Dynamic Load Balancing Using Gossip Protocol Including Automated Negotiation

Ms.N.Gowthami[1], Ms.V.Loganayagi[2], Ms.S.Gowri Saranya[3]
[1,2]*Final M.E CSE, DhanalakshmiSrinivasan Engineering College, Perambalur*
[3]*Assistant Professor, Department of CSE, DhanalakshmiSrinivasan Engineering College, Perambalur*
[1]gowthamicse.info@gmail.com
[2]logu.vadivel@gmail.com
[3]gowri.dsec@gmail.com

Abstract: A decentralized design whereby the component of the middleware run on every processing node of the cloud environment. At present, if the client is requesting the service to the cloud service provider, the resource is allocated only if the requested resource currently available in it. Otherwise it must wait, hear the gossip protocol used to spread the status information from the server to the cloud tenant. But the waiting time exceeds, traffic increases and also the customer satisfaction get lost. At present the gossip protocol is used for the request forwarding and using dynamic load balancing algorithm to dynamically balance the load to treat all the machines equally and the PTN mechanisms are used to increase the customer satisfaction. Because the resource allocation is not successful without the customer satisfaction, for that trade off algorithm and the concession making algorithm is used. This will be efficient for the cloud providers to minimize the cost of accessing the cloud application. In the future make the resource allocations are made efficient in all kinds of bandwidth available.

Keywords— Dynamic load balancing, gossip protocols, Price and Time Slot Negotiation (PTN), concession algorithm, tradeoff algorithm, cloud computing.

I. INTRODUCTION

The Cloud service provider owns and administers the physical infrastructure, on which the cloud services are provided. The site owners provide the service to their respective users via that are hosted by the cloud service provider. Cloud tenant running a collection of the virtual applications that are hosted on the cloud infrastructure, the services are provided to the end users through the public internet.

Dynamically balancing the load with respect to the sites in the large cloud environment and the effective adoptionof the load changes for the fair resource allocation [1]. The gossip protocol is used for exchanging the request with the randomly selected nodes. It can gossip the request quickly than the original request reaches the cloud service provider. So that the service provider can able to respond faster to the user using the cloud service Dynamic load balancer is used to dynamically balance the load.

PTN mechanisms are used for providing the service level agreement (SLA) between the cloud service provider and the user. By proving the negotiation, the customer satisfaction gets increased. An SLA is a guarantee that defines the set of quality of service (QoS) constraints such as price or time constraints. The

price and time has inverse relationship both the price and time slot has to be negotiated simultaneously.

II. Background

A. Gossip Protocol

The gossip protocol is used for spreading the information with the neighboring nodes. Thus the status of the service provider server is known to all intermediate nodes[5]. When executing a round-based gossip protocol, each node selects a subset of other nodes to interact with, whereby the selection functions are often probabilistic [11].

Nodes interact via 'small' messages, which are processed and trigger local state changes[10].Node interaction with gossip protocol follows the so-called push-pull paradigm, whereby two nodes exchange state information, process this information and update their local states during a round [3],[9]. Compared to alternative distributed solutions, gossip-based protocols tend to be simpler, more scalable and more robust.

B. CYCLON Protocol

Unstructured overlay form an important class of peer to peer networks. The construction of the overlays, which is essentially a membership managementInexpensive membership management while retaining random graph properties[8].

III. Proposed Work

A. Dynamic Load Balancing

Load Balancing is an essential technique for spreading out the total workload across the available processor of the cloud tenant to achieve better performance of the distributed computation with dynamic load balancing the work load may migrate from one processor to another during the run time [2]. The dimension exchange method is used for distributed load balancing. The splitting of workload between a pair of processor is called the dimension exchange.

Algorithm 1 for Dynamic Load Balancing

Ni: List of primary nodes
SNj: List of supporting nodes
PI: Priority index maintained by supporting node
Pk: List of processes in Process Queue
i, j, k ϵ N, j<i // N← Natural Number
Initially: 1. Assuming each node is having some load,
2. Some supporting nodes may have loaded
Ni ← Load, Sj← Load // load defines some process is there
Procedure: Main ( )
{
I. Suppose a node Nt is heavily loaded with load ξ where 0< t< i , ξ ϵ Pk
// Two situations are possible as in step II.
II. If (Search_node) // Search_node will find
// out whether a light weighted primary node is
//available, if primary node is available it will give
//index of available node.
{
Available_Node ← ξ // Overload is assigned
//to Available node given by Seach_node ()
Load (Nt) =Load (Nt) – ξ
}
Else
{
Call Seacrh_S_node ( ) // Search_node will
//find out a light weighted or minimum
//weighted supporting node with index as
//Available_S_node.
Available_s_node ← ξ
Load (Nt) =Load (Nt) – ξ
IN_S (Available_s_node, ξ)
}}
Procedure: IN_S (SN_node, ξ)
{
I. Priority is assigned to SN_node as PI
(SN_node) = t
II. If t > PI (RP) // RP is the currently
//process on Selected supporting node.
{
// P_List is list of pending process shorted
//according to priority.

```
P_List ← RP // RP is added to Pending
//List
Now RP ← ξ // ξ is now allotted to
//the Selected Supporting node.
}
Else
P_List ← ξ // ξ is added to Pending
//List
}.}
Procedure: Search _node ()
{
I. for each Ni except node initiating the search_
node procedure
{
Check the node with minimum load
//Minimum load includes the number of
//processes as well as structure or configuration
//of node and node should be able to accept
//node.
}
II. If Desired Node available
{
return (index of available node) // index
//defines the property to identify the Node
}
}
Procedure: Search _S_node ()
{
I. for each SNj except node initiating the search
node procedure
{
Check the Supporting node with minimum
load
// Minimum load includes the number of
//processes as well as structure or
//configuration of node
}
II. return (index of available Supporting node)
// index defines the property to identify the
// Node
}
```

B. Concession Making Algorithm

The algorithm determines the amount of concession total for each negotiation round, which corresponds to the reduction in the agent expected total utility. The agent in this work adopt the time dependent strategies is to determine the amount of concession required for the next proposal.

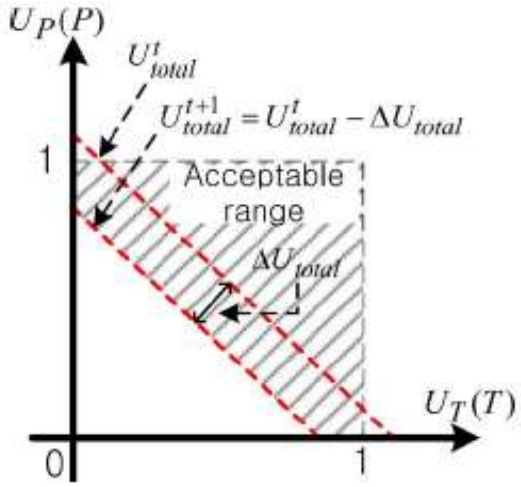

Fig 1Graph for Concession making algorithm

The concession making algorithm is for an agent. In the graph, the total utility line moves to 0 with a concession. Let t, τ , and λ be the negotiation round, the negotiation deadline, and negotiation strategy, respectively[6]. The negotiation agent determines the amount of concession ΔUtotal, and also determines its expected utility in the next round[2], [4].

$$\Delta U_{\text{total}} = U^t_{\text{total}} \cdot \left(\frac{t}{\tau}\right)^{\lambda}$$

$$U^{t+1}_{\text{total}} = U^t_{\text{total}} - \Delta U_{\text{total}}.$$

C. Tradeoff Algorithm

The tradeoff algorithm uses the burst mode proposal to increase the negotiation speed. The grants are allowed to concurrently make multiple proposals. With each proposal consisting of different pairs of price and time slot that generating the same aggregated utility [7].The concurrent proposals differ from each other

only in terms of individual price and time slot utilities [5].

IV. Experiments And Discussion

The experiment includes the following such as cloud server, manager, cloud consumer, the agent and the intermediate nodes. The allocation of the resource without any proper load balancing technique is shown in the fig 2

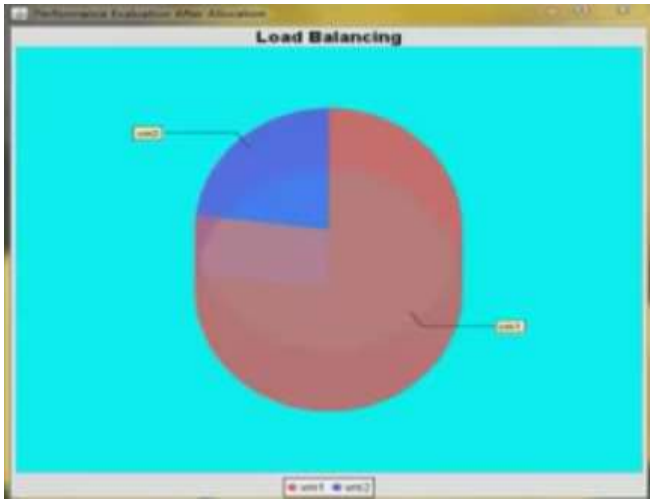

Fig 2 Allocation graph without load balancing

In the Fig 3 the resource is balanced between the two virtual machines. Thus the both machines can able to utilize the cloud resource efficiently. Thus all the machines in the cloud are treated equally. The dynamic load balancing algorithm is used to balance the load.

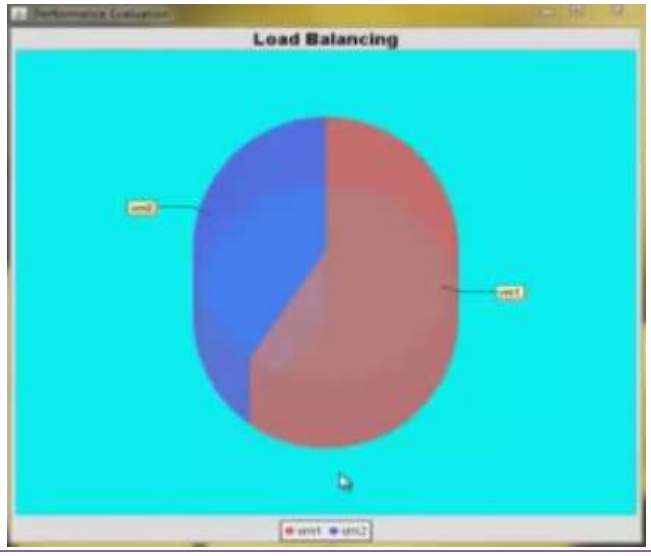

Fig 3 Allocation graph after dynamic load balancing

The automated negotiation is included for providing some concession, discount and reduction for improving the customer satisfaction. This is an agreement between the consumer and the provider. The agreement must be beneficial to both the provider and consumer; this is shown in the Fig 4

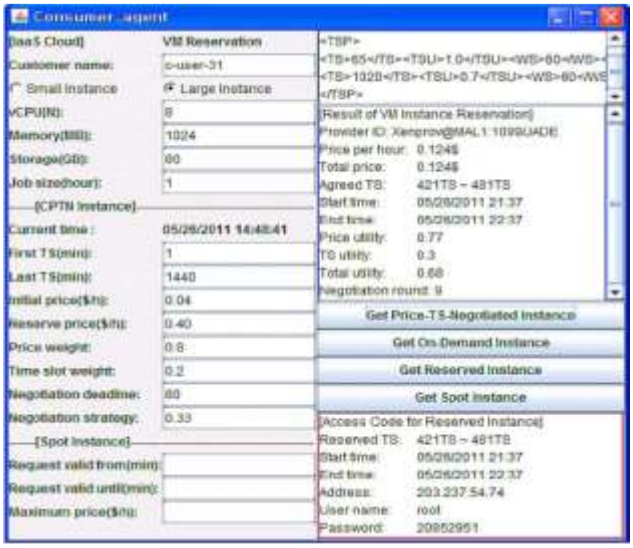

Fig 4 Outputof Automated Negotiation

V. Conclusion And Future Enhancement

The cloud service provider allocates the resource to the client if the client made the request through the public internet. The efficient adaptation to the load changes and scalability of the middleware of the middleware layer in terms of both the number of machines in the cloud is identified. Every cloud tenant is treated equally so that the load is distributed among all consumers.

The efficiency of resource allocation is based on customer satisfaction. PTN mechanisms are used for providing the service level agreement between the cloud service provider and the user. By providing the negotiation, the customer satisfaction gets increased.

References

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," *in Proc. 1999 Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. 126

[2] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *J.Parallel and Distrib. Computing*, vol. 7, no. 2, pp. 279–301, 1989.

[3] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguade, "Utility based placement of dynamic web applications with fairness goals," *in 2008 IEEE Network Operations and Management Symposium*.

[4] FetahiWuhib, Rolf Stadler, and Mike Spreitzer, " A Gossip Protocol for Dynamic Resource Management in Large Cloud Environment " *IEEE Transactions on Network and Service Management*, vol. 9, no. 2, June 2012.

[5] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J.Applied Mathematics*, vol. 17, no. 2, pp. pp. 416–429, 1969.

[6] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. Computer Syst.*, vol. 23, no. 3,pp. 219–252, 2005.

[7] E. Loureiro, P. Nixon, and S. Dobson, "Decentralized utility maximization for adaptive management of shared resource pools," *in 2009 International Conference on Intelligent Networking and Collaborative Systems*.

[8] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," *in 2007 International Conference on World Wide Web*.

[9] S. Voulgaris, D. Gavidia, and M. van Steen, "CYCLON: inexpensive membership management for unstructured p2p overlays," *J. Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.

[10] F. Wuhib, M. Dam, R. Stadler, and A. Clem, "Robust monitoring of network-wide aggregates through gossiping," *IEEE Trans. Network and Service Management*, vol. 6, no. 2, pp. 95–109, June 2009.

[11] F. Wuhib, M. Dam, and R. Stadler, "A gossiping protocol for detecting global threshold crossings," *IEEE Trans. Network and Service Management*, vol. 7, no. 1, pp. 42–57, Mar. 2010.