Manipal Journal of Science and Technology

Volume 5 | Issue 1 Article 6

6-1-2020

A Novel Greedy Based Algorithm for Resource Allocation in Cloud Computing Environment

Vijendra Pratap Singh
Banaras Hindu University, Varanasi, Uttar Pradesh, India, vijendrap.singh4@bhu.ac.in

Research Article

A Novel Greedy Based Algorithm for Resource Allocation in Cloud Computing Environment

Sunil Kumar, Vijendra Pratap Singh*, Pangambam Sendash Singh

Email: vijendrap.singh4@bhu.ac.in

Abstract

Resource allocation in a cloud computing environment means a mechanism to allocate different cloudlets to the available cloud resources on the basis of some criteria like cloudlets' characteristics and/or requirements. It is one of the major challenges for cloud providers. Cloud providers have introduced many models for resource allocation. In this paper, we have proposed a greedy-based resource allocation algorithm. Here, first of all, the given cloudlets are classified into two categories: computational cloudlets and interactive cloudlets on the basis of some parameters. Next according to their category, they are allocated differently. The performance metrics of this newly proposed algorithm are calculated and compared with traditional resource allocation techniques viz. FCFS and SClF. For validation of our algorithm, we set up experimental environments and implemented them in CloudSim. We found that in different scenarios our newly proposed resource allocation algorithm works better than traditional techniques in terms of resource utilization, completion time, throughput, etc.

Keywords: Cloud computing, Resource allocation, Virtual machine, Greedy algorithm.

Introduction

Cloud computing is an internet-based computing service that provides resources based on *service-level agreements (SLAs)* established between the cloud providers and cloud end users [1]. Resources may be of the following types viz. processing elements (PEs), CPUs, RAM, storage, bandwidth (BW), etc. Cloud providers provide a pool of abstract, virtualized, and scalable resources to the cloud users based on a *pay-as-you-go (PAYG)* model *on-demand* basis. Cloud resources are provided as three different types of services: *Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS) and Platform-as-a-Service (PaaS)* [2]. The IaaS provides a fabric layer or hardware of cloud environments like CPUs, memory, storage, network bandwidth, etc. In other words, we can say that resource service provides hardware resources such as computing processors, network resources, storage, and memory, as a service for remote clients. The PaaS provides a computing environment where users can develop their own software/applications. SaaS is providing already created applications to the users. In this paper, we focus on IaaS where cloud providers provide resources in terms of virtual machine (VM) instances.

Resource (VM) allocation means assigning the different cloudlets to appropriate VMs. These resource allocation algorithms aim to minimize the execution time and maximize throughput with better utilization of the available resources. Cloud providers often face many decision problems during resource allocation. Nowadays, resource allocation has also become complicated due to limited cloud resources and increasing cloud users. There are some traditional allocation techniques like First Come First Serve (FCFS) technique, Shortest Cloudlet First (SClF) technique, etc. Different cloud providers have also introduced different resource allocation algorithms.

Sunil Kumar[1], Vijendra Pratap Singh[1], Pangambam Sendash Singh[1]

[1] *Department of Computer Science, Banaras Hindu University, Varanasi, Uttar Pradesh, India.*

Manuscript received: 21 April 2020
Revision accepted: 20 May 2020

* Corresponding Author

How to cite this article: Sunil Kumar, Vijendra Pratap Singh, Pangambam Sendash Singh. "A Novel Greedy Based Algorithm for Resource Allocation in Cloud Computing Environment", Manipal J. Sci. Tech., vol.5(1), 32-39, 2020.

Published by Impressions@MAHE, 2020

Greedy algorithm [3] is a class of optimization algorithms in which the current best choice is chosen for solving the problem without worrying about the future, with the hope that it will give the optimal solution. But in most cases, the Greedy algorithm does not give the globally optimal solution; it gives the approximated optimal solution. In this paper, we introduce a Greedy-based VM allocation algorithm that can be used for resource allocation in a cloud environment. The detailed algorithm is explained in section 3. The newly proposed algorithm is implemented and simulated for different input data sets on the CloudSim-3.0.3 framework at the Java platform [2] and it is compared with traditional allocation techniques like FCFS and SClF on the basis of performance metrics like makespan, mean execution time, and throughput. And we find that this allocation algorithm works better than FCFS and SClF.

The rest of this paper is organized as follows. In section 2 we will give some related works. Traditional allocation techniques FCFS, SClF as well as some basic terminologies are described in section 3. Our newly proposed algorithm is explained in section 4. Experimental details and simulation results are presented in section 5. Lastly, in section 6; we give a brief conclusion and future directions.

Related works

In a cloud environment, a data center is a set of hosts where numbers of virtual machines exist in the same hosts [2]. Each virtual machine has some computing resources which are allocated to the specific cloudlets/tasks according to their needs in a way that it maximizes the throughput and minimizes the makespan. There are many job scheduling algorithms based on traditional algorithms (e.g., FCFS, Priority, Round-Robin, SClF, etc) which are used efficiently for resource allocation. In [4], the authors proposed a model for Grid Computing which is known as Swift Scheduler. According to this model, the assignment of jobs to the resources is done as First it stores all the user jobs and resources in their respective queue. After that, it arranges the job queue in ascending order according to their length and calculates their expected processing time for all resources. Cloudsim - a Java based platform is introduced in [2] to model, simulate cloud computing environments and analyze their results. Different allocation policies, different available classes and features of this simulator are explained in detail. In [5], the authors established a mechanism for scheduling that follows the Lexi search approach. The main benefit of using the Lexi search approach is to find an optimal feasible assignment. The proposed algorithm is working as: First, the requests or user tasks are stored in a job pool or central middleware, and then these jobs are partitioned. Two algorithms: LCFP (Longest Cloudlet Fastest Processing Element) and SCFP (Shortest Cloudlet Fastest Processing Element) are proposed in [6]. According to LCFP, the lengthier cloudlets assign to the VM which has the maximum processing elements to minimize the makespan. According to SCFP, the shorter cloudlets assign to the VM which has the maximum processing elements to reduce the flowtime. A survey of some existing scheduling algorithms on the basis of different parameters, scheduling strategy, and performance metrics etc. is found in [7]. In [8], the authors proposed an algorithm with some improvement in the traditional Max-Min algorithm which is used for task scheduling. In a traditional Max-Min algorithm, it selects the job which has the highest completion time and assigns it to the resource where it executes in less time. The proposed algorithm in the paper selects the task which has minimum completion time and assigns it to a resource where it executes on average or nearest greater than average time. An improved load balancing algorithm by using the Min-Min algorithm is presented in [9] in order to minimize makespan and maximize resource utilization. According to the algorithm, it divides all the submitted jobs into two groups: one group represents VIPs task means higher priority and the second group represents User's task means lower priority. After dividing, it executes both job groups according to the Min-Min algorithm and computes the makespan. Then, it selects the VIPs task and chooses the task which has minimum completion time and assigns it to VM which is heavily loaded and computes its completion time. Using this algorithm, it easily manages the load on each VM such that the value of makespan for all tasks is less. In [10], the authors proposed an algorithm for load balancing which uses the concept of Honey bee behaviour. The main purpose of this algorithm is to maximize the throughput such that the load on each VM should be

balanced. In [11], the authors proposed a resource scheduling algorithm in cloud computing based on genetic algorithm techniques. In this algorithm, cloudlets are grouped on the basis of data and requested resources and it is prioritized. Resource selection is done on the basis of cost and turnaround time using a greedy approach. Using this algorithm energy consumption can be saved and resource utilization can also be increased. Various scheduling algorithms and issues related to them in a cloud computing environment are discussed in [12]. It is also mentioned that existing scheduling algorithms give high throughput and they are cost-effective, but they do not consider the reliability and availability of the resource. So they need to be improved. In [3], the authors proposed a greedy-based job scheduling technique in cloud computing. Using this algorithm completion time of the submitted cloudlets can be decreased and user satisfaction can be increased. In [1], the authors proposed a greedy mechanism for VM provisioning and VM allocation in a cloud computing environment. In this algorithm, a dynamic VM provisioning and allocation problem for an auction-based model is formulated and a promising result in terms of revenue of the cloud providers is achieved. A detailed survey on some scheduling algorithms and resource scheduling based on service level agreements on cloud computing is done in [13]. Merits and demerits of the algorithms are also explained.

Background knowledge

I. Some basic terminologies

- **Cloudlets:** Cloudlet is a representation of task/job/process in CloudSim framework. It takes the job and submits to CloudSim. DeviceCloudletCloud.
- **Virtual Machines (VMs):** It runs inside a Host, sharing hostList with other VMs. It processes cloudlets. This processing happens according to a resource allocation policy, defined by the CloudletScheduler. Each VM has an owner, which can submit cloudlets to the VM to be executed.
- **Hosts:** It is a physical entity in the cloud datacenter. It can be maintained through the hostList in Cloudsim.
- **Datacenters:** Datacenter class is a Cloud Resource whose hostList are virtualized. It deals with the processing of VM queries (i.e., handling of VMs) instead of processing Cloudlet-related queries.
- **Cloud users:** This is end users of the cloud services.
- **Datacenter Brokers:** Datacenter Broker represents a broker acting on behalf of a user. It hides VM management, as VM creation, submission of cloudlets to these VMs and destruction of VMs.
- **Processing Element (PE):** Number of CPUs or cores assigned to the virtual machines.
- **Million Instructions Per Second (MIPS):** It is a unit to represent the processing/execution capability of the PE/CPU.
- **Bandwidth (BW):** It is a unit to represent the communication/interactive capability of VMs.

II. Traditional Algorithms:

There are many different existing resource allocation algorithms. The most common traditional algorithms are First Come First Serve (FCFS) and Shortest Cloudlet First (SClF) algorithms. These two traditional resource allocation or scheduling algorithms are explained below.

- **First Come First Serve (FCFS) Algorithm:** This is the simplest algorithm in which cloudlets are allocated to the available VMs in the order in which they arrive. It aims to allocate the cloudlets with the least waiting time. FCFS mechanism is already available in the CloudSim platform. One of the disadvantages of this algorithm is that it is non-preemptive.
- **Shortest Cloudlet First (SClF) Algorithm***:* This algorithm aims to allocate the available VMs to the cloudlet having a minimum length. In this algorithm first cloudlets ascended in the order of their cloudlet length and they are allocated to the available VMs in an FCFS manner. One of the disadvantages of this algorithm is that the waiting time of those cloudlets having greater cloudlet lengths may become very large.

Proposed Algorithm

Let us assume that our proposed resource allocation algorithm in a cloud environment is as $VM = \{VM_1, VM_2, \ldots VM_m\}$ be the set of m virtual machines. $T = \{T_1, T_2, \ldots, T_n\}$ be the set of n cloudlets (tasks). All the machines are unrelated and parallel and are

Published by Impressions@MAHE, 2020

denoted as R in the model. *npmtn*: Non-preemptive cloudlets that mean the processing of that cloudlet on a virtual machine cannot be interrupted (assuming that failure does not occur). We denote the finishing time of a cloudlet T_i by CT_i. Our aim is to reduce the makespan which can be denoted as CT_{max}. So, our proposed Greedy model is $R|npmtn|CT_{max}$.

In our new algorithm, we make some assumptions. All the cloudlets are assumed to be independent of each other; i.e., there is no data dependency between the cloudlets, they can be executed in parallel using the available resources (VMs) in any order. It is also assumed that there is no limitation on the number of resources (VMs) that can be provided by the cloud provider. Lastly, the output size of all the cloudlets is also assumed to be known in advance.

Our new Greedy-based VM allocation algorithm mainly consists of three basic steps viz. classification of cloudlets into two categories, allocation of cloudlets of different categories to their respective VMs by using our new proposed algorithm, comparing the performance metrics with traditional techniques like FCFS and SClF. Detail algorithm is explained below:

1. Based on the attributes of the submitted cloudlets, cloudlets are pre-processed to classify into two different categories: Class-I (Computational type Cloudlets) and Class-II (Interactive type Cloudlets). Computational type Cloudlets can be efficiently run on high capacity VMs while Interactive type Cloudlets can be run efficiently on high bandwidth (communication bandwidth means latency) VMs.
2. Computational type cloudlets and Interactive type cloudlets are descended in the order of their expected execution time and expected bandwidths respectively. Expected Execution Time (ExpEt) and Expected Bandwidth (ExpBW) of the cloudlets are calculated by using the following relations.

$$ExpEt = \frac{Cloudlet\ Total\ Length}{Average\ MIPS\ of\ the\ VMs} \quad (1)$$

ExpBW = No.of PEs*Cloudlet length+Input file size+Output file size (2)

3. Next, the available VMs are descended in the order of their capacities. The capacity of a VM is given by the following relation.

$$Capacity = No.of\ PEs * MIPS + Bandwidth$$
$$C_j = PE_{num_j} * PE_{mips_j} + VM_{bw_j} \quad (3)$$

4. Computational type cloudlets are submitted to these descended VMs, and they are executed. While doing so, cloudlets having larger Expected Execution Time (ExpEt) are assigned to high capacity VMs and cloudlets having smaller Expected Execution Time (ExpEt) are assigned to low capacity VMs.
5. Again, on the basis of the bandwidths of the VMs, VMs are descended.
6. Interactive type cloudlets are then submitted to these reordered VMs, and they are also executed. While doing so, highly interactive cloudlets are assigned to VMs having high bandwidths (communication bandwidths) and less interactive cloudlets to VMs having lesser bandwidths.

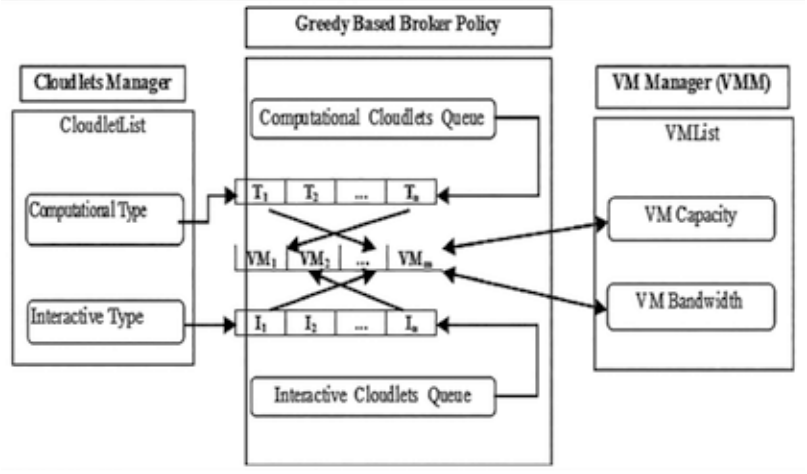

Fig 1: Architectural representation of proposed scheduling algorithm

7. Finally, with different input sets, the performance metrics are calculated by simulating our newly proposed algorithm, and they are compared with traditional techniques: -FCFS and SClF.

Experimentation

I. Performance metrics

Throughput, makespan and mean execution time of the cloudlets are used as performance metrics in this new Greedy-based VM allocation algorithm. This algorithm uses the following two well-known QoS parameters makespan and means execution time of cloudlets.

- **Throughput:** Throughput is the rate of successful execution of the cloudlets. In other words, it can be defined as the total number of successfully executed cloudlets per unit time. For the better utilization of the resources, throughput should be made higher.

$$Throughput = \frac{Total\ number\ of\ successfully\ executed\ cloudlets}{Time\ taken\ in\ execution\ of\ these\ cloudlets} \quad (4)$$

- **Makespan:** Makespan is a measure of resource utilization and throughput. For better utilization of the resources and to maximize throughput, makespan should be minimized. Makespan is defined as the overall cloudlet completion time. We denote the completion time of cloudlet Ti on VMj as CTij.

$$Makespan = Maximum\ completion\ time\ of\ the\ cloudlets \quad (5)$$

$$Makespan = max\{CT_{ij} | i \in T, i = 1,2,\dots n and j \in VM, j = 1,2,\dots M\}$$

- **Mean execution time:** It is the average of the execution time of all the cloudlets. It is given by the relation:

$$Execution\ time = Finish\ execution\ time - Start\ execution\ time \quad (6)$$

$$Mean\ execution\ time = \frac{Overall\ execution\ time}{Total\ no.of\ cloudlets} \quad (7)$$

$$MET = \frac{\sum_{i=1}^{n} (FET_i - SET_i)}{n} \quad (8)$$

II. Experimental results

This new algorithm is implemented in the CloudSim-3.0.3 platform. CloudSim is a Java based framework that can be used to model and simulate cloud computing environments. While simulating this algorithm, we add new classes and methods according to our needs; Cloudlet class and VM class are also extended for calculating new parameters such as Expected Execution Time, Expected Bandwidth, Capacities of the VMs and other performance metrics. This new algorithm is simulated three times using different input parameters.

We use one data center that can run multiple hosts with host parameters given in Table I.

Table I: host parameters

Parameters	Values
MIPS	15000
RAM	4096
STORAGE	1000000
BW	10000

- **Observation no. 1:** Here two hosts and three VMs are used. The hosts and VM parameters are given below:

No. of Hosts: 2
No. of Processing elements of Host#0: 2
No. of Processing elements of Host#1: 4
No. of VMs: 3

Table II: VM Parameters For OBS.1

VM ID	MIPS	NO. OF PES	RAM	BW	SIZE	VMM
0	2000	1	512	1000	10000	Xen
1	1900	3	512	1100	10000	Xen
2	1800	2	512	1400	10000	Xen

Using the above parameters, FCFS, SClF and the new proposed algorithm is simulated by varying the number of cloudlets. The simulation result is shown below:

Table III: Results Of Observation No.1

No. of Cloudlets	Makespan			Mean Execution Time			Throughput		
	FCFS	SCLF	Proposed	FCFS	SCLF	Proposed	FCFS	SCLF	Proposed
10	30.43	32.07	12.48	13.05	13.96	6.31	0.33	0.31	0.8
20	61.07	49.82	19.92	25.62	24.04	11.68	0.33	0.4	1
30	86.87	69.89	34.53	38.31	36.17	17.78	0.35	0.43	0.87
40	125.34	111.22	45.13	53.09	49.37	23.01	0.32	0.36	0.89
50	147.58	129.31	59.14	63.59	58.21	28.43	0.34	0.39	0.85

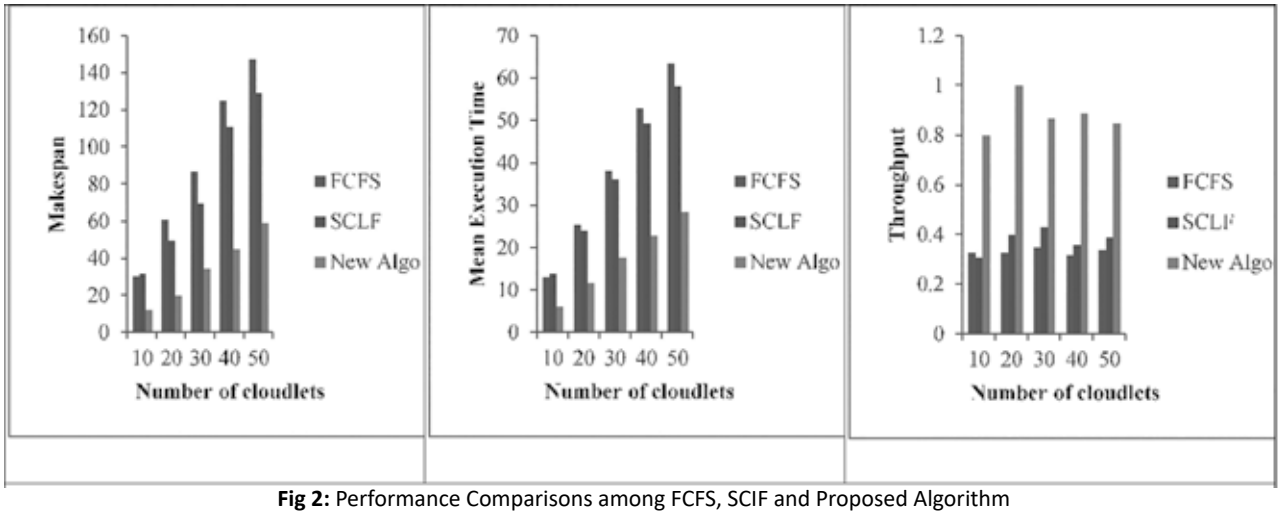

Fig 2: Performance Comparisons among FCFS, SCIF and Proposed Algorithm

- **Observation no. 2:** Next, number of hosts and number of VMs are increased. The hosts and VM parameters are given below:

No. of Hosts: 3
No. of Processing elements of Host#0: 3
No. of Processing elements of Host#1: 5
No. of Processing elements of Host#2: 2
No. of VMs: 6

Table IV: VM Parameters For OBS.2

VM ID	MIPS	NO. OF PES	RAM	BW	Size	VMM
0	2000	1	512	1000	10000	Xen
1	1900	3	512	1100	10000	Xen
2	1800	2	512	1400	10000	Xen
3	2000	3	512	1200	10000	Xen
4	1900	1	512	1200	10000	Xen
5	1800	1	512	1400	10000	Xen

Using the above parameters, FCFS, SClF and the new proposed algorithm is simulated by varying the number of cloudlets. The simulation result is shown below:

Table V: Results Of Observation No. 2

No. of Cloudlets	MAKESPAN			MEAN EXECUTION TIME			THROUGHPUT		
	FCFS	SCIF	Proposed	FCFS	SCIF	Proposed	FCFS	SCIF	Proposed
10	15.27	11.45	8.18	6.53	6.55	4.44	0.66	0.87	1.22
20	41.14	29.97	18.98	14.34	15.43	7.69	0.49	0.67	1.05
30	49.47	54.2	29.89	21.71	21.6	10.54	0.61	0.55	1
40	68.25	51.27	37.47	27.04	26.76	13.97	0.59	0.78	1.07
50	83.91	72.42	42.41	35.29	34.57	16.83	0.6	0.69	1.18

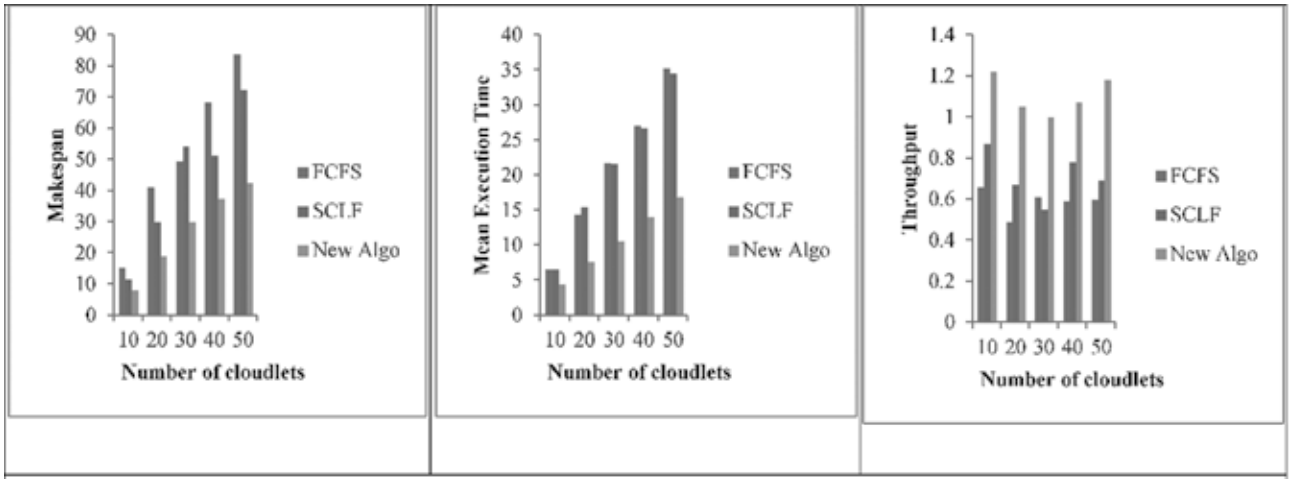

Fig 3: Performance Comparison among FCFS, SCIF and Proposed Algorithm

- Observation no. 3: Here, three hosts are used. The number of cloudlets is fixed to fifty. And by varying the number of VMs employed, FCFS, SClF and the new proposed algorithm is simulated. Host parameters, VM parameters and simulation results are given below:

No. of Hosts: 3
No. of Processing elements of Host#0: 2
No. of Processing elements of Host#1: 1
No. of Processing elements of Host#2: 3
No. of Cloudlets: 50

Table VI: VM Parameters Of Observation No. 3

VM ID	MIPS	NO. OF PES	RAM	BW	SIZE	VMM
0	2000	1	512	1000	10000	Xen
1	1900	3	512	1100	10000	Xen
2	1800	2	512	1400	10000	Xen
3	2000	3	512	1200	10000	Xen
4	1900	1	512	1200	10000	Xen
5	1800	1	512	1400	10000	Xen
6	2000	3	512	1100	10000	Xen
7	1900	2	512	1000	10000	Xen
8	1800	3	512	1100	10000	Xen

Table VII: Results Of Observation No. 3

#VMs	VM-ID	MAKESPAN			MEAN EXECUTION TIME			THROUGHPUT		
		FCFS	SClF	Proposed	FCFS	SClF	Proposed	FCFS	SClF	Proposed
2	0,1	192.8	188.9	93.13	1.76	91.52	45.47	0.26	0.26	0.54
4	0,1,2,3	88.94	99.99	41.16	37.64	39.82	19.12	0.56	0.5	1.21
7	0,1,2,3,4,5,6	70.7	64.63	37.48	26.57	28.07	13.66	0.71	0.77	1.33
8	0,1,2,3,4,5,6,7	50.74	56.02	33.29	21.67	23.88	11.62	0.99	0.89	1.5
9	0,1,2,3,4,5,6,7,8	68.45	56.45	32.4	25.92	19.78	9.91	0.73	0.89	1.54

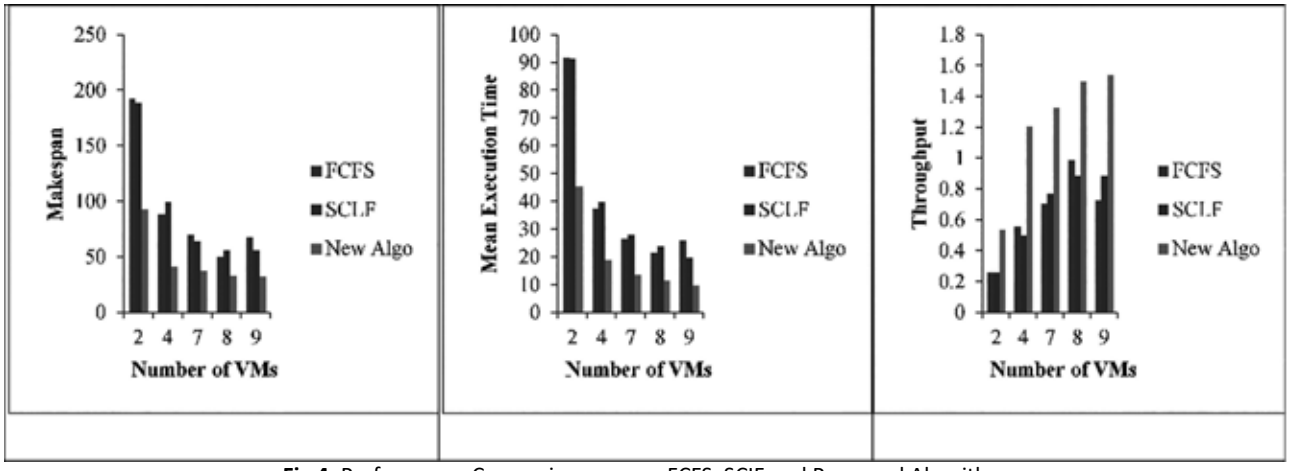

Fig 4: Performance Comparison among FCFS, SCIF and Proposed Algorithm

Conclusion And Future Directions

Resource allocation algorithms in a cloud computing environment aim to minimize the execution time and to maximize throughput with better utilization of the available resources. On the basis of these simulation results, it is observed that this newly proposed resource allocation algorithm works better than traditional scheduling techniques like FCFS and SClF. The makespan, overall maximum completion time and mean execution time of this new algorithm are less than those of the traditional ones, throughput is also better than these traditional algorithms, which means there is a better utilization of the resources. There are also certain topics that should be considered so that this newly proposed algorithm can be further improved. We need to check the best criteria of classification of the cloudlets into different classes and parameters that can be taken into account for proper resource allocation.

References

[1] M. M. Nejad, L. Mashayekhy, and D. Grosu, "Truthful Greedy Mechanisms for Dynamic

Published by Impressions@MAHE, 2020

Virtual Machine Provisioning and Allocation in Clouds," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 2, pp. 594–603, Feb. 2015, doi: 10.1109/TPDS.2014.2308224.

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Softw. Pract. Exp., vol. 41, no. 1, pp. 23–50, Jan. 2011, doi: 10.1002/spe.995.

[3] J. Li, L. Feng, and S. Fang, "An Greedy-Based Job Scheduling Algorithm in Cloud Computing," J. Softw., vol. 9, no. 4, Apr. 2014, doi: 10.4304/jsw.9.4.921-925.

[4] K. Somasundaram and S. Radhakrishnan, "Task Resource Allocation in Grid using Swift Scheduler," Int. J. Comput. Commun. Control, vol. 4, no. 2, p. 158, Jun. 2009, doi: 10.15837/ijccc.2009.2.2423.

[5] M. Paul, D. Samanta, G. Sanyal, and W. Bengal, "Dynamic job Scheduling in Cloud Computing based on horizontal load balancing," Int. J. Comput. Technol. Appl., vol. 2, no. 5, pp. 1552–1556, 2011.

[6] S. Sindhu and S. Mukherjee, "Efficient Task Scheduling Algorithms for Cloud Computing Environment," 2011, pp. 79–83.

[7] Vijindra and S. Shenai, "Survey on Scheduling Issues in Cloud Computing," Procedia Eng., vol. 38, pp. 2881–2888, 2012, doi: 10.1016/j.proeng.2012.06.337.

[8] U. Bhoi and P. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing," Int. J. Appl. or Innov. …, vol. 2, no. 4, pp. 259–264, 2013, [Online]. Available: http://ijaiem.org/Volume2Issue4/IJAIEM-2013-04-30-130.pdf.

[9] Huankai Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," in 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH), Feb. 2013, pp. 1–8, doi: 10.1109/ParCompTech.2013.6621389.

[10] D. B. L.D. and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Appl. Soft Comput., vol. 13, no. 5, pp. 2292–2303, May 2013, doi: 10.1016/j.asoc.2013.01.025.

[11] H. A. Ravani, H. A. Bheda, and V. J. Patel, "Genetic Algorithm Based Resource Scheduling Technique in Cloud Computing," vol. 1, no. 7, pp. 168–174, 2013.

[12] P. Salot, "A SURVEY OF VARIOUS SCHEDULING ALGORITHM IN CLOUD COMPUTING ENVIRONMENT," Int. J. Res. Eng. Technol., vol. 02, no. 02, pp. 131–135, Feb. 2013, doi: 10.15623/ijret.2013.0202008.

[13] R. D. L. N. Srinivasu, "A Review and Analysis of Task Scheduling Algorithms in Different Cloud Computing Environments," Int. J. Comput. Sci. Mob. Comput., vol. 4, no. 12, pp. 235–241, 2015.