IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 4, April 2016.
www.ijiset.com
ISSN 2348 – 7968

Virtual Machines Scheduling In Private Cloud Using UPF MRU

Parth Sharma[1]
Software Engineering Department (M. Tech)
SRM University
Chennai (Tamilnadu) 603203, India
parth_rajesh@srmuniv.edu.in

G.Senthil Kumar[2]
Software Engineering Department (Sr. Assistant Professor)
SRM University
Chennai (Tamilnadu) 603203, India
senthilkumar.g@ktr.srmuniv.ac.in

Abstract — Cloud computing is a fifth generation concept after the Mainframes, personal computers, client server computing and the web. The notion of Cloud computing has not only reshaped the field of distributed systems but also technically changed how service providers utilize computing today. Cloud computing can be seen as pool of virtualised computer resources in the perspective of customer. Cloud computing systems fundamentally provide access to large pools of data and proportional resources via a variety of interfaces similar like grid and HPC resource management and programming systems. One of the vital issues in cloud computing system is the scheduling of virtual resources and virtual machines (VMs). Current virtual machine companies provide services that do not take into account priority. While fees may be on a high basis, products received discrete only in terms of size or available features rather than superior priority. The proposed algorithm tries to resolve this by taking both time and priority into account when scheduling virtual machine requests from the user. This proposed model is going to address this issue using open source cloud computing environment eucalyptus, an open source software framework for cloud computing that implements what is generally indicated as Infrastructure as a Service (IaaS), systems that give users the ability to run and control entire virtual machine instances installed across a variety physical resources. This efficient resource allocation algorithm will provide a better utilization of CPU, Memory and Disk.

Key words - Cloud. Virtual Machine Scheduling. User-Priority Scheduling. Task scheduling.

I. INTRODUCTION

Clouds are a big pool of easily useful and obtainable virtualized resources (such as hardware, development platforms and/or services). These resources can be effectively reconfigured to adjust to a variable load (scale), allowing also for maximum resource utilization. This bunch of resources is typically exploited by a pay-per-use model in which guarantees are presented by the Infrastructure Provider by means of adapted SLA. Cloud computing is a new paradigm of computing with shared infrastructure which allow users to use resources on demand. Cloud Computing is generally been defined in three models Software as a Service (SaaS), Platform as a Service (Paas) and Infrastructure as a Service (IaaS). In these IaaS is the most attractive and important for the business perspective. It provides the specialty to run software on user's own virtual machine. On the basis of deployment of cloud there are four type of model public, private, community and hybrid cloud. This project is based on the private cloud. The private cloud has its own limit of virtual machines and resources and processors. For this cloud environment we can use UEC (Ubuntu Enterprise cloud 10.04) which is a complete open source eucalyptus which provide cloud computing framework. Basically it used for most in academic research organizations. There are many clouds computing options available for free or purchase, each with their own varying levels of services and fees. Each cloud user that pays yearly (rather than month-to-month), pays a fee, or pays a higher fee, should be granted a high priority. Currently, this is not the case. Rather, some existing scheduling algorithms grant users that pay on a month-to-month basis the same priority as those who pay on a yearly basis. The proposed algorithm wants to prioritize the scheduling of the virtual machine according to the users' chosen priority level. The objective of this paper is to tender a new virtual machine scheduling algorithm. This new algorithm is directed at cloud computing systems in order to increase efficiency. It schedules virtual machine requests by users according to their priority, which is determinate by the account level purchased by the user. In addition, the proposed algorithm improves resource utilization by maximizing the CPU utilization. This algorithm is the UPF_MRU algorithm (User Priority First and Maximize Resource Utilization).

II. RELATED WORK

For providing private cloud environment, there are mainly two open source system for infrastructure and virtual machine on the physical machine and provisioning the resources like allocation and scheduling. However, these both could not support the efficient policies to consolidate or provisioning. The overall goal of the system is to increase the throughput at the end; for this there is dynamic and adaptive real time virtual machine technique for High Programming Computing. Some other also presented vGreen design to utilize the resources and virtual machine efficiently without any failure and saving the system energy

ISSN 2348 – 7968

and increase the overall performance. There are some dynamic approaches to create virtual cluster that can deals with the parallel job and serial job. The proposed UPF MRU (User Priority First and Maximize Resource Utilization Scheduling Algorithm) takes care of all the important entities in virtual machine like allocation of resource and monitoring and management of resource properties like CPU, memory size, storage, workload and access privileges. The proposed approach is convey a concern that does not currently get conveyed – scheduling virtual machine service requests according to priority while conserving a high resource utilization rate and thus increasing proficiency. An increase in proficiency helps to reduce redundant energy, time, or resource waste which often leads to the slowing of a system. Also, this approach take account of the situation that virtual machine requests with high priority should be given to the calculating nodes and get the service first. Meanwhile, the scheduler monitors the free resources and increases the resource utilization. Finally, this point of view is one that keeps many stakeholders in mind - the environment, the company (service provider), and, of course, the user. The more resource utilization that occurs, the more proficient the system becomes. The more proficient a system is, the less power and electricity there is that gets wasted. The Paper has been divided into three parts – preparation, implementation, and verification. The scope of this paper will include all three parts. The parts of these sections are as follows:

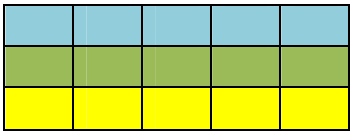
1. Come up with the existing virtual machine scheduling algorithm concept. Create proposal detail of the priority scheduling algorithm for the cloud environment.
2. Allocate each virtual machine request to the queue which represents its priority – high, medium, low. Ensure that the calculating node is always allocated the virtual machine with the highest priority. Elevate the priority of a queue after a set time period. Increase the CPU utilization through continual benchmarking during development.
3. Run the system multiple times to ensure relevant results.

A. Traditional Scheduling Algorithm

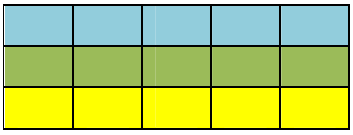
Eucalyptus allows for a node to deploy many virtual machines, but only one virtual machine per CPU core on a computing node. It uses two types of Algorithms Greedy (First Fit) and Round Robin Algorithm. Greedy allocates the node the suitable resources to a new request. Round Robin records the last position of scheduler visited. The Resources are considered as circular linked list.

B. Proposed Scheduling Algorithm

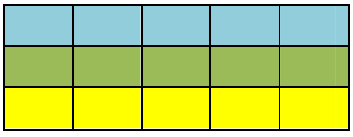
The solution to this lack of priority awareness in virtual machine scheduling is solved by the proposed algorithm. This proposed algorithm is based on the algorithm presented in the paper ―An Efficient Approach for Virtual Machines Scheduling on a Private Cloud Environment‖ which uses 3, single queues to represent the priority levels of the user requests. The proposed algorithm, however, provides: 3 queue-sets for priority. These queue-sets will be ranked according to priority – high, medium, and low.3 queues per queue-set for the number of CPUs. Each queue in a queue-set will indicate the number of CPUs to run the user request on. Each queue in a queue-set will indicate the number of CPUs to run the user request on. The CPUs will be 1 (yellow), 2 (green), and 4 (blue).



Queue-set 1 (High priority)



Queue-set 2 (Medium priority)



Queue-set 3 (Low priority)

Figure: 1

For each CPU queue in the high priority queue-set, the user request that arrives first is processed first –first come first served. If, however, there are medium priority requests

ISSN 2348 – 7968

waiting to be processed, then the high priority requests are processed for a specific time frame, e.g., t10, then medium priority requests are elevated to high priority and linked to the end of the current high priority list. So also, if there is a low priority request waiting, after a specific time interval, the low priority queue is elevated to medium priority where it waits for its turn to be elevated to high priority:

The process concept view of UPF-MRU:-

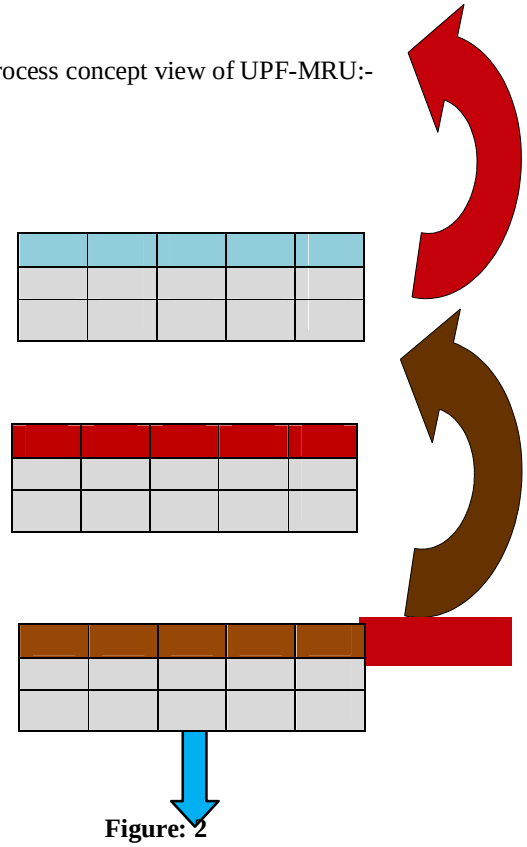


Figure: 2

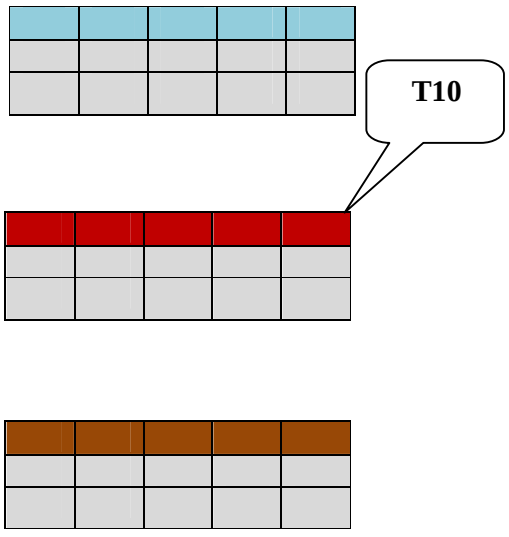


Figure: 3

(Queue is represented as empty for illustration purposes. In real life, requests would probably continue to arrive.)

The proposed algorithm differs primarily from other algorithms as it takes into account user priority levels as well as efficiency through maximizing the CPU utilization (~90%).This solution to virtual machine scheduling in a cloud computing environment is better than previously proposed or current scheduling algorithms as its primary focus is both efficiency (through high CPU utilization) and incorporating the priority ranking of each user request. Thus, not only will higher priority requests get higher priority when scheduled, but overall, the system will be more efficient while doing so. This algorithm will target a CPU utilization rate of ~90% (+/- 10%). Real-world systems estimate a range of 40%-90%. Using one data set, the UPF_MRU algorithm can optimize the priority scheduling algorithm by increasing the CPU utilization percentage. The proposed algorithm will not result in a decrease in the CPU utilization rate as compared to the same data with no efficiency optimization. The UPF_MRU scheduling algorithm can be used for virtual machine scheduling in cloud system. The high CPU utilization rate will result in a more efficient system. Input data, or test data, will be manually, and sporadically, input using the test data created for the project. This test data contains requests for high, medium, and low priority users. The data will be input via a standard console input window. Each request will contain only the information about the task to be scheduled. The virtual machine and scheduler will administer to the remaining information – the user ID, priority, arrival time, number of CPUs required, and the time required to complete the request. The UPF_MRU algorithm will utilize first come, first served scheduling and will handle the priorities of the user requests. The design consists of: 3 queue-sets, 1 queue-set per priority level, 3 queues per queue set, 1 queue per CPU number – 1, 2, 4.R = {NC, VM, N} is the set of resources on cloud infrastructure, where:-Node controller: NC = {nci}, $1 \leq i \leq H$ NC is the set of H node controllers in the proposed cloud system. H is the total amount of the node controllers. Each node controller nci has:-nci.CPU Performance is the value of CPU bandwidth of the node controller,-nci. Memory Size is the value of memory size of the node controller. Virtual machines: VM = {vmi}, $1 \leq i \leq M$ VM is the set of M virtual machines. M is maximum available VMs in NC. N is a H×M matrix. Nik represents maximum number of VMs in whole system, $1 \leq i \leq H$ and $1 \leq k \leq M$. nik = {0 if nci has not vmk , {1 if nci has the vmk Simultaneous VM request from users' as VMs, where: VMs= {VMi}, $1 \leq i \leq n$ VM is the set of n requested VM instances to be scheduled in infrastructure resource. Each requested VMi has:

ISSN 2348 – 7968

VMi.Req_Software: software required, this can be a set of software,-VMi.Req_CPU: CPU bandwidth required,-VMi.Req_Mem: memory size required, and -VMi.Req_Time: expected execution time required. The simultaneous VMs request from users 'constraints are expressed as follows: VMs≤ Nik, 1 ≤ i ≤ H and 1 ≤ k ≤ M

III. EXPERIMENTAL RESULTS

For experiment we used ANSI C and tools which we used are Xcode (for code creation), Astah Community (for UML modeling), MacBook Pro, MacBook Air and AutoCAD. The output will be generated by the program after every user request. That is, when a request has been entered, the user will receive immediate feedback in the form of a console output. This output will include the pertinent information about the request – i.e., "User <name> arrived at <time> and processing length was <time length>". In order to determine if the hypothesis has been validated or disproven, the CPU utilization will be checked. This will entail recording the initial state of the system's CPU(s) using a computer's CPU, such as with an Activity Monitor (Mac OS X for systems with 4+ cores), and then, while running the simulation, recording the system's CPU usage. Correctness will be proven by checking that the user requests have been processed in the correct order. That is, the first request is to be processed first, if that request is of a low priority, when a high priority request arrives and is queued, it will replace the earlier but low priority request. This information will be available by means of "show" functionality and will present the following information via the console:

Current time is <time>
Total user requests: <#of Requests>
Completed user requests: <completed Requests>
User <user_1> arrived at <time_1> and finished at <time_1>
User <user_2> arrived at <time_2> and finished at <time_2>
…
User <user_n> arrived at <time_n> and finished at <time_n>

In addition, to verify the program's output is correct, six typical virtual machines (scheduled by hand) were used as the test input. Three different sets of test data was created for the purpose of simulating three different types of input situations. This test data was the basis for testing the proposed algorithm. The three data sets can be broken down in the following three ways to represent three different user situations:

Test Data 1 - Simulates the situation where user requests for virtual machines come randomly.

Test Data 2 - Simulates the same virtual machine requests from users but in a different order than Test Data 1.

Test Data 3 - Same arrival order as Test Data 1 with the exception that the needed time of each virtual machine request is ten times longer than the needed time of each virtual machine request in Test Data 1.Simulated by decreasing the time quantum to1/10. After running each set of the test data, the results were visually depicted using a simple line graph and compared against the hypotheses.

For each graph, the horizontal axis (X) represents the number of the virtual machines while the vertical axis (Y) represents the percentage of the CPU utilization. Results of Test Data 1, Results of Test Data 2, Results of Test Data 3

Output analysis - In each diagram, there are two lines, a red and a blue line. The blue line represents the utilization percentage of the CPU when using a traditional priority scheduling algorithm. The red line, on the other hand, represents the utilization percentage of the CPU when using the UPF_MRU algorithm (the proposed Highest Priority First and Maximize Resource Utilization algorithm). Analyzing the following pairs, it can be seen that:

1, 2: Greater numbers of virtual machine requests with smaller CPU requirements in the waiting queue result in a distinct promotion.

1, 3: Based on the fact that the blue lines are identical, a decrease of the time quantum of system will lead to a greater CPU utilization.

An abnormal case for this situation, virtual machines in a cloud based system, would be if two or more users request the services of a virtual machine at exactly the same time. This situation is not the normal situation that is accounted for by the algorithm for this situation could not occur. When two or more users request the services of a virtual machine, the request is first received by the cluster which in turn schedules the requests to different nodes. The new algorithm is focused on the schedule on the node level and not the cluster. Thus, since the cluster is not able to assign two different requests at the exact same time to the node, the algorithm is unaffected by this abnormal case.

ISSN 2348 – 7968

IV. RESULT AND DISCUSSION

A. Test Data Sets

Table: 1

	Test Data 1			
User ID	**Priority**	**Arrival Time**	**Need Time**	**CPU**
1	3	0	10	2
30	2	5	22	4
2	3	8	5	2
15	1	12	30	1
45	1	14	8	4
36	2	20	18	4
24	3	36	10	1
11	3	40	16	1
9	1	41	8	2
31	2	46	12	4
59	2	50	13	2
42	1	52	16	1
1	3	0	10	2
30	2	5	22	4
2	3	8	5	2

Table: 2

	Test Data 2			
User ID	**Priority**	**Arrival Time**	**Need Time**	**CPU**
30	2	0	22	4
31	2	4	12	4
36	2	6	18	4
1	3	12	10	2
2	3	18	5	2
15	1	22	30	1
103	2	25	30	4
33	3	34	7	2
45	1	44	8	4
24	3	46	10	1
11	3	50	16	1
9	1	51	8	2
59	2	54	13	2
42	1	56	16	1
99	3	64	40	4

Table: 3

	Test Data 3			
User ID	**Priority**	**Arrival Time**	**Need Time**	**CPU**
1	3	0	100	2
30	2	5	120	4
2	3	8	50	2
15	1	12	300	1
45	1	14	80	4
36	2	20	180	4
24	3	36	100	1
11	3	40	160	1
9	1	41	80	2
31	2	46	120	4
59	2	50	130	2
42	1	52	160	1
103	2	55	300	4
99	3	58	400	4
33	3	64	70	2

ISSN 2348 – 7968

B. Input/output listing

	Input 1 With Corresponding Out Put				Priority Scheduling without efficiency	Priority Scheduling with efficiency
User ID	**Priority**	**Arrival Time**	**Need Time**	**CPU**		
1	3	0	10	2	<Priority Scheduling without efficiency> Here is the information of finished VM: User 1 arrived at 0 and finished at 11. User 2 arrived at 8 and finished at 18. User 45 arrived at 14 and finished at 87. User 30 arrived at 5 and finished at 124. User 24 arrived at 36 and finished at 134 User 36 arrived at 20 and finished at 140 User 33 arrived at 64 and finished at 146 User 9 arrived at 41 and finished at 151. User 31 arrived at 46 and finished at 169 User 59 arrived at 50 and finished at 176 User 11 arrived at 40 and finished at 183 User 42 arrived at 52 and finished at 200 User 15 arrived at 12 and finished at 223 User 103arrived at 55 and finished at 235 User 99 arrived at 58 and finished at 245 total system cpu is: 980 total use cpu is: 678 percentage is:0.69183	<Priority Scheduling with efficiency> Here is the information of finished VM: User 1 arrived at 0 and finished at 10 User 2 arrived at 8 and finished at 24 User 24 arrived at 36 and finished at 50 User 11 arrived at 40 and finished at 56 User 15 arrived at 12 and finished at 60 User 59 arrived at 50 and finished at 75 User 33 arrived at 64 and finished at 84 User 9 arrived at 41 and finished at 85 User 42 arrived at 52 and finished at 95 User 30 arrived at 5 and finished at 129 User 45 arrived at 14 and finished at 130 User 31 arrived at 46 and finished at 147 User 36 arrived at 20 and finished at 156 User 103 arrived at 55 and finished at188 User 99 arrived at 58 and finished at 198 total system cpu is: 792 total use cpu is: 688 percentage is:0.868687
30	2	5	22	4		
2	3	8	5	2		
15	1	12	30	1		
45	1	14	8	4		
36	2	20	18	4		
24	3	36	10	1		
11	3	40	16	1		
9	1	41	8	2		
31	2	46	12	4		
59	2	50	13	2		
42	1	52	16	1		
103	2	55	30	4		
99	3	58	40	4		
33	3	64	7	2		

Table: 4

ISSN 2348 – 7968

	Input 2 With Corresponding Out Put				Priority Scheduling without efficiency	Priority Scheduling with efficiency
User ID	**Priority**	**Arrival Time**	**Need Time**	**CPU**	<Priority Scheduling without efficiency> Here is the information of finished VM: User 2 arrived at 18 and finished at 46 User 1 arrived at 12 and finished at 79 User 30 arrived at 0 and finished at 87 User 31 arrived at 4 and finished at 89 User 33 arrived at 34 and finished at 103 User 45 arrived at 44 and finished at 146 User 24 arrived at 46 and finished at 151 User 9 arrived at 51 and finished at 156 User 36 arrived at 6 and finished at 159 User 59 arrived at 54 and finished at 183 User 11 arrived at 50 and finished at 191 User 42 arrived at 56 and finished at 201 User 103 arrived at 25 and finished at 228 User 15 arrived at 22 and finished at 232 User 99 arrived at 64 and finished at 245 total system cpu is:980 total use cpu is: 678 percentage is:0.691837	<Priority Scheduling with efficiency> Here is the information of finished VM: User 2 arrived at 18 and finished at 49 User 30 arrived at 0 and finished at 58 User 31 arrived at 4 and finished at 59 User 24 arrived at 46 and finished at 74 User 33 arrived at 34 and finished at 78 User 1 arrived at 12 and finished at 80 User 9 arrived at 51 and finished at 85 User 59 arrived at 54 and finished at 90 User 11 arrived at 50 and finished at 95 User 15 arrived at 22 and finished at 104 User 42 arrived at 56 and finished at 107 User 36 arrived at 6 and finished at 125 User 45 arrived at 44 and finished at 130 User 103 arrived at 25 and finished at 164 User 99 arrived at 64 and finished at 181 total system cpu is:724 total use cpu is: 687 percentage is:0.948895
30	2	0	22	4		
31	2	4	12	4		
36	2	6	18	4		
1	3	12	10	2		
2	3	18	5	2		
15	1	22	30	1		
103	2	25	30	4		
33	3	34	7	2		
45	1	44	8	4		
24	3	46	10	1		
11	3	50	16	1		
9	1	51	8	2		
59	2	54	13	2		
42	1	56	16	1		
99	3	64	40	4		

Table: 5

ISSN 2348 – 7968

	Input 3 With Corresponding Out Put				Priority Scheduling without efficiency	Priority Scheduling with efficiency
User ID	Priority	Arrival Time	Need Time	CPU	<Priority Scheduling without efficiency>	<Priority Scheduling with efficiency>
1	3	0	100	2	Here is the information of finished VM:	Here is the information of finished VM:
30	2	5	220	4		
2	3	8	50	2	User 2 arrived at 8 and finished at 632	User 24 arrived at 36 and finished at 152
15	1	12	300	1	User 33 arrived at 64 and finished at 1081	User 11 arrived at 40 and finished at 216
45	1	14	80	4	User 45 arrived at 14 and finished at 1150	User 2 arrived at 8 and finished at 236
36	2	20	180	4	User 9 arrived at 41 and finished at 1203	User 42 arrived at 52 and finished at 254
24	3	36	100	1	User 1 arrived at 0 and finished at 1206	User 15 arrived at 12 and finished at 375
11	3	40	160	1	User 24 arrived at 36 and finished at 1376	User 1 arrived at 0 and finished at 386
9	1	41	80	2	User 31 arrived at 46 and finished at 1571	User 45 arrived at 14 and finished at 860
31	2	46	120	4	User 59 arrived at 50 and finished at 1653	User 31 arrived at 46 and finished at 1061
59	2	50	130	2	User 11 arrived at 40 and finished at 1853	User 36 arrived at 20 and finished at 1301
42	1	52	160	1	User 42 arrived at 52 and finished at 1874	User 30 arrived at 5 and finished at 1419
103	2	55	300	4	User 36 arrived at 20 and finished at 1936	User 103 arrived at 55 and finished at 1583
99	3	58	400	4	User 30 arrived at 5 and finished at 2081	User 99 arrived at 58 and finished at 1683
33	3	64	70	2	User 15 arrived at 12 and finished at 2340	User 33 arrived at 64 and finished at 1714
					User 103 arrived at 55 and finished at 2350	User 9 arrived at 41 and finished at 1724
					User 99 arrived at 58 and finished at 2450	User 59 arrived at 50 and finished at 1773
					total system cpu is:9800 total use cpu is: 6780 percentage is:0.691837	total system cpu is:7092 total use cpu is: 6780 percentage is:0.956007

Table: 6

In figure, there is a graph for each of the three test data sets: Test data 1, test data 2, and test data 3. In each graph, there are two lines, a blue line and a red line. The blue line depicts the results of the test data set without efficiency whereas the red line depicts the test data set with efficiency. Reviewing the results of each graph, it can be seen that the blue line supports hypothesis, that a priority scheduling algorithm can be used in a cloud-based computing system. Now, in reviewing the red line, the second hypothesis is supported, that the UPF_MRU algorithm can increase the CPU utilization percentage from that of the basic scheduling algorithm.

ISSN 2348 – 7968

C. Results From Test Data Sets

Number of Virtual Machines (X-Axis)	Test Data1 w/o Efficiency (Y-Axis)	Test Data 1 with Efficiency (Y-Axis)
1	0.5	0.5
2	0.84375	0.84375
3	0.797297	0.842857
4	0.552239	0.580769
5	0.6	0.626712
6	0.677419	0.692308
7	0.635922	0.707921
8	0.584034	0.752577
9	0.57874	0.777778
10	0.615108	0.801802
11	0.605263	0.837838
12	0.571429	0.799587
13	0.636364	0.834437
14	0.697479	0.86911
15	0.691837	0.868687

Table: 7

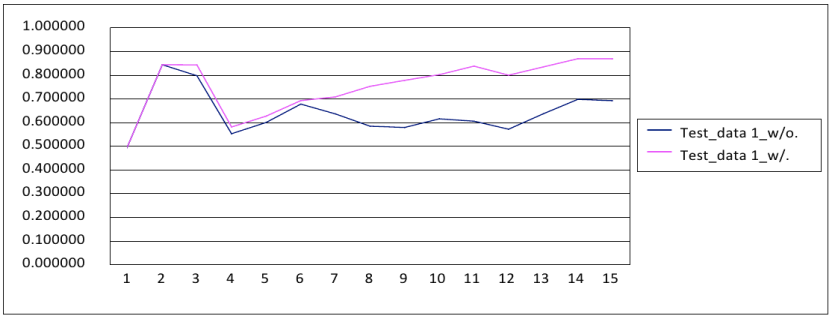


Figure: 4

ISSN 2348 – 7968

Number of Virtual Machines (X-Axis)	Test Data2 w/o Efficiency (Y-Axis)	Test Data 2 with Efficiency (Y-Axis)
1	1	1
2	1	1
3	1	1
4	0.919355	0.919355
5	0.88806	0.959677
6	0.690722	0.741848
7	0.76378	0.80123
8	0.75	0.802326
9	0.764085	0.813869
10	0.730263	0.888
11	0.684524	0.860294
12	0.676136	0.929688
13	0.664021	0.950758
14	0.631707	0.938406
15	0.691837	0.948895

Table: 8

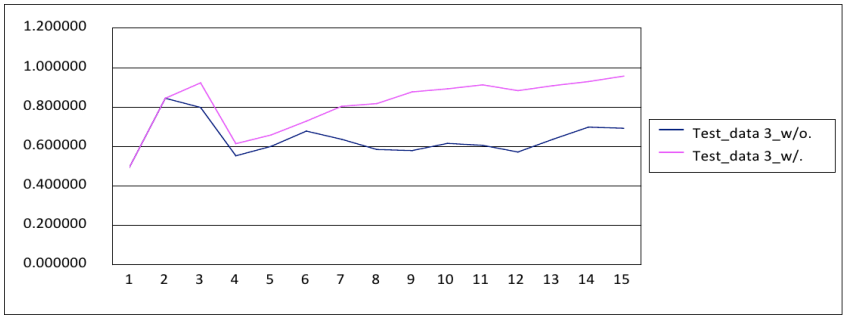


Figure: 5

ISSN 2348 – 7968

Number of Virtual Machines (X-Axis)	Test Data 3 w/o Efficiency (Y-Axis)	Test Data 3 with Efficiency (Y-Axis)
1	0.5	0.5
2	0.84375	0.84375
3	0.797297	0.921875
4	0.552239	0.61371
5	0.6	0.657857
6	0.677419	0.727841
7	0.635922	0.803013
8	0.584034	0.816886
9	0.57874	0.876043
10	0.615108	0.891554
11	0.605263	0.911794
12	0.571429	0.882434
13	0.636364	0.907753
14	0.697479	0.928332
15	0.691837	0.956007

Table: 9

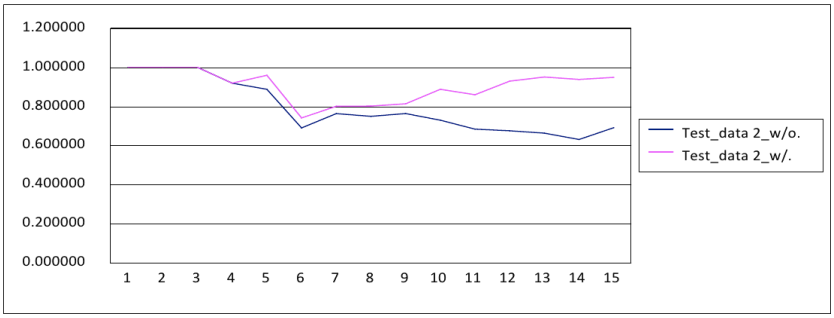


Figure: 6

ISSN 2348 – 7968

D. Proof of Output Correctness

	Input 1 With Corresponding Out Put				Priority Scheduling without efficiency	Priority Scheduling with efficiency
User ID	**Priority**	**Arrival Time**	**Need Time**	**CPU**	<Priority Scheduling without efficiency> Here is the information of finished VM: User 1 arrived at 0 and finished at 11 User 2 arrived at 8 and finished at 18 User 45 arrived at 14 and finished at 62 User 30 arrived at 5 and finished at 72 User 36 arrived at 20 and finished at 75 User 15 arrived at 12 and finished at 93 total system cpu is: 372 total use cpu is: 236 percentage is: 0.634409	<Priority Scheduling with efficiency> Here is the information of finished VM: User 1 arrived at 0 and finished at 10 User 2 arrived at 8 and finished at 24 User 45 arrived at 14 and finished at 38 User 15 arrived at 12 and finished at 60 User 30 arrived at 5 and finished at 76 User 36 arrived at 20 and finished at 83 total system cpu is: 332 total use cpu is: 236 percentage is: 0.710843
1	3	0	10	2		
30	2	5	22	4		
2	3	8	5	2		
15	1	12	30	1		
45	1	14	8	4		
36	2	20	18	4		

Table: 10

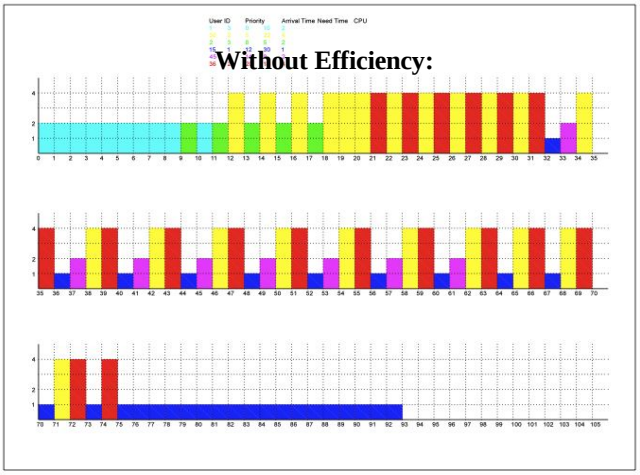


Figure: 7

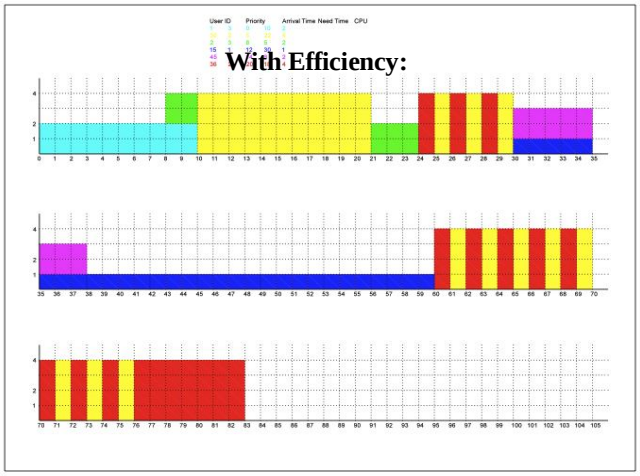


Figure: 8

V. CONCLUSION

Overall, scheduling virtual machines in the cloud lack an algorithm that combines priority and efficiency. This combination would be an algorithm that can maintain the order in which virtual machine requests are received while also maintaining a high CPU utilization rate. To this end, the UPF_MRU algorithm was created. The priority consideration is achieved with elements of the FCFS and Round Robin algorithm. Virtual machine requests that come in first will have priority over those who come in second (FCFS). Furthermore, higher priority users (determined by the plan's cost, benefits chosen, etc.) have priority over medium or low priority users. Once a request has been processed, it is rotated to the end of the queue to be serviced again when its time comes around again (Round-Robin). The algorithm focused on efficiency by increasing the utilization of the CPU. This increased utilization of the CPU is achieved by decreasing the size of the time quantum of the system. As the time quantum decreases, the CPU utilization percentage increases. As compared with a basic priority scheduling algorithm, the UPF_MRU algorithm has a greater utilization percentage. Thus, the UPF_MRU algorithm can make the scheduling of virtual machines in a cloud-based system more efficient while retaining the highest priority first model. Based on the research conducted prior to embarking on this paper, there does not seem to be cloud scheduling algorithms that are focused on priority and efficiency. Creating a scheduling algorithm that captures the user's practical needs and requirements would be extremely useful in virtual machine systems. In addition, as there do not appear to be any differences in cloud user plans with regard to priority; this would a means of encouraging commerce competition for the most efficient CPU. This in turn would help companies to focus on the efficiency of their CPU in addition to their other considerations.

VI. REFERENCES

1. Bin Lin, Peter A. Dinda, Dong Lu. 2004. User-driven Scheduling of Interactive Virtual Machines. IEEE, 380-387, DOI:10.1109/GRID.2004.64
2. Yubin Xia, Yan Niu, Yansong Zheng, Ning Jia, Chun Yang, Xu Cheng 2008. Analysis and Enhancement for Interactive Oriented Virtual Machine Scheduling, Volume: 2, DOI: 10.1109/EUC.2008.143
3. D.Nurmi, R.Wolski, C.Grzegorczyk, G.Obertelli, S.Soman, L.Youseff, and D. Zagorodnov 2008. IEEE, 124-131, DOI:10.1109/CCGRID.2009.93
4. G. Dhiman, G.Marchetti, T. Rosing 2009 VGreen: A System for Energy Efficient Computing in Virtualized Environments, IEEE, 243-248. DOI:10.1145/1594233.1594292
5. N. Bobroff, A.Kochut and K.Beaty 2009. Dynamic placement of virtual machines for managing SLA violations, IEEE, 119-128. DOI:10.1109/INM.2007.374776
6. Chen H., Jin H., Hu K. 2010. Affinity-aware Proportional Share Scheduling for Virtual Machine, IEEE, 75-80. DOI:10.1109/GCC.2010.27
7. L.Wangy, G.V.Laszewskiy, M.Kunzez and J.Taoz 2011. Schedule Distributed Virtual Machines in a Service Orientedent Environment, IEEE, 230-236. DOI:10.1109/AINA.2010.47
8. Hsu Mon Kyi, Thinn Thu Naing 2011. An Efficient Approach for virtual machines scheduling on a private cloud environment, IEEE, 365-369. DOI:10.1109/ICBNMT.2011.6155958
9. Ding, Yidong Li, Lihua Ai, and Siwei Luo 2012. An Adaptive Resource Scheduling Mechanism based on User Behaviour Feedback in Cloud Computing, Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2012 13th International Conference on, 543-547. DOI:10.1109/PDCAT.2012.37
10. Zheng, Z., Zhang, Y., Wang, Z., Zhang, X. 2012. The Multi-processor Load Balance Scheduler Based on XEN, Systems and Informatics (ICSAI), 2012 International Conference on, 905-909. DOI:10.1109/ICSAI.2012.6223154
11. Huankai Chen, Professor Frank Wang, Dr Na Helian 2013. User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing, IEEE,1-8.DOI:10.1109/ParCompTech.2013.6621389
12. Karan D. Prajapati, Pushpak Raval, M. B. Potdar 2012. Comparison of Virtual Machine Scheduling Algorithms in Cloud Computing, International Journal of Computer Applications, (0975-8887) Volume 83 – No 15, December 2013
13. Rutvij H. Jhaveri, Yash P. Dave, Avani S. Shelat 2014. Various Job Scheduling Algorithms in Cloud Computing: A Survey, Information Communication and Embedded Systems (ICICES), 2014 International Conference on, 1-5. DOI:10.1109/ICICES.2014.7033909