INDEPENDENT MEDIATOR BASED LOAD-BALANCING SYSTEM

[1]Abhishek Kumar, [2]Dr. Achintya Singhal
Research Scholar, SSSUTMS SEHORE, MP
Associate Professor, Computer Science, Banaras Hindu University, VARANASI,UP

Abstract: Cloud computing shares information and give numerous resources to clients. Clients pay just for those resources as much they utilized. Cloud computing stores the information and circulated resources in the open environment. The measure of information stockpiling increments rapidly in open environment. In this way, load balancing is a fundamental test in cloud environment. Load balancing is conveyed the dynamic workload over different hubs to guarantee that no single hub is overloaded. It helps in legitimate usage of resources .It additionally enhance the execution of the framework. However this part of cloud computing has not been given careful consideration yet. In spite of the fact that load balancing is being considered as an imperative viewpoint for other unified internet based computing environments, for example, appropriated computing, parallel computing and so forth. Numerous algorithms had been proposed for finding the arrangement of load balancing issue in these fields. In any case, not very many algorithms are proposed for cloud computing environment. Since cloud computing is fundamentally not quite the same as these different kinds of environments, isolate load balancing algorithm should be proposed to provide food its necessities. This work proposes an Independent Mediator Based Load-Balancing Algorithm (IMLB) which gives dynamic load balancing to cloud environment. The proposed system has been actualized and found to give tasteful outcomes.

Keywords: Load Balancing Algorithm, Cloud computing, Load agent,

Introduction

The cloud computing is an appropriated internet based worldview, intended for remote sharing and utilization of various resources and administrations like stockpiling, computational abilities and applications and so forth with high unwavering quality over the extensive systems. In any case, because of dynamic approaching solicitations, dynamic asset allotment is required in it. This innate dynamism in cloud computing requires productive load balancing instruments. Load balancing concerns dissemination of resources among the clients or demands in uniform way with the goal that no hub is overloaded or sitting inert. Like in, all other internet based circulated computing errands, load balancing is a vital perspective in cloud computing. Without load balancing arrangement, productivity of some overloaded hubs can strongly corrupt at times, prompting infringement of SLA. In conventional disseminated computing, parallel computing and matrix computing environments load balancing algorithms are sorted as static, dynamic or blended planning algorithms in view of their temperament [6] where:

a)Static Load Balancing Algorithm is suitable for small distributed environments with high Internet speed andignorable communication delays.

b)Dynamic Load Balancing Algorithm focuses on reducing communication delays and execution time and thus are suitable for large distributed environments.

c)Mixed Load Balancing Algorithm focuses on symmetrical distribution of assigned computing task and reducing communication cost of distributed computing nodes.

Literature review

Load balancing is one of important problems of heterogeneous computer networks. To address this problem,many centralized approaches have been proposed in the literature but centralization has proved to raise scalability tribulations. Randleset. al [8] provided a comparative analysis of various dynamic load balancing algorithms (Honeybee foraging, Biased Random Sampling, and Active Clustering). Their analysis has highlighted that honeybee algorithm has maximum throughput with increased system diversity as compared to other two algorithms. The honeybee algorithm is motivated from the behaviour of biological bees that move in search of their food. Similarly in load balancing there are virtual servers offering virtual services. Every server requiring services calculates the profit and posts it on its advert board. The servers interested in serving the request also calculate their profit and compare it with the colony profit. If case of high colony profit interested server serves the current virtual server otherwise returns to the scout behaviour i.e. to choose another server randomly.

Hu. et. al. [5] proposed genetic algorithm based scheduling mechanism for load balancing among virtual machines. This mechanism selects the least loaded virtual machine for load transfer and optimizes the high migration cost. However due to large number of virtual machines and frequent service requests in the data centre, there is chance of inefficient service scheduling. Xuet. al. [4] introduced a model for load balancing in public cloud by using game theory. This algorithm is based on cloud portioning. They divided the cloud into three categories idle, normal and overloaded on the basis of load degree. Zero load degree represents an idle cloud whereas if it lies between zero and highest value then the cloud status is normal otherwise the cloud is overloaded. Here method of selecting range for load degree has been left unaddressed. Wang et. al. [13] has proposed two static algorithms incloud environment. One is for Opportunistic load balancing in which incoming tasks received by a node haveminimum execution time which is calculated by service manager. Second is Load Balance Min Min which improves the resource utilization by maintaining the load balance. However both these algorithm are not suitable for CC as they do not support dynamic environments.

Osman et al. [19] proposed a system to migrate legacy and network application by providing a virtualization layer on top of the operating system and transferring a process group. They achieve lower downtime of service, but still use stop-and-copy approach. Nakaiet. al [20], introduced an approach for client-based load distribution that adaptively changes the fraction of the load that each client submits to each service replica to minimize overall response times. Bhaskaret. al. [2] proposed a mechanism working in two phases. In first phase it finds the CPU

utilization and memory required for each instance and also finds the memory available for each virtual machine. In second phase, it compare the available resources with required resources, if required resources are available then proceed further otherwise discard the request. Drawback of this mechanism is that it lacks in scalability. Xuet. al[15] has introduced an agent based model using decision theory. The aim of this model is to reduce the computational cost involved in load balancing. The migration concept used in this architecture transfers the load from overloaded nodes to under loaded nodes.

Proposed work

From the above analysis, there is a need of an algorithm which can offer most extreme resource usage, greatest throughput, least reaction time, dynamic resource planning with adaptability and dependability. This work proposes an Independent Mediator Based Load-Balancing Algorithm (IMLB) to address above issues. At whatever point a VM ends up plainly overloaded, the specialist organization needs to disperse the resources in such a way, to the point that the accessible resources will be used in a legitimate way and load at all the virtual machines will stay adjusted. IMLB component involves three operators: Load Agent, Channel Agent and Migration Agent. Load and channel operators are static operators though relocation operator is a subterranean insect, which is an extraordinary class of versatile specialists. The purpose for sending ants is their capacity to pick briefest/best way to their goal. Subterranean insect specialists are spurred from natural ants which look for a way from their settlements to the sustenance source. At the same time they emit a substance called pheromone on ground [16] along these lines leaving a trail for different partners to take after. However this concoction vanishes with time. At first the ants begin looking through a nourishment source haphazardly, subsequently they may take after various ways to a similar source, however with entry of time, thickness of pheromone on the most brief way increment and therefore all devotee ants begin following that way bringing about increment of pheromone thickness significantly further. An engaging property of ants is that they move from source to goal for gathering wanted data or playing out an errand however they don't really return to their source rather they demolish themselves at the goal just in this manner lessening pointless activity on the network. Since load balancing in CC would require looking for under loaded servers and resources, subterranean insect operators suit the reason and satisfy it properly without putting extra weight on network. Depiction of different specialists conveyed in IMLB is as per the following:

Load Agent (LA): It controls data strategy and keeps up all detail of a server farm. The significant work of a load operator is to compute the load on each accessible virtual machine after allotment of another activity in the server farm.

VM_Load_Fitness table: It is utilized for keeping up record of particulars of every virtual machine of a server farm. It contains virtual machine id, status of its memory devoured alongside CPU use, wellness esteem and load status of all VMs.

Channel Agent (CA): It controls the exchange strategy, determination arrangement and area approach. On accepting the demand from load agent, the channel specialist will start some movement operators to other server farms for looking through the virtual machines having comparative arrangement. It additionally keeps the record of all messages got from these operator's response table.

Migration Agent (MA): These agents are started by channel agent. It will move to other server farms and speak with load agent of that server farm to enquire the status of VMs display there, searching for the coveted setup. On getting the required data it impart the same to its parent channel agent. A while later, it will remain at goal area, sitting tight for self-annihilate message from parent CA channel agent. The status of movement operator might be live or demolished based on its relevance.

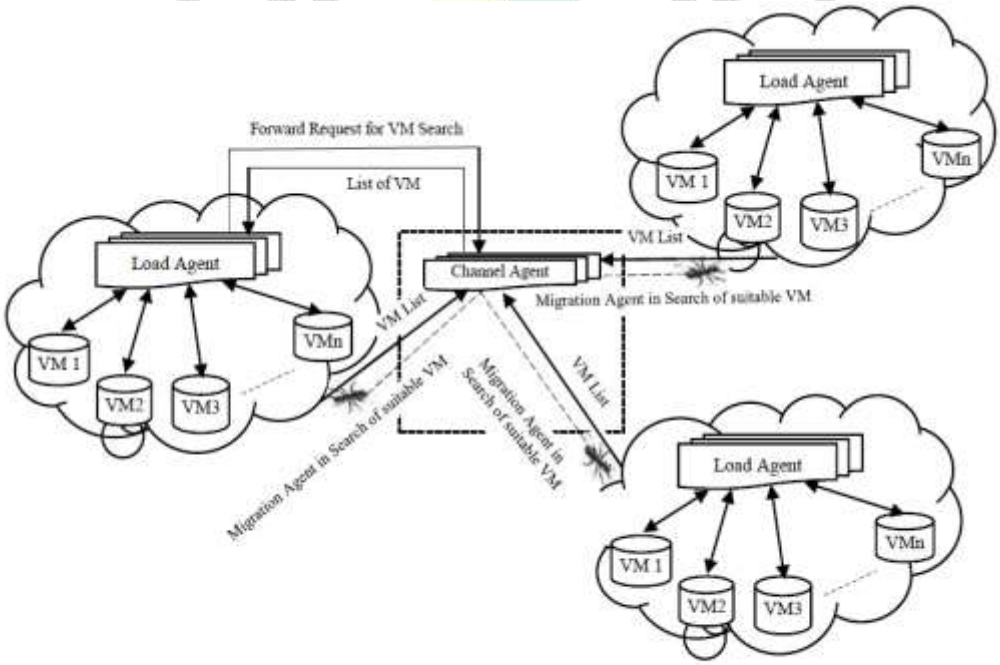

Fig.1: High level view of IMLB

Load agent acts proactively to calculate load status of different VMs accessible in a DC. Intermittently it decides the workload of virtual machines as far as accessible memory, accessible CPU use, and expected reaction time. A while later it figures the wellness estimation of each virtual machine which straightforwardly relative to the memory of a machine and can be assessed by condition 1, 2 and 3:

$$\mu \text{ available} = \mu \text{ total} - \mu \text{ used} \quad \text{.....1}$$

$$\gamma\,(\%) = \frac{\mu \text{ available}}{\mu \text{ total}} \times 100 \quad \text{.....2}$$

The percentage of fitness values will gives the status of a virtual machine.

$$\gamma = \left\{ \begin{array}{l} \leq 25\% \text{ critical allocation} \\ > 25\% \text{ normal allocation} \end{array} \right\} \quad \text{.....3}$$

Presently at whatever point a request arrives in a server farm, subsequent to distributing resources to it load agent will refresh VM_Load_Fitness table to reflect exhibit status of all VMs. For this load agent computes level of μ and ʎ since these components influence handling of approaching requests. Based on estimation of μ accessible, wellness esteem (ϑ) for every hub is created. For whatever length of time that ϑ of a hub is more noteworthy than a limit (25%), for this situation VM status is ordinary. As and when wellness estimation of a VM turns out to be not exactly or equivalent to limit esteem, load balancing should be performed. Load agent on watching basic status of a VM will suggest and send the detail of that VM to the channel agent. At that point channel agent will start the relocation agents to other server farms for looking through the virtual machines having comparable determinations. Movement agents being ants will travel one way. On achieving a goal server farm, movement agent will initially send an affirmation message to its parent channel agent. A while later it will check with load agent of that server farm for accessibility of virtual machines having comparable design as wanted. On the off chance that no such VM exists at that server farm, relocation agent sends a <Not-Applicable> message back to its parent station agent and sits tight for <self_destroy> direction from it. In any case, on the off chance that at least one VMs having wanted design are discovered, relocation agent additionally checks their μ and ϑ sends it to channel agent.

On getting another request in the server farm, the load agent will delineate particular with the accessible virtual machines. In the event that the wellness estimation of a VM is typical, load agent continues future for assignment generally load agent will call channel agent for point of view server farms having VMs with comparable arrangement for load balancing. As of now, channel agent examines reaction analysis table and discovers <MAid, DCid,,VMid>for coordinating the request. In the event that more than one reasonable record is discovered, it picks the record with biggest. Channel agent at that point speaks with comparing movement agent to affirm current ϑ of VM under thought. On getting reaction from movement agent, channel agent again investigations all appropriate VMs, if still same VM has most astounding ϑ , it passes that record to load agent for additionally load balancing, generally channel agent again speak with relocation agent for new reasonable VM. At that point channel agent would send this data to load agent for additionally preparing.

Algorithm of Load Agent

Load_ Agent( ):

Input: Receive request from user;

Output: Allocate_ resources_with_IMLB;

Case I:

```
{
If (VM_Load_Table==empty( ) )
Then allocate_requested_resources;
Maintain_VM_Load_Table;
μ available =μ total - μ used
μ available
ϑ (%) = ————————— × 100
μ total


If (ϑ>25) then
{
allocation_status:=Normal;
}
Else
{
allocation_status:=Critical;
initiateChannel_Agent(VMinitial);
}
```

Case II:

```
If (VM_Load_Table≠ empty)
Scan VM_Load_Table;
If(Load_Status(VMi) ==Critical)
```

{
Call Channel_Agent(VMLoad_Balance);
Receive <DCid,VMidforload_ transfer>;
Transfer_request to DCid;
}
Else
Allocate_request to VMid;
Update VM_Load_Table;
}

From that above algorithm, the average response time is 97ms in case I and 113ms in case II.So it is clear that IMLB takes optimum time when virtual machine becomes overloaded.It is revealed that IMLB algorithm provides desired results.

Conclusion

This work concentrates on load balancing in distributed computing condition. Load balancing in distributed computing has been overlooked, yet quick development in number of cloud clients has raised interest for load balancing systems. This work has proposed an independent agent based load balancing component which gives dynamic load balancing to cloud condition. Significant commitment of this system is proactive load count of VM in a DC and at whatever point load of a VM comes to close limit esteem, load agent starts scan for a competitor VM from different DCs. Keeping data of applicant VM in advance, decreases benefit time. Result acquired through execution of this algorithm works attractively.

References

[1] Bhaskar. R, Deepu. S.R and Dr. B.S. Shylaja (2012, September). Dynamic Allocation Method for Efficient Load Balancing in Virtual Machines for Cloud Computing Environment. Advanced Computing: An International Journal (ACIJ), 3(5), pp. 53-61.

[2] Dr. PK Sinha, SR Dhore,(2010),Multi-Agent Optimized Load balancing Using Spanning Tree for Mobile Services, International Journal Of Computer Application , 1(6).

[3] G. Xu, J. Pang, X. Fu. (2013, Feb). A Load Balancing Model Based on Cloud Partitioning for the Public Cloud. Tsinghua Science and Technology.[Online].*18(1)*,pp. 34-39.

[4] J. Hu, J. Gu, G. Sun, T. Zhao. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment in *Proc. PAAP*, 2010, pp. 89-96.

[5] K. B. Mahieddine. An Evaluation of Load Balancing Algorithms for Distributing System Athesis of Doctor of Philosophy submitted in The University of Leeds, School of Computer Studies, October 1991.

[6] M. Amar, K. Anurag, K. Rakesh, K. Rupesh, Y. Prashant (2011). SLA Driven Load Balancing For Web Applications in Cloud Computing Environment, Information and Knowledge Management, 1(1), pp. 5-13.

[7] M.Randles,D.Lamb,AT.Bendiab. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing. In *Proc. ICAINAW*, 2010, pp.551-556.

[8] R. Ezumalai,G. Aghila, R. Rajalakshmi,(2010, Feb). Design and Architecture for Efficient Load balancing with Security Using Mobile Agents.International Journal of Engineering &Technology(IACSIT). [Online]. 2(1), pp. 149-160.

[9] S Jing and K She (2011 April). A Novel Model for Load Balancing in Cloud Data Centre. Journal of Convergence Information Technology. 6(4),pp. 29-38.

[10] S. Ray and A.D. Sarkar (2012, October). Execution Analysis of Load Balancing Algorithms in Cloud Computing Environment. International Journal on Cloud Computing: Services and Architecture (IJCCSA). 2(5), pp. 1-13.

[11] S. S. Moharana, R. D. Ramesh &D.Powar,(2013, May). Analysis of Load Balancers In Cloud Computing. International Journal of Computer Science & Engineering (IJCSE). [Online]. 2(2), pp.: 101-108.

[12] S.C. Wang,K.Q. Yan, W.P.Liao, S.S. Wang. Towards a Load Balancing in a three-Level Cloud Computing Network. In *Proc. ICCSIT*, 2010, pp.108-113.

[13] T. Desai, J. Prajapati,(2013, Nov). A Survey of Various Load Balancing Techniques And Challenges In Cloud Computing. International Journal of Scientific & Technology Research, [Online]. 2(11), pp.158-161.

[14] Y.Xu, L. Wu, L. Guo, Z.Chen, L.Yang, Z.Shi. An Intelligent Load Balancing Algorithms Towards Efficient Cloud Computing. In *Proc. AAAI Workshop*, 2011, pp. 27-32.

[15] Z Zhang and X Zhang. A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation.I*n Proc. ICIMA*, 2010, pp. 240-243.

[16] Z.Chaczko, V. Mahadevan, S.Aslanzadeh, C. Mcdermid. Availability and Load Balancing In Cloud Computing. In *Proc. ICCSM* , 2011, pp.134-140.

[17] Clark, C., Fraser, K., Hand, S., Jacob, G.H.Live migration of virtual machines. In: 2nd ACM/USENIX Symposium on Network Systems, Design and Implementation (NSDI), pp. 273–286 (2005).

[18] Osman, S., Subhraveti, D., Su, G., Nieh, J. The design and implementation of ZAP: a system for migrating computing environments. ACM SIGOPS Oper. Syst. Rev. **36**(SI), 361–376 (2002).

[19] Nakai, A., Madeira, E., Buzato, L.E. Improving the QoS of web services via client-based load distribution. In: Proceedings of the 29thBrazilian Symposium on Computer Networksand Distributed Systems (SBRC2011) (2011).

[20] Cardellini, V., Colajanni, M., Yu, P.S. Request redirection algorithms for distributed web systems. IEEE Trans. Parallel Distrib. Syst. **14**(4), 355–368 (2003).

[21] A Keren and A Barak (2013 January). Opportunity Cost Algorithms for Reduction of I/O and Interposes Communication Overhead. In a Computing Cluster. *IEEE Trans.* 14 (1), pp. 399-446.

[22] Kalaivanan, M., and K. Vengatesan. "Recommendation system based on statistical analysis of ranking from user." International Conference on Information Communication and Embedded Systems (ICICES), , pp. 479-484. IEEE, 2013.

[23] P. Sanjeevikumar Vengatesan K, R. P. Singh, S. B. Mahajan," Statistical Analysis of Gene Expression Data Using Biclustering Coherent Column", International Journal of Pure and Applied Mathematics, Volume 114, Issue 9, Pages 447-454