IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-16, May 2019 E-ISSN: 2347-2693

A Study of Virtual Machine Resource Scheduling Algorithms in Cloud Computing Environment

Divya Tantri[1*], Karan George[2], S. Rajarajeswari[3]

[1,2,3]Department of Computer Science, Ramaiah Institute of Technology, Karnataka, India

Corresponding Author: divya.tantri@gmail.com

DOI: https://doi.org/10.26438/ijcse/v7si16.1833 | Available online at: www.ijcseonline.org

Abstract— Cloud computing is a *fast-growing* technology that is being used extensively in the market today. Most cloud service providers, provide clients with one of two things either *resources* or *services* as entitled to the client by the SLA. In *resource allocation* a user may want to gain access to multiple resources at the same time, thus we require a method for the *optimal* provision of resources. With the sudden massive growth in cloud computing there is no dearth of techniques and algorithms that can be used for resource allocation. In this paper we will study the various resource scheduling algorithms being utilized in the *virtualization* industry and compare them on common factors to discern the comparatively optimal resource scheduling algorithm.

Keywords—Resources,services,optimal,virtualization,resource allocation

I. INTRODUCTION

Cloud service providers in today's world provide a large number of services which can be availed to the customers benefit. Virtualization is one of the most beneficiary cloud application field. Virtualization today refers to the creation and the use of resources such as storage, server, desktop and operating systems which run over an abstracted physical hardware. Virtualization has played a major role in the of modern cloud computing technology, usually in cloud computing, users share the data amongst themselves that is stored in the cloud., but with the advent of virtualization users no longer share data over the cloud but also share the infrastructure. Virtualizations major benefit lies in standardizing the process of version updation. Having a large number of machines with the standard versions installed and updating them to a newer version will be a hassle for any company with many machines. This problem is easily overcome by using virtualization as the resources are no longer maintained by the user but by the third-party cloud service providers who will ensure system version updates take place seamlessly on all machines. The term resource in this paper refers to the use of the virtual machines allocated by the service provider.

This paper will consist of meticulous investigation of the currently existing algorithms, followed by a small study depicting the use of these algorithms in a cloud computing environment and finally a comparative study based on common comparison factors

II. CATEGORICAL DIVISION OF THE CLOUD SCHEDULING ALGORITHMS

The existence of various cloud scheduling algorithms in the present cloud computing environment has warranted a categorical division in order to reduce similar algorithms into a category and make the comparative study through while eliminating an algorithm as the optimal one. Resource scheduling algorithms are needed when clients request multiple resources at the same time. In order to prevent resource starvation and ensure allocation we use these algorithms. They mainly aim at implementing an efficient form of load balancing in order to improve the overall resource utilization. Resources that are allocated improperly cause a huge resource drain on the cloud service providers who will have to pay additional amounts of money to maintain these resources. The categorical division broadly classifies the algorithms into 3 main categories

Category A: Dynamic Scheduling Algorithms

As the name suggest Dynamic Scheduling Algorithms involve allocating the resources not based on any predefined plan but allocating the resources dynamically and on the run. This prevents most resource blocking that is encountered when a plan causes a particular resource to be pre-allocated to a user thus preventing optimal resource utilization

Category B: Agent based scheduling Algorithms
Agent based scheduling Algorithms involve the use of an external third-party agent to allocate the resources. The agent will be responsible to monitor the resources present and allocate resources according to certain predefined criteria that will have to be met before the resource is allocated. The task will not always fall into the hands of a single third-party agent but might involve the use of multiple agents in forming a decision about the allocation.

Category C: Cost Optimization based scheduling Algorithms These scheduling algorithms enable cost-effective elasticity by scaling at resource level of each virtual machine. Server can apply optimal resource allocation techniques on user requests, to gradually improve the bandwidth and processing abilities. On analyzing the deadlines for every task, we choose services with the lowest execution cost and time.

III. DYNAMIC SCHEDULING ALGORITHMS

A. MCDA using PROMETHEE Method

Multi criteria decision analysis based on the preference ranking organization method for enrichment evaluations. This method is mostly preferred when the resource architecture is scattered or distributed. Resource management is done by independent nodes. These autonomous nodes perform resource management and configurations by analyzing multiple criteria. The algorithm simply redistributed the resources if it was not being used by any other virtual machine. The initial algorithm was implemented in C and results were observed in terms of physical machine implementations and CPU allocation. The figure below represents the life cycle of a task using the MCDA algorithm.

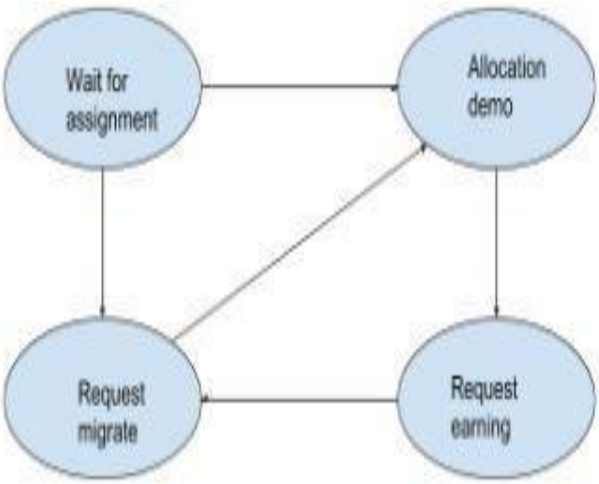

Figure 1.0: Life cycle of a task

B. Linear Scheduling for Tasks and Resources (LSTR)

LSTR follows a very simple linear allocation of resources to the various tasks by utilization of a queue technique. The algorithm utilizes two separate queues that maintain tasks in them according to a certain threshold value. The threshold value is usually calculated by summing up all the needed resources for a particular task. The tasks are then split among the two arrays as tasks that come above and below the threshold value respectively. This is then followed by linearly scheduling the tasks and resources for completion in each individual array starting with the array that contains elements (tasks) that fall below the threshold value. This technique is used to maximize the quality of service.

Pseudocode for underlying LSTR algorithm

1. Create array tempA[] and tempB[] to store resource requirements from resource array R[]
2. Initialize tempA[] with R[i] such that R[i] has resource length less than threshold
3. Initialize tempB[] with R[i] such that R[i] has resource length greater than threshold
4. Sort the resources from tempB[] and place into sorted list

5. Sort the resources from tempA[] and place into sorted list
6. Submit the resources from the sorted list to the respective Virtual machines

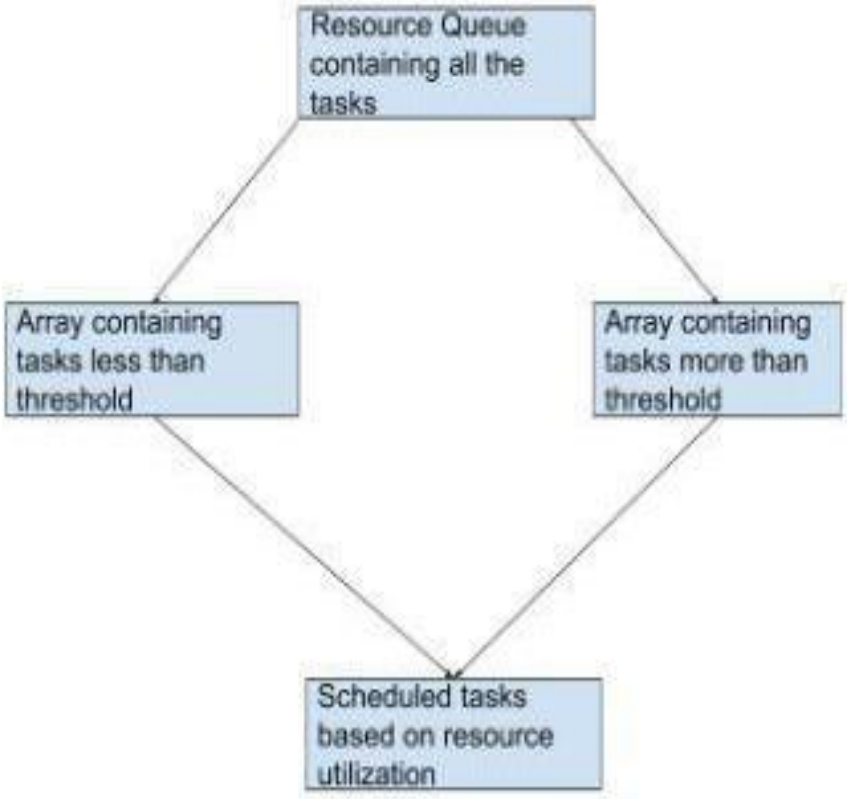

Figure 1.1: Scheduling in LSTR

C. Profmin VM max/min available space

In this section we present two separate algorithms that work in have similar workings except in the opposite sense. Profmin VM max available space algorithm works to allocate tasks to virtual machines as follows. Upon arrival of a task the algorithm first looks to see if there are any VMs available, if there are no VMs available the task is simple allocated to the virtual machine with the largest available free storage or compute space. This allocation however has a drawback similar to the first fit algorithm used in cache scheduling. By directly allocating the largest free space to the incoming task we are losing out on the capability to accommodate possible larger tasks in this space that may arrive in the future.

The Profmin VM min available space works in the opposite sense that is, upon arrival of a task the task is allocated to the virtual machine that has the least available space but can still accommodate the task. This negates the previously mentioned drawback but has a drawback of its own. We allocate compute resources here in a miserly manner and if the task requires additional compute resource it cannot be computed in that minimal space.

Pseudocode for Min-min algorithm is given below 1. For all the submitted tasks (ti) in Metatask (MT)
2. For all the resources (R) in metatask (MT)
3. Compute completion time j CTij= ETij + r
4. End of step 2 loop. j
5. End of step 1 loop.
//Phase 2: Assigning task ti
6. For each task in MT, find the task t with minimum completion time to the resource having minimum completion time. i
7. Assign t with minimum completion time and that resource on which it is calculated i to resource Rj
Remove task t that has m minimum completion time i

8. Update resource R from MT j ready time of r
9. Update completion time of all unmapped tasks in MT j
10. Repeat step 6-10, until all the tasks in metatask (MT) have been mapped
11. End of step 6 loop.

Pseudocode for Min-Max algorithm is given below

1. For all the submitted tasks (ti) in metatask (MT)
2. For all the resources (R)
3. Compute completion time j CTij= ETij + r
4. End of step 2 loop. j
5. End of step 1 loop.

//Phase 2: Assigning task ti

6. For each task in MT, find the task t with maximum completion time to the resource which gives minimum completion time. i
7. Assign t with maximum completion time and thatresource on which it is calculated i to resource Rj
8. Remove task ti from MT. that has m minimum completion time
9. Update resource Rj ready time (rj)
10. Update completion time of all unmapped tasks in (MT)
11. Repeat step 6-10, until all the tasks in metatask (MT) have been mapped
12. End of step 6 loop.

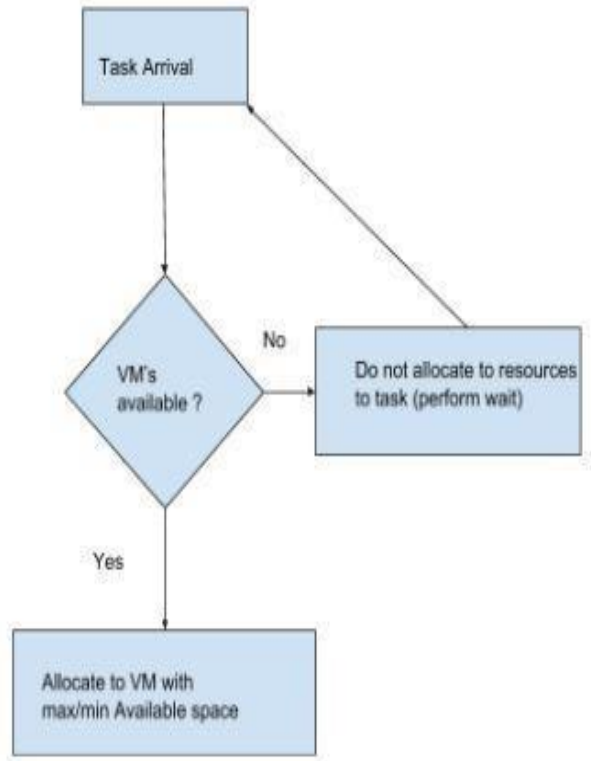

Figure 1.2: Life cycle of Profmin max/min available space

D. Dynamic Resource Allocation Using VM

This algorithm consists of several parts including a controller that is centralized and a node manager that is local and used to control smaller number of machines when compared to the controller. The algorithm first evaluates the capabilities and configurations of the physical machines available and sends this data to the central controller. When a VM has to be initialized a scheduler evaluates the resource needs of the virtual machine including the storage and compute needs. This data is obtained from the local node manager. If the resource utilization crosses a particular threshold value, the VM's are moved around using information from the LNM to reduce the load and enforce appropriate load balancing.

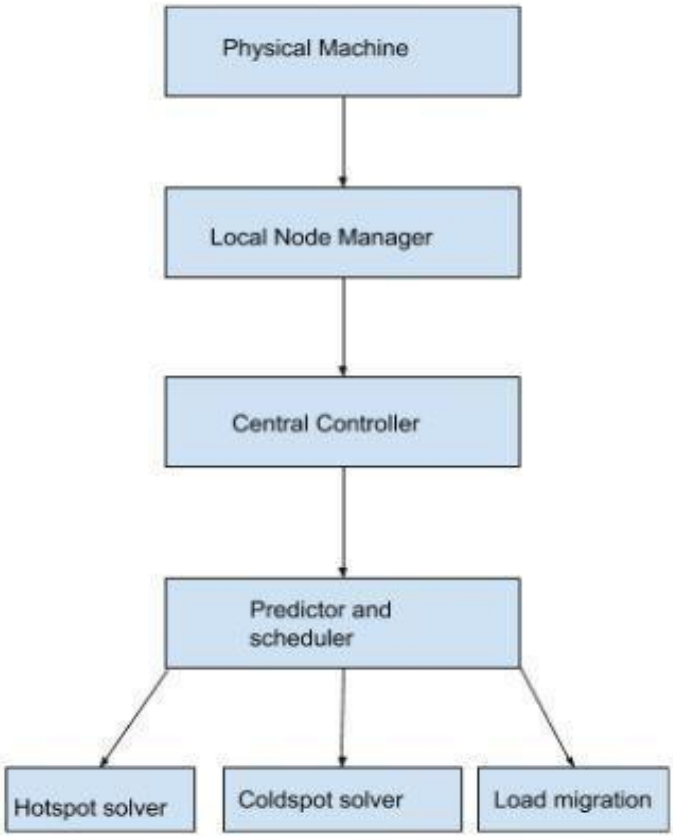

Figure 1.3: DRA-VM life cycle

E. Round Robin Based Resource Selection

The algorithm as the name suggests utilizes a round robin mechanism for resource selection a scheduling. The request for the virtual machine resource allocation begins at the end user who uses cloud service abstractions to transfer high level resource details to the resource manager. Upon receiving these requests, the resource manager allocates a weight based on burst time and puts the request into a queue. This is followed by the round robin execution of all the tasks in the queue that assumes a weighted format which gives more time to higher weighted tasks in comparison to the other tasks. Round robin is mainly utilized to enforce load balancing and load migration.

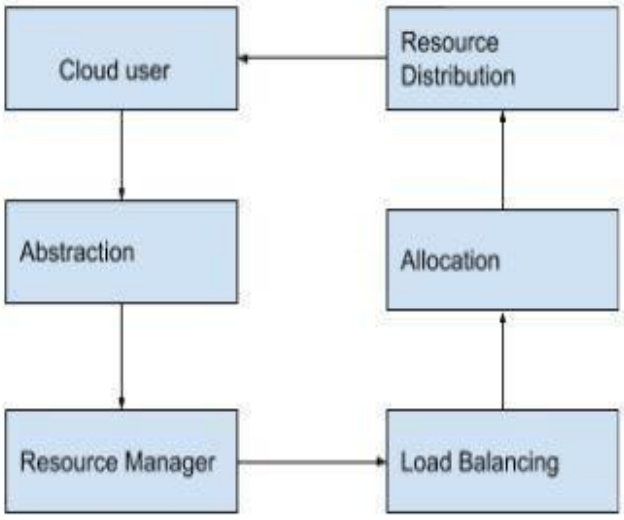

Figure 1.4: Round Robin resource allocation framework

IV. AGENT BASED SCHEDULING ALGORITHMS

A. Negotiation Strategy for Buyer and Seller

A widely used strategy in the cloud computing environment today that is employed even for amazon's EC2 instances it consists of interactions between 2 entities as the name suggests which are the buyers and the seller's respectively. The procedure is similar to that of a typical negotiation where the buyer will try to negotiate a good offer to propose to the seller. If the offer is accepted by the seller then a contract is drafted. This mechanism has been automated nowadays and involves agents on both the sides performing a negotiation for the contract price and other such details.

Figure 2.0: FSM for the negotiation

B. Agent Based Scheme Architecture

An Agent based technique that is primarily used in the wireless resource platforms. It is used to optimize the QOS factors and to provide optimal resource scheduling. The architecture is similar to previously described agent-based algorithms where the user agent collects the requirements for virtualization including the compute and storage virtualization. This is followed by an agent-based server that gathers the information from various service providers and selects the optimal one based on the request made by the user agent. The application service provider that is selected by the agent then provides the requested resources.

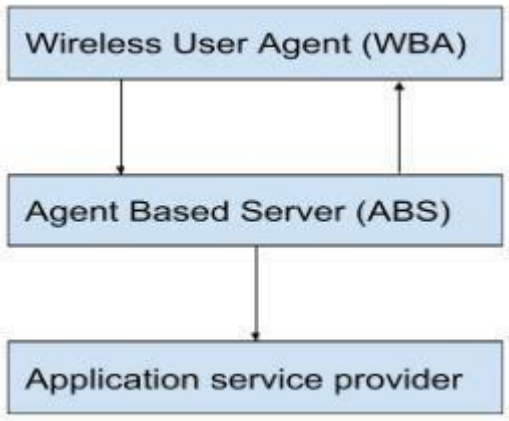

Figure 2.2: Agent Based scheme for wireless cloud

C. Multi Agent System Architecture

The multi-agent system features (dynamic, flexibility, autonomy, and learning) are features a Cloud Computing System needs for the self-management of its resources.

There is a Consumer Agent (CA) that notes down a consumer's requirements. A Broker Agent (BA) acts as a mediator

between Consumer Agent and Service Provider Agent (SPA) and sends the estimates to the consumer for agreement, once agreed it forwards it to the SPA.SPA is composed of multiple RA's. Resource Agent accepts request from SPA, manages web service by providing requirements to the end user.

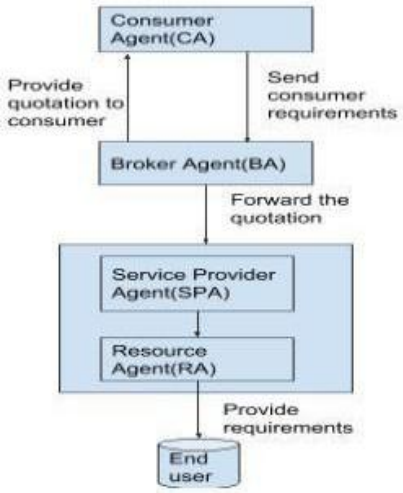

Figure 2.4: Multi-Agent System Architecture

D. Agent based Resource Allocation Model (ARAM) A Resource Allocation Model is like a mechanism designed to ensure that the requirements of an application are taken into account correctly by the provider's infrastructure. This model consists of a local cluster server and a master cluster server.

E.

The local cluster server sends resource information status to master cluster server, which in turn checks with the grid information server to make a new database containing information on the status and cost of resources. These requirements arrive at a Resource Broker. If demands of these requirements cannot be fulfilled by the broker, it moves to another broker. The allocation information is sent to the master cluster server, and then subsequently to the local cluster server.

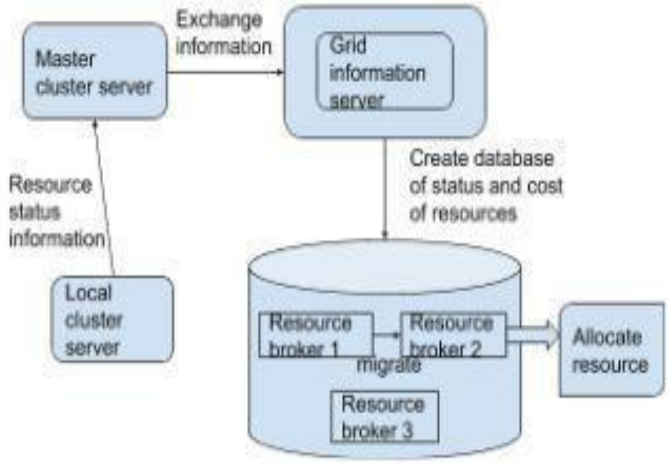

Figure:2.5.*ARAM System model*

F. Adaptive Resource Allocation Model

As the customers and the data centers are located across areas, when a customer requests for a service, his request is allocated to the appropriate data center based on parameters like execution response time and fast allocation time. We propose an adaptive resource allocation model that allocates the customer's job to a data center. Finding an appropriate data center might be a challenge, hence we adopt two methods:

a) the distance (network delay) between a customer and data centers, and 2) the workload of each data center.

This promises us faster and apt allocation of customers' requests.

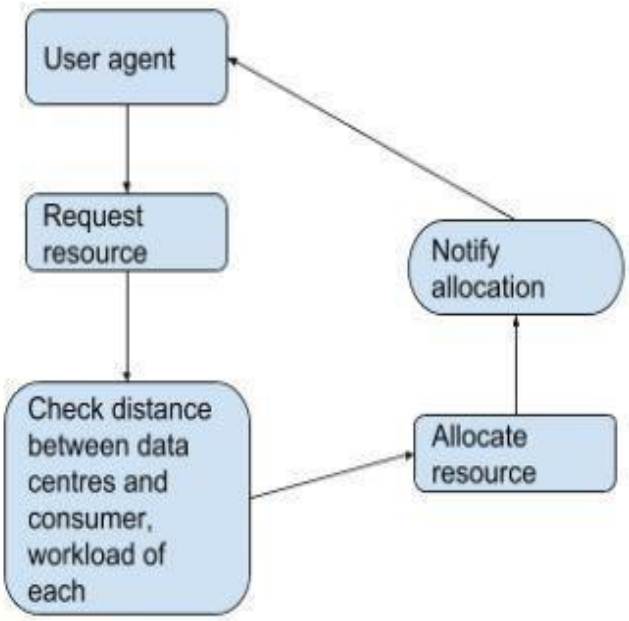

Figure:2.6. *Adaptive Resource Allocation model using network delay*

G. Market Based Model

In this model, there isn't much scalability. Buyers and service providers communicate through trade exchange procedures. Service providers use k-pricing and buyers place their bids. In the system, the winners of the bidding round are determined to efficiently allocate resources on the basis of job urgency based on execution time deadline.

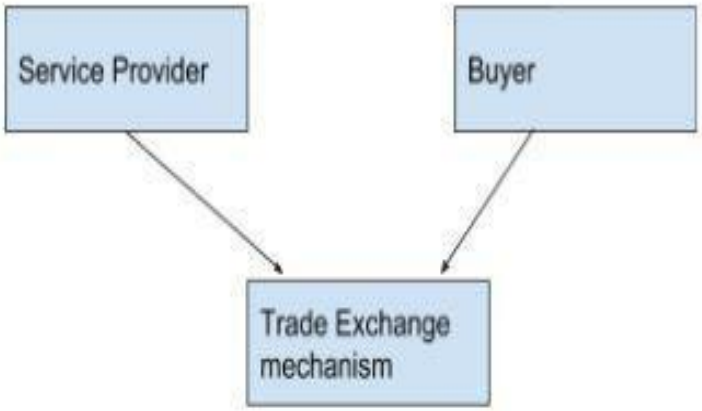

Figure:2.7. *Bidding mechanism of market-based model*

V. COST OPTIMIZATION BASED SCHEDULING ALGORITHMS

A. Lightweight Scaling Algorithm

Lightweight Scaling algorithm provides elasticity by scaling at resource level of virtual machine by configuring them. It offers automatic scaling up or down resource allocation at run time which leads to the load balancer to redistribute the resources upon every incoming request. The most practical approach is managing elasticity based on allocation and deallocation of the virtual machine instances to the application.

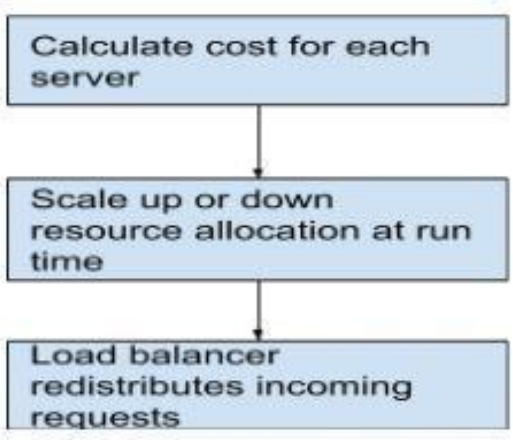

Figure 3.1: Lightweight Scaling steps

B. Optimal Resource Allocation Technique (ORAT) Based on the proposed architecture, the incoming requests were optimally allocated resources by the server. This gradually improves bandwidth and resource utilization at every step. This algorithm can vastly reduce the request loss possibility and therefore, decrease the total resource obligatory, compared to any predictable allocation method. Optimal server consolidation depends on resource provisioning that meets workloads needs. Slow runs of a system can be avoided by utilization of resources in the cloud. This also helps in load balancing.

C.

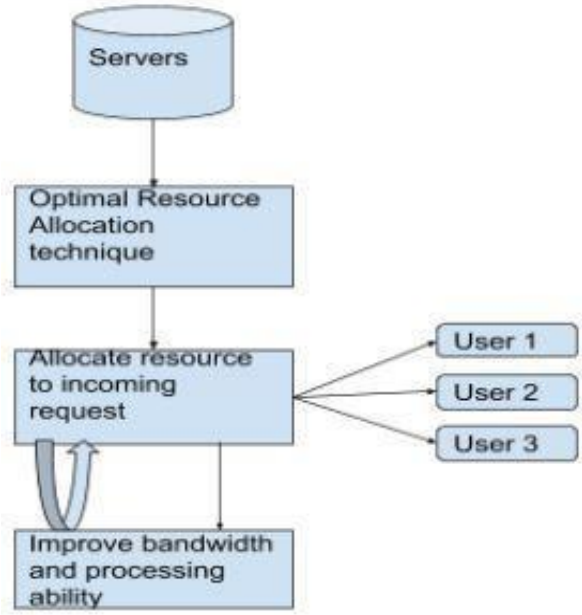

Figure 3.2: Optimal Resource Allocation Technique cycle

D. Compromised Time Scheduling Algorithm

This algorithm takes into consideration cost of execution and execution time. The pre-step is to schedule any incomplete tasks. The deadline for every task is calculated and the execution time and cost of a service is estimated. Based on the lowest cost and execution time, a task is assigned to a resource.

This algorithm tries to accommodate instance-intensive cost- constrained workflows by compromising execution time and cost with user enabled input as they come, after considering cloud computing characteristics.

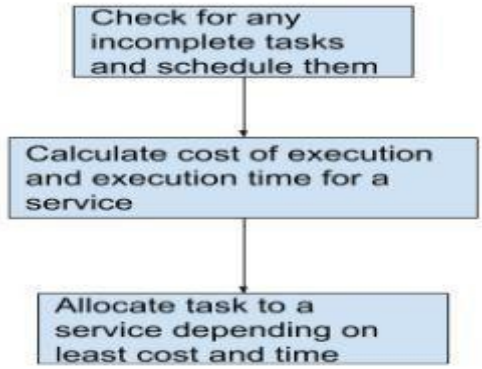

Figure 3.3: CTC workflow

VI. COMPARATIVE ANALYSIS

Since most of the above algorithms are deployed using diverse parameters the comparative study will be based on certain common factors including resource utilization, cost load balancing.

Resource utilization: An optimal resource scheduling algorithm should make use of resources given to it in an optimal manner

***Virtual Machine Cost*:** Per user cost of VM should be minimized by using proper sharing strategy

Load Balancing: Balancing load among available VM's reduces the chance of deadlocks and increases system throughput

VII. ANALYSIS OF CATEGORY I ALGORITHMS

MCDA using PROMETHEE method

Resource Utilization: Due the use of multiple criteria decision analysis the algorithm takes into account all the factors involved in optimization only the allocation and utilization of resources. It does this by making sure the outcome of each decision maximizes number of iterations and number of physical machine resources being used.

Virtual Machine Cost: As the VM's are allocated purely on the basis of maximizing resource utilization it does not take into account cost of each physical machine being used hence VM cost is high

Load Balancing: MCDA does not apply weighted preference for any of the tasks to allocate the resources it does not implement any form of load balancing.

LSTR

Resource Utilization: LSTR is a linear scheduling algorithm and as such makes sure that the tasks that require the most resources according to time of execution are executed first thus allowing for optimal resource utilization. The linear algorithm first executes the task that utilizes resources greater than a threshold and thus allows high resource utilization

Virtual Machine Cost: Yet again being a linear algorithm the focus of LSTR is on maximizing the physical resource usage according to time required for a task. As such it doesn't take into account VM cost.

Load Balancing: The algorithm does not implement any form of load balancing except scheduling the tasks in a particular linear order for execution based on a threshold value.

Profmin VM max/min available space

Resource Utilization: The algorithm is characterized by allocation of space onto physical machines for optimal resource utilization using caching algorithms. Resource utilization is relatively optimal as larger tasks are allocated to machines with more space thus boosting utilization

VM cost: The algorithm makes maximum use of available space and scans all the VM's looking for available space to allocate task. As such VM's with optimal cost are the ones with just enough space to accommodate the task. These are the VM's selected and thus cost is kept to an optimal value

Load Balancing: The algorithm maximizes space utilization on each and every one of the virtual machines but does not implement any methodology for load balancing

Dynamic Resource Allocation Using VM

Resource Utilization: Since the local node manager collects enough information from all the physical resources locally attached to it and relays it to a central system, the central system has sufficient information to optimize resource utilization by optimizing factors such as CPU load and utilization time.

VM cost: Since the algorithm uses a central agent to collect resource data, it primarily focuses on allocation of these resources to appropriate virtual machines to maximize utilization and as such does not account for whether the VM being allocated is properly shared or not

Load Balancing: The presence of the Hotspot solver and the Coldspot solver to migrate the loads, the algorithm has a provision for appropriate load balancing. The hotspot solver causes load migration when heavy loads are on the same system while the coldspot solver does the exact opposite. Hence implementing sufficient load balancing.

Round Robin Based Resource Selection

Resource Utilization: The round robin algorithm executes the tasks in a circular fashion after they are initially arranged according to the resource utilization of each task. Hence the provides a sub optimal resource utilization . It does this by improving parameters such as response time and number of services being used by each virtual machine.

VM Cost: The absence of a central database creates a lack of centralized data of all the VM's the cost cannot be optimized using round robin algorithm.

Load Balancing: The initial task performed by the algorithm is an allocation of weights to the individual tasks . Tasks that are highly weighted are usually the ones with more load and hence and executed more often providing load balancing.

VIII. ANALYSIS OF CATEGORY II ALGORITHMS

Negotiation Buyer and Seller

Resource Utilization: Negotiation between automated agents will cause the resources to be allocated from the cloud service providers side based on client requirements stated in an SLA. As such the seller's agent knows the resource requirements beforehand and optimizes the utilization of each resource.

VM cost: Most sellers such AWS make sure the user has an optimal experience and in doing so always acquires a profit for the resources provided. Hence VM cost is not optimal

Load Balancing: The negotiations between the buyer and seller do not implement implicit load balancing unless specifically mentioned in the requirements as stated in the service level agreement.

Agent Based Scheme Architecture

Resource Utilization: Since it is primarily used in wireless platforms the methodology uses a central agent that focuses on factors such as providing optimal resource scheduling but doesn't focus on utilization of the resources. Hence resource utilization is sub-optimal.

VM cost: he absence of a central database creates a lack of centralized data of all the VM's cannot be properly shared among all the tasks and therefore VM cost is not optimal.

Load Balancing: The entire purpose of an agent server is to obtain the needs for individual users as an abstract and perform resource scheduling while distributing the load and providing better quality of service. Hence the algorithm provides load balancing.

Multi Agent System Architecture
Resource Allocation: This algorithm self manages resources by noting down the consumers requests by being dynamic, flexibility, autonomy, and learning. The SPA sends estimates to the CA and only if the consumer accepts, the allocation occurs successfully.
VM Cost: The cost for having a consumer agent, a broker agent as a mediator and a service provider agent is high and not cost efficient.

Load Balancing: Load Balancing doesn't occur because the service provider sends out estimates to consumer agent for agreement and therefore reallocation.

Agent Based Resource Allocation Model
Resource Allocation: The requirements of applications are correctly made note of and as the requirements arrive, the database is checked with respect to every resource broker. If one resource broker cannot satisfy the requirement, it migrates to the next one for successful allocation

VM Cost: VM cost is low because the resource brokers allow migration in case they can't satisfy a requirement of an application

Load Balancing: Load balancing isn't efficient because of migration and not re distribution of tasks.

Adaptive Resource allocation model
Resource utilization: Here resource utilization occurs because the consumer requests are systematically distributed to different data centers and make sure resources are utilized efficiently.

VM Cost: The distance between the consumer and a data center can play a factor in increasing the cost if they are far apart and a closer data center isn't available for resource allocation of the consumer's job.

Load Balancing: The consumers job sometimes gets congested at nearer data center's rather than farther ones, hence load balancing doesn't occur

Market Based Model
Resource utilization: In market-based model, bidding mechanism occurs. The job with shortest time deadline occurs first and is calculated as the winner of the k pricing mechanism offered by the service providers and the others are not.

VM Cost: There isn't much scalability in this algorithm with respect to cost because only the winner of the bidding process is allocated resources and the others have to keep waiting. This doesn't provide an optimal cost solution.

Load Balancing: Load Balancing occurs because there is no congestion of workload on the servers and each winner is allocated a resource systematically. This provides an efficient load balancing technique for the servers.

IX. ANALYSIS OF CATEGORY III ALGORITHMS

Lightweight scheduling
Resource utilization: This algorithm provides elasticity at resource level by scaling up or down resources as the requests arrive. This fails to provide efficient resource utilization.

VM Cost: This is a cost-efficient algorithm because it calculates the cost for every server and hence re- allocates requests by scaling them up or down as the requests arrive.

Load Balancing: Load balancing occurs efficiently because once the cost of servers is calculated and the requests arrive, the resource allocation is scaled up or down and asks the load balancer to redistribute the resources upon every incoming request.

Optimal Resource Allocation Technique
Resource Utilization: As the name suggests this is an optimal resource allocation technique which optimally allocates the resource as the request arrives and at every step it increases the bandwidth and processing capabilities

VM Cost: The cost to increase bandwidth at every step and its processing abilities along with allocating an optimal resource allocation method to every request as it arrives is a lot.

Load Balancing: Load balancer doesn't re allocate resources but assigns each resource to a request optimally with respect to resource utilization instead.

Cost Compromised Technique

Resource Utilization: In cost compromised technique, the resource utilization doesn't occur efficiently because this algorithm seeks to improve cost and execution time.

VM Cost: The execution cost is calculated along with the execution time and the task with lowest execution time and cost is assigned a resource. This algorithm considers cost constrained workflows.

Load Balancing: This algorithm aims to be cost efficient and instance intensive, hence the load balancer assigns tasks with least cost an execution time to a resource.

Table 1

Resource Scheduling Algorithm	Algorithm Parameters	Resource Utilization	VM cost	Load Balancing	Category
LSTR	Time, Resource Utilization	✓	X	X	I
Profminvmmax Avaispace, ProfminvmMin Avaispace	Response time, Service Initiation, Number of Initiation	✓	✓	X	I
Round Robin Based Algorithm	Response Time, No. of services, CPU Usage	✓	X	✓	I
MCDA using PROMETHEE Method	No. of iterations, No. of Physical Machines	✓	X	X	I
DRM-VM	CPU Load, Time, Number of VM's	✓	X	✓	I
Negotiation Strategy of buyer & seller	Success rate of buyer, No. of resource to acquire	✓	X	X	II
Agent Based Scheme Architecture	No. of Users, No. of successful request	X	X	✓	II
Multi Agent System Architecture	Probability of Failure, No. of successful completion	✓	X	X	II
ARAM	Resource utilization, No. of JA's	✓	✓	X	II
Agent – based Adaptive Resource Allocation	Number of Consumer, Success Rate, Average Allocation Time	✓	X	X	II
Market Based Model	Trading Between buyer and Seller	X	X	✓	II
Lightweight Scaling	Number of Server, Bandwidth, Time,	X	✓	✓	III
ORAT	Resource Utilization, Data Transfer Rate, Number of Requests, Request Loss Probability	✓	X	✓	III
CTC	Execution Cost, Execution Time	X	✓	✓	III

X. PROPOSED IMPROVED ALGORITHM

As clearly reviewed by the comparative analysis algorithms that focus on resource utilization often fail to implement load balancing. The system we intend to propose will be a slightly more optimal version of the LSTR linear algorithm. The fact that it is a linear algorithm prevents it from implementing effective load balancing. We propose an improved version of LSTR in the following section.
The algorithm proposed will be combination of LSTR and round robin method.

Pseudocode for proposed improved algorithm

1. Create array tempA[] and tempB[] to store resource requirements from resource array R[]
2. Initialize tempA[] with R[i] such that R[i] has resource length less than threshold
3. Initialize tempB[] with R[i] such that R[i] has resource length greater than threshold
4. Sort the resources from tempB[] and place into sorted list
5. Sort the resources from tempA[] and place into sorted list
6. Allocate weights to the elements in the sorted list based on other algorithm parameters.
7. Pass the sorted and weighted list through a round robin implementation to achieve load balancing

The proposed algorithm will further improve the initial LSTR algorithm. Linear algorithms are often overlooked for optimization due to their simplicity. In the proposed method LSTR though being a simple method will perform comparatively better with lesser compute resources in comparison to other heavier agent and cost-based optimization algorithms.

A second proposal is an optimization to the Profminvmmax Avaispace, ProfminvmMin Avaispace algorithm. The algorithms function by fitting the VM onto the physical device with the largest or smallest possible fit respectively. This results in sub-optimal performance as once the VM is allocated to the physical machine the space is consumed entirely and cannot be used to fit a larger more optimal process that could provide more resource utilization. This is very similar to the problem initially faced when caching was developed as a solution to boost memory speed. We can apply a similar solution to this problem using the best fit algorithm to allocate space on the VM.

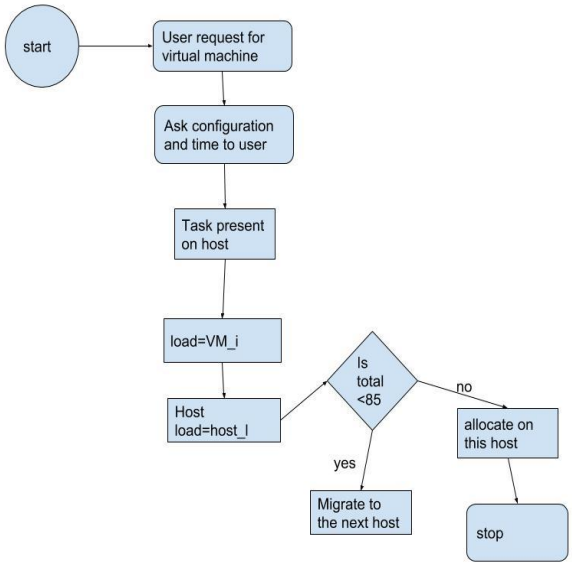

Figure 4.0: Proposed Algorithmic Workflow

Best Fit : This policy allocates the process to the host in which the process gets best fit and every time checks from the first host.

Step 1: User request for virtual machine.
Step 2: Cloud provider ask about configuration required by the consumer.
Step 3: Calculate the load of VM that we are going to allocate: Vm_l.
Step 4: Calculate the load of each host: host_l
Step 5: Calculate total load of each host Total_l = Vm_l + host_l
Step 6: Find Lowest Total Load
Step 7: If Total_l = <85 % Yes: Allocate Vm on this host and stop No: Find next lowest load.

Example:
Best fit algorithm will allocate VM to data centers for fulfilling the requirement of user

RESOURCE REQUIREMENTS FOR VMs

Vi	Vi Rt(i)
4	0.49
8	0.48
3	0.47
2	0.43
0.5	0.35
10	0.34
1	0.15
7	0.15
6	0.13
9	0.07

ALLOCATION OF VMs

server consolidation time	server 1	server 2	server 3	server 4
t-1	v1,v9	v6,v8	v5,v10	v2,3
t-1				v4,7
t				v4(m)
t		v8(m)		v8
t		v3		v3(m)
t	v2(M)	v2		
t	v5(b)	v5(b)		
t		v10		
t	v1(m)	v1		
t		v7(b)		
t			v6(N)	
t	v9(m)		v9	

m-profmin algorithm used
N-new server space created after
saturation b-best fit algorithm used
M-profmax algorithm used

XI. CONCLUSION

On comparing various cloud resource scheduling algorithms based on their nature and point of focus, we can see that there is no algorithm which satisfies all three parameters taken into consideration for the analysis and hence there is always room for improvement.

We proposed optimization technique for two algorithms and have future plans to implement the same. We need to be able to satisfy diverse needs of the user to allocate resources. The different types of agent-based resource allocation techniques seem more reliable than the others. It can efficiently solve the real-time task scheduling problem in cloud.

REFERENCES

[1] J. K. Author, "Title of dissertation," Ph.D. dissertation, Abbrev. Dept.,
[2] Peter Sempolinski, Douglas Thain, "A Comparison And Critique Of Eucalyptus, Open Nebula And Nimbus", Cloud Computing Technology And Science (CloudCom),DOI 10.1109/CloudCom.2010,Page(s):417-426
[3] Ms.NITIKA, "Comparative Analysis Of Load Balancing Algorithms In Cloud Computing ", International Journal Of Engineering And Science, 2011,ISSN:2319- 1813,Page(s): 34-38
[4] Shridhar G.Domanal and G.Ram Mohana Reddy "Load Balancing In Cloud Computing Using Modified Throttled Algorithm", IEEE International Conference on Cloud Computing in Emerging Markets, Oct, 2013,Page(s): 1-5
[5] Hamid Shoja ,Hossein Nahid, Reza Azizi " A Comparative Survey On Load Balancing Algorithms In Cloud Computing", International Conference On Computing, Communication And Networking Technologies, July,2014 , Page(s):1-5
[6] Veerawali Behal, Anil Kumar, "Cloud Computing: Performance Analysis Of Load Balancing Algorithms In Cloud Heterogeneous Environment", 5th International Conference Confluence The Next Generation Information Technology Summit, Sept,2014,Page(s): 200-205
[7] George Chang, Shan Malhotra, Paul Wolgast, "Leveraging The Cloud For Robust And Efficient Lunar Image Processing",IEEE, 2011, Page(s):1-8

