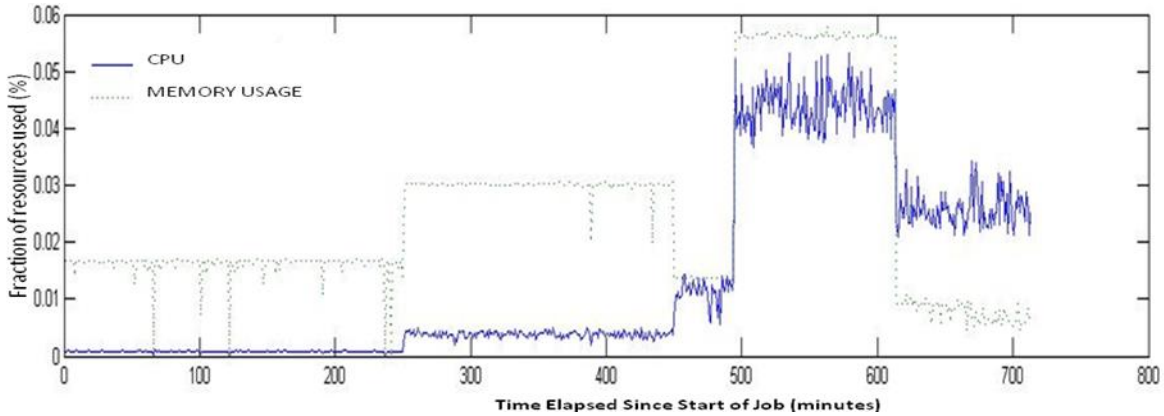


Figure 4. Resource usage pattern of google trace

4) Resource allocator

This component is responsible for allocating resources such as network, bandwidth, storage and databases to the cloud users depending upon the resource availability and resource usage pattern. LAM is applied at the level of this component. It ensures high QoS while adhering to the SLAs.

a. LAM scheduling model

LAM scheduling model assumes that machines are provisioned to the clients as soon as a request is made depending upon the user types and their respective priorities. The targeted system is composed of a set $H = \{h_1, h_2, ...h_k\}$ physical of host machines for creating virtualized resources. Each host h_k is composed of a set of virtual machines $M = \{M_1, M_2, \ldots M_i\}$. Each of these machines M_i is characterized by different resource types such as memory (m), bandwidth (b) and CPU (c). Thus, each M_i is composed of (m_i, b_i, c_i). Every service provider contains several machines in the form of virtual machines with varying resource types. These virtual machines can be started as well as stopped depending on the dynamically changing workload.

b. Task model

Tasks arriving at the service provider's level can be defined as a set $T = \{t_1, t_2 \ldots\}$. Each of these tasks submitted by a user can be represented by twin parameters, nj and ej i.e. $t_j = \{n_j, e_j\}$, where n_j and e_j are network latency and execution time respectively. Execution time of each task is calculated using length of task. This approach for calculating execution time in virtualized cloud environment has been used in literature [47]. It is assumed that the tasks arriving at the service provider's side are of varying lengths with different resource usage requirements and durations. It should also be noted that it is not feasible to know the length of each task accurately in advance. However, we can make an estimate about the jobs behavior by monitoring the access pattern of resources. Let l_j denote duration of each task (task length per size) on a virtual machine M_i and c_j denote amount of CPU usage and m_j amount of memory used by a particular task. Thus,

$$e_j = \frac{l_j}{(c_j)} \quad (1)$$

From network latency and execution time, we can calculate the total time to execute a task tj using (2).

$$e_{t_j} = n_j + e_j \quad (2)$$

c. Problem formulation

Table 3 shows the parameters and notations used in this paper. Each of the application submitted to the service provider consists of a set of T tasks. R is a set of resources such that $R_{availcpu}$, $R_{availmem}$ and $R_{availBW}$

are the amount of CPU, memory and bandwidth available at the service provider's end. R_{cpu}, R_{mem} and R_{BW} are the CPU, memory and bandwidth demanded by the consumers respectively. Let TS be task success ratio, TE be number of tasks that have been executed and TT is total number of tasks that have been submitted. Then task success ratio is given by (3).

$$TS = \left(\frac{TE}{TT}\right) \tag{3}$$

Let RR be resource utilization rate. The resource utilization rate at each host node can be calculated as (4).

$$RR = \left(\frac{\sum_{k=1}^{|H|}(\sum_{j=1}^{|T|} l_{jk})}{\sum_{k=1}^{|H|} c_k}\right) \tag{4}$$

Where, c_k is CPU capability of host and l_{jk} is length of task at each host k.

Table 3. Parameters

Symbol	Description
Mi	virtual machines available
P	priority values
C	decision criteria's
m	Memory of machine
b	Bandwidth of machine
c	CPU usage (fraction of CPU usage) of machine
T	total tasks
t	available tasks
nj	network latency for each task
ej	execution time for each task
etj	total execution time for each task
lj	duration of each task (task length per size)
xa	x-coordinate position of Cloud users
ya	y- coordinate position of Cloud users
xi	x-coordinate position of the data center
yi	y-coordinate position of the data center
di	distance between cloud users and datacenter
R	set of all the resources
rj	resources belonging to pool of resources R
vj	value associated with each resource j
RR	resource utilization rate
|H|	number of hosts
|T|	number of Tasks

Therefore, the problem can be stated as follows: Find a mapping from T onto a subset R, in order to optimize the amount of resources used. Thus, the mathematical model of the stated problem has two objective functions; the first objective is maximizing the task success ratio of all the tasks that are submitted for execution, and the other objective is minimizing the resource consumption.

$$\text{Minimize } \left(\frac{\sum_{k=1}^{|H|}(\sum_{i=1}^{|T|} l_{ik})}{\sum_{k=1}^{|H|} c_k}\right) \tag{5}$$

$$\text{Maximize } \left(\frac{TE}{TT}\right) \tag{6}$$

Subject to:

$$R_{cpu} < R_{availcpu} \tag{7}$$

$$R_{mem} < R_{availmem} \tag{8}$$

$$R_{BW} < R_{availBW} \tag{9}$$

3. LATENCY AWARE MAX-MIN ALGORITHM (LAM)

LAM is based on DMMM algorithm [48]. According to DMMM algorithm shown in Algorithm 1, if T = {t_1, t_2...t_j} be 'j' cloud user tasks, that are to be assigned resources R = {r_1, r_2...r_m}, where 'm' is the number of resources. If X_{ik} is a value calculated from {v_1, v_2 ...v_m} as the outputs of a decision matrix where X_{ik} = maximum (v_1, v_2 ...v_m). Then, DMMM algorithm selects the resource with value X_{ik} and assigns this resource to task which takes minimum time for its execution.

Algorithm 2 shows the pseudo-code of LAM algorithm. LAM first finds the maximum value associated with each resource by calling ALGO_FindResourceVal algorithm, and it then, finds out the task that requires minimum total time and assigns resource having maximum value to task having minimum execution time. This algorithm will iterate until all the tasks have been assigned resources. Algorithm 3 gives the pseudo code of algorithm for calculating maximum value associated with each resource. In algorithm ALGO_FindResourceVal, the set P={P1, P2 ...Pn} is a set of priority values associated with each user type, where n is the number of users. C = {C_1, C_2…C_{dc}} are decision criteria's and dc is the number of decision criteria's or constraints that will be adopted by the service providers.

ALGORITHM 1 DMMM ALGORITHM

```
1. Input:  T = {t1, t2,…,tj}, R = {r1, r2,…,rm}
2. Begin:
3.     For all  ti ∈ T
4.        For all rj ∈ R
5.           Xik =max(v1,v2,...,vm) /*finding the maximum value associated with each task*/
6.        End For
7.     End For
8.     Do while T ≠ Null
9.       ti(ei)= min(t1(e1), t2(e2),..., tj(ej)) /*finding out the task which requires minimum
   time for execution*/
10.           ti→ rj(Xik) /*assigning resource with maximum value to task with minimum
   execution time*/
11.          T= T-{ti}
12.          End Do While
13.        End
```

Theorem 1

The time complexity of ALGO_FindResourceVal algorithm is O (N^2)

Proof

For calculating the maximum value associated with each resource, the time complexity is O (N^2). It takes O (1) (line 6, Algorithm 3) to calculate value for a particular criterion. It takes O (N) (lines 5-7, Algorithm 3) and O (N^2) (lines 4-8, Algorithm 3). For calculating maximum value, it takes O (1) (line 10, Algorithm 3). Therefore, the complexity of our ALGO_FindResourceVal algorithm is O (N^2) +O (1) =O (N^2).

Let (x_a, y_a) be coordinate positions of cloud users and (x_i, y_i) be coordinates of the data center, both may be dispersed across different geographical locations. Let {d_1, d_2 ...d_n} be distance of data center1, data center2 and so on from the cloud user. Therefore, distance d_i, where 0<i<n+1 is given and 'n' is the total number of data centers owned by cloud service providers dispersed at varied geographical locations is given by (10).

$$d_i = \sqrt{((x_i - x_a)^2 + (y_i - y_a)^2)} \quad (10)$$

ALGORITHM 2 LAM

```
1. Input:  T = { t1, t2, ... tj}, R = {r1, r2,…,rm}
2.    Begin:
3.       For all rj ∈ R  /*perform step 4 for all the resources belonging to pool of resources
   R*/
4.          Xik =ALGO_FindResourceVal /*calculating the maximum  value associated with each
   resource */
5.       End For;
6.    Sort Xik in ascending order /*sorting the resources on the basis of their values*/
7.      Do while T≠ Null /*repeat steps 8-10 for all the tasks */
```

```
8.        ti(eti)= min(et1,et2,…etj) /*finding out task having minimum execution time*/
9.         ti → rj(Xik) /*assigning resource with maximum value to task with minimum execution
   time*/
10.           T = T -{ ti }
11.           End Do while
12.         End
```

Now distance D is calculated by (11),

$$D = \min\{d_1, d_2, \ldots d_n\} \quad (11)$$

The latency in cloud is not a function of distance alone; it includes other factors like time delay between different network entities as well. These entities can be the hosts, data centres, SaaS providers or end users. Latency is an important parameter to be considered in the cloud environment because it has a direct impact on the customers' overall satisfaction. An unsatisfied end-user is more likely to switch to other cloud providers. Thus, network latency is given by (12),

$$n_j = \left(\frac{D}{\beta}\right) + \sum_{i=1}^{n}\sum_{j=1}^{n} td_{ij} \quad (12)$$

where, Network bandwidth (β) is defined as speed of network in bits per time units and td_{ij} is the time delay between two network entities.

4. RESULTS AND DISCUSSIONS

To exhibit the performance efficiency obtained by LAM, we compared it with three benchmark scheduling algorithms Greedy-R, Greedy-P and FCFS [49]. Apart from this, we also compared it to other resource allocation techniques in cloud proposed in literature such as dynamic resource allocation [25] and preference-based resource allocation schemes in cloud computing systems [18].

a. Greedy response (Greedy-R) Scheduling: In order to maximize the response time of a system the task having quickest time of execution first is assigned to the most powerful cloud resource
b. Greedy parallelization (Greedy-P) Scheduling: In order to maximize the response time of a system and perform task parallelization, the task having quickest time of execution first is assigned to the cloud resource that is least powerful.
c. First come first serve (FCFS) Scheduling: In this form of scheduling, tasks are assigned to any of the available cloud resources as soon as they arrive.
d. Dynamic resource allocation scheme in cloud computing [25]: In this scheme, resources are allocated to users based on characteristics of jobs, with low priority jobs preventing delay of high priority jobs and dynamic allocation of resources for a user job within a job deadline.
e. Preference-based resource allocation in cloud computing systems [18]: It is a demand-based preferential resource allocation scheme and allocates resources based on users payment capacity.

4.1. Experimental setup and metrics

Experimental setup for performance evaluation is described in this section. All the experiments were conducted

ALGORITHM 3 ALGO_FindResourceVal

```
1. Input:  P ={ P1, P2, ... Pn}, C = {C1, C2,…,Cdc}
2. Output: Max_Value
3.      Begin:
4.              For all Ci ∈ C /*perform step 6 for all the criteria's*/
5.                  For all Pj ∈ P/*perform step 6 for all the priority values*/
6.                      Valij = Ci* Pj /*calculating associated value*/
7.                  End For
8.             End For
9.  Max_Value=Max(Valij) /*finding maximum value for a resource*/
10.      Return Max_Value  /*returning the maximum value associated with a resource*/
11.      End
```

Performance Analysis Tool: Since, the targeted system for this study is a generic environment for cloud computing, results need to be evaluated on a large-scale cloud environment. However, conducting experiments repeatedly in order to compare our algorithms with others on such a large scale, in a real environment can be a very daunting task. Thus, in order to ensure repeatability and adaptive tuning of the proposed framework, experiments have been conducted using cloudSim toolkit [34]. We chose cloudSim because contrary to other similar simulation toolkits such as SimGrid and GangSim, it also helps in modelling of virtualized resources on demand [50]. Furthermore, it also supports simulation of dynamic loads.

Workload setup: In order to make conclusion about our simulations the experiments were conducted using a real workload trace i.e. Google workload trace.This also validates the practical usage of LAM algorithm. different VM types are modelled as per Amazon EC2 [14] instance types and the different parameters used are presented in Table 4. The resources used in the experiment include virtual machines comprising of different processing, memory and network bandwidth requirements. The different types of tasks include tasks of length 400, 1000 and 2000 MIPS. Since the trace logs record consists of around 25 million tasks and it is quite difficult to conduct experiments on such large number of tasks. Therefore, in our experimental setup, 20*104 tasks spanning were used as representative.

Environment: The experimental environment consists of about 80 nodes. Each node is modelled to have one CPU core with performance of 3000 MIPS, 4 GB RAM, 100 GB/s network bandwidth and 100 GB storage. Each host can have multiple VMs. There are five different types of virtual machines used.

Metrics used: The performance of LAM has been calculated by using metrics, typical to a large-scale cloud system. Since optimal utilization of resources at the data centre is the main objective of the proposed problem, therefore, the following metrics were used. These metrics stem from the ones used by Z. Xiaomin *et al*. [47].

a. Task success ratio: Task success ratio is defined as ratio of tasks executed to the total number of tasks submitted. It can be calculated with the help of (3).
b. Resource utilization rate: The resource utilization rate at each host node is defined as the total amount of resources being utilized by the system and is given by (4).
c. Task arrival interval: Task arrival interval determines the difference in arrival time between two tasks.

Table 4. VM types used in the experiment

VM instance type	CPU (MIPS)	RAM (Gib)	Storage (MB)	Bandwidth (MBPS)
S1	200	8	1 * 32	450
S2	400	3.75	1 * 80	500
S3	600	15	2 * 80	750
S4	800	15.25	1 * 32	1000
S5	1000	15	1 * 60	1000

4.2. Analysis of results

A series of experiments were conducted in order to find out the impact of the difference metrics on the performance of LAM. All the results are summarized in Figure 5 and Figure 6.

4.2.1. Effect of number of tasks

Task success ratio: We analyzed the effect of varying the number of tasks on the task success rate. The results are demonstrated in Figure 5(a). It can be observed from the results that task success ratio of all the algorithms do not vary much with increase in the number of tasks. This uniformity in the success ratio is attributed to the scalable [51] and elastic nature of the cloud environment. The results also show that LAM has high task success ratio in contrast to algorithms in literature. Therefore, it can be concluded that LAM is a preferred approach for critical tasks where the tasks are required to be successfully executed with no scope of failure.

Resource consumption: Figure 5(b) shows the effect of the number of tasks on resource consumption. It can be inferred that FCFS algorithm shows the best results in terms of resource consumption followed by LAM. However, LAM outperforms Greedy-P, dynamic resource allocation scheme and Preference based resource allocation in terms of both task success ratio and resource consumption. It shows similar resource consumption to Greedy–R, but has a higher task success ratio. It also shows that even though LAM has higher resource consumption than FCFS, but still is a preferred technique as the task success ratio is higher.

4.2.2. Effect of task arrival interval

The rate at which tasks arrives at the service provider also has an impact on performance. Therefore, in order to assess the effect of task arrival interval, we have taken the value of tasks in the range of [0, 20].

Task success ratio: Figure 6 depicts the effect of arrival interval of tasks on its success ratio and resource consumption. From Figure 6(a), it can be observed that task success ratio of LAM is approximately 31, 35 and 36 percent more than greedy-p, greedy-r and FCFS algorithm respectively. It also shows that LAM algorithm outperforms other approaches for dynamic cloud environment like dynamic resource allocation scheme and preference-based resource allocation by 9 and 17 percent respectively.

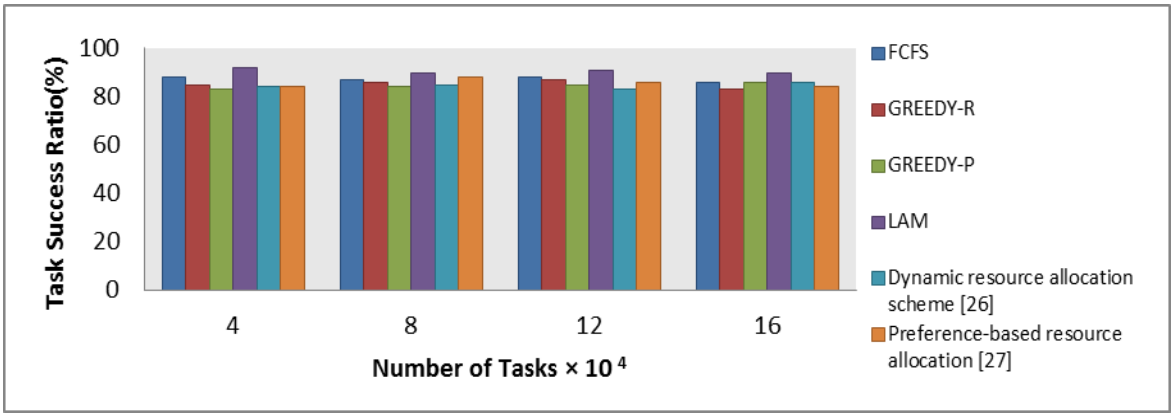
Resource consumption: This experiment analyzes the impact of arrival interval of tasks on resource consumption. Figure 6(b) shows the results of the experiment that was conducted. According to it, greedy-r utilizes minimum amount of resources followed by Greedy-P and LAM, but still LAM performs better than dynamic resource allocation scheme and preference-based resource allocation by consuming only 54 percent of the resources as compared to the other two that utilize 60 and 61 percent resources, respectively.Thus, consumption is optimal in LAM as compared to other algorithms for varying arrival interval. It also shows that its task success ratio is higher than the other algorithms.

4.2.3. Evaluation of scalability of the algorithm

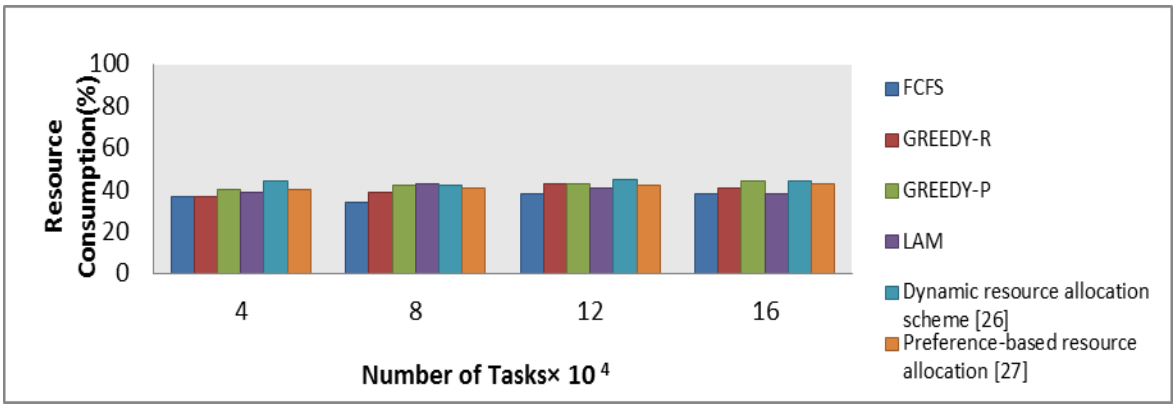
The scalability of LAM can also be inferred from Figure 5(a) and Figure 5(b). It was observed that as size of the tasks increases, LAM algorithm gives comparable performance both in terms of resource consumption as well as task success ratio. Thus, it can be established that LAM is scalable and appropriate for xecuting large-scale applications in IaaS clouds.

4.2.4. Robustness

Robustness of LAM can be inferred from the fact that it consumes lesser number of resources and has high task success ratio with change in arrival interval and number of tasks. Therefore, it can be inferred that the proposed approach is a viable solution for allocation of resources in a cloud environment.

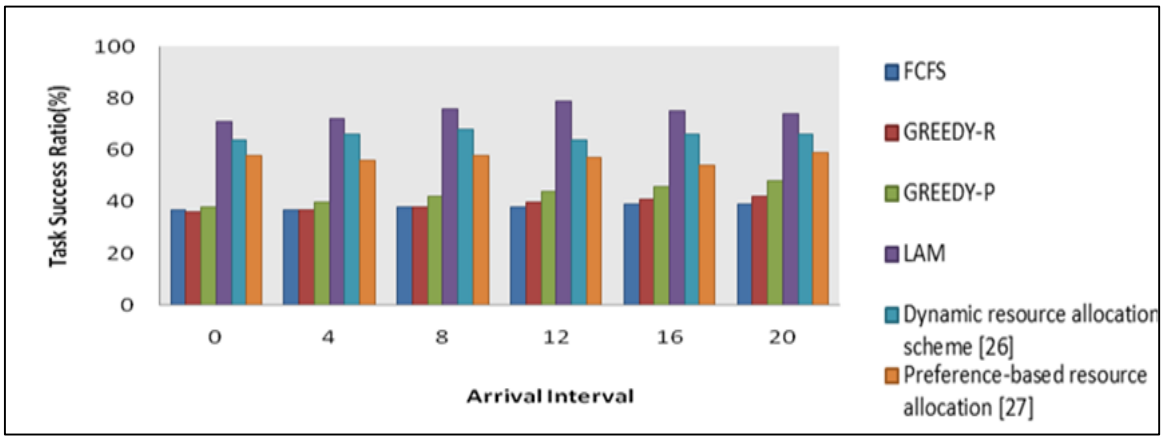


(a)

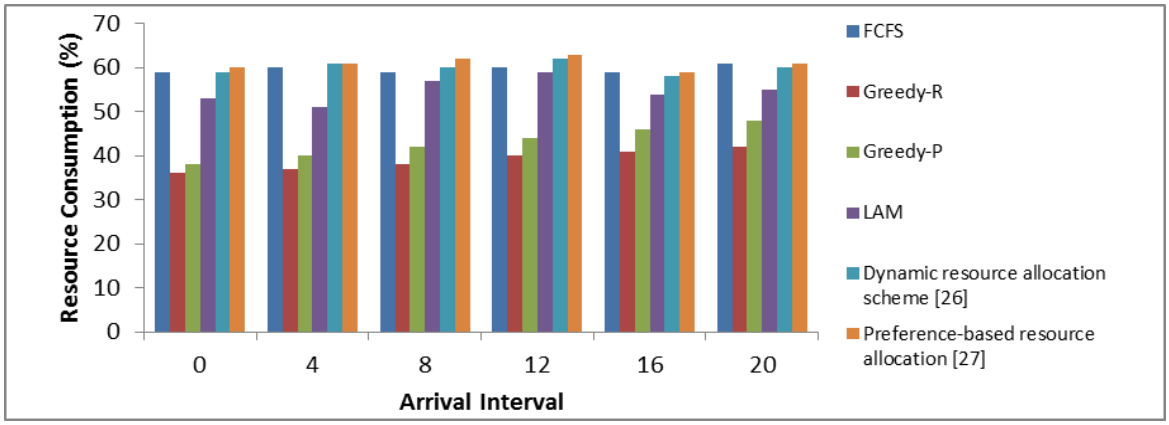


(b)

Figure 5. Effect of number of tasks on task success ratio and resource consumption,
(a) Task success ratio v/s number of tasks, (b) Resource consumption v/s number of tasks



(a)



(b)

Figure 6. Effect of task arrival interval on task success ratio and resource consumption,
(a) Task success ratio v/s arrival interval, (b) Resource consumption ratio v/s arrival interval

5. CONCLUSION

This work presents a strategy for allocating resources at the service provider's side. The scenario was modeled as a resource allocation problem with the objective of minimizing resource consumption and maximizing task success ratio. The proposed approach incorporates basic elements of cloud computing such as pay per usage, elasticity and dynamic requirements and usage of resources. A resource allocation algorithm called LAM was proposed. It is a novel technique as it utilizes resources in an optimal manner and has several benefits and advantages such as better management of resources, improved quality of services, scalability, robustness, infrastructure level performance enhancement and ability to attract more customers i.e. better QoS while abiding by the SLAs. Furthermore, it shows an improvement in resource consumption when compared with other resource allocation algorithms such as Greedy-R, Greedy-P and FCFS, dynamic resource allocation and preference-based resource allocation. The authors strongly believe that LAM represents a significant step towards enabling service provider to make important decisions regarding provisioning of resources. With techniques presented in this paper, the service providers can distribute resources in an optimal manner leading to more profit and a better infrastructure performance. The proposed method can be easily incorporated into a real-world system as the experiments have been performed on a real world production workload i.e. Google cluster trace using system setup similar to a public cloud such as Amazon EC2. As a future work, we would like to explore with other different strategies like particle swarm optimization, genetic algorithm and min-min algorithm for resource allocation.

ACKNOWLEDGEMENTS

This research was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University through the Fast-track Research Funding Program.

REFERENCES

[1] W. van der Aalst and E. Damiani, "Processes Meet Big Data: Connecting Data Science with Process Science," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 810-819, 2015.

[2] K. A. Shakil, et al., "BAMHealthCloud: A biometric authentication and data management system for healthcare data in cloud," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 1, pp. 57-64, 2020.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," NIST, pp. 1-7, 2011.

[4] W. Huang, et al., "Mobile internet big data platform in china unicom," *Tsinghua Science and Technology*, vol. 19, no. 1, pp. 95-101, 2014.

[5] M. S. Mahmud, et al., "A Survey of Data Partitioning and Sampling Methods to Support Big Data Analysis," *Big Data Mining and Analytics*, vol. 3, no. 2, pp. 85-101, 2020.

[6] Y. Xing and Y. Zhan, "Virtualization and cloud computing," in *Lecture Notes in Electrical Engineering*, vol. 143, no. 1, pp. 305-312, 2012.

[7] Geeta and S. Prakash, "Role of virtualization techniques in cloud computing environment," in *Advances in Intelligent Systems and Computing*, vol. 760, pp. 439-450, 2019.

[8] E. Triantaphyllou, "Multi-Criteria Decision Making Methods: A Comparative Study," Springer, pp. 5-21, 2000.

[9] C. W. Tsai, et al., "A Hyper-Heuristic Scheduling Algorithm for Cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 236-250, 2014.

[10] Z. Wan, "Sub-millisecond level latency sensitive cloud computing infrastructure," in *2010 International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT 2010)*, pp. 1194-1197, 2010.

[11] K. Kumar, et al., "Resource allocation for real-time tasks using cloud computing," in *Proceedings - International Conference on Computer Communications and Networks (ICCCN)*, pp. 1-7, 2011.

[12] Y. Liu, et al., "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398-2410, 2016.

[13] S. K. Garg, et al., "SMICloud: A framework for comparing and ranking cloud services," in *Proceedings - 2011 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011)*, pp. 210-218, 2011.

[14] "Amazon EC2 Instance Types," Amazon Web Services (AWS), 2016.

[15] K. H. Kim, et al., "Power-aware provisioning of Cloud resources for real-time services," in *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, MGC'09 held at the ACM/IFIP/USENIX 10th International Middleware Conference*, pp. 1-6, 2009.

[16] I. S. Moreno, et al., "Analysis, Modeling and Simulation of Workload Patterns in a Large-Scale Utility Cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 208-221, 2014.

[17] M. Satyanarayanan, et al., "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, 2009.

[18] N. Kumar and S. Saxena, "A Preference-based Resource Allocation in Cloud Computing Systems," *Procedia Computer Science*, vol. 57, pp. 104-111, 2015.

[19] C. Mergenci and I. Korpeoglu, "Generic resource allocation metrics and methods for heterogeneous cloud infrastructures," *Journal of Network and Computer Applications*, vol. 146, p. 102413, 2019.

[20] T. L. Sterling, et al., "Beowulf cluster computing with Linux," MIT Press, 2002.

[21] D. Ergu, et al., "The analytic hierarchy process: Task scheduling and resource allocation in cloud computing environment," *Journal of Supercomputing*, vol. 64, no. 3, pp. 835-848, 2013.

[22] W. Ming, et al., "Resources Allocation Method on Cloud Computing," in *International Conference on Service Sciences*, Wuxi, pp. 199-201, 2014.

[23] Y. Zhao, et al., "A Resource Allocation Scheme for SDN-Based 5G Ultra-Dense Heterogeneous Networks," in *2017 IEEE Globecom Workshops*, 2018.

[24] D. Poola, et al., "Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds," in *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pp. 858-865, 2014.

[25] A. T. Saraswathi, et al., "Dynamic Resource Allocation Scheme in Cloud Computing," *Procedia Computer Science*, vol. 47, pp. 30-36, 2015.

[26] Z. Li, et al., "Cloud resource allocation for cloud-based automotive applications," *Mechatronics*, vol. 50, pp. 356-365, 2018.

[27] M. Farrellee, "Condor: Grid Scheduler and the Cloud," *Open Source Cloud Computing Forum*, 2009.

[28] M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222-235, 2014.

[29] M. Alam and K. A. Shakil, "An NBDMMM Algorithm Based Framework for Allocation of Resources in Cloud," *arXiv: 1412.8028v1*, 2014.

[30] Vineetha V., "View Point. Performance Monitoring in Cloud," *Infosys*, pp. 1-8, 2012. [Online]. Available: http://docplayer.net/967541-View-point-performance-monitoring-in-cloud-www-infosys-com-abstract-vineetha-v.html.

[31] R. Begam, et al., "TIMER-Cloud: Time-Sensitive VM Provisioning in Resource-Constrained Clouds," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 297-311, 2020.

[32] P. Zhang, et al., "An Intelligent Optimization Method for Optimal Virtual Machine Allocation in Cloud Data Centers," *IEEE Transactions on Automation Science and Engineering*, pp. 1-11, 2020.

[33] H. Sun, et al., "Research and Simulation of Task Scheduling Algorithm in Cloud Computing," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 11, pp. 6664-6672, 2013.
[34] R. N. Calheiros, et al., "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Practice and Experience*, vol. 41, pp. 23-50, 2011.
[35] J. H. C. Reiss and J. Wilkes, "Google cluster-usage traces format schema 2014-11-17 external.pdf - Google Drive," Google Inc., 2014.
[36] M. Rahman, et al., "A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids," in *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, pp. 35-42, 2007.
[37] F. Lopez-Pires, "Many-Objective Resource Allocation in Cloud Computing Datacenters," in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pp. 213-215, 2016.
[38] Platform Computing, "LSF Version 4.1 Administrator's Guide," 2001.
[39] T. Tannenbaum, et al., "Condor: a distributed job scheduler," in *Beowulf Cluster Computing with Windows*, pp. 307-350, 2001.
[40] F. López-Pires and B. Barán, "Many-Objective Virtual Machine Placement," *Journal of Grid Computing*, vol. 15, no. 2, pp. 161-176, 2017.
[41] G. Coyle, "The Analytic Hierarchy Process (AHP)," Practical Strategy: Structured Tools and Techniques, Open Access Material, Glasgow, Pearson Education Ltd, 2004.
[42] W. E. Yang, et al., "An outranking method for multi-criteria decision making with duplex linguistic information," *Fuzzy Sets and Systems*, vol. 198, pp. 20-33, 2012.
[43] R. W. Saaty, "The analytic hierarchy process-what it is and how it is used," *Mathematical Modelling*, vol. 9, no. 3-5, pp. 161-176, 1987.
[44] M. Yurdakul, "Measuring a manufacturing system's performance using Saaty's system with feedback approach," *Integrated Manufacturing Systems*, vol. 13, no. 1, pp. 25-34, 2002.
[45] D. Ergu, et al., "Further Discussions on Induced Bias Matrix Model for the Pair-Wise Comparison Matrix," *Journal of Optimization Theory and Applications*, vol. 161, no. 3, pp. 980-993, 2014.
[46] K. A. Shakil, et al., "Exploring non-homogeneity and dynamicity of high scale cloud through Hive and Pig," *Indian Journal of Science and Technology*, vol. 8, no. 35, pp. 1-6, 2015.
[47] X. Zhu, et al., "Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168-180, 2014.
[48] M. Alam and K. A. Shakil, "A Decision Matrix and Monitoring based Framework for Infrastructure Performance Enhancement in A Cloud based Environment," Institute of Doctors Engineers and Scientists, 2013.
[49] J. O. Gutierrez-Garcia and K. M. Sim, "A family of heuristics for agent-based elastic Cloud bag-of-tasks concurrent scheduling," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1682-1699, 2013.
[50] A. Beloglazov, et al., "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, pp. 755-768, 2012.
[51] V. Padhye and A. Tripathi, "Scalable Transaction Management with Snapshot Isolation for NoSQL Data Storage Systems," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 121-135, 2015.