An Efficient Technique of Load Balancing in Cloud Computing using Priority based and Round Robin based Scheduling

Priyanka Prajapati
M Tech Scholar
Alpine Institute of Technology , Ujjain

Prof. Amit Kumar Sariya
H.O.D. (Computer Science and Engineering)
Alpine Institute of Technology , Ujjain

Abstract—Cloud computing offers utility oriented services to users. It also reduces the maintenance cost of the infrastructure as well as initial investment cost for users or start-up companies. Due to the effectiveness of these services, many companies or institutes are trying to adopt this technology for offloading scientific or industrial applications. Virtualization in cloud computing is used to build virtual infrastructure which is managed by a load balancer. Efficient design of the load balancer can reduce over or under pro- visioning. Load balancer contains virtual machine allocation technique which performs an important role with respect to power consumption of the data centre. In this paper, we propose a system architecture and the virtual machine allocation algorithm for the load balancer in a scientific federated cloud. We have tested the proposed approach in a scientific federated cloud. Experimental results show that the proposed algorithm not only increases the utilization of resources but also reduces the energy consumption.

Keywords-Cloud Computing; Virtual Machine; Load Balancing; High Utilization of Resources; Power Consumption

I. INTRODUCTION

Cloud computing offers infrastructure, platform and soft- ware as a service for the execution of complex Computation, data storage and software execution [1]. The data centers use virtualization technology to share the computing resources through virtual machines. Virtual machine allocation technique creates virtual machines on the physical machines to maintain a Service Level Agreement (SLA). This technique is responsible for over or under provisioning problem. Furthermore, over provisioning of resources may lead to the higher power consumption. Considerable amount of percentage of total data center's expenditure is required for energy consumption [2].

Physics experiments require huge amount of computational power which is provided by many data centers. CDF [3] is a physics experiment of particle accelerator called Tevatron. Global Science experimental Data hub Center (GSDC) of KISTI [4] supports this experiment by providing computing resources which are running 24 hours of 7 days, but not always resources are fully utilized because of various reasons. Therefore, under utilization of resources as well as over provisioning problem affect power consumption.

Dynamic allocation of computational resource and server consolidation can reduce the power consumption as well as increase the resource utilization of data center. In order to manage the virtual infrastructure, bunch of software's work as a middle ware between virtualization technology or hypervisor and user. OpenNebula [5] is an open-source software for management of virtual clusters. It supports the hypervisors like Xen, KVM and VMware. We have used KVM [6] as a hypervisor in the implementation [8].

This paper proposes a system architecture for the scientific federated cloud as well as priority-based virtual ma- chine allocation algorithm for load balancer. The proposed methodology in this paper not only solves the problem of load balancing but also reduces power consumption. It is tested on scientific federated cloud and the result shows that the proposed technique reduces considerable energy consumption and increases the resource utilization [9] [10] [11].
The paper is organized as follows. Motivation is described in Section II. Section III presents the proposed system architecture. Load balancer is included in the Section IV which also contains the proposed algorithm. Section V gives experimental results. Section VI concludes the paper.

II. MOTIVATION

In order to support CDF experiment, GSDC has allocated the computing resources statically (dedicated). Figure 1 shows the monthly usage of the resources [12] [13] [14].

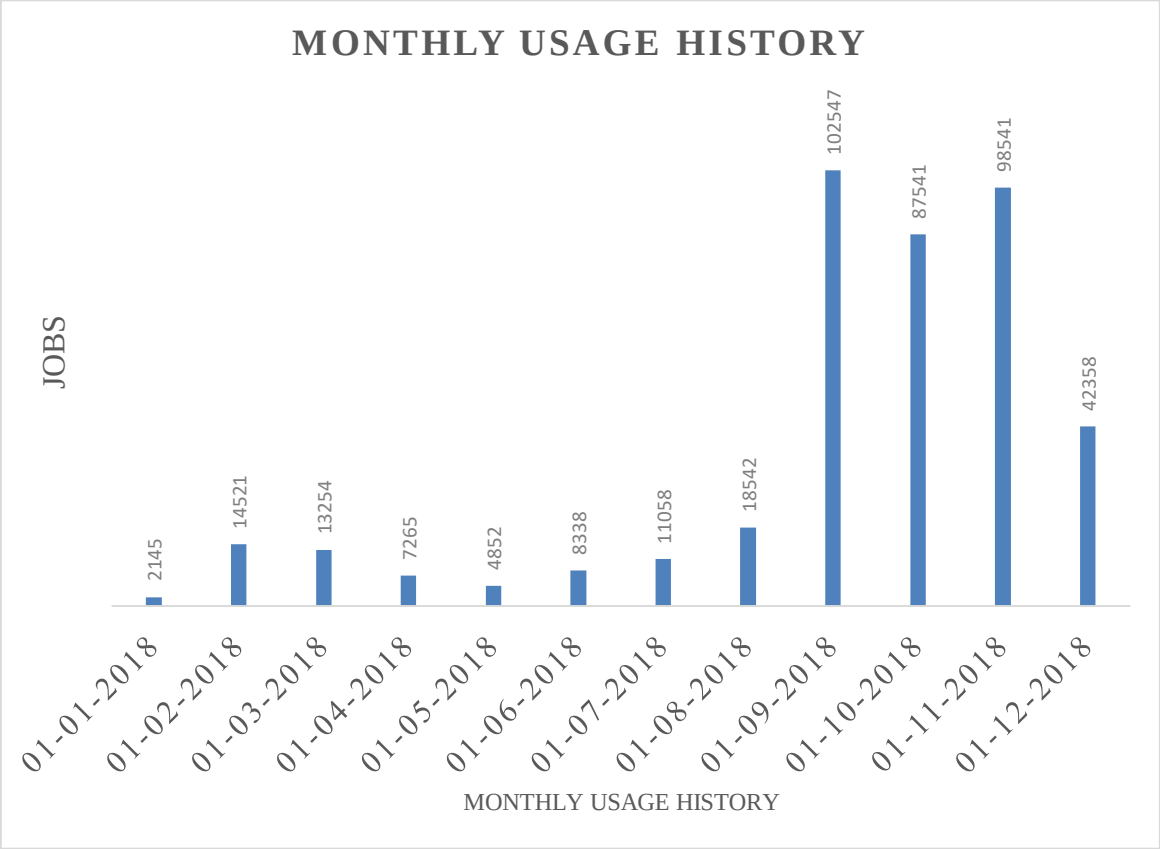

Figure 1. Monthly Usage History

From Figure 1, we can conclude that there is a problem of under utilization of resources. The computing farm can run around 80,000 jobs in a month, depending on the available resources as well as job's execution time. Due to the static allocation of these resources to the CDF experiment, GSDC is unable to use it for other purpose. Dynamic allocation technique for computational resources is not only reduce the power consumption of the data center but also increase resource utilization.

III. SYSTEM ARCHITECTURE

This section explains the proposed system architecture. Figure 2 shows the system architecture which is centralized management system. It can be installed in the data center which is used to manage other data centers. It consists of three layers, first one is scientific application. Users submit jobs to the federated cloud infrastructure through a batch system. Condor [7] batch system is used in our implementation. This batch system has a queue, in which jobs can wait until the virtual resources are available.

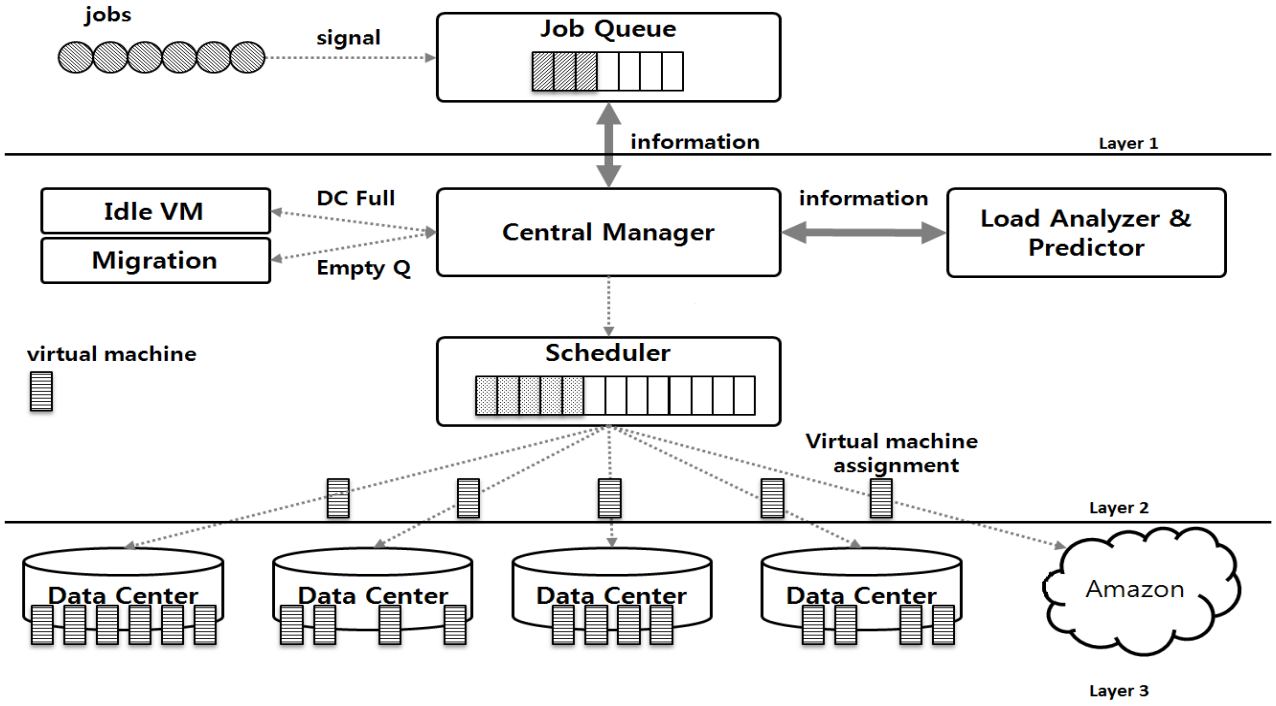

Figure 2. System Architecture

Next layer is a load balancer. The load balancer reads the Condor queue status and calls the module of creation, deletion or migration. The load balancer maps the virtual machine and physical machine together. Creation module decides location for creation of a new virtual machine in a data center. Deletion module includes the deletion of the idle virtual machine. Migration module transfers the virtual machine from one physical machine to another physical machine to improve utilization of resources and to reduce power consumption.

Last layer is a scientific federated cloud which consist of free and paid data centers with physical machines (PMs). In this implementation, we have used Amazon as a paid cloud. All data centers except Amazon assign the infrastructure statically to this proposed system.

Figure 3 describes the flow diagram of the proposed system. Each physical machine of a data center has definite number of slots for virtual machine e.g. 1 machine has 16 slots which means only 16 virtual machines can be run by that physical machine as per our setting. Load balancer section explores the detail information about each module.

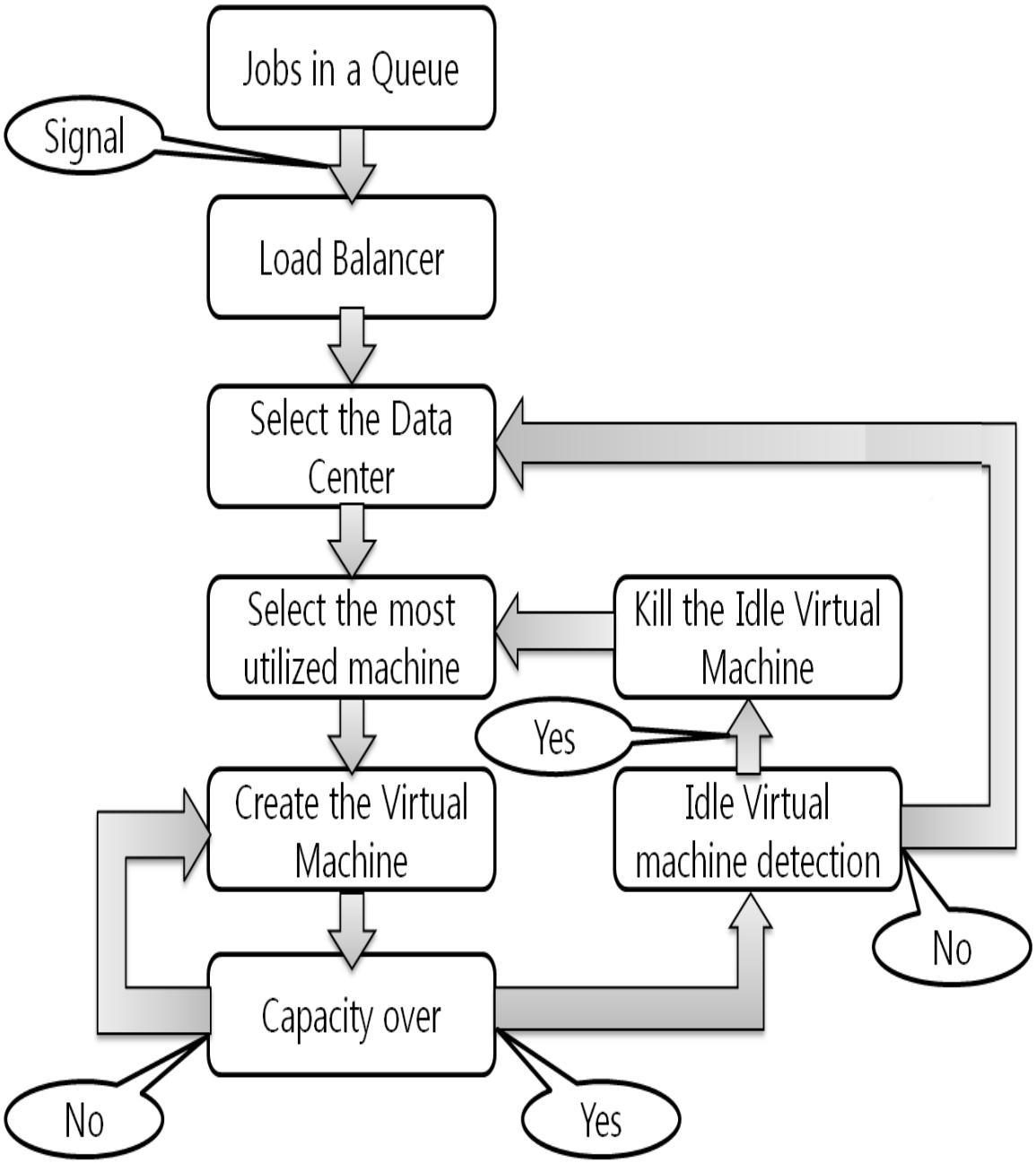

Figure 3. Automation Process

IV. LOAD BALANCER

This module is used to manage virtual machines with creation, deletion and migration module. This section consists of information about data structure and modules which are used in the proposed system.

A. Data Structure

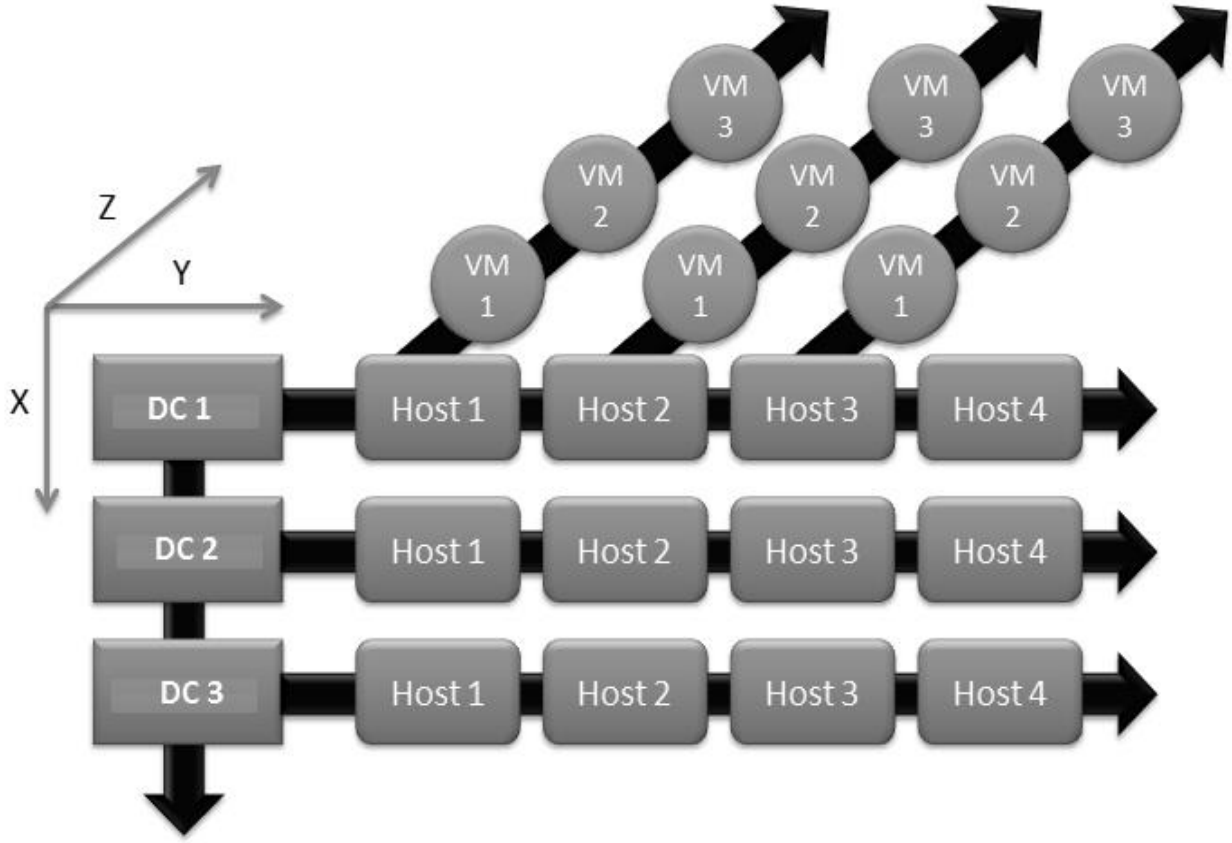

Figure 4. Data Structure

It is a three dimensional structure as per shown in the Figure 4. X-axis represents the data center *i* (DC_i) while Y-axis describes the host or physical machine with id *j* included in that data center *i* (DC_iH_j). Z-axis describes the virtual machines with id *k* ($DC_iH_jV_k$) deployed on a host with id *j* within a data center *i*. This three dimensional data structure is used for taking all the decisions including creation, deletion or migration of virtual machine. In a data center's data structure, some of the information is added like number of host or physical machines present in that data center *i* (DC_iNH), maximum number of virtual machines can be deployed in that data center *i* (DC_iMV), priority of the data center *i* (DC_iPr). DC_iFV *is* available slots to deploy new virtual machine in a data center i. *m is* the number of data center. *MAX* represents the maximum number of virtual machine can be created under this overall system. Equation 1 helps to find the value of *MAX*.

$$MAX = \sum_{I=0}^{M} DC_I MV \qquad (1)$$

Priority of the data center *i* (DC_iPr) can be decided by number of parameters. The host machine's data structure also contains information like maximum number of virtual machines which can be deployed on the specific host ma- chine *j* of the data center *i* (DC_iH_jMV) and available slots present in the host for the deployment of new virtual machine (DC_iH_jFV). *n* is the number of host machine present in a data center. Virtual machine's data structure includes the additional information like virtual machine state ($DC_iH_jV_kS$) and information of the job which is assigned to this virtual machine like status of the job ($DC_iH_jV_kJS$). Equation 2 and 3 gives the maximum number of virtual machines can be deployed and maximum number of free slots available in a data center *i*, respectively.

$$DC_iMV = \sum_{J=0}^{N} DC_iH_jMV \qquad (2)$$

$$DC_iFV = \sum_{J=0}^{N} DC_iH_jFV \qquad (3)$$

B. Priority Decision Policy

Priority of a data centre (DCiPr) is decided by the number of parameters. In this technique, we have used cost, transfer rate, performance and availability of the data centre. Performance is calculated in terms of completed jobs per unit's time and availability is measured by considering down time of the data centre. In this section, we describe the matrix-based data canter selection policy. Every data centre has its own costing mechanism for each virtual machine as well as performance and availability. There are two types of matrices as shown in Table I and Table II. First one is the cost vs transfer rate (FM CvT) and the other is performance vs availability (FM PvA). These matrices are calculated at runtime. Transfer rate is related with the network delay and cost is for utilizing resources for a specific time. Performance is the number of jobs completed per unit time while availability describes the resources available for the computation. V (FM CvT) and P (FM PvA) is a function to select the data centers based on user requirement. For example, if the user is more concern about cost then first 3 data centers are selected

using function V. In case of function P, if the user is required high performance then first 3 data centers are selected. After getting these two sets of data centers from function P and V, next step is to find common data centers from both sets. Q (Set3) assigns the priorities to these data centers based on user's requirement.

Table I Cost vs Transfer Rate

Data Center ID	1	2	3	4
Cost ($)	0.1	0.15	0.17	0.2
Transfer rate (Gbps)	10	1	6	2

Table II Performance vs Availability

Data Center ID	1	2	3	4
Performance (jobs/time)	1200	1180	990	988
Availability (%)	98	95	92	94

These priorities will not be constant for every user and can be changed depending on resource availability and the user's requirement.

Algorithm 1 Data Centre Selection

```
1: procedure DC SECTION (LIST)
2: for i ← 1 m do . Fill the values in the matrix
3: FM _CvT ← CostPerVM_ DCi
4: FM _ CvT ← TransferRate_ DCi
5: end for
6: Set1 ← V (FM_ CvT) . Selection of data centers
7: for i ← 1;m do . Fill the values in the matrix
8: FM_PvA ← Performance_ DCi
9: FM PvA Availability DCi
10: end for
11: Set2 ← P(FM_ PvA) . Selection of data centers
12: Set3 ← Set1∩Set2 . Select common data centers
13: Set3 ← Q(Set3) . Assign the priorities
14: return Set3
15: end procedure
```

C. Utilization of Resources

Server consolidation is running multiple underutilized virtual machines on a single physical machine to reduce power consumption. Each physical machine can launch certain number of virtual machines as per our environment. These are called slots per physical machine. Most utilized host contains less number of free slots available to launch new virtual machine. This policy is not applicable to Amazon. If the waiting job queue is empty or available virtual machines are more than the required virtual machines to manage the existing work load then idle virtual machines can be deleted. Therefore, deletion of such idle virtual machines helps to reduce the over provisioning problem and also reduces the overall power consumption. Virtual machine creation and then booting that virtual machine needs considerable amount of time. This time is called virtual machine launching overhead which is around 2 to 4 minutes.

D. Priority-Based Algorithm

After considering these challenges, we propose load balancing algorithm named Priority-based virtual machine al- location. This algorithm manages the virtual machine based on status of the waiting job queue. There is a possibility that while booting newly created virtual machine, jobs can be completed by other existing virtual machines. Therefore, creation of the virtual machine is the most important decision. To create virtual machine, the algorithm needs to wait for time *t* to fetch the queue. This time *t* value depends on the number of waiting jobs in a queue or job's average execution time or virtual machine launching overhead time.

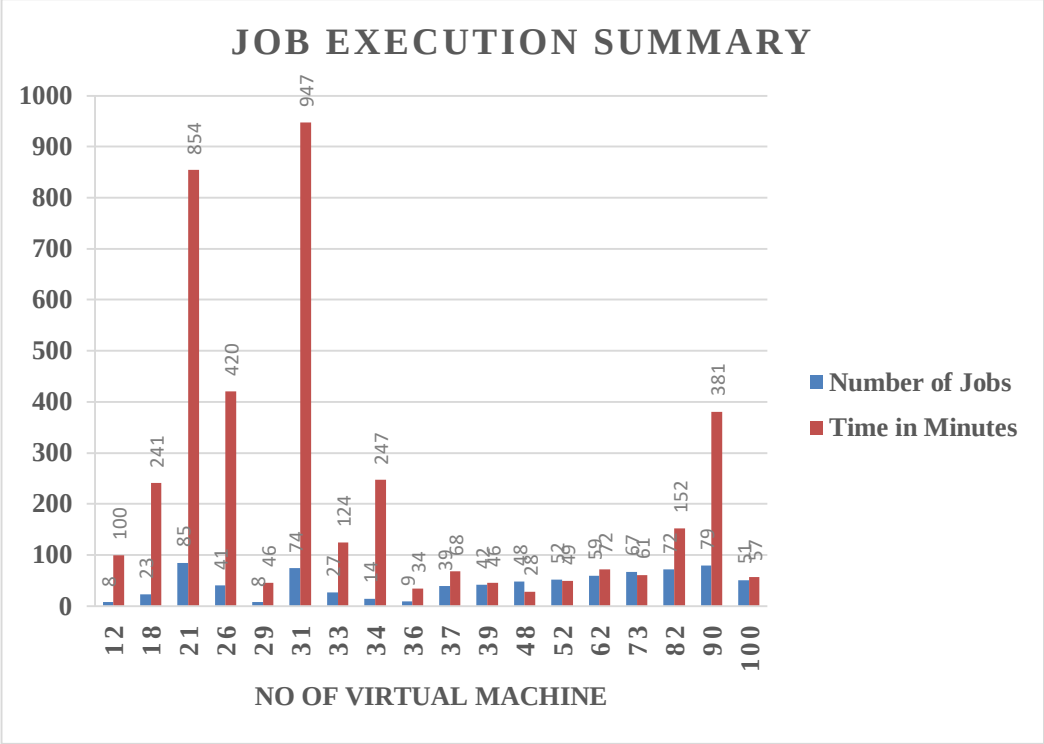

Figure 5. Job Execution Summary

In order to get average execution time we have used historical data, Figure 5 shows the number of jobs as well as the Time in Minutes of scientific CDF jobs. X- Axis represents the time and Y-axis represents the Minutes as well as number of jobs. The main concept behind t value is to avoid over provisioning.

Table III Priority

Data Center ID	Priority	Priority	Priority	Priority
Data Center 1 (Free)	1	3	2	1
Data Center 2 (Free)	2	1	3	2
Data Center 3 (Free)	3	2	1	3
Cloud 4 ((Amazon, Paid))	4	4	4	4

Whenever the data center's capacity to create new virtual machine gets over in a federated cloud environment then priority has to be changed by calling change priority function. This function changes the priority of the data center. Table III describes how the priorities can be changed. Algorithm 1 assigns the priority to the data center. Paid cloud is always at low priority, because most of the data centers are free to use in a scientific environment. The policy stats that virtual machines are deployed on a paid cloud only when there are no available slots in free data centers.

Algorithm 2 describes the priority-based virtual machine allocation in a scientific federated environment. It traverses through all the data centers till the data center's priority is equal to 1. Then it finds the most utilized Machine_id. CREATE_VM (DC_iH_j) function creates the virtual machine on the host j of data center i. After creation of virtual machine, the number of available slot which is present in data center (D C_i F V) as well as in host data structure (DC_iH_jFV), is reduced by 1. Algorithm 2 is called after each interval of time t.

Algorithm 2 Priority-based Virtual Machine Allocation

1: procedure PRIORITY-BASED_VM_ ALLOCATION ()
2: for DC_i; (i ← 1;m) do
3: if DC_iPr == 1 then. Searching for priority
4: Cloud ← DC_i
5: break
6: end if
7: end for
8: DC_i ← Cloud
9: min ← 0 . Initialization

10: for DCiHj ; (j ←1; n) do . for all hosts
11: if min > DCiHjFV or min == 0 then
12: if DCiHjFV > 0 then
13: Machine id← DCiHj
14: min ← DCiHjFV
15: end if
16: end if
17: end for
18: DCiHj ← Machine id
19: DCiFV ← DCiFV □ 1
20: DCiHjFV← DCiHjFV □ 1
21: CREATE VM (DCiHj)
22: end procedure

E. Round Robin Police

Algorithm: Resource Selection and Monitoring using Round Robin Police.
Input: Task 't_i' in the task pool, $t_i \epsilon T = \{1, 2....n\}$, Resource '$r_j$' in the cloud data center, $r_j \epsilon R = \{1, 2...m\}$, Cost Matrix ($\mathbf{C}_{n\times m}$) table.
Begin

Begin
1. **for** $t_i \in T$ **do**
2. **for** $r_j \in R$ **do**
3. Consider the Cost Matrix table ($\mathbf{C}_{n\times m}$)
4. Check the resource availability for all the tasks
5. **For** each task t_i calculate min (e_k) between resources **do**
6. $La \leftarrow t_i\ (e_k) - min\ (e_k)$
7. **for** each resource *rj* calculate min(e_k) between the tasks **do**
8. $Lb \leftarrow t_i\ (e_k) - min\ (e_k)$
9. Find $r_j(e_k) = 0$
10. **For** each task t_i re-compute *min* (e_k) among resources **do**
11. Prepare List (L_c) with execution times in resources except $t_i(e_k)$ is zero
12. Prepare List (L_d) with execution times consider by both tasks and in resources
13. Calculate $L_c - min\ (e_k)$
14. Calculate $L_d + min\ (e_k)$
15. Reconstruct cost matrix ($\mathrm{Cl}_{n\times m}$)
16. **If** tasks are in ready queue **then**
17. Task scheduler re-compute all tasks and then repeatedly do: (apply Round Robin scheduling algorithm)
18. Specify the time quantum value based on execution time of tasks upon assigned resources
19. Allocate the task to the resources which is best fitted based on assignment mechanism
20. Insert the task into next round of queue, if the time quantum value is expired
21. Maintain slot table for local mapping to record execution schedule of resources
22. **Else**
23. Break
24. **EndIf**
25. Calculate average execution time of tasks
26. Calculate throughput
27. Calculate resource utilization rate
28. Calculate scheduling success rate
29. **End for**
30. **End for**
31. **End for**
32. **End for**
End

V. EXPERIMENTS

We conducted several test cases to test the proposed methodology and performance of the system. This section describes the information about setup environment.

A. Setup

In order to setup the environment, we have used servers which have 4 dual-socket (4-core) Intel Core i3-2.6Gh with 8 GB of DDR4/1600MHz RAM and 64 bit Scientific Linux Release 6.3 operating system. For cloud environment, we have used Cloudsim. Another data center in a federated environment used same type of servers with high speed network connectivity.

Amazon is used as a paid or public cloud. This is the test bed to test the proposed system. Virtual machine has configuration of 1 CPU and 1024 MB RAM.

B. Test Cases

First test case is related with the importance of the automation system in a cloud environment. In order to test this, we have created number of jobs with execution of simple date, echo and sleep command. These jobs are executed on virtual machines. Figure 6 represents the jobs execution time with respect to virtual machines. As shown in Figure 6, we can conclude that as the number of virtual machines increases the total execution time required for these jobs reduces with the limitation of execution time and system throughput. Therefore, the automation of the system can help to increase the throughput of the system. In Figure 7 represents utilization of CPU depends upon number of virtual machine.

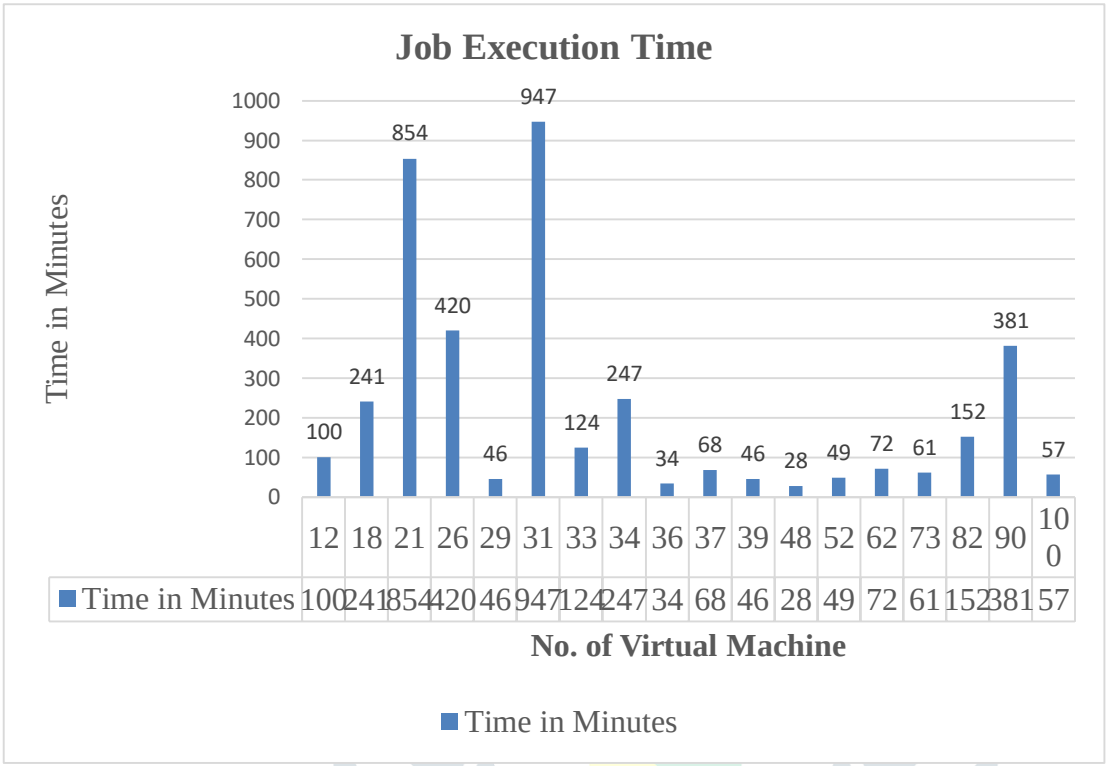

Figure 6. Job Execution Time

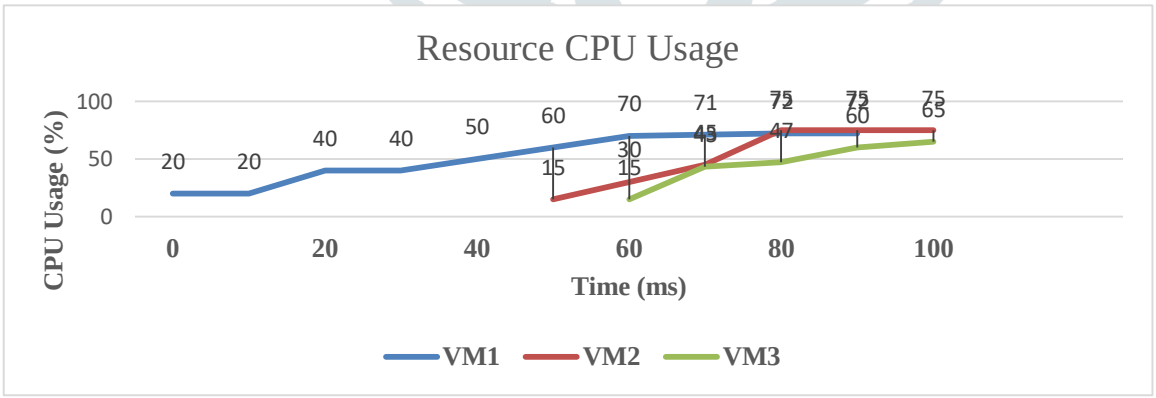

Figure 7. Utilization of CPU

VI. CONCLUSION

This paper has presented the priority-based virtual ma- chine allocation algorithm for a scientific federated environment. In order to manage the virtual cluster, this algorithm provides the creation and deletion of the virtual machine dynamically. This algorithm has capability to manage efficient utilization of resources in scientific federated environment. All test cases prove that our proposed algorithm can effectively handle the virtual infrastructure, virtual machine provisioning and power consumption of a data center.

In the future work, migration policy decides for transfer the virtual machine from one host machine to another host machine or

from one cloud to another cloud. Migration policy helps to take decision regarding server consolidation and to reduce the power consumption of resources. Virtual machine migration policy can apply to any virtual machine depending on the external or internal reasons. Internal rea- sons can be performance of the virtual machine. External reason can be server consolidation of underutilized virtual machines to reduce power consumption.

REFERENCE

[1] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al. "A view of cloud computing". Communications of the ACM 53, no. 4: 50-58, 2010.

[2] Neetesh Kumar et al.. "An Energy Aware Cost Effective Scheduling Framework for Heterogeneous Cluster System," ELSEVIER, Future Generation Computer Systems , 2017, 71: 73-88.

[3] CDF, http://www-cdf.fnal.gov/, May 2014.

[4] GSDC, http://en.kisti.re.kr/supercomputing/, May 2014.

[5] OpenNebula, http://opennebula.org/about:about, May 2014.

[6] KVM, http://www.linux-kvm.org/page/Main Page, May 2014.

[7] Sukrati Jain, Ashendra K. Saxena , "A survey of load balancing challenges in cloud environment," in IEEE System Modeling & Advancement in Research Trends (SMART), International Conference, 10.1109/SYSMART.2016.7894537, pp 234-238 , 25-27 Nov. 2016.

[8] Mamta Khanchi, Sanjay Tyagi, "An Efficient Algorithm for Load Balancing in Cloud Computing" International Journal of Engineering Sciences & Research Technology, June, 2016.

[9] Arun Pratap Singh, Prof. Pritesh Jain, Upendra Singh, "Survey of Load Balancing Algorithms in Cloud Computing", International Journal of Scientific Development and Research (IJSDR), Volume 2, Issue 9, September 2017, pp. 209-215..

[10] Neetesh Kumara et al. "A Green SLA Constrained Scheduling Algorithm for Parallel/Scientific Applications in Heterogeneous Cluster Systems," ELSEVIER, Sustainable Computing: Informatics and Systems 22: 2019, pp. 107-119.

[11] Singh, S., Abraham, T., Narayana lyengar, S.: A Review: Different Improvised Min-Min Load Balancing Algorithm in Cloud Computing Environment. Journal of Computer and Mathematical Sciences, (2016)

[12] Patel, R., Patel, S., Patel, D., Patel, T.: Improved GA Using Population Reduction Load Balancing in Cloud Computing. 2016 Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI) (2016)

[13] Vanithaa, M., Marikkannu, P.: Effective resource utilization in cloud environment through a dynamic well Organised load balancing algorithm for virtual machines. Computers and Electrical Engineering (2017)

[14] Violetta N. Volkova1, Liudmila V. Chernenkaya , "Load Balancing in Cloud Computing" , IEEE , 2018 , pp. 387-390.

[15] Pappu Singh Chouhan, Prof. Makrand Samvatsar, Upendra Singh, "Study of Resource Allocation Techniques in Cloud", International Journal of Scientific Development and Research (IJSDR), Volume 2, Issue 9, September 2017, pp. 236-240.

[16] Priya Gupta, Prof. Makrand Samvatsar, Upendra Singh "Dynamic Resource Allocation Scheme Using Priority Base Scheduling on Cloud Computing," IEEE International conference on Electronics, Communication and Aerospace Technology page(s) 1-5, 2017.

[17] Priya Gupta, Prof. Makrand Samvatsar, Upendra Singh "Review of Resource Allocation Techniques In Cloud Computing", International Journal of Scientific Development and Research (IJSDR), IJSDR | Volume 2, Issue 7 pp 241-245, July 2017.

[18] Kapil Thakkar, Roopesh Sharma, Upendra Singh, "Weight Balancing Techniques in Cloud Computing", International Journal of Scientific Development and Research (IJSDR) , Volume 2, Issue 7 , July 2017 , pp. 332-337.

[19] Pappu Singh Chouhan ,Makrand Samvatsar ,Upendra Singh , "Energetic Source allotment scheme for cloud computing using threshold-based" , International conference of Electronics, Communication and Aerospace Technology (ICECA) , IEEE , 2017 , pp. 1-8

[20] Arun Pratap Singh, Prof. Pritesh Jain , Upendra Singh , "Load Balancing in Cloud Computing Using Modified Optimize Response Time" , International Conference on Inventive Research in Computing Applications (ICIRCA 2018) , IEEE , 2018, pp. 1-6.

[21] Kapil Thakkar, Prof. Roopesh Sharma, Upendra Singh , "Load Balancing in Cloud Environment Using DAG and Honey Bee Algorithm" , International Journal of Scientific Development and Research (IJSDR) , Volume 2, Issue 7 , July 2017,