DIAGONAL SCALING IN CLOUD COMPUTING

[1.] **Aruna Mailavaram**
Asst. Prof. in CSE, Mahatma Gandhi Institute of Technology, Hyderabad.

[2.] **B.Padmaja Rani**
Prof. in CSE,JNTUHCEH, Hyderabad.

Abstract: Diagonalscaling is a combination of both horizontal and vertical scaling to create a flexible cloud system that can handle different work load effectively. The advantage of use of diagonal scaling is to handle traffic spikes and resource- intensive tasks. This improves system performance, increased scalability and flexibility and better resource usage. Organization can benefit from vertical scalability for high CPU or memory-intensive workloads, horizontal scalability for applications with high concurrent connections and diagonal scalability for applications with unpredictable traffic demands.

Introduction:

When it comes to cloud computing, the architecture is essentially a massive grid of "cloud servers" linked together in order to perform multiple tasks at once, and which leverages virtualization to make the most of the processing capacity available on each server [10].Cloud resource consumption, reaction speed, and flexibility of the web service can all be improved by balancing the load. Efficient load balancing reduces the time it takes to complete tasks and boosts the system's performance [11]. Task scheduling and virtual machine monitoring are critical for achieving load balancing in a cloud environment. In computer science, scheduler is an efficiency problem due to the cloud's host and virtualized design [12]. Thus, it is impossible to calculate and forecast the mapping of resources in cloud computing. As a result, an effective task scheduling method with dynamic load balancing is needed to ensure that fewer VMs are managed with overloaded or under-loaded operations [13]. Keeping asense on the virtual machine and balancing the workload is another crucial responsibility. Once a task is scheduled, the proposed model performs a scheduler operation on a virtual machine and moves the work to an under-loaded virtual machine [14].

Although the term "cloud computing" encompasses many various areas of the modern world, the two most important are

- Horizontal Scalability
- Vertical Scalability

Horizontal Scalability: (Scaling Out)

With Horizontal Scaling, new servers can be added to existing virtual machines to divide the work more evenly, enhancing performance. Rather than increasing the capacity of a single server, the strategy is to reduce the server's load [18]. Scaling out is another name for Horizontal Scaling. For Horizontal Scaling, load balancing, categorization, and a distributed data are all essential [19]. As a result of Horizontal Scaling, each node contains only a portion of the data. The figure 2 represents the horizontal scalability model.

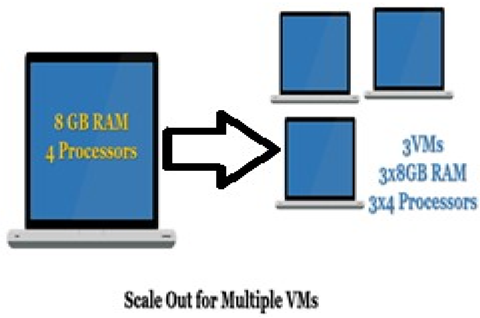


Fig 1: Horizontal Scalability Model

Vertical Scalability: (Scaling Up)

It's referred to as the Scale-up method. A single machine's capacity can be increased by taking advantage of additional resources in the very same logical server or unit, which is known as vertical scaling [20]. Allowed to add resources like processing capacity, storage, and memory to an already existing hardware or software system [21] is referred to as vertical scaling. The figure 3 represents the vertical scalability model.

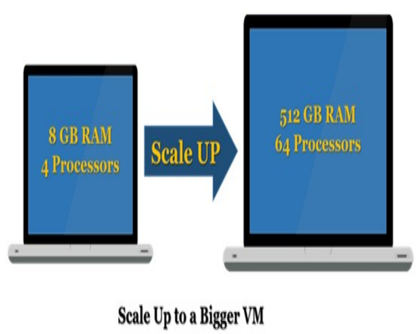


Fig 2: Vertical Scalability Model

Diagonal Scaling : In diagonal scaling we use the idea of both vertical and horizontal scaling. Organizations use scale up and scale out to deal with erratic surges.

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

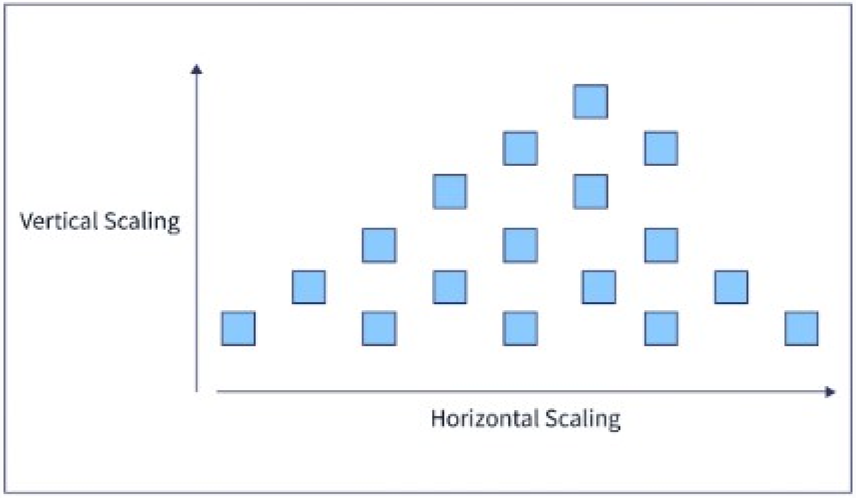


Fig 3 : Diagonal Scaling

Literature Survey :

According to Mohitkumar, load balancing is achieved in cloud environment in two steps: first one is to distribute the task among the node, second one is to monitor the virtual machine and perform the load balancing operation using task migration or virtual machine migration approach. The aim of task scheduling is to create a schedule and assigned each task to node (virtual machine) for specific time period so that all task are executed in minimum time span. Task scheduling is NP complete problem in the field of computer science because number of task and length of task change very rapidly in cloud environment. It is difficult to calculate all possible task-resource mapping in cloud environment and find an optimal mapping is not easy task. Therefore we need an efficient task scheduling algorithm that can distribute the task in effective way so that less number of virtual machine should be in overloaded or under loaded condition.Srinivasa Rao Gundu[1] et al. [2020] discussed types of load balancing algorithms along withtheir pros and cons. The static algorithms gather information about every user in detail andmaintain information about all the resources used by the cloud service provider and physicaland virtual machines.Zhenjiang Li[2] et al. used an energy awareness mechanism to reduce the energy consumedby the cloud resources to balance the cloud. The system dynamically calculates the number ofactive servers involved in cloud computing to achieve a particular task and computes theenergy consumed by them based on the throughput. It decides the load balancing factor.Many factors involve the computation of energy consumption, but this system has mainlyconsidered the planning and configuration of the cooling system involved in between themaster and slave nodes of the cloud.Xianyong Wei [3] et al. scheduled an optimistic task using a popular genetic algorithm known as "Ant Colony Optimization" (ACO), which is improved to achieve better globaloptimization rather than the local optimization to balance the load at the cloud server. Theobjective of any genetic heuristic algorithm is to define an optimization function that involvesmulti parameters like waiting time, degree of resources and cost of the task.Chunlin Li [4] et al. proposed a complex workflow operation in the geo-distributed cloud toachieve good load balancing. In the cloud environment, the cloud execution is a crucialparameter that decides the load balancing element. So, this system assigns all the tasks to beexecuted in a cloud as a queue, and a

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

shortest path task scheduling algorithm is proposed tominimize the load at virtual machines by converting a DAG representation into a hyper-graphto partition the tasks and allocate the resources using Dijkstra algorithm.

Problem formulation : We have to assign all the tasks to the virtual machines in an efficient manner to reduce make span, improve average resource utilization ratio, minimize over loaded and under loaded virtual machines and proposed algorithm will make all the tasks processed with in dead line.

Capacity of VM is CVM = p*q+BW{j,k}

Where p is the processing speed and q is he umber cpu and BW is the band width between j and k virtual machines.

Now we find out the load at the data center i.e., $DC = \sum_{i=1}^{m} CVM_j$

Load information at the VM can be found by LVM = number of task *∑task length/p*q.

So, total load at Data Center is $TL_{VM} = \sum_{j=1}^{m} L_{VM}$.

Expected Execution Time at Virtual Machine is calculated as

EET = Task Length/p*q.

Makespan

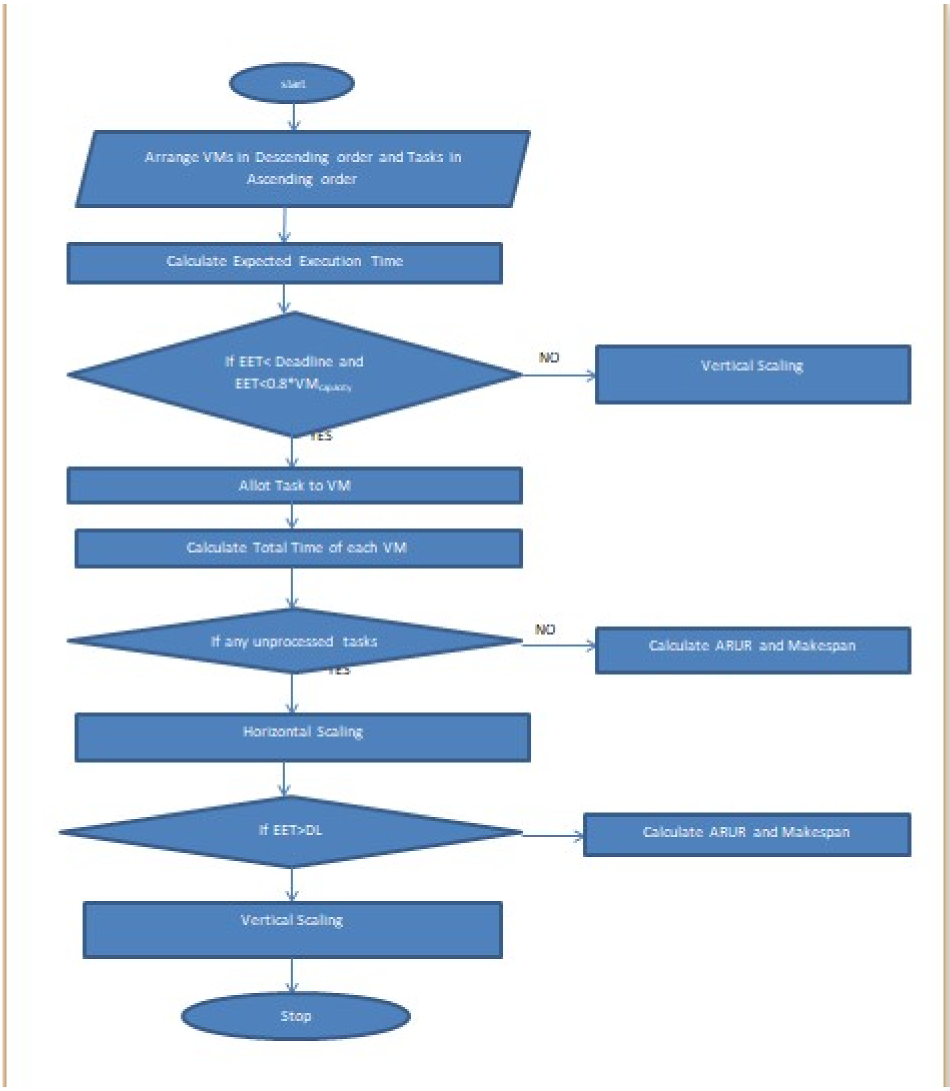
Makespan Time = $Max(\sum^{m}_{i=1} ET)$.

Average resource utilization ratio (ARUR) is calculated by the formula

ARUR = (mean time/ makespan)*100 where mean time = ∑ Time taken by resource (VM_j) to finish all the job /number of resource where j= { 1, 2, 3,…..n}
The range of average resource utilization ratio is 0 to 1, maximum value forARUR is 1 (resource utilization is 100%) and worst value is 0 (resource is in ideal condition).

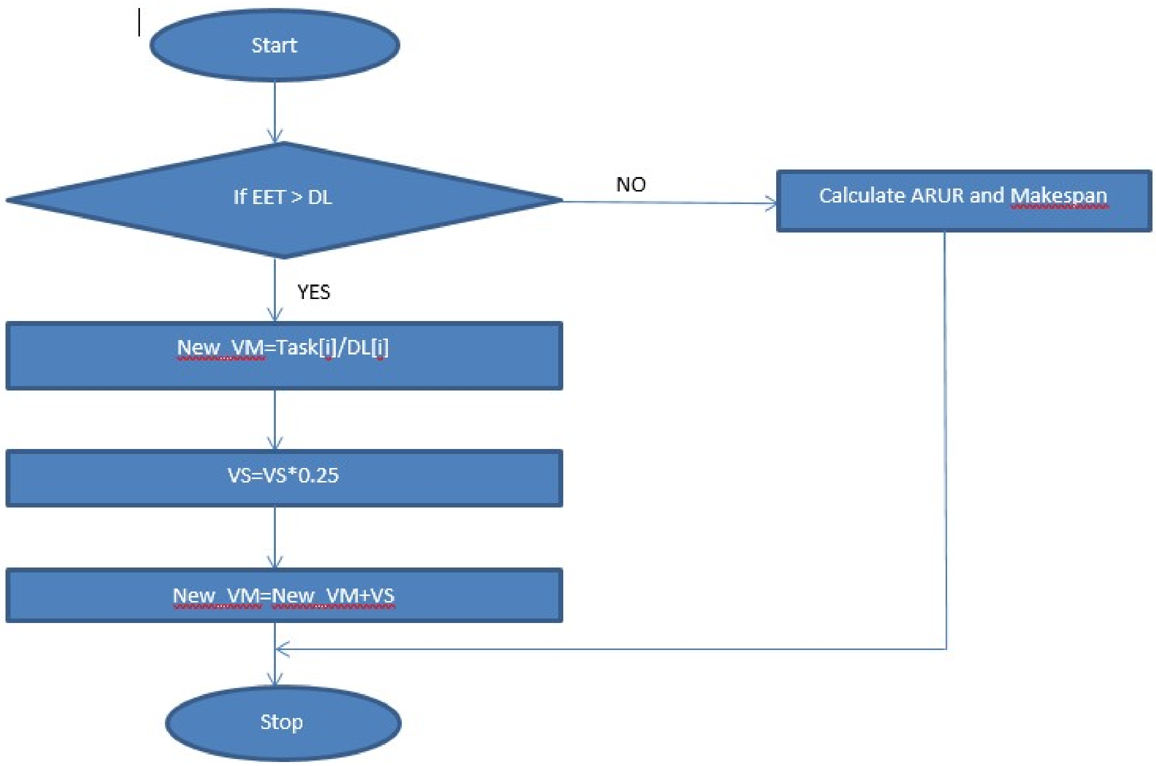
Flow Chart for Vertical and Horizontal Scaling :

Vol. 21, No. 1, (2024)
ISSN: 1005-0930



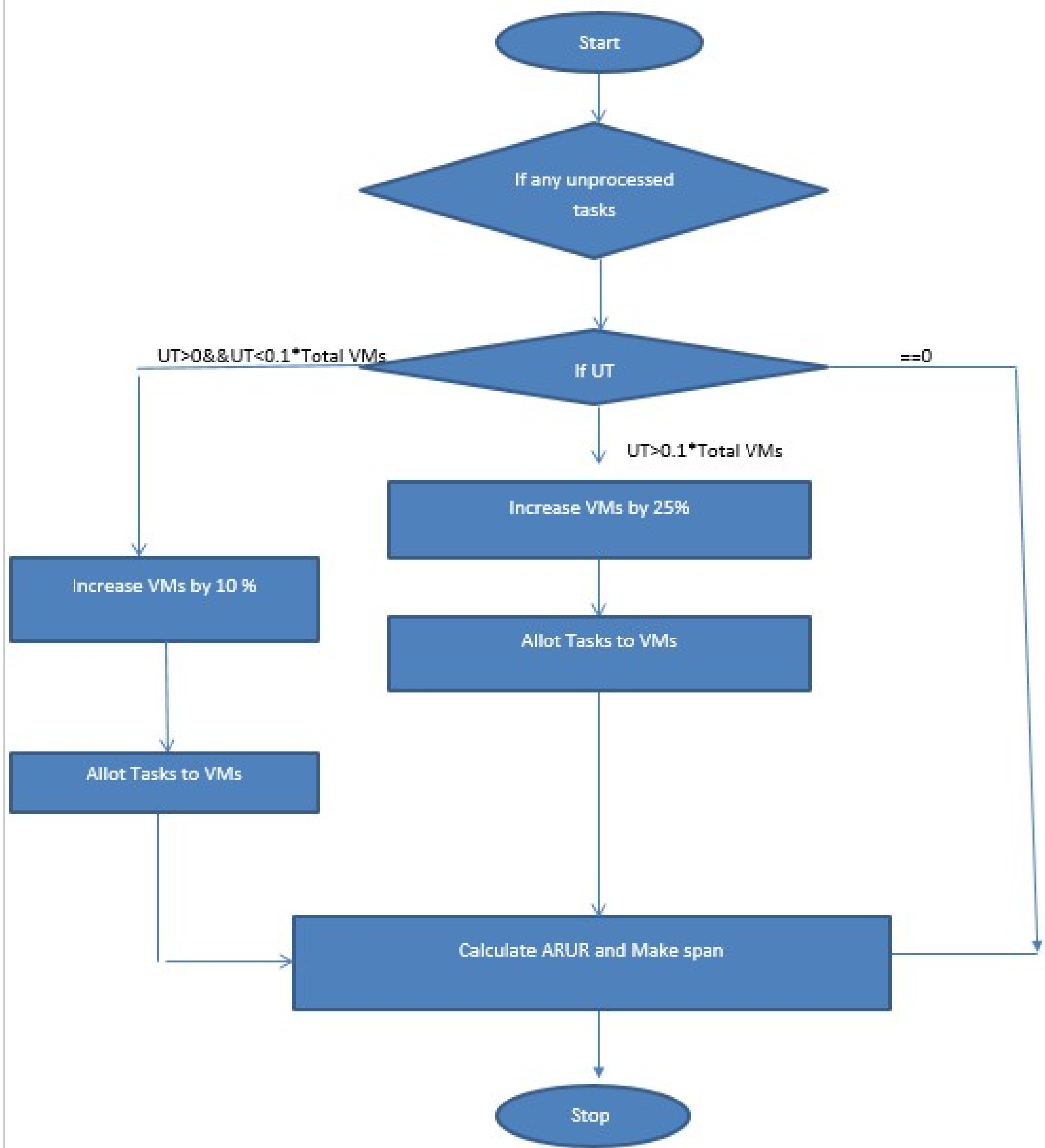
Flowchart for Vertical Scaling :

Vol. 21, No. 1, (2024)
ISSN: 1005-0930



Flowchart for Horizontal Scaling :

Vol. 21, No. 1, (2024)
ISSN: 1005-0930



Proposed Algorithm:Diagonal Scaling
Start
Input : VM[], Task[], TT[], DL[], UT, VS, HS, MT, ARUR, m, n, I, j, k.
Arrange :
Tasks in Ascending Order.
VMs in Descending Order.
Calculate Expected Execution Time for each Task

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

TT[j] = TT[j-1] + EET
If(EET[i] < Dead Line)
do
Assign Task[i] to VM[j]
Go to next task
Until TT[i] < 0.8*$VM_{capacity}$
Go to next VM
Repeat step 4 to 7
Else
Go to Vertical scaling until all VMs are assigned with some task
Do
If any unprocessed tasks left over
Go to Horizontal Scaling
If EET[i] < DL
Go to Vertical Scaling
Else
Calculate Makespan, Average Resource Utilization Ratio
Stop.

Algorithm : Vertical Scaling
Start.
If EET[i] > DL
New_VM[j] = Task[i]/DL[i]
VS = New_VM[j] * 0.25
New_VM[j] = New_VM[j] + VS
Else
Calculate ARUR and Make span
Stop.

Algorithm : Horizontal Scaling
Start
If any unprocessed tasks UT
Do
If UT>0 && UT < 0.1* Total VMs
Increase VMs by 10 percentage
Else if UT > 0.1 * Total VMs
Increase VMs by 25 percentage
Else
Calculate ARUR, Makespan
Stop

Here wehave proposed a novel load balancing algorithm which reduces the make span time, increases the Average Resource Utilization Ratio by considering that all tasks reach within the

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

deadline. So, in order to do this, we used cloud sim, a simulation tool. We have created VMs with different processing speeds and tasks of different lengths. All the tasks are arranged in ascending order, VMs in descending order. Initially we starting from 20 percent of VMs since we are keeping those VMs in spare to handle high length tasks. All the tasks were allotted VMs since the tasks of small lengths were allotted to high capacity VMs, initially for one VM three to four tasks were allotted as the length of the increases and MV capacity decreases only one task was allotted to VM by checking its overload condition if any tasks are remaining we are allotting them to the left over tasks for still remaining tasks we are going for horizontal scaling that is adding more number of tasks and vertical scaling by increasing system capacity. Considering the same synthesized data and found the results better.

Task ID	**Task Length(MI)**	**Deadline of Task {ms)**	**VMs (MIPS)**
17	98049	1050	760
18	107218	300	740
19	147281	1150	720
15	153285	800	700
7	182561	2000	680
16	205633	600	660
9	215744	1300	640
13	222784	550	620
10	253197	1250	600
14	253745	1000	580
4	325750	420	560
3	325800	410	
5	325970	700	
6	333911	660	
11	334013	450	
2	339760	920	
12	344630	400	
0	381771	400	
1	392397	745	

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

8	396156	300	

After Diagonal scaling we got better results compared to both vertical and horizontal scaling separately .

Task ID	**Task Length**	**Deadline of Task**	**Execution Time**
17	98049	1050	136
18	107218	300	284
19	147281	1150	488
15	153285	800	218
7	182561	2000	478
16	205633	600	302
9	215744	1300	326
13	222784	550	348
10	253197	1250	408
14	253745	1000	422
4	325750	420	420
3	325800	410	410
5	325970	700	440
6	333911	660	439
11	334013	450	428
2	339760	920	424
12	344630	400	400
0	381771	400	400
1	392397	745	456

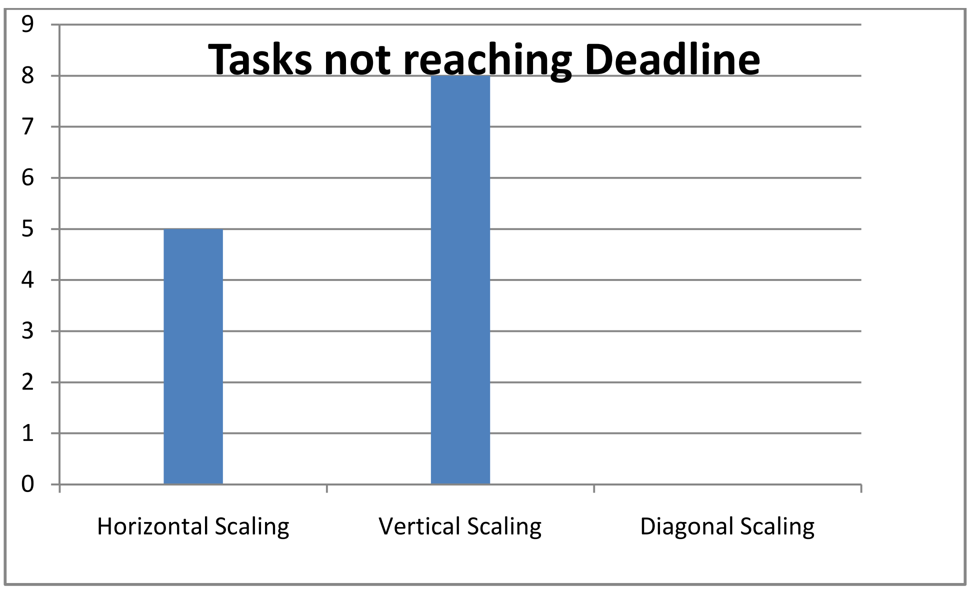
Vol. 21, No. 1, (2024)
ISSN: 1005-0930

8	396156	300	300

All the tasks have reached the deadline.

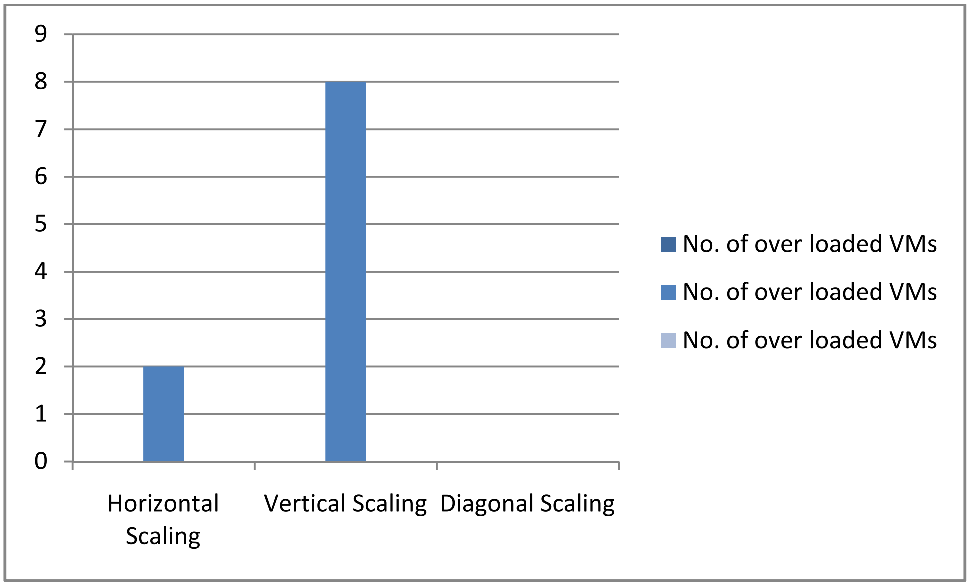
Results : Diagonal scaling have reached better results compared to other algorithms it achieved nearly five parameters, and proven that the results are better in terms of makespan, total number of tasks not reaching the deadline, number of under loaded VMs, number of over loaded VMs, Average Resource Utilization Ratio.

Tasks not reaching Deadline	
Horizontal Scaling	5
Vertical Scaling	8
Diagonal Scaling	0

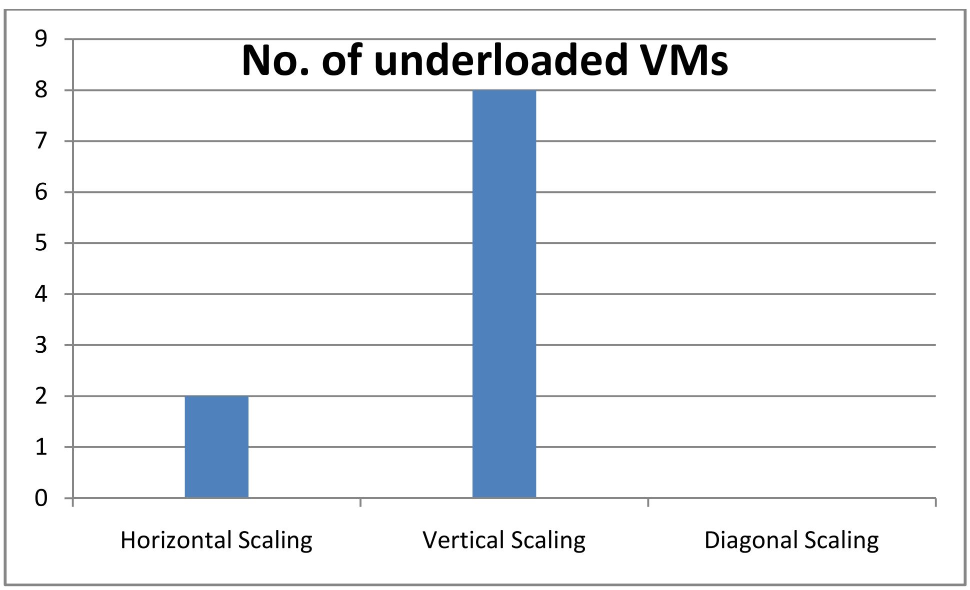


No. of over loaded VMs	
Horizontal Scaling	2
Vertical Scaling	8
Diagonal Scaling	0

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

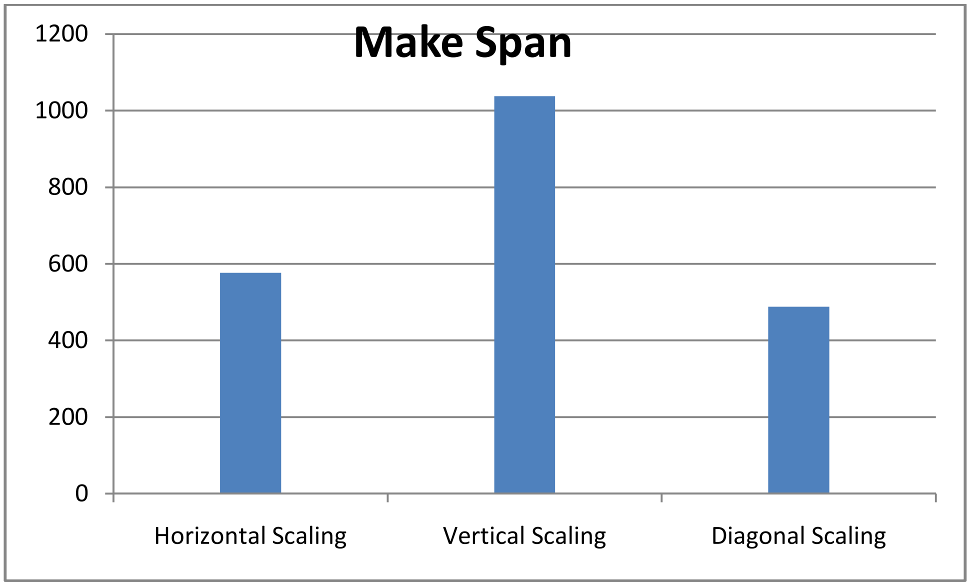


No. of underloaded VMs	
Horizontal Scaling	2
Vertical Scaling	8
Diagonal Scaling	0

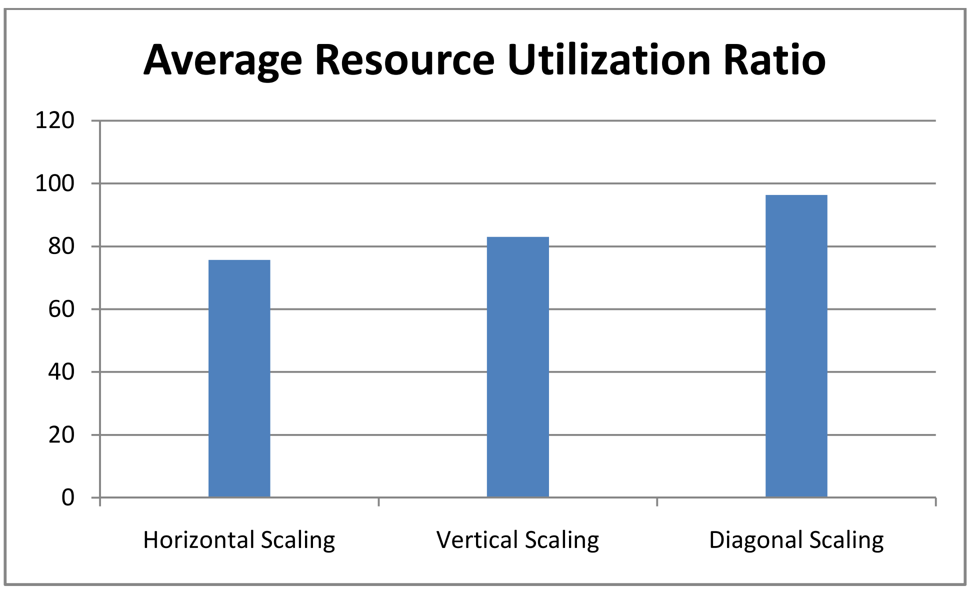


Make Span	
Horizontal Scaling	577

Vertical Scaling	1038
Diagonal Scaling	488



ARUR	
Horizontal Scaling	75.65
Vertical Scaling	83.01
Diagonal Scaling	96.4



Vol. 21, No. 1, (2024)
ISSN: 1005-0930

Conclusion : Wehave developed a dynamic load balancing algorithm which uses both vertical and horizontal techniques to improve the performance of the system . In my previous paper we did both vertical and horizontal separately and achieved that the horizontal scaling is better than vertical scaling. Here in this algorithm the combination of both is giving better results compared to many other load balancing algorithms. The comparative analysis will be done in my next upcoming paper. The tasks reaching dead line is very high and we are able to achieve reduced makespan.

References :

1. Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing Mohit Kumara,*,S.C.Sharma. https://doi.org/10.1016/j.procs.2017.09.141.

2.Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. https://doi.org/10.1016/j.compeleceng.2017.11.018..

3.Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing.https://doi.org/10.1016/j.procs.2017.12.092

4.Dynamic load balancing algorithm to minimize the make span time and utilize the resources effectively in cloud environment.

https://doi.org/10.1080/1206212X.2017.1404823.

5. Load balancing algorithm to minimize the make span time in cloud environment. ISSN1746-7233,England,UK World Journal of Modelling and Simulation. Vol.14(2018)No.4,pp.276-288.

6. A comprehensive survey for scheduling techniques in cloud computing.

https://doi.org/10.1016/j.jnca.2019.06.006.

7. Dynamic Auto-scaling and Scheduling of Deadline Constrained Service Workloads on IaaS Clouds. 10.1016/j.jss.2016.05.011

8. Optimized Task Scheduling and Resource Allocation on Cloud Computing Environment Using Improved Differential Evolution Algorithm. http://dx.doi.org/10.1016/j.cor.2013.06.012

9. Heuristic-based load-balancing algorithm for IaaS cloud.https://doi.org/10.1016/j.future.2017.10.035.

10. Load balancing in cloud computing: A big picture. https://doi.org/10.1016/j.jksuci.2018.01.003

11. A Priority Based Job Scheduling Algorithm Using IBA and EASY Algorithm for Cloud Metaschedularhttps://www.researchgate.net/publication/312814322