Weighted Diffusive Load Balancing Algorithm In Cloud Environment

[1]Jaydeep R Viradiya , [2]Milankumar Sanandia

Abstract

Load balancing is a computer networking methodology to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. In load balancing algorithms parameters like throughput, fault tolerance, resource utilization, *scalability etc, are considered. In this paper firstly analysis of the diffusive load balancing algorithm for VM allocation and then compared it with a proposed weighted diffusive load balancing algorithm to achieve better response time and increase performance.*

Keywords: Virtual Machine, Load Balancing, Cloudsim.

1. Introduction

Cloud computing services can be used from diverse and widespread resources, rather than remote servers or local machines. There is no standard definition of Cloud computing. Generally it consists of a bunch of distributed servers known as masters, providing demanded services and resources to different clients known as clients in a network with scalability and reliability of datacenter. The distributed computers provide on-demand services. Services may be of software resources (e.g. Software as a Service, SaaS) or physical resources (e.g. Platform as a Service, PaaS) or hardware/infrastructure (e.g. Hardware as a Service, HaaS or Infrastructure as a Service, IaaS). Amazon EC2 (Amazon Elastic Compute Cloud) is an example of cloud computing services [3].

Load balancing is a computer networking method to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy [1].

2. Existing load balancing technique

[A] Decentralized content aware load balancing: - In [5] Nidhi Jain Kansal find out that a new content aware load Balancing policy named as workload and client aware policy (WCAP). It uses a unique and special property (USP) to specify the unique and special property of the requests as well as computing nodes. USP helps the scheduler to decide the best suitable node for the processing the requests. This strategy is implemented in a decentralized manner with low overhead. By using the content information to narrow down the search, this technique improves the searching performance and hence overall performance of the system. It also helps in reducing the idle time of the computing nodes hence improving their utilization.

[B] Join-Idle-Queue - In [5] Nidhi Jain Kansal find out that a Join-Idle-Queue load balancing algorithm for dynamically scalable web services. This algorithm provides large scale load balancing with distributed dispatchers by, first load balancing idle processors across dispatchers for the availability of idle processors at each dispatcher and then, assigning jobs to processors to reduce average queue length at each processor. By removing the load balancing work from the critical path of request processing, it effectively reduces the system load, incurs no communication overhead at job arrivals and does not Increase actual response time.

[C] Scheduling strategy on load balancing of virtual machine resources - In [5] Nidhi Jain Kansal find out that a scheduling strategy on load balancing of VM resources that uses historical data and current state of the system. This strategy achieves the best load balancing and reduced dynamic migration by using a genetic algorithm. It helps in resolving the issue of load-imbalance and high cost of migration thus achieving better resource Utilization

.

[D] Central load balancing policy for virtual machines - In [5] Nidhi Jain Kansal find out that a Central Load Balancing Policy for Virtual Machines (CLBVM) that balances the load evenly in a distributed virtual machine/cloud computing environment. This policy improves the overall

performance of the system but does not consider the system that is fault-tolerant.

[E] A Task Scheduling Algorithm Based on Load Balancing - In [5] Nidhi Jain Kansal find out that a two-level task scheduling mechanism based on load balancing to meet dynamic requirements of users and obtain high resource utilization. It achieves load balancing by first mapping tasks to virtual machines and then virtual machines to host resources thereby improving the task response time, resource utilization and overall performance of the cloud computing environment.

[F] Biased Random Sampling - M. Randles et al. [3] investigated a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. The performance of the system is improved with high and similar population of resources thus resulting in an increased throughput by effectively utilizing the increased system resources. It is degraded with an increase in population diversity.

Method :-As give in [6] these method is dynamic in nature. To achieve these we have to analyze the degree distribution of nodes in a stochastic network system with a fixed number of nodes and fixed average number of edges. A node's in-degree refers to the free resources of the node. The job assignment and resource updating processes required for load balancing are encoded in the network structure. Therefore, when a node receives a new job, it will remove one of its edges to decrease its in-degree.
Similarly when the node completes the job it will add the edge to the node. Hence, the generated graph using this protocol will be a strongly connected directed graph.
`The increment and decrement of node's in-degree is performed via Biased Random Sampling (BRS). Random sampling is the process whereby the nodes in the network are randomly picked up with equal probability. The sampling starts at some fixed node, and at each step, it moves to a neighbour of the current node, which is chosen randomly according to an arbitrary distribution.

[G] Active Clustering - M. Randles et al. [3] investigated a self-aggregation load balancing technique that is a self aggregation algorithm to optimize job assignments by connecting similar services using local re-wiring. The performance of the system is enhanced with high resources thereby increasing the throughput by using these resources effectively. It is degraded with an increase in system diversity.
Method:-As given in [2] it is based on the iterative execution by each node in the network of the following steps:

- At a random time the node elects itself as the initiator node and elects a matchmaker node among its neighbours.
- The matchmaker node chooses one neighbour that is compatible with the initiator and makes the two establish a new link.
- Finally, the matchmaker removes a link between itself and the chosen neighbour.

[H] Diffusive load balancing algorithm

Diffusive load balancing algorithm is described by fig 1 in that load is distribute in round robin manner to the virtual machine.

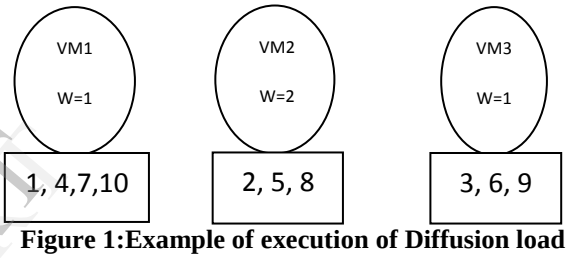

Figure 1:Example of execution of Diffusion load balancing algorithm

In figure 1 there are three virtual machine and also assign weight based on virtual machine configuration and assign the 10 job on these virtual machine. Based on diffusive load balancing algorithm it just schedule job in roundrobin manner to available virtual machine. So VM1 gets four jobs, VM2 gets three jobs and VM3 gets three jobs.

3. Proposed algorithm

3.1. Introduction to proposed system

Diffusive load balancing technique distributes the load among all nodes without node configuration. Proposed system will distributes the load with node configuration like based on weight allocated to the server node.

From the fig 2 we can understand this technique that based on weight allocate to particular node it will receive the jobs. In fig 4.1 VM2 has weight = 2 so it has a twice job then VM1 and VM2

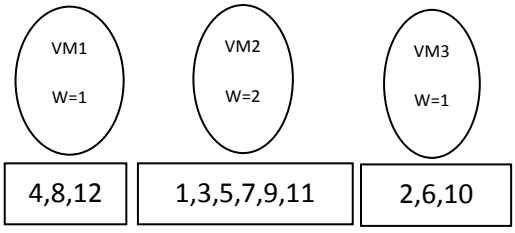

Fig 2 Load distribute in weighted diffusive LB

Weight calculation is comparative and it considers many factor like memory, Number of Cpu, MIPS (Micro Instruction per Second), bandwidth etc of VM.

For Example

- VM with one cpu, 1GB of memory, 1TB of Storage space, 1000000 bandwidth will have weighted count=1
- VM with two cpu, 4GB of memory, 2TB of Storage space and 1000000 bandwidth will have weighted count=2
- VM with four cpu, 8GB of memory 4TB of Storage space and 1000000 bandwidth will have weighted count=4 and so on.

Benefits:-

- Improves the performance of the system.
- Decreases overhead associated.
- Improve response time.

Disadvantage of existed system:-

- It just assign the jobs to particular VM in rounrobin manner without VM consideration so it increase the response time of the jobs.
- when VM is not capable to take a load then also these technique give the job to that VM so at the processing time it just drop the cloudlet and did not give the response.
- Decrease the performance of the system.

3.2. Description of Algorithm

Below there is a description of the weighted diffusive load balancing algorithm.

STEP 1: Make VM's of different category on datacenter with appropriate memory, storage, bandwidth etc.

STEP 2: Calculate weight factor for all VM, which are created on datacenter, on the basis of its computing power like its number of cpu, speed of processor etc.

STEP 3: WeightedDiffusiveLoadBalancer maintains an index table of VMs, associated weighted count and the number of requests currently allocated to the VM. At start all VM's have 0 allocations.

STEP 4: When a request to allocate a new VM from the DataCenterController arrives, it parses the table and identifies the least loaded VM.

STEP 5: After Identifying the least loaded VM's in different datacenters, it allocate requests to the most powerful VM according to the weight assigned. If there are more than one, the first identified is selected.

STEP 6: WeightedDiffusiveLoadBalancer returns the VM id to the DataCenterController.

STEP 7: The DataCenterController sends the request to the VM identified by that id.

STEP 8: DataCenterController notifies the WeightedDiffusiveLoadBalancer of the new allocation.

STEP 9: WeightedDiffusiveLoadBalancer updates the allocation table increasing the allocations count for that VM.

STEP 10: When the VM finishes processing the request, and the DataCenterController receives the response cloudlet, it notifies the WeightedDiffusiveLoadBalancer of the VM de-allocation.

STEP 11: The WeightedDiffusiveLoadBalancer updates the allocation table by decreasing the allocation count for the VM by one.

STEP 12: Continue from step 4.

4. Experiment setup

The proposed algorithm is implemented in cloudsim simulation toolkit[7]. Java language is used for implement new weighted diffusive load balancing algorithm. Below there is a some of the assumption to compare the proposed algorithm with existing one.

Table 1: Simulation consideration

Parameter	Value
Simulation toolkit	Cloudsim
Number of host	100
Number of Datacenter	1
Host storage	1000000MB
VM storage	10000MB

4.1. Response time Vs Different VM

Below there is a reading of simulation of diffusive LB and weighted diffusive LB for average response time with varying the type of VM.

Table 2: Constant parameter for graph response time Vs different VM

Cloudlet	1000
Host	100
Datacenter	1

Table 3: Simulation result for response time Vs different VM

Different VM	Avg Response time	
	Diffusive LB	Weighted Diffusive LB
2	1245.43	1102.42
3	1018.10	840.10
4	879.38	675.30
5	770.90	574.10
6	706.74	500.74

In graph average response time become decrease as the diversity increase for both but Weight Diffusive LB have a little improvement then diffusive LB.

Figure 3 shows that increase in diversity in terms of different VM decrease the average response time of 1000 cloudlet in both of technique. But in weighted diffusive LB result of average response time is better than diffusive LB.

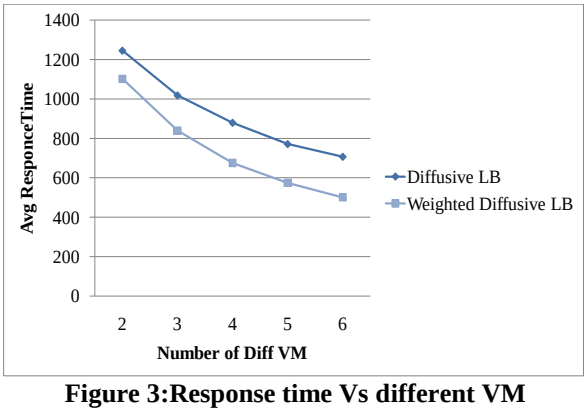

Figure 3:Response time Vs different VM

4.2. Number of cloudlet Vs performance

This experiment is conduct for check the performance of the system with respect to proceed cloudlet based on input cloudlet. Below there is a constant parameter consideration during result were taken.

Table 4: Constant parameter for number of cloudlet Vs performance

Cloudlet	1000
Diff Cloudlet	3
Diff VM	4
Host	100
Datacenter	1

Table 5: Simulation result for number of cloudlet Vs performance

Number of cloudlet	Proceed cloudlet	
	Diffusive LB	Weight diffusive LB
1000	733	987
1500	1109	1491
2000	1479	1990
2500	1854	2485
3000	2200	2985

Figure 4 shows that number of proceed cloudlet using weighted diffusive LB is higher than diffusive LB.

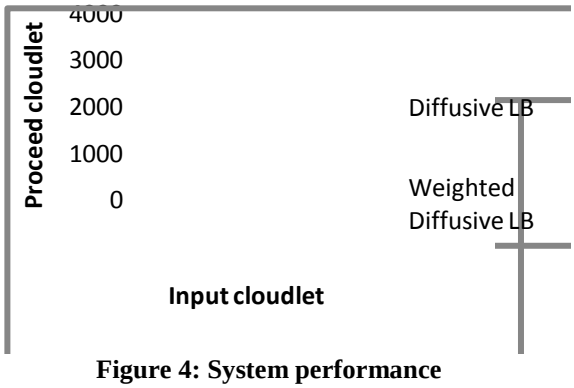

Figure 4: System performance

5. Conclusion

In this paper new VM load balancing algorithm has been developed based on weighted diffusive technique on cloud computing environment in cloudsim toolkit, using java language. In the proposed algorithm cloudlet(job) is assign to particular VM based on the configuration of the VM, according to the experiment we conclude that if it select the least loaded VM for execution of job then it also increase the performance of the system also decrease the response time of the jobs.

Proposed algorithm mainly focus on load balancing on virtual machine using weighted diffusive technique so load balancing can also be extented on host machine using weighted diffusive technique. It can also be extended to heterogeneous environment.

6. References

[1] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Approach, TATA McGRAW-HILL Edition 2010.

[2] E. Di Nitto, D.J. Dubois, R. Mirandola, F. Saffre and R. Tateson, "Applying Self-Aggregation to Load Balancing: Experimental Results." In Proceedings of the 3rd international Conference on Bioinspired Models of Network, information and Computing Systems (Bionetics 2008), Article 14, 25 – 28 November, 2008.

[3] Martin Randles, David Lamb, A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops,pp-551-556.

[4] Ali M. Alakeel, "A Guide to Dynamic Load Balancing in Distributed Computer Systems", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[5] Nidhi Jain Kansal, Inderveer Chana, "Cloud Load Balancing Techniques: A Step towards Green Computing", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012, pp 238-246

[6] O. Abu- Rahmeh, P. Johnson and A. Taleb-Bendiab," A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks", INFOCOMP - Journal of Computer Science, ISSN 1807-4545, 2008, VOL.7, N.4, December, 2008, pp. 01-10.

[7] Saurabh Kumar Garg and Rajkumar Buyya, "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations", 2011 Fourth IEEE International Conference on Utility and Cloud Computing,pp:105-113

[8] Ram Prasad Padhy, P Goutam Prasad Rao "LOAD BALANCING IN CLOUD COMPUTING SYSTEMS" Thesis from National Institute of Technology, Rourkela-769 008, Orissa, India May, 2011.

[9] Ali M. Alakeel, "A Guide to Dynamic Load Balancing in Distributed Computer Systems", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[10] Antonio Corradi, Letizia Leonardi and Franco Zambonelli, "Diffusive Load-Balancing Policies for Dynamic Applications", IEEE Concurrency, march 1999,pp:22-31

[11] F. Saffre, R. Tateson, J. Halloy, M. Shackleton, and J. L. Deneubourg,"Aggregation Dynamics in Overlay Networks and Their Implications for Self-Organized Distributed Applications" ,The Computer Journal, 2008.

[12] J. Pasteels, J. Deneubourg, and S. Goss. "Self-organization mechanisms in ant societies (i): trail recruitment to newly discovered food sources". In From individual to collective behavior in social insects, pp 54-155.

[13] D. Devescovi, E. Di Nitto, D. J. Dubois, and R. Mirandola. "Self-Organization Algorithms for Autonomic Systems in the SelfLet Approach", In International Conference on Autonomic Computing and Communication Systems, Autonomics 2007. ACM, 2007.