A Proactive Approach for Resource Provisioning in Cloud Computing

Ankita Jain, Arun Kumar Yadav, Brijesh Kumar Chaurasia

Abstract--- Cloud computing has become a preferred service supplier for information technology, on the users demands resources can be provided there over internet. On the internet workload are changing frequently but continuously changing pattern is still there. Currently for automatic scaling of resource management, low cost and improving resource utility in the cloud, workload prediction scheme has become a very bright packer. Recently, there are various approaches available for workload prediction which is based on the single model prediction approach. However because of the internet providing a large scale heterogeneity data over the cloud, it is very difficult to find out a satisfactory result by mean of a traditional model. We have proposed a proactive approach for resource allocation by analyzing large scale heterogeneity data in cloud. Our implementation shows a better result of resource prediction accuracy with low cost and less time consuming than previous approaches.

Index Terms--- Master Server, Slave Server, Workload, Resource utilization, Virtual Machine.

I. INTRODUCTION

In the previous decades, cloud computing has turned into the advance stage of services for giving benefit. Enormous organizations, for example, Amazon, Microsoft, Google and IBM have as of now offered Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) or Software-as-a Service (SaaS) arrangements in the cloud market [1]. Numerous specialist organizations pick cloud computing platform to decrease cost and vitality in light of its asset provisioning points of interest.

Not quite the same as the traditional computing paradigm, cloud computing has numerous prevalent qualities in which elasticity is the key one to diminish cost. Versatility more often than not alludes to dynamic provisioning assets in the cloud. Cloud computing stage can naturally scale asset to take care of the demand of different applications with the elasticity. Right now, more cloud specialist organizations, for example, AWS EC2 (AWS, 2016) and Google App Engine (Google, 2016) have given versatile administrations. By utilizing these administrations, they can give on-request computing force and capacity limits progressively. With a specific end goal to empower the scalability, it is important to build up a component to scale up or on the other hand down Virtual Machine (VM) naturally [2] in the cloud. As a critical list, scaling time for the most part contains start-up time to instate another VM, and is sufficiently long to be concerned. It will decrease the scaling time if the measures of resources are resolved while the VMs are conveyed and propelled early.

To accomplish the scalability said over, the prediction approaches assume an imperative part to decide the measure of resources. One of most difficult job is the manner by which to progress the prediction precision. An extensive variety of prediction demonstrating systems is being utilized for evaluating the normal client requests in the cloud. The prediction strategies can be classified into two sorts as per the qualities of prediction technique. One kind of prediction strategies utilizes the classical models, including various models of time series , (for example, ARIMA, second moving normal model (SMA), exponential moving strategy (EMA), autoregressive demonstrate (ARM), et al.), Neural Network (NN), SVM, Markov models, Bayesian models et cetera [3],[4],[5]. Aside from those methods, a number of works concentrated on recognizing extraordinary pattern of the workload [6]. All in all, a single prediction model is utilized by common prediction methods that implies, for a specific workload feature single prediction model are taken. Be that as it may, the heterogeneity of resource compose and the changeability of resource prerequisite by applications raise new difficulties to workload prediction in the cloud. In the view of use, workloads have different changing examples impacted by application write. For instance, workloads related with online administrations demonstrate some particular transient pat2 terns, for example, periodic, and busting [7] [8]. These examples normally rely upon the intrinsic qualities of uses and also on the workload force. As fundamental normal for workloads, and self-comparability [9],[10], have been looked into normally. During fetching of VM_i from the service provider, physically resources of the VM_i will [11] be consuming 1\4 amount of space at SS_i during execution.

Under-provisioning assets will cause service level objective (SLO) violation, which are frequently connected with significant money related punishments. Over-provisioning squanders resources that could be put to different employments. To maintain a strategic distance from the two issues, the measure of assets apportioned to applications ought to be balanced progressively, which brings two principle challenges: (1) choosing how much asset to distribute is non-minor since application resource needs frequently change with time and describing runtime application conduct is troublesome; (2) application resources requirement should be predicted ahead of time so the administration framework can alter resources allocation in front of the requirements[12].

Manuscript received February 01, 2019

Ankita Jain, M.Tech. Research Scholor, ITM University, Gwalior, Madhya Pradesh, India. (e-mail: ankitajain4727@gmail.com)

Arun Kumar Yadav, Deptt. of CSE, ITM University, Gwalior, Madhya Pradesh, India. (e-mail: arun26977@gmail.com)

Brijesh Kumar Chaurasia, Deptt. of CSE, ITM University, Gwalior, Madhya Pradesh, India. (e-mail: bkchaurasia.itm@gmail.com)

Besides, resource administration frameworks ought not to require earlier information about applications, for example, application conduct profiles, and running the resource administration framework itself (as well as predictive algorithms) ought not be exorbitant.

II. RELATED WORK

For resource management in cloud computing, the research of workload prediction and dynamic resource allocation is the most basic research challenges there. A relaxation tradeoff among the SLAs (e.g. response time, throughput etc.) and different constrains (e.g. price effectiveness and so forth.) desires to be accomplished. To gain the dynamic resource provisioning ,resource prediction schemes is the essential step .recently there are few techniques have been researched on the workload characteristic and workload prediction method with different angles. This segment gives an overview of a number of the related strategies.

For SAAS providers there are presenting the actualization of a cloud workload prediction module which is based on the ARIMA(Autoregressive Integrated Moving Average) model .there are prediction based ARIMA model are presented which is generated the accuracy of the future workload prediction[13]. In term of resource utilization and quality of service parameter they compare the impact of completed accuracy. The simulation result of the model shows an average accuracy of up to 91%. In this implementation there should be integrate a architecture against poor quality of service about the dynamic prediction approach. There should be plan to robust model for workload prediction also that will be predict height in resource utilization that is not find out with the ARIMS model.

For the satisfaction of forthcoming resource demand in this paper they presenting provisioning strategies by using the combination of neural network and linear regression .by the use of neural network and linear regression they developed a prediction-based resource measurement and resource allocation strategy [14]. Experimental results show that the proposed approach gives more adaptive useful resource control for programs hosted in the cloud environment, and vital mechanism to acquire on call for useful resource allocation inside the cloud. This prediction model is not enabling to implement to predict for wide range workload generator.

A prototype system has been implemented there that help to improve the quality of services of the cloud. There totally research efforts show about improving the quality of services in the cloud platform [15]. Mainly two problems such as generic cost sensitive prediction problem and heavy dynamic environment are considered to formulate here by purpose a heterogeneous measure to quantity the prediction error for find out the effectiveness evolution of the proposed solution they presented an integrated prototype system to enhance the QOS of cloud. This approach provide the less prediction accuracy they improve the accuracy by providing customer-oriented personalized prediction. Any such target can be fulfilled by leveraging the consumer profile information. The solution for capacity planning is a type of mid-time period prediction presented here. The irregularity of small granularity time series makes the prediction inherently tough.

Workload prediction approach and cloud architecture are presented here that is based on the linear regression model. An auto scaling mechanism proposed for scale the virtual resource in different asses levels in cloud service by the help of combines the real-time scaling and pre-scaling [16]. Auto scaling approach result demonstrate that this approach satisfy the user (SLA) service level agreement and as well as also keeping a low cost of scaling. To predict the workload they used a linear regression model and also implement an approach in both the real time scaling and the pre-scaling to scale the cloud services. in this paper there should be simulate to further evaluation and improved the auto-scaling approach with multimedia conference system and data sets from SNS system deployed in our cloud services. There also focus on the cost-aware-auto-scaling approach.

A workload prediction approach is implemented here which divide the workload according to the workload features and set apart different prediction model. the classification of workload are based on the 0-1 programming model in which they convert the workload classification into 0-1 programming problems[17].they presented a optimization algorithm and formulate an optimization problems to maximize prediction precision. This approach provided more accuracy by using real traces typical online services and also improves the uniformity of the prediction error. This approach cannot experiment with wide variety of workloads in the cloud services. There should intend to propose more prediction models for higher range workload pattern in the service cloud.

they presented a innovative clustering based resource estimation approach that make a cluster of the same group of task that have similar characteristics . The ancient workload information for duties in a cluster is used to estimate the sources wanted via new jobs based on the cluster(s) to which they belong [18]. We broaden a workload version based totally at the dataset that is then used to estimate the workload styles of several randomly selected jobs from the hint log. There should be focus on enhancing the clustering and the corresponding workload prediction methods and investigating the resource management schemes that can satisfactory make use of the prediction effects and also plan to explore the hierarchical clustering method which captures the records of nearby clusters that can be used to help improve the prediction scheme.

A PRESS scheme for cloud systems are presented here which is based on the resource allocation in the cloud services. Predictive elastic resource scaling (PRESS) scheme unobtrusively extracts fine-grained dynamic patterns in software resource needs and automatically adjust their resource allocations [19]. This approach leverages mild-weight sign processing and statistical gaining knowledge of algorithms to reap on-line predictions of dynamic software resource necessities.

This approach reduces the resource waste and service

level object violations by elastic resource scaling. They applied the PRESS approach on pinnacle of XEN in the NCSU virtual computing lab, the usage of RUBiS benchmarks pushed by using actual-life workload traces, and a resource-utilization trace of a Google software workload. This PRESS system should be implemented with wide variety of heterogeneous data patterns over the cloud computing.

For Iaas cloud there are presented an adaptive resource utilization prediction approach. By using prediction system there are proposing a real-time resources in the cloud services. Prediction system observe the real-time utilization of resources and also feed those values into different and several buffers which are categorized on the types of resources and period span size [20]. A results show a better performance as compare to another approach such as ARIMA for a data set that fails normality test. Moreover, other prediction strategies like deep studying based networks may be used, whilst noticing both the temporal and spatial complexities of the technique in assessment with ARIMA and AR-NN. Also hybrid strategies can be used depending on the size of education information, period for prediction, and type of prediction patterns.

The main aim of this research work is analyzing the cloud workloads and moreover evaluating the performances by two mainly used prediction schemes such as Markov modeling and Bayesian modeling with 7hour of Google cluster data [21]. An important final results of this research work is the categorization and characterization of the cloud workloads in an effort to assist main into the user demands for prediction parameter modeling. This paper defines the categorization and characterization metrics of the cloud workloads that are critical inside the technique of schooling and modeling the prediction techniques. two widely used prediction techniques are analyzed and their prediction efficiencies in estimating memory intensive and CPU in depth workloads are evaluated in Matlab environment. The experiments screen that the CPU extensive workloads show increased prediction error percent than the memory intensive workloads. The accuracy of the prediction strategies relies upon on the efficiency of the prediction modeling and the reliability of such techniques can be more advantageous by means of incorporating the weight dynamics and by refining the insignificant facts from the modeling parameters. There should be developing effective prediction technique which provides a novel prediction model with higher reliability and accuracy.

For resource provisioning and workflow scheduling in the infrastructure as a service(Iaas) cloud environment there are implementing an augmented shuffled frog leaping algorithm (ASFLA).the ratio of performance of the ASFLA has been compares with shuffled frog leaping algorithm (SFLA) and state of art PSO algorithm[22]. The efficacy of ASFLA has been assessed over a few well-known scientific workflows of assorted sizes using a custom java primarily based simulator. The simulation effects display a marked improvement inside the overall performance standards of accomplishing minimal execution cost and assembly the time schedule time limits. The experimental evaluation of ASLA, SFLA and PSO shows that the former outperforms the alternative algorithms in reducing the overall execution value of the considered workflows. In a heterogeneous environment on the cloud system, there includes evolution of the proposed approach.

A resource allocation model has been proposed here on the basis of dynamic parameters. Mostly traditional models are based on the static parameters such as CPU utilization, threshold, resource, and workload that provides lack of efficiency and less handy about the over-provisioning and under-provisioning situation. A dynamic resource allocation schemes compete that all over the issues [23]. The proposed model is carried out on CloudSim and experimental outcomes show the proposed model can enhance resource utilization and time. If they compared to standard distributed computing systems, cloud computing structures are more reliable, dynamic, and scalable. In current trend the assignment is managing the resources to preserve the scalability in dynamic surroundings. The need is to enhance the performance of cloud computing systems by means of provisioning and allocation of on-demand resources to reduce the time

A resource management framework for self-management of cloud resources called SCOOTER has been proposed. A useful resource management framework called scooter has been proposed and it has an capacity to manage assets robotically through residences of autonomic management, which can be self-recovery (discover and react to sudden faults), self-optimizing (maximize resource usage and strength efficiency and minimize execution cost, execution time, resource competition and SLA violation fee), self-configuring (capability to readjust sources) and self-protective (detection and safety of cyber-assaults) automatically with minimum human involvement [24]. Through a case study and experimental results show that the proposed scheme performs better than another previous autonomic resource management framework in term of various quality of service(Qos) parameters such as execution cost, energy consumption, execution time, SLA violation rate, fault detection rate, intrusion detection rate, resource utilization, resource contention, throughput and waiting time. SCOOTER efficiently schedules the provisioned cloud resources automatically for execution of heterogeneous workloads and maintains the SLA which fulfill the user satisfaction. In future, scooter can also be prolonged by way of growing pluggable scheduler, in which resource scheduling can be modified effortlessly based totally on the brand new requirements.

III. PROPOSED ARCHITECTURE FOR RESOURCE ALLOCATION

We presenting the architecture of proposed provisioning approach based on the workload data analyzing of large scale heterogeneity data over the cloud. In this architecture, the workload prediction management can automatically provide a quick resource according to each task request by solving the optimization problem.

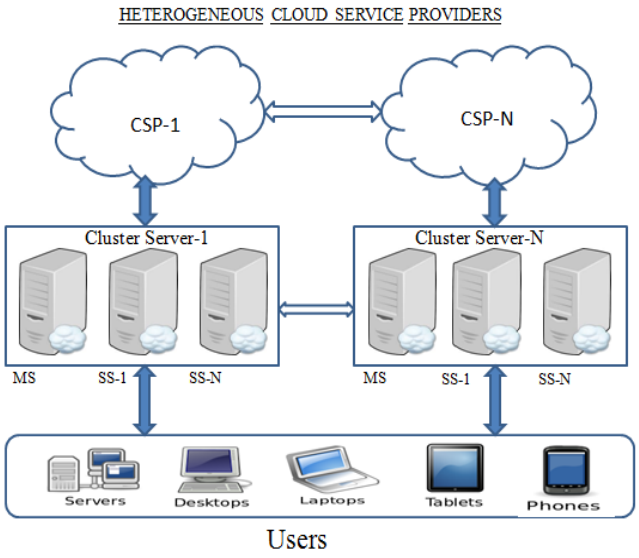

Fig. 1: Architecture of proactive resource provisioning

The predominant components of the proposed resource allocation architecture are depicted in fig. 1. This consists of components as users, master servers, slave servers, heterogeneous cloud service providers, cluster server, cloud provisioning, and load balancer also.

Components of proposed architecture

Users: For built a server, there heterogeneous physical resources with high computing capability are connected together in the cloud computing environment. These Server cluster is a high performance computing machine which are assign to each users by service provider. There is defining a third level architecture where all the users (mobile, desktop, static, and laptop) are exists in the third level of the architecture. Firstly user request to nearby cluster server where each virtual machine allocated to each slave server in cloud computing environment. Each virtual machine is allocated by CSP_i (cloud service providers) as per the user requirement and subscription. Each virtual machine exists in the slave server of clusters server which is allocated by the cloud service provider. First user will be request for the resources to nearby cluster server where master server will be provide the resource allocation in the VM_i as per the availability of resources under the slave server.

Master Server: under the cluster servers, master server is a higher performance computing machine which handles all the resource requests generated from the user's side. MS_i maintains the prelisting resources at the slave server and monitor how many resources like CPU, RAM and storage are utilized in the VM_i which is allocated to each SS_i. There is also a threshold value of resource utilization such as resources are utilizing 90% in a virtual machine and 30% in another virtual machine of slave servers and utility of the resource in each SS_i is monitor by the MS_i. if any new request occurs by the users then the master server will be check the availability of how many CPU, RAM, and Storage available in the VM_i , that types of records will be listed with reference of SS_i and this is called prelisting. Users forward the request to the nearer server that request will be go for MS_i and that will be analyzed the resources and will be forward this as per the users requirement (service). Master server handles the list of free and allocated resources of the VM_i under the SS_i and also updates both lists if any changes happened there and quickly provide the resource as per the users demand by analyze these lists. In case if a new request is generated by the users for the resources then master server will be check for the resource availability and will be logically collect the resource and allocates them to demanding users. In another case if any server will be fail and free resources are not available then master server will be request to nearby server for additional resources and will be send a request to cloud service provider also.

MSi also responsible for the load balancing of the resources in the CS_i. Load balancer is also a component of MS_i which is actually analyzed the how much total utilization of the resources, how much threshold value is there and on the basis of these values load balancer will be decides that, where a new VM_i should be execute on SS_i.

Slave server: cluster server includes both the master server and n number of slave servers. As per the user requirement CSP_i allocates the virtual machine to the each server. In the slave server, virtual machine can be assign anywhere as per the availability by cloud service provider. Virtual machines include the CPU, RAM and Storage resources mainly under the slave server which is created by the cloud service providers. The master server monitors the resource availability as well as the free and allocated resources in the slave server and also handles a prelist of these resources. Each VM_i has a unique identity which is assign to the slave servers in the cluster servers.

Heterogeneous CSP: CSP_i is actually responsible for provisioning the physical and any types of resources to the users in form of VM_i. As per the users requirement service providers creating a VM_i and also provide a unique ID to each virtual machine and pass the references to master server and MS_i decides that, in which SS_i , that VM_i will be execute .As well as heterogeneous CSP_i is maintaining one of the component known as resource provisioning. this component is actually responsible for receiving request, analyzing available resources and provisioning of resources, In case, when the demand of resources are not available at the service provider locally then the CSP_i will borrow the requested resources from the nearby CSP_i. There are following jobs performs by the CSP_i such as ,offering the physical resources, creating the virtual machines, providing a unique ID to each virtual machine to execute over the slave server. All these virtual machines at the slave server are controlled by the master server. As well as heterogeneous CSP_i also consider that if a demand requested resources is no available locally Then the requested resources will be borrow with the nearby service providers.

IV. PROPOSED ALGORITHM

The proposed algorithm, proactive approach for resource allocation in heterogeneous cloud computing environment is given below.

The main objective of proactive approach is to maximize the utilization of resources through direct allocation of unused pre-allocated resources to new requesting user and

give response to the user in less time by using direct allocation through SS_i and also increase the performance of the system.

Following parameters, we have used in proposed algorithm such as cloud service provider(CSP_i), cluster servers(CS_i), master servers(MS_i), slave server(SS_i), virtual machines(VM_i), resources(R_i) and maximum capacity(Max_Cap) etc. .The consolidation of parameters is given below in table.

Table I: List of parameters

Abbreviations	Description
CSP_i	Cloud service provider which allocates VM_i.
CS_i	No. of Cluster server that include both MS_i and SS_i.
MS_i	No. of Master server that maintain a list about SS_i.
SSi	No. of slave server which include no. of VM_i.
VM_i	No. of virtual machines which includes resources.
R_i	No. of resources like CPU, RAM and STORAGE (R1, R2 ….R3) respectively.
Max_Cap	Stand for maximum capacity .
Ref	Stand for reference
Avl	Stand for available
Req	Stand for request
Utl	Stand for utilization
Free $R_i(C)$	Stand for Free resources of VM_i.
Exe	Stand for execution
Free $R_i(Z)$	Stand for Free resources of SS_i
N_Req_i	Stand for new request by user site
SVM_i	Stand for sub VM_i

In our proposed methodology we consider a modify proactive approach for VM allocation to produce maximum utilization of resources. The algorithm-1 represents the proactive approach that requires MS_i list and SS_i list as input and provides allocation of VM_i on SS_i through CSP_i as output. The main objective of our algorithm is VM_i allocation on SS_i through CSP_i. Within this resource allocation is done in such a way that initially no VM_i is executing on SS_i then MS_i will be send request to CSP_i for VM_i allocation. CSP_i creates VM_i and send that reference to the MS_i; if SS_i is already empty then MS_i will select SS_i and pass the reference of VM_i. If SS_i is not empty then MS_i will check the list of Free $R_i(Z)$ on SS_i. If SS_i has at least 1\4 part of requested resources then MS_i will allocate the Ref (VM_i) to SS_i and also update the list of VM_i on SS_i and send ref of VM_i to requested user.

Algorithm 1: VM_i allocation on slave servers (SS_i) through CSP_i

Input: MS_i list and SS_i list Output: efficient VM_i allocation

1. Repeat for {i=0,1,2…….n}
2. For each SS_i in CS_i do
3. For each VM_i in SS_i do
4. For each SS_i list on MS_i do
5. If all SS_i=empty or resources (R_i) at VM_i (SS_i)< N_Req_i then
   i. MS_i sends N_Req to CSP_i
   ii. check for R_i(N_Req) on CSP_i for i=0,1,2…….n
   [End if statement]
6. CSP_i creates the VM_i and send back Ref(VM_i) to MS_i
   a. If all SS_i=empty then
      i. MS_i allocate the Ref(VM_i) to SS_i
      ii. update the list of VM_i on SS_i
      iii. Forward the Ref(VM_i) to user.
      else
   b. MS_i check Free $R_i(Z)$ on SS_i
   c. If Free $R_i(Z)(SS_i)$>1\4(R_i)(Ref(VM_i)) then
      i. MS_i allocate the Ref (VM_i) to SS_i
      ii. update the list of VM_i on SS_i
      iii. Forward the Ref (VM_i) to user.
7. Return

It's consumed more time if the procedure of VM_i allocation is happened through CSP_i.Our proposed algorithm-2 is showing the VM_i allocation through MS_i by creating a SVM_i on SS_i. There if number of N_Req_i is requesting for resources then MS_i will check availability on Free $R_i(C)$ and that should be "Free $R_i(C)$>= N_Req_i" , if condition is satisfied then MS_i will creates a SVM_i and send the reference Ref(SVM_i) to CSP_i otherwise MS_i will check Free R_i(C) at VM_i and Free $R_i(C)$ at VM_{i+1} for remaining resources of N_Req_i if these Free $R_i(C)$ at (VM_i + VM_{i+1})>= N_Req_i then MS_i will creates SVM_i after combining the Free $R_i(C)$ at VM_i and VM_{i+1}. Hence there have checked the Free R_i of VM_i (Free $R_i(C)$) and have been satisfied for rest of N_Req_i by creating a SVM_i but for allocation of SVM_i there should be requires 1\4(R_i) of SVM_i at Free $R_i(Z)$. if Free $R_i(Z)$>= 1\4(R_i)(SVM_i) then MS_i can allocates SVM_i to SS_i and send the Ref(SVM_i) to CSP_i and Update the list of SVM_i on SS_i and Forward that Ref(SVM_i) to requested users also.

So our both algorithms are efficient enough to allocate the resources, minimizing the response time by direct provisioning from server side and maximize the utilization of resources.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Algorithm 2: VM_i allocation on slave servers (SS_i) through MS_i

Input: MS_i list and SS_i list Output: efficient VM_i allocation

1. Repeat for {i=0,1,2…….n}
2. For each SS_i in CS_i do
3. For each VM_i in SS_i do
4. For each SS_i list on MS_i do
5. For each N_Req_i do
6. MS_i check Free $Ri(C)$ for N_Req_i exe
   - A. If Free $R_i(C)$>= N_Req_i then
     - a. MS_i will creates SVM_i
     - b. Forward the $Ref(SVM_i)$ to CSP_i.

     else
     - a. MS_i will check Free R_i(C) at VM_i and Free $R_i(C)$ at VM_{i+1} for remaining resources of N_Req_i.
     - b. If (Free $R_i(C)$ at VM_i)+ (Free $R_i(C)$ at VM_{i+1})>= N_Req_i then
       - i) MS_i will create SVM_i after combining the Free $R_i(C)$ at VM_i and VM_{i+1}.
       - ii) MS_i check Free $R_i(Z)$ on (SS_i) for SVM_i allocation
       - iii) If Free $R_i(Z)(SS_i)$>=1\4$(R_i)(SVM_i)$ then
         - MS_i allocates SVM_i to SS_i
         - MS_i send the $Ref(SVM_i)$ to CSP_i
         - Update the list of SVM_i on SS_i
         - Forward the $Ref(SVM_i)$ to user.

       else

   Execute Algorithm-1.

   [End of if Statement]

   [End of if Statement]

   [End of if Statement]

8. Return.

V. SIMULATION SETUP

The proactive algorithm for resource allocation in cloud computing has been proposed in this section. The main objective of our approach is to track the status of available resources, allocated resources in the server and this approach also contributing to maintain the load balancing over the slave server.

Each cluster server includes a master server and many more slave server. Each slave server serves a virtual machine which is allocated by the CSP_i. In the slave server each virtual machine has a unique ID. The main roll of master server is to listing all over resource states of each VM_i at slave server. Each virtual machine contains the following resources like CPU, RAM and STORAGE such as R_1, R_2 and R_3 respectively.

CSP_i allocates one and more virtual machine at a slave server with unique ID and slave server also have own resource capacity to executes VM_i resources. Each virtual machine takes a fix amount of space for the logically allocation at the slave server. Physically VM_i are placed in the data center and references of VM_i passed on master server by CSP_i during demanding request by users.

A condition is arising there, if a new user requesting for resources to MS_i then how wiil be allocation of the resources happened? For that initially master server will be contact to CSP_i about requested VM_i and CSP_i will be send reference of VM_i and then MS_i will decides that where should be allocate that VM_i on the SS_i. We know that MS_i listing many types of information about resources of VM_i like how much VM_i executing and where are executing in cluster as well as that is also listing about how much resources are executing of each VM_i and where are executing and how much resources are free and utilized in VM_i. in this condition if requests is arising from users side for resources then MS_i will check list of free resources on VM_i and will be check for the availability of resources if resources will be free then master server will be allocates resources from pre executed VM_i to requested user. Each VM_i have its own capacity for resources allocation, free resources of VM_i we can calculate by subtract total utilized resources from the total resources of VM_i at SS_i. That is representing in a equation form such as eq. 1 shows the free resources at virtual machines.

$$C(R_i) = A(R_i) - B(R_i) \qquad (1)$$

Where C denotes the free resources of the VM_i in respect of R_i that is resources and A stand for total resources available of the VM_i and B stand for utilized resources of the VM_i at SS_i. If i=1 then we calculating free resources of resource type-1 that is CPU and if we are taking i=2 then we will be calculating free resources of resource type-2 that is RAM and if we are taking i=3 then we will be calculating free resources of resource type-3 that is STORAGE. There are given below a tabular representation to find out free resources of VM_i at the slave server.

Table II: Resources at virtual machines (VM_i)

		Total R_i(A)			Utl R_i(B)			Free R_i(C)		
SS_i	VM_i	R1 CPU	R2 RAM (GB)	R3 Storage (GB)	R1 CPU	R2 RAM (GB)	R3 Storage (GB)	R1 CPU	R2 RAM(GB)	R3 Storage (GB)
SS1	VM1	4	2	1024	2	1	512	2	1	512
	VM2	5	3	1024	2	1	256	3	2	768
SS2	VM1	5	3	2048	2	2	1024	3	1	1024
	VM2	5	4	2048	3	2	512	2	2	1536
SS3	VM1	6	4	2048	2	2	256	4	2	1792
	VM2	5	4	2048	2	2	1024	3	2	1024

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

The free resources of VM_i at SS_i is calculated by table-2. In which first column showing n-Number of server in the cluster and second column presenting n-Number of VM_i which is allocated by the CSP_i at SS_i. Next one column shows the total resource capacity of VM_i which are denoted by "A" and another next presenting the total utilized resources of the VM_i which is denoted by "B". With the help of these two values (A&B) here calculating the value of Free R_i(C) that is a total free resource of the VM_i on ss_i.

Suppose, a slave server contains n number of VM_i such as SS_1 (VM_1, VM_2 ...VM_n) and each VM_i include R_1, R_2 and R_3 resources. When we fetch VM_i from the service provider then physically, resources of VM_i will be consuming 1\4 (R_i) amount of space at SS_i during execution. For example SS_i consumes n-Number of VM_i such as (VM_1, VM_2 ...VM_n) and which includes resources like CPU, RAM, and STORAGE (R_1, R_2 ...R_3) respectively. VM_i consumes suppose R_1=1, R_2=1, and R_3=256 physically on CSP_i then during execution of VM_i at the SS_i that will be consume R_1=0.25, R_2=O.25 and R_3=64 physical amount of SS_i, that is calculated only for resources of single VM_i at SS_i. There is also a tabular representation below for calculating total physical amount of space consuming by VM_i during execution on SS_i.

Table III: resource consumptions on SS to execute VM_i

VMs on SSs	SS_1			SS_2			SS_3		
	R_1 CPU	R_2 RAM (GB)	R_3 Storage (GB)	R_1 CPU	R_2 RAM (GB)	R_3 Storage(GB)	R_1 CPU	R_2 RAM(GB)	R_3 Storage (GB)
VM_1	4	2	1024	5	3	2048	6	4	2048
VM_2	5	3	1024	5	4	2048	5	4	2048
Total requested physical space by VM_i(B)	9	5	2048	10	7	4096	11	8	4096
Resource consumed on SS(1/4of VM resources) (C)(C=B\4)	2.25	1.25	512	2.5	1.75	1024	2.75	2	1024

Total resource consumption on SS_i by VM_i during execution is calculated on above table. There each VM_i contains our own resources which are located on the CSP_i physically and CSP_i allocate the reference of VM_i to MS_i and MS_i decides where that VM_i should be executing on SS_i. SS_i also has its own resources and resource capacity for VM_i execution. When VM_i is executing then they will be consume only 1/4 of VM_i resources on SS_i.

In reference of R_1, R_2 and R_3, VM_i shows own actual capacity of resources for allocation at SS_i in Table3. By that we are calculating the total of requested physical space that is represented by (B) in 5th row of above Table. Finally presenting calculation about consumed resources (1/4 amount of VM_i resources) on SS_i and that is denoted by (C) in the last row of above Table. There are formulating an equation about to calculate resource consumption on SS_i during execution of VM_i. Eq.2 shows the resource consumptions on SS_i to execute VM_i.

$$C(R_i) = B(R_i)/4 \qquad (2)$$

Now consider about free resources of SS_i after execution of VM_i. Above mentioned that SS_i have its own resources and also physical resource capacity and each SS_i reserve 10% capacity for own purpose or own calculation . remaining 90% capacity of SS_i provided for VM_i execution. Suppose max capacity of SS_i (SS_1, SS_2 …SS_n) is 10, 10 and 55200 in reference of resources, like R_1, R_2 …R_3. Then SS_i will be provides 90% of his its actual capacity and that will be 9, 9 and 46080 remaining part for each SS_i resources. After consuming resources of SS_i by VM_i resources, there are need to evaluate all free resources of SS_i. We have calculated all total consumed resources on SS_i by 1\4 of VM_i resources in above table-2. So with the help of those evaluated values of (C) by table-3, we can calculate total free resources of SS_i. There are following representation for calculates total free resources on SS_i.

Table IV: Resource utilization at SS_i

SS_i	Max_Cap (X)			Utl R_i (Y)			Free R_i(Z) (Z=X-Y)		
	R1 CPU	R2 RAM (GB)	R3 Storage (GB)	R1 CPU	R2 RAM (GB)	R3 Storage(GB)	R1 CPU	R2 RAM(GB)	R3 Storage (GB)
SS_1	9	9	51200	2.25	1.25	512	6.75	7.75	50688
SS_2	9	9	51200	2.5	1.75	1024	6.75	7.25	50176
SS_3	9	9	51200	2.75	2	1024	6.25	7	50176

Maximum capacity of SS_i in reference of R_i are calculated in table-4, which is denoted by (X) and also showing total utilized resources of SS_i which are consumed by VM_i resources during execution. These values of total consumed resource on SS_i we have already calculateed on table-3that is (C=B\4) in reference of SS_i. With the help of these values, we are evaluating the total free resources of SS_i and that is mentioned in table-4. Eq. 3 shows the total free resources of SSi.

$$Z(R_i) = X(R_i) - Y(R_i) \qquad (3)$$

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Now we know that how much resources are free and how much resources being utilized of VM_i on SS_i as well as we also know about the free physical resources of SS_i that mean in the basis of these information we can search quickly for free resources and can allocate easily to requested users. If new users demanding for VM_i then MS_i will be consider that which server have availability of free resources of VM_i, if free resources are available then MS_i logically creates SVM_i directly on SS_i and allocate these SVM_i to new requested users. There are following table representing how many users requesting for SVM_i.

Table V: New user demands for resources

Requested VM_i	New Requested Resources		
	R_1 CPU	R_2 RAM (GB)	R_3 Storage (GB)
SVM_1	3	2	0
SVM_2	4	2	1024
SVM_3	3	2	0
SVM_4	4	3	1024

In this manner each sub VM_i will be have a unique ID and master server will send information about new SVM_i to CSP_i. VM_i also creates a new list of logically created sub VM_i and store the information about reference of VM_i on SS_i through which new SVM^i is created. Suppose there are following new users which is requesting for VM_i allocation. For providing the physical resources master server will be initiate and look at the free resources for satisfying the request of resources for the new VM_i. Initially for a new user, MS_i will create SVM_1 and for next demanding user, MS_i will be creating another SVM_2 and so on. If we are unable to satisfying the N_Req_i by allotment on a particular VM_i of SS_i then we will be check for availability of free resources on another VM_i of another SS_i , if there will be availability then combines both VM_s and creates a SVM_i and after that allocates to requested users.

There are following tabular representation that is showing, how an MS_i satisfies the user demands by create a SVM_i on SS_i.

Table VI: Creation and allocation of new SVMs by MS_i

New SVM_i	Free Resources of VM_i used by new SVM_i	SS_i
SVM_1	VM_2	SS_1
SVM_2	VM_1	SS_3
SVM_3	VM_2	SS_3
SVM_4	VM_1	SS_1
	VM_2	SS_2

The value of new requested VM_i by user site are considering through table-6 which have 4 new VM_i demands. For providing the physical resource master server will initiate and will be checking for free resources for fulfill the request of resources of new VM_i. In above table-6 MS_i used remaininng resources of VM_2 on SS_1 to create a new SVM_1 and that SVM_1 is allocated to requested $user_1$. Clearly we can see that SVM_2 and SVM_3 is executing on SS_3 and for rest of SVM_4 there are combined the resources of VM_1 at SS_1 and VM_2 at SS_2.

VI. RESULTS AND COMPARATIVE ANALYSIS

Simulation Tool (CloudSim)

We used cloudSim toolkit version 3.0.1 simulator to implement our proactive algorithm to execute tasks along with Window 7 OS, core i3 2.40 GHz processor and NetBeans IDE 7.2.1. It provides a computing environment for demand resource provisioning and their management. It can simulate in a heterogeneous environment, dynamic workload with application. The simulation result obtained and then comparing with scooter algorithm on the basis of the some parameters like execution time, execution cost, waiting time, and resource utilization.

There we consider a scenario have different task with varying size on different number of VM_i. We have created these VM_s on the basis of 2500 and 5000 workload in cloud. In this scenario we have generated number of cloudlets that is allocated to different VM_i on host and evaluated result are comparing with some of specified parameters through with another one algorithm parameters.

Simulation Result

after implementing our proposed architecture and algorithm on cloudSim the result obtained by using various workload that is maintained in above paragraph. The result obtained in heterogeneous environment by proactive algorithm is compared on the basis of some parameters, execution time, execution cost, waiting time and resource utilization with existed algorithm like scooter. existed algorithm "scooter" implemented with various parameter but we have selected some parameters of that and implement with 4 selected parameter and compared obtained result on the basis of these parameters. It clearly shows that our results are slightly better than the existing algorithm.

We have taken number of cloudlets in table-7 are executing in various numbers of virtual machines on host through server and shown the comparison between our proposed algorithm and existing algorithm in the basis of 4 parameters. Initially number of CloudLets executed on different host then total time taken by cloudlet in executing individually from server is 560.67, 531.2, 514.06, 497.9, 475.35, 1130.15 second. It is slightly less time as compared to the time taken by scooter i.e. 561.66, 531.21, 515.03, 499.97, 476.16, 1103.10 seconds respectively to executes the coudLets from server for 6 VM_s over three hosts. In second, number of cloudlets are executing on various host then total cost taken by cloudlets in executing individually from server is 132.53$, 175.35$, 190.32$, 196.45$, 207.35$, 468.9 4$ with respect to individual cloudlets. It is slightly less cost as compared to the cost taken by scooter i.e. 132.69$, 176.39$, 191.35$, 198.36$, 209.46$, 468.95$ respectively to execute the cloudlet from server. In third, number of cloudlets are executing on different various host then during execution total waiting time of cloudlet individually from server is 7.35, 6.10, 5.35, 4.92, 4.36 , 6.53 seconds.

It is slightly less time as compared to the time taken by scooter i.e. 7.526, 6.13, 5.265, 4.96, 4.42, 6.62 respectively

during execution of cloudlet from server. In fourth, number of cloudlets is executing on various host then total utilization of the tasks taken by cloudlets in executing individually from server is 50, 32.77, 52.22, 45, 31.92, 46.66 percent. It provides slightly less resource utilization as compared to the existed algorithm scooter i.e. 82.466, 84.569, 85.941, 86.792, 88.41, 89.68. with the comparison of both algorithm we can say that there are not a big difference, the result are slightly better than existed algorithm result and the resource utilization of existed algoorithm have more as comparison to our algorithm-1.

Table VII: Result from the execution of algorithm-1

	Cloudlet ID	execution time(sec)	execution cost(C$)	waiting time(sec)	resource utilization (%)
Proactive	CL0	560.67	132.53	7.35	50
	CL1	531.2	175.35	6.10	32.77
	CL2	514.06	190.32	5.35	52.22
	CL3	497.9	196.45	4.92	45
	CL4	475.35	207.35	4.36	31.94
	CL5	1103.10	468.9	6.53	46.66

We have taken number of cloudlets are executing on various number of virtual machine on host which is allocated virtually by virtual machine. In this way cloudlet is requested for allocation then VM_i checks for free resources then it provided quick allocation virtually on host. Host layer taken a less amount of time during allocation as compare to the server layer and also increase the optimization of resources. on the basis of algorithm-1 our implemented result is slightly better with some parameters but provides a less resource utilization as compared to the existed algorithm. But the result of algorithm-2 in table-8 are given a better performance and resource utilization of resources as compare to existed algorithm result.

Table VIII: Result from the execution of algorithm-2

	Cloudlet ID	execution time(sec)	execution cost(C$)	waiting time(sec)	Resource utilization (%)
Proactive	CL6	595.16	60.81	5.51	80.55
	CL7	561.66	94.13	4.81	87.50
	CL8	1288.63	263.25	9.04	82.65
	CL9	1350.7	227.97	12.81	83.34
	CL10	1330.23	328.37	12.94	91.67

We are focused on maximum resource utilization of resources as compare to scooter algorithm in table-8. number of cloudlets is executing on various host then maximum utilization of the VM_i at host taken by cloudlets in executing individually in server is 80.55, 87.50, 82.65, 83.34, 91.67 percent. It provides slightly better result of resource utilization as compared to the existed algorithm scooter i.e. 80.554, 81.854, 82.65, 80.33, 81.56.

The comparative analysis of our proactive approach to the scooter approach is shown below on fig-2 and fig-3 on the basis of algorithm-1 and algorithm-2.

Within this we shown our resource allocation of proactive approach which is not given a better result with respect to resource utilization as compared to existed algorithm but with some parameters our proactive approach is generated a slightly better result.

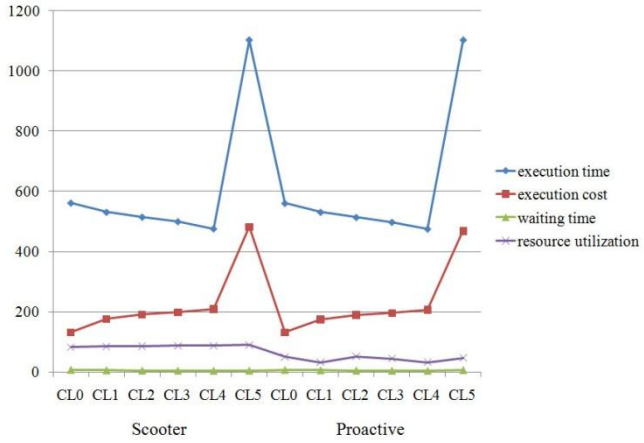

Fig. 2: Comparative Analysis on Algorithm-1

Our comparative analysis of algorithm-2 showing a better result of resource utilization in fig-3 as compare to existed scooter algorithm.

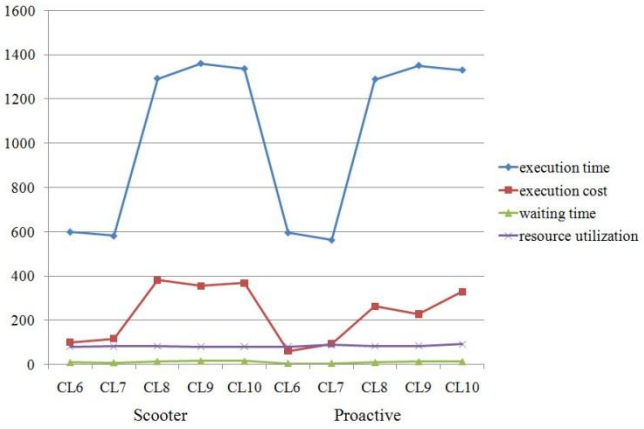

Fig. 3: Result of Comparative Analysis of Algorithm-2

Resource utilization is always being a huge concern for any of allocation policy. Clearly we can see that as per the comparative analysis of fig-1 and fig-2, our algorithm is better on to these parameters compare to scooter algorithm.

VII. CONCLUSION AND FUTURE DIRECTION

In this paper, a proactive approach for resource allocation has been proposed. Various optimization schemes have many approaches for resource allocation in heterogeneous cloud environment for quick resource provisioning and maximum utilization of resources. Due to ineffectiveness of the proposed resource allocation policies, there was less optimization and utilization of resources which decrease the overall performance of cloud data centers. Proposed proactive approach in this paper, allocate the resource effectively without violating SLA and QoS.

Our approach provides direct allocation of VM_i from server site using already allocated VMs resources, instead of allocations through CSP_i. Results shown in analysis section clearly state that proposed approach with direct allocation of VM through server site gives the better optimization of resources, maximum utilization of resources and better

performance in data centers.

The proposed proactive approach have the limitation where, if the user is requesting for their remaining resources then Master Server (MS_i) at cluster will be additional time to search and provide free resources, which will increase the waiting time and second case, if MS_i is unable to get the requested amount of free resources at cluster then the request will be forwarded to CSP_i to provide the resources. Definitely, the waiting time will be increasing much more in second case. So, we will try to upgrade our proposed approach in future to resolve these limitations.

REFERENCES

1. Fox, Armando, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, and Ion Stoica. "Above the clouds: A Berkeley view of cloud computing." Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28, no. 13 (2009): 2009.
2. Roy, Nilabja, Abhishek Dubey, and Aniruddha Gokhale. "Efficient autoscaling in the cloud using predictive models for workload forecasting." In Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp. 500-507. IEEE, 2011.
3. Chen, Zhijia, Yuanchang Zhu, Yanqiang Di, and Shaochong Feng. "Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network." Computational intelligence and neuroscience 2015 (2015): 17.
4. Sapankevych, Nicholas I., and Ravi Sankar. "Time series prediction using support vector machines: a survey." IEEE Computational Intelligence Magazine 4, no. 2 (2009).
5. Bankole, Akindele A., and Samuel A. Ajila. "Cloud client prediction models for cloud resource provisioning in a multitier web application environment." In Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on, pp. 156-161. IEEE, 2013.
6. Caron, Eddy, Frédéric Desprez, and Adrian Muresan. "Forecasting for Cloud computing on-demand resources based on pattern matching." PhD diss., INRIA, 2010.
7. Ghorbani, Mahboobeh, Yanzhi Wang, Yuankun Xue, Massoud Pedram, and Paul Bogdan. "Prediction and control of bursty cloud workloads: a fractal framework." In Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis, p. 12. ACM, 2014.
8. Calzarossa, Maria Carla, Luisa Massari, and Daniele Tessera. "Workload characterization: A survey revisited." ACM Computing Surveys (CSUR) 48, no. 3 (2016): 48.
9. Yin, Jianwei, Xingjian Lu, Xinkui Zhao, Hanwei Chen, and Xue Liu. "BURSE: A bursty and self-similar workload generator for cloud computing." IEEE Transactions on Parallel and Distributed Systems 26, no. 3 (2015): 668-680.
10. Eldin, Ahmed Ali, Ali Rezaie, Amardeep Mehta, Stanislav Razroev, Sara Sjostedt de Luna, Oleg Seleznjev, Johan Tordsson, and Erik Elmroth. "How will your workload look like in 6 years? analyzing wikimedia's workload." In Cloud Engineering (IC2E), 2014 IEEE International Conference on, pp. 349-354. IEEE, 2014.
11. Wang, Kai, Minghong Lin, Florin Ciucu, Adam Wierman, and Chuang Lin. "Characterizing the impact of the workload on the value of dynamic resizing in data centers." Performance Evaluation 85 (2015): 1-18.
12. Calzarossa, Maria Carla, and Daniele Tessera. "Modeling and predicting temporal patterns of web content changes." Journal of Network and Computer Applications 56 (2015): 115-123.
13. Calheiros, Rodrigo N., Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. "Workload prediction using ARIMA model and its impact on cloud applications' QoS." IEEE Transactions on Cloud Computing 3, no. 4 (2015): 449-458.
14. Islam, Sadeka, Jacky Keung, Kevin Lee, and Anna Liu. "Empirical prediction models for adaptive resource provisioning in the cloud." Future Generation Computer Systems 28, no. 1 (2012): 155-162.
15. Jiang, Yexi, Chang-Shing Perng, Tao Li, and Rong N. Chang. "Cloud analytics for capacity planning and instant vm provisioning." IEEE Transactions on Network and Service Management 10, no. 3 (2013): 312-325.
16. Yang, Jingqi, Chuanchang Liu, Yanlei Shang, Bo Cheng, Zexiang Mao, Chunhong Liu, Lisha Niu, and Junliang Chen. "A cost-aware auto-scaling approach using the workload prediction in service clouds." Information Systems Frontiers16, no. 1 (2014): 7-18.
17. Liu, Chunhong, Yanlei Shang, Li Duan, Shiping Chen, Chuanchang Liu, and Junliang Chen. "Optimizing workload category for adaptive workload prediction in service clouds." In International Conference on Service-Oriented Computing, pp. 87-104. Springer, Berlin, Heidelberg, 2015.
18. Patel, Jemishkumar, Vasu Jindal, I-Ling Yen, Farokh Bastani, Jie Xu, and Peter Garraghan. "Workload estimation for improving resource management decisions in the cloud." In Autonomous Decentralized Systems (ISADS), 2015 IEEE Twelfth International Symposium on, pp. 25-32. IEEE, 2015.
19. Gong, Zhenhuan, Xiaohui Gu, and John Wilkes. "Press: Predictive elastic resource scaling for cloud systems." In Network and Service Management (CNSM), 2010 International Conference on, pp. 9-16. Ieee, 2010.
20. Zia Ullah, Qazi, Shahzad Hassan, and Gul Muhammad Khan. "Adaptive Resource Utilization Prediction System for Infrastructure as a Service Cloud." Computational intelligence and neuroscience 2017 (2017).
21. Panneerselvam, J., Liu, L., Antonopoulos, N., Bo, Y., 2014. Workload analysis for the scope of user demand prediction model evaluations in cloud environments. In: Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on. IEEE, pp. 883–889.
22. Kaur, Parmeet, and Shikha Mehta. "Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm." Journal of Parallel and Distributed Computing 101 (2017): 41-50.
23. Seth, Sonam, and Nipur Singh. "Dynamic Threshold-Based Dynamic Resource Allocation Using Multiple VM Migration for Cloud Computing Systems." In International Conference on Information, Communication and Computing Technology, pp. 106-116. Springer, Singapore, 2017.
24. Gill, Sukhpal Singh, and Rajkumar Buyya. "Resource Provisioning Based Scheduling Framework for Execution of Heterogeneous and Clustered Workloads in Clouds: from Fundamental to Autonomic Offering." Journal of Grid Computing (2018): 1-33