Hybrid Scheduling Algorithm for Efficient Load Balancing In Cloud Computing

Navpreet Singh
M.tech Scholar, CSE & IT Deptt., BBSB Engineering College, Fatehgarh Sahib,
Punjab, India
(IKG - Punjab Technical University, Jalandhar) navpreetsaini26@gmail.com
Dr. Kanwalvir Singh Dhindsa
Professor, CSE & IT Deptt.,
BBSB Engineering College, Fatehgarh Sahib,
Punjab, India
(IKG - Punjab Technical University, Jalandhar) kanwalvir.singh@bbsbec.ac.in

ABSTRACT

In cloud computing environment, various users send requests for the transmission of data for different demands. The access to different number of users increase load on the cloud servers. Due to this, the cloud server does not provide best efficiency. To provide best efficiency, load has to be balanced. The highlight of this work is the division of different jobs into tasks. The job dependency checking is done on the basis of directed acyclic graph. The dependency checking the make span has to be created on the basis of first come first serve and priority based scheduling algorithms. In this paper, each scheduling algorithm has been implemented sequentially and the hybrid algorithm (round robin and priority based) has also been compared with other scheduling algorithms.

Keywords: **Closest data center, Optimized response time, Dynamic Load, Round-Robin scheduling, Priority based scheduling.**

Date of Submission: March 29, 2017 Date of Acceptance: April 11, 2017

1. INTRODUCTION

CLOUD computing is a technology that delivers computing as service, rather than a product, where shared resources, information and software are provided over the network. The cloud is a pool of heterogeneous resources [1]. Its providers deliver the applications over the internet, which are accessible from any web browser connected to internet. The quality of the service provided has improved since the burden of managing the resources and their implementations has been shifted to the provider of the service. Hence cloud computing model has been of a huge benefit to the end users, IT buyers, software developers, system administrators and other corporate clients since it features low operational cost, ensures availability of pool of resources, free from capital cost and security. Cloud computing resources are provisioned to the end users in the form of services as pay-per-view basis.

The service providers own and manage the Data Centers at various locations, and these data centers may be configured with different hardware depending on its utilization. The hardware also keeps on changing with time depending upon the user requirement.
Cloud load balancing refers to distributing client requests across multiple application servers that are running in a cloud environment [3]. Cloud service providers maintain the service level agreement by scheduling the available resources efficiently and the time performance optimization is achieved by deploying the application on proper virtual machines according to the service level agreement. Efficient allocation of virtual machines is provisioned in two different steps: (a) static planning initially: group the set of virtual machines; classify them, and deployment on physical host. (b) provisioning of resources dynamically: depending upon the workload; virtual machines are created and additional resources are allocated dynamically.
Cloud load can be balanced by serving the requests received to the nearest data center. This approach is called Closest Data Center approach. It reduces the network costs and has serving capability to a limited area of request generation.

2. REVIEW OF LITERATURE

Many existing literature works have been reviewed as described below:
Xiao et.al[2] proposed an architecture, in which, to achieve the service level objective, all available hardware resources are pooled in a common shared space in cloud computing infrastructure, from which the hosted applications can access the resources as per their needs. Das et.al[3] proposed a utility function as a general two tier architecture for dynamic and autonomous resource allocation. The function consisted of a local agent that was responsible for calculating the utilities, for current or fore casted workload. The results were then transferred to global arbiter, which computes near-optimal configuration of resources.Segal et.al[4] described an architecture for dynamic scaling of web application. It consisted of front end load balancer, a number of web application virtual machine. Provisioning and DE-provisioning of virtual machines were controlled

by a dynamic scaling algorithm based on the relevant threshold of web application.

Jiyin et.al[5] proposed an adaptive resource allocation mechanism for cloud system with pre-emptible task execution to increase cloud utilization. However, this approach was not apt for time optimization and cost optimization.Buyya et.al[6] through a case study presented how Cloud Analyst tool can be used to model and evaluate a real world problem. The case study was of a social networking application hosted on the cloud. It illustrated the work as how a simulator can be used to effectively identify the usage patterns which can affect the data centers hosting the application.Soklic [7] studied comparisons between various load balancing techniques and proposed that static load balancing algorithms are more stable but at the same time, dynamic distributed algos are always considered better than the static balancing algorithms.

Kaur [8] discussed VM Load balancer algorithm to find the suitable virtual machine in a short period of time. Author proposed to count the max length of the virtual machine to allocate a new request. If the length of the current virtual machine is not sufficient, then a new virtual machine would be added. Jieqing et.al[9] proposed an algorithm for adding capacity to the dynamic balance mechanism for the cloud. The algorithm obtained better load balancing degree by taking lesser time all loaded tasks.

3. CLASSIFICATION OF LOAD BALANCING ALGORITHMS

Load balancers implement type specific algorithms to make load balancing decisions. The decision determines to which remote server a new job is to be forwarded. Few of the algorithms for load balancing are studied in this section. Depending on system state, load balancing algorithms can be divided into two types as static and dynamic [10]. A static load balancing algorithm does not take into account the previous state or behavior of a node while distributing the load. On the other hand, a dynamic load balancing algorithm checks the previous state of a node while distributing the load, such as CPU load, amount of memory used, delay or network load, and so on.

Static Algorithm: Static algorithms are appropriate for systems with low variations in load. In static algorithm, the traffic is divided evenly among the servers. This algorithm requires a prior knowledge of system resources. The performance of the processors is determined at the beginning of the execution. Therefore, the decision of shifting of the load does not depend on the current state of system. However, static load balancing algorithms have a drawback. In that, the tasks are assigned to the processor or machines only after it is created and those tasks cannot be shifted during its execution to any other machine for load balancing. Advantages: - Performs better in terms of complexity issue.

Disadvantage: - Compromises with the result as decision is solely made on statically gathered data. Also, the algorithms are non-preemptive.

Types of Static Algorithm: -

Round Robin scheduling: Load is distributed evenly to all the nodes. Equal load is assigned to each node in circular order without any priority and back to first node when the last node is reached. It is easy to implement, simple and starvation free.

Threshold algorithm: Load is assigned immediately on creation of the node. Each node has a load limit. When load state of a node exceeds its limit, a message is sent to all remote nodes regarding a new load state.

Randomized algorithm: A node is selected randomly. When the load exceeds the node's limit, it is migrated to a randomly selected new neighbor. It does not send any load message to other remote nodes. But it causes much communication overheads due to random selection of nodes.

Dynamic Algorithm: In dynamic algorithm, the server with the least load in the whole network or system is searched and preferred for balancing a load. For this, real time communication with network is needed which can increase the traffic in the system. Here, current state of the system is used to make decisions to manage the load. Dynamic algorithms respond to the actual current system state in making load transfer decisions. Since current state of the system is used to make dynamic load balancing decisions, processes are allowed to move from an over utilized machine to an underutilized machine in real time dynamically.

Advantages: - No single web server will be overloaded. Also, better performance report as it considers current load of system to choose next data center. The algorithms are preemptive.

Disadvantage: - Higher run time complexity. Communication overheads occur more and more as number of processes increase.

Types of Dynamic Algorithm: -

Priority based scheduling: The priorities are calculated during the execution of the system. Higher the priority of the request earlier will be resource allocation to the request. The goal of dynamic priority is to adapt to dynamically changing progress.

Central Queue Algorithm: Any new and pending activities are stored in a cyclic FIFO queue. This algorithm needs high communication among needs. Whenever a new request is received, first activity is removed from the queue.

Least Connection Algorithm: It decides the load distribution depending upon the present number of connections on a node. A load balancer maintains a log of number of connections on a node. Load increases with every new activity or a request, whereas load decreases when an activity finishes. Nodes with lesser connections are selected first.

4. CRITERIA FOR EFFICIENT LOAD BALANCING IN CLOUD COMPUTING

Cloud computing is process of execution of various tasks over the network in such a manner that user does not know any information about hardware components. Load balancing at different datacenters must be achieved in such a way so that minimum response time has been achieved by the system. In the process of execution of different tasks, datacenters allocate virtual machines to different resources that utilize different components of virtual machines so that minimum datacenter cost and minimum response time has been achieved by the system. In the process of cloud computing, load balancing policy has been designed using hybrid round robin scheduling with priority based approach. In the proposed work, resources have been provided different levels of priorities for allocation of virtual machines. Datacenters have different numbers of physical machines and these physical machines contain different number of virtual machines. The datacenters that have highest number of virtual machines can process large number of resources for output.

To schedule jobs for execution, the algorithms are very vital. In the cloud computing domain, one of the most challenging problems is the job scheduling algorithms. In the table below are discussed some of the existing algorithms for job scheduling.

Scheduling is one of the most vital tasks in cloud computing atmosphere. In the Table 1 below, various programming algorithmic rules and various parameters have been analyzed. The table above suggests an algorithmic rule for improving resource availableness and computing in cloud computing environment.

S.No.	Features	Round Robin scheduling	Priority Based scheduling	Hybrid scheduling
1.	**Load distribution**	Load is distributed evenly among all nodes. Equal load is assigned to each node in circular node without any priority and will be back to first node if last node has been reached.	Each request is assigned a priority and the one with highest priority is served first. Requests with same priority are served in FCFS manner. t is a non-preemptive scheduling algorithm.	It is a hybrid scheduling algorithm which is a combination of RR and PB. Requests are first given the priority and are then executed in round robin fashion.
2.	**Implementation**	It is easy to implement, is simple and is starvation free.	It is complex in nature.	It is more complex and is starvation free.
3.	**Interprocess communication**	It does not require interprocess communication.	It requires interprocess communication.	Requires much interprocess communication.
4.	**Disadvantage**	Cannot give expected result when the jobs are of unequal processing time.	Processes with lower priority are not given much opportunity to execute.	Still a better scheduling than RR and PB individually.

Table 1: Comparison of scheduling algorithms

5. PROPOSED ALGORITHM

The proposed algorithm can be formally described by the pseudo code as follows:

```
New Process P arrives.
P enters the ready queue.
Update the Service Rate and Arrival Rate.
Process P is loaded from ready queue into CPU to be executed.
IF (Ready Queue is Empty), BT(P)
    Time Quotient update SR and AR
End IF
IF (Ready Queue is Not Empty), AVG (Sum BT of processes in TQ ready queue)
    Update SR and AR.
End IF.
CPU executes P by TQ Time.
Update SR and AR.
IF (P is not terminated)
    Return P to the ready queue with its updated Burst Time.
    Update SR and AR.
End IF.
```

(where BT – Burst Time, TQ – Time Quotient, SR – Service Rate, AR – Arrival Rate)

6. EXPERIMENTAL SETUP

Cloud resource allocation has been done using proposed load balancing policy, that uses priority constraints for allocation of different VM on to resources.
Various parameters have been used for simulation of cloud services using different load balancing policies.

Parameter	Value Used
UB Name	UB1
Region	2
Requests Per User Per Hour	60
Data Size Per Request	100
Peak hour start(GMT)	3
Peak hour end (GMT)	9
Avg Peak Users	400000
Avg Off Peak Users	40000
DC 1 – No Of VM	80
DC 2 – No Of VM	40
DC 3 – No Of VM	20
VM Image Size	10000MB
VM Memory	1024 MB
VM Bandwidth	1000 bps
DC 1 – No. of Physical Machine	20
DC 2 – No. of Physical Machine	10
DC 3 – No. of Physical Machine	3
DC – Memory Per Machine	2048MB
DC – Storage Per Machine	40000MB
DC – Available BW Per Machine	40000 bps
DC – No. of Processors Per Machine	4
DC – Processor Speed	1000MIPS
DC – VM Policy	Time Shared
User Grouping Factor	10000
Request Grouping Factor	1000
Executable Instruction Length	250
Load Balancing Policy	Round Robin, PB, ESCEL and "Hybrid RR and Priority"

Table 2: Cloud Analyst Configuration for simulation

*(UB - user base, GMT – Greenwich Mean Time, DC - Data Centre, PB - Priority Based, RR - Round Robin, VM - Virtual Machine, ESCEL - Equally Spread Current Execution Algorithm, ms – millisecond, MB – MegaByte, bps – bits per second, MIPS – million instructions per second)

Table 2 above represents various parameters that have been used for simulation of the proposed work. These parameters have been used for simulation of different cloud service broker policies with different load balancing policies.

7. RESULTS AND DISCUSSION

Various scheduling algorithms have been implemented to evaluate the efficiency of overall response time and the service time in response to various service requests received. The results of these evaluations have been discussed here.

Table 3 below represents overall response time and Data Centre processing time for resources using Closest Data Centre service broker policy.

	RR	PB	ESCEL	**Proposed Hybrid**
Over all response time (ms)	300.864	300.78	300.63	**300.62**
DC Service Time (ms)	3.46	3.04	2.809	**2.77**

Table 3: Overall response time and DC Processing time using Closest Data center.

Hybrid algorithm has optimized overall response time as well as the data center service time which is 300.62 ms and 2.77 ms respectively for Closest data center service broker policy. It is because of the proposed hybrid algorithm, that the requests are served in the minimal duration of time. Hybrid algorithm is most efficient in case of Closest Data center approach because in this approach, every service request is first forwarded to the closest data center, hence resulting in lesser response time. Also, its priority is set high.

Table 4 below represents overall response time and datacenter processing time for resources using optimized response time datacenter service broker policy.

	RR	PB	ESCEL	**Proposed Hybrid**
Over all response time (ms)	301.09	300.95	300.86	**299.15**
DC Service Time (ms)	3.47	3.15	2.83	**2.80**

Table 4: Overall response time and DC Processing time using Optimized Response Time.

Hybrid algorithm using Optimized response time broker policy achieve optimal response time and data center service time for all the request received. As it is clear from table 4 above, that the suggested Hybrid algorithm has the lowest overall response time of 299.15 ms as well as the lowest time taken to service the data centre request i.e. 2.80 ms, hence making it the most suitable algorithm. This is because, unlike Round Robin scheduling, the request having higher priority is served first irrespective of its completion time and its request age.

Table 5 below represents overall response time and datacenter processing time for resources using dynamic load datacenter service broker policy.

	RR	PB	ESCEL	**Hybrid**
Over all response time (ms)	300.98	300.95	300.88	**300.02**
DC Service Time (ms)	3.80	3.68	3.50	**3.40**

Table 5: Overall response time and DC Processing time using Dynamic Load.

Hybrid algorithm is better than other mentioned scheduling algorithms because when load is divided dynamically, it gives better response time as compared to all other algorithms. The overall response time taken and Data Center Service Time is most efficient in case of hybrid algorithm because unlike PB scheduling, it automatically increases the priority for the old processes having low initial priority, hence executing them eventually. The different load balancing and service broker policies have been used for resource scheduling using different datacenters virtual machines.

Fig. 1 below depicts the graphical representation of overall response time of the user base for resource processing by using different load balancing and service broker policies. The overall response time of the Hybrid scheduling algorithm to serve the request in case of various service broker policies is much less as compared to other scheduling algorithms which makes it an efficient load balancing technique.

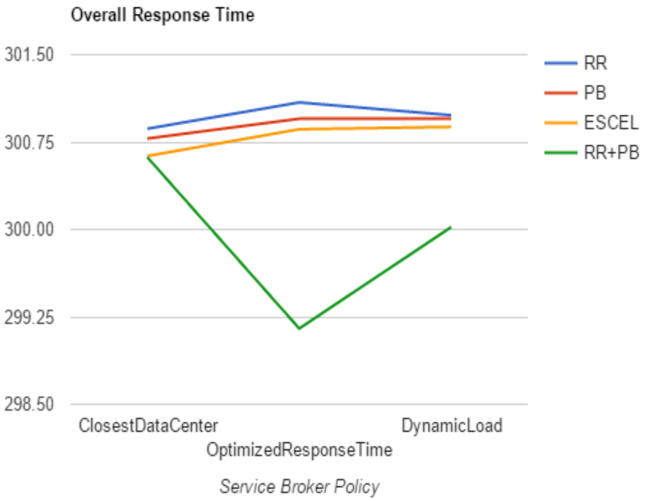

Fig.1: Over all response time using different Load balancing and service broker policy.

The Fig.1 above shows that the overall response time to service a request with Closest Data Center approach is least in case of the Hybrid scheduling algorithm that is 300.02 ms, which is lesser as compared to other scheduling algorithms. This is because, every service request is first forwarded to the closest data center, hence resulting in lesser response time. Also, its priority is set high.

Also, in case of Optimized Response Time approach, the service request with a higher priority is executed first, hence taking the least response time of 299.15 ms, which is much lesser as compared to other scheduling algorithms.

In the Dynamic Load approach for service execution, Hybrid scheduling algorithm takes overall response time of 300.02 ms which is much lesser as compared to other scheduling algorithms. This is because, in this approach, the load is divided among data centers dynamically. If a server is having no load or lesser load, it is assigned a request to service it. Hence making the process starvation free and efficient.

Fig.2 below depicts graphical representation of datacenter service times of various datacenters for resource processing by using different load balancing and service broker policies.

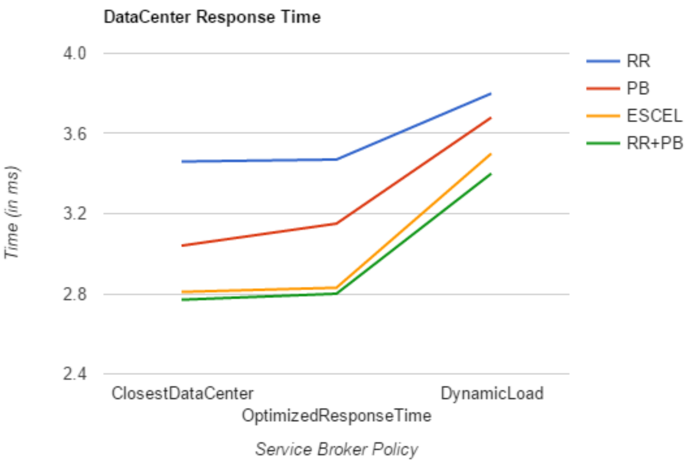

Fig. 2: Datacenter response time using different Load balancing and service broker policy

The response time taken by the datacenter to serve the request in case of the Hybrid scheduling algorithm in various service broker policies is much less as compared to other scheduling algorithms which makes it an efficient load balancing technique.

From fig.2 above, it is clear that in the Closest data center approach, the Data center response time is least is case of Hybrid scheduling algorithm, which is 2.77ms, lower as compared to other scheduling algorithms. This is because, in this approach, any service request received by the broker is assigned to the closest data center to be processed. Hence it takes the least time to service it. Using the Optimized Response Time approach, Hybrid scheduling takes only 2.80 ms for a Data center to respond to a request. It is lesser as compared to other scheduling algorithms for the same approach because in Hybrid scheduling, the priority of a process increases automatically with the age of the request received. Hence making it more efficient as compared to other algorithms. The proposed Hybrid scheduling algorithm is a dynamic scheduling algorithm. That means, the requests are alloted to different servers dynamically for execution. The requests are divided depending upon the load of a server and its capacity to service a request. Because of this approach, datacenter response time taken for Hybrid algorithm is 3.40 ms which is least as compared to other scheduling algorithms for the same approach.

Here various types of scheduling have been discussed. And also, the results of implementation of various types of scheduling algorithms in CloudSim have been shown in the Fig.1 and Fig.2 above. With this experimental setup, the performance of scheduling algorithms have been analyzed. The proposed Hybrid algorithm of Round Robin and Scheduling combined has shown impressive results as compared to other algorithms in achieving maximum utilization of resources in minimal duration of time. According to the results of hybrid Round Robin and Priority Based Scheduling algorithm, all the scheduling criterion is perfect as compared to other scheduling algorithms.

Table 6: Comparison of load balancing scheduling algorithms

S.No.	Features	RR	PB	Hybrid
1.	**Load distribution**	This is a priority free algorithm.	Each process is given a priority.	Each process has a priority.
2.	**Request execution**	Execution is done in a circular order, one after another.	Higher priority requests are served first.	Larger priority processes have large time slices.
3.	**Improvement**	It doesn't have starvation but for large number of processes, wait time can often be too long.	Indefinite blocking for low priority requests.	Requests are executed on basis of their priority and the priority increases gradually for aging requests.

The proposed hybrid scheduling algorithm is a combination of round robin and priority based scheduling algorithm. So it features all the properties of both algorithms and also satisfies any limitation in any of the two algorithms. As it is clear from table 6 above that, the proposed hybrid algorithm provides each process a priority and priority increases for the aging requests, it makes this algorithm more efficient. By analyzing performance evaluation parameters, it can be concluded that proposed approach provides much better results than previous approaches.

7. CONCLUSION

In cloud computing scenario, number of tasks has to be assigned on various processes to handle load on the cloud. These tasks have been divided into sets and the dependency checking is done for prevention of dead lock state or to prevent demand of various extra resources for allocation. Make span has been developed based on the allocation. Tasks must be checked for dependency by using directed Acyclic Graph. The tasks that are divided by following the Hybrid scheduling are more efficient as compared to all existing load scheduling algorithms. It also ensures the efficient and fair distribution of all computing resources.

REFERENCES

[1] K. Kishor, V. Thapar, "An efficient service broker policy for Cloud computing environment", International Journal of Computer Science Trends and Technology (IJCST) - Vol. 2, Issue 4, July-Aug 2014.

[2] Qiang Li, Qinfen Hao, Limin Xiao, "Adaptive management of virtualized resources in cloud

[3] T.C. Chieu, A. Mohindra, A.A. Karve, A. Segal, "Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment," in IEEE International Conference on e-Business Engineering, pp. 281-286, Dec. 2009.

[4] L. Jiyin, Q. Meikang, J.W. Niu, Y. Chen, and Ming, Adaptive Resource Allocation for Preeemptable Jobs in Cloud Systems, IEEE International Conference on Intelligent Systems Design and Applications, pp. 31-36, 2010.

[5] B. Wickremasinghe, N. Rodrigo Calheiros, and R. Buyya, Cloud Analyst: A CloudSim-based Visual Modeller for Analyzing Cloud Computing Environments and Applications, IEEE, January 2011.

[6] Milan E. Soklic "Simulation of Load balancing algorithms" ACM-SIGCSE Bulletin, Vol.34, Issue 4, December 2002.

[7] Jaspreet Kaur, "Comparison of Load balancing algorithms in a Cloud", International Journal of Engineering Research and Applications", Vol. 2, Issue 3, May-June 2012.

[8] Zhang Bo, Gao Ji, Ai Jieqing, "Cloud Loading Balance algorithm", Information Science and Engineering, Second International Conference, Vol.2, No.5, 4-6 Dec. 2010.

[9] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 6, pp. 1107-1117, 2013.

[10] Di Caro, G. Dorigo, M. Mobile, "Agents for adaptive routing" in Proceedings of the Thirty-First Hawaii International Conference on System Sciences, Kohala Coast, HI, USA, Vol. 7, pp. 74–83, January 1998.

[11] Bo, Z., Ji, G., Jieqing, A., "Cloud Loading Balance algorithm", Proceedings of the 2010 2nd International Conference on Information Science and Engineering, Hangzhou, China, pp. 5001–5004, December 2010.

[12] Wu Lee, Chan, Huang, "Dynamic load balancing mechanism based on cloud storage", Computing, Communications and Applications Conference, Hong Kong, China, pp. 102–106, January 2012.

[13] A. Bhadani, Chaudhary, "Performance evaluation of web servers using central load balancing policy over virtual machines on cloud", Third Annual ACM Bangalore Conference, Bangalore, India, pp. 16-19, January 2010.

[14] K. Nishant, P. Sharma, V.Krishna, C. Gupta, K.P. Singh, N. Nitin, R. Rastogi, "Lord Balancing of Nodes in Cloud Using Ant Colony Optimization", 2012 UK Sim 14th International Conference on Computer Modelling and Simulation, Cambridge, UK, pp.3-8, March 2012.

[15] A.P. Deshmukh, Prof. K.Pamu, "Applying Load Balancing: A Dynamic Approach", (IJARCSSE), Vol.2, Issue 6, June 2012.