ISSN:0975-9646

Dynamic Load Balancing Mechanism In Multiservice Cloud Storage

Karishma B. Badgujar, Prof. Pravin R. Patil
Department of Computer Engineering
Pune Institute of Computer Technology
Pune, India

Abstract— Today's world, Internet is most popular technology. Everyday about 400 billion peoples introduced with internet. Traffic is growing continuously over internet. Popularity of internet is lies in web applications. So it is necessary to manage large traffic over internet. All data of internet is being shared on cloud. The cloud can store large amount of data. But the data redundancy and duplication may cause the misbalance of memory space of the cloud. The system administrator cannot give assurance of the every single node which is being participated inside the process of data integration on cloud. To avoid the load of duplicate data on the cloud we developed novel data center architecture: INS (Index Name Server). It helps to improve the performance of the cloud system.

Keywords—INS, P2P, Cloud system, Load Balance.

I. INTRODUCTION

Cloud Computing is most emerging technology in Networking. Cloud is nothing but pool of virtualized computer resources. Cloud is an internet-based development where virtualized and dynamically scalable resources are provided as a Service. Though Cloud Computing has glorious features there are certain crucial issues which are needed to be resolved. One of them is load balancing. In existing systems static approach is used. Due to static approach quality of utilization of resources can't be achieve. The system administrator no longer can give assurance of the optimal status of each node in the cloud system, which might responsible to integration bottleneck and wastage of resource. Due to this the flexibility and utility of cloud storage system is no longer remain because the system keeps handling duplicate and redundant data. Here we use the index database to find out the various sources of user demand, and analyse environmental quality monitoring parameter and level of busy parameter to distribute the data and to achieve the load balancing system. Thus, while transmitting prior data, our proposed scheme can avoid the network congestion which occurs due to duplicate data and less waiting time at the same node inside the cloud.

II. LITERATURE SURVEY

Load balancing is process of dividing workload among nodes to increase the response time. Load balancing is process of removing dependency from single node and to adding resources in such way workload can be distributed over them. Cloud computing is most emerging technology. Cloud is nothing but pool of virtualized resources which provides on demand services. In the business model using software as a service, users are provided access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software". Cloud Computing is to effective as it can give high availability and scalability. Effectiveness of cloud is lies in load balancing. Load balancing is most complex task in Cloud Computing. There are number techniques to achieve load balancing.

For any load balancing algorithm, it is very important to analyze the traffic flow in real-time scenarios over different geographic regions, and then balance the overall workload accordingly. All regions over the globe have a different time zone and have certain peak hours during which the network load is supposed to be at its peak. Therefore, load balancer must be capable of handling the traffic in peak hours in every location so as to achieve maximum resource utilization and throughput.

III. EXISTING SYSTEM

A. Load Balancing Approach with Centralized Load Balancer and Two Back-end Servers

In "Applying Load Balancing: A Dynamic Approach", there is a design with one Load Balancer communicating with all the nodes and monitoring their load. This load balancer reports the load to two back-end servers.

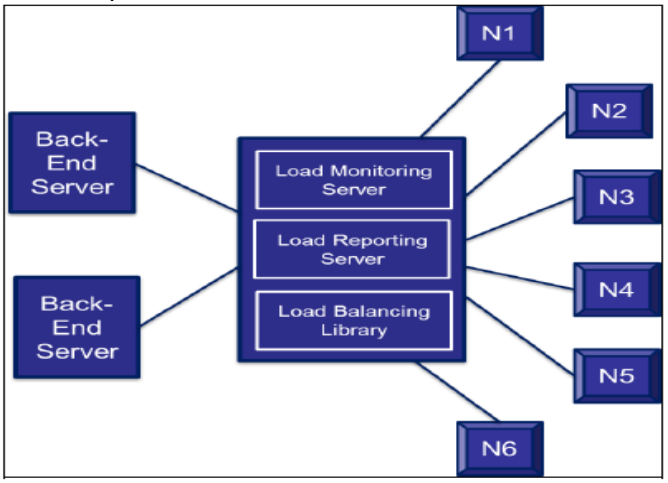

Fig 1: Load balancing in dynamic approach.

The servers finally make the balancing decision and return the address of the suitable node to which the overload

should be transmitted. Additional servers are used for the sake of reliability. If one server fails, the other can take its job and the system continues to work properly. This design is illustrated in the Fig.1. The central load balancer has three parts: Load monitoring server, Load reporting server and load balancing library while the two back-end servers only comprise of Load monitoring server and Load reporting server. The Load Reporting Server is used to collect the machine load information on which it is running. The collected data is sent to Load Monitoring Server which is located on same machine. Load Monitoring Server stores the collected data in data structure. Then an all-to-all broadcast of the load information is carried out. Then, at the central load balancer, when request comes, the Load Balancing Library finds least loaded machine and return the address of that machine.

The drawback of the design is the huge communication overhead involved as it follows a global strategy of load balancing. Secondly the cost of this all-to-all broadcast of load information is high[1].

B. *A load balancing approach using one supporting node with each primary node and using a priority scheme to schedule tasks at supporting nodes*

Let us now have a look on another approach it uses one supporting node (denoted as SNi) with each primary node (denoted as Ni) as depicted in Fig. 5. In case of overload at node Ni, an interrupt service routine generates an interrupt and the overload is transferred to its supporting node and it also uses a priority scheme, if the priority of the incoming process at the supporting node is greater than that of the currently running process, then the current process is interrupted and assigned to a waiting queue and the incoming process is allowed to run at the supporting node. Otherwise the current process continues and incoming process is in waiting state until the current process is completed.

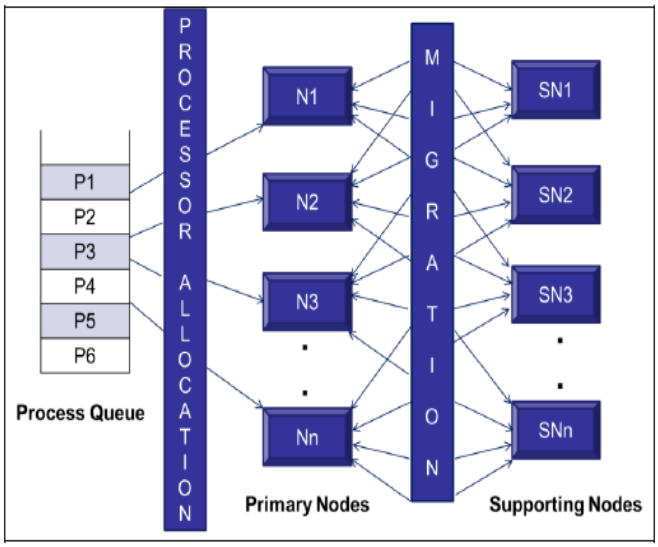

Fig 2: Load Balancing in Task Supporting Node Approach.

This approach has a drawback of its complexity and the cost of such a huge infrastructure. The priority scheme makes it more dynamic and suitable for distributed systems as well as handling real time tasks[2].

C. *Efficient Load Balancing in Cloud Computing using Fuzzy Logic*

This paper designed a load balancing algorithm based on round robin in Virtual Machine (VM) environment of cloud computing in order to achieve better response time and processing time. The load balancing algorithm is done before it reaches the processing servers the job is scheduled based on various parameters like processor speed and assigned load of Virtual Machine (VM) and etc. It maintains the information in each VM and numbers of request currently allocated to VM of the system. It identify the least loaded machine, when a request come to allocate and it identified the first one if there are more than one least loaded machine.

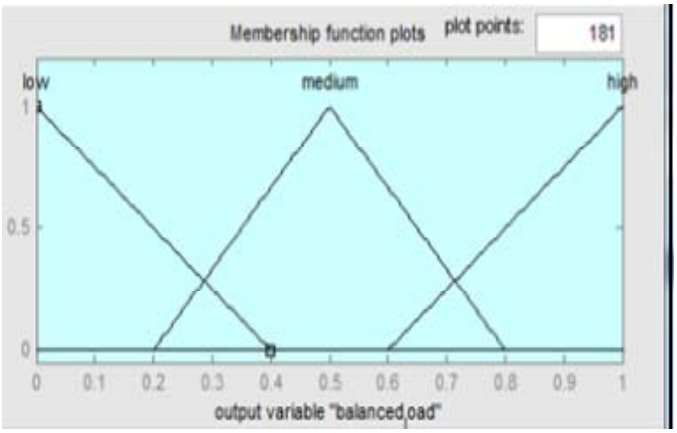

Fig 3: Membership Output Function of Fuzzy Logic Balanced Load

This system uses following algorithm:

```
Begin
    Connect_to_resources()()
    L1
    If(resource found)
        Begin
            Calculate connection_string()
            Select fuzzy_connection()
            Return resource to requester
        End
    Else
        Begin
            If(Anymore resource available)
              Choose_next_resource()
              Go to L1
           Else
        Exit
    End
End
```

They tried to implement the new load balancing technique based on Fuzzy logic. Where the fuzzy logic is natural like language through which one can formulate their problem. The advantages of fuzzy logic are easy to understand, flexible, tolerant of imprecise data and can model nonlinear

functions of arbitrary complexity, and is used to approximate functions and can be used to model any continuous function. Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic and the mapping provides a basis from which decisions can be made, or patterns recognized [3].

D. Optimal Load-Balancing

This paper is about load-balancing packets across multiple paths inside a switch, or across a network. It is motivated by the recent interest in load-balanced switches. Load-balanced switches provide an appealing alternative to crossbars with centralized schedulers. A load-balanced switch has no scheduler, is particularly amenable to optics, and – most relevant here – guarantees 100% throughput. A uniform mesh is used to load balance packets uniformly across all 2-hop paths in the switch.

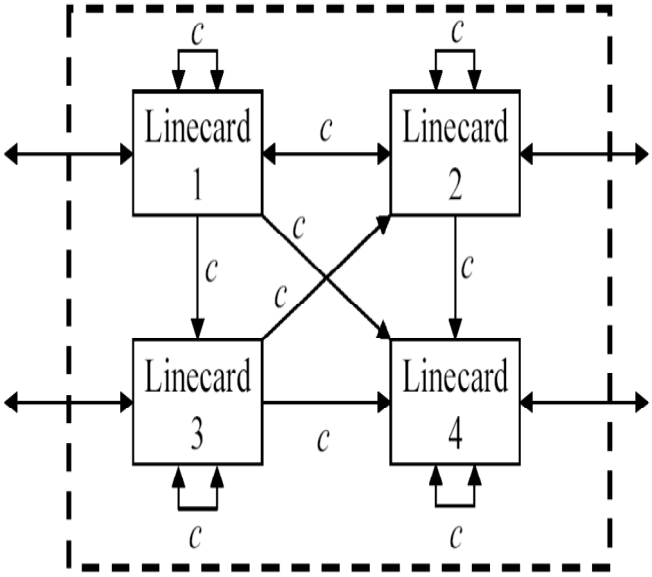

Fig 4: Optimal Load Balancing.

In this paper we explore whether this particular method of load balancing is optimal in the sense that it achieves the highest throughput for a given capacity of interconnect. The method we use allows the load-balanced switch to be compared with ring, torus and hypercube interconnects, too. We prove that for a given interconnect capacity, the load-balancing mesh has the maximum throughput. Perhaps surprisingly, we find that the best mesh is slightly non-uniform, or biased, and has a throughput of N/(2N − 1), where N is the number of nodes[11].

E. Structured Peer-to-Peer Systems Using Load Balancing with Imperfect Information

This paper, assume that the entire hash space provided by a DHT is [0, 1], and each virtual server in the DHT has a unique ID selected independently and uniformly at random from the space [0, 1]. Let N be the set of participating peers, and V be the set of virtual servers hosted by the peers in N in the DHT. Denote the set of virtual servers in peer i by Vi. Each peer i Є N in our proposal estimates the load, which is denoted by Ti, that it should perceive, where Ti is an estimation for the expected load per unit capacity, and Є is a predefined system parameter. If the current total load of i is greater than Ti (i.e., i is overloaded), then i migrates some of its virtual servers to other peers. Otherwise, i is under-loaded, which does nothing but waits to receive the migrated virtual servers. For an overloaded peer (e.g., peer i), i picks those virtual servers for migration, such that 1) i becomes under-loaded, and 2) the total movement cost, MC, in (2) is minimized due to the reallocation. If i is an under-loaded peer, then i may be requested to receive a migrated virtual server, and i accepts such a virtual server if the added load due to the virtual server will not overload itself; otherwise, i rejects such virtual server.

Algorithm: REALLOCATION (i) Peer i computes the reallocation of its local virtual servers, where Load (i) $= \sum_{v \in V_i} L_v$

$A = \sum_{v \in V_i} L_v / \sum_{i \in N} C_i$

1: $A \leftarrow \text{Ln } n. \frac{\int_{y=0}^{C_{max}} (yF_Y(y)\,dy)}{\int_{x=0}^{C_{max}} (xF_x(x)\,dx)}$;
2: $T_i \leftarrow A \times C_i + Є$;
3: switch Load(i) do
4: case > T_i
5: $U_i \leftarrow \varphi$;
6: while Load(i) > T_i and $V_i \neq U_i$ do
7: $v \leftarrow \arg\min\{L_v | v Є V_i - U_i\}$;
8: find j Є I satisfying Eq.(4) to accommodate v;
9: if j accepts v then
10: $V_i \leftarrow V_i - \{v\}$;
11: $Ui \leftarrow U_i \cup \{v\}$;
12: break;
13:case ≤ T_i
14: while Load(i) < T_i do
15: receive v to host;
16: $V_i \leftarrow V_i \cup \{v\}$;
17: break;

F. Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems

This paper presents an efficient, proximity-aware load balancing scheme by using the concept of virtual servers. It contents 1) Relying on a self-organized, fully distributed key tree structure constructed on top of a DHT, load balance is achieved by aligning those two skews in load distribution and node capacity inherent in P2P systems—that is, have higher capacity nodes carry more loads; 2) proximity information is used to guide virtual server reassignments such that virtual servers are reassigned and transferred between physically close heavily loaded nodes and lightly loaded nodes, thereby minimizing the load movement cost and allowing load balancing to perform efficiently; and 3) our simulations show that our proximity-aware load balancing scheme reduces the load movement cost by 11-65 percent for all the combinations of two representative network topologies, two node capacity profiles, and two load distributions of virtual servers.

It consist of three algorithms, named as,

1) Check a KT node algorithm

```
Procedure check_KT_node(KT-node X)

1: if (X.region ⊆ the responsible region of X.host) then
2:      delete_KT_children(X)
3: else
4:      add_KT_children(X)
5: end if

Procedure delete_KT_children(KT_node X)

1: for i=1 to k do
2: if (X.child[i]≠NULL) then
3:      delete X.child[i]
4:      X.child[i]=NULL
5: end if
6: end for

Procedure add_KT_children(KT_node X)

1: for i=1 to k do
2: if (X.child[i]==NULL && X.child[i]→region ⊄ the responsible region of X.host) then
3:      c = new KT_node
4:      c→region = the i^th fraction of X.region
5:      X.child[i] = c
6:      c→parent = X
7:      plant_KT_node(c)
8: end if
9: end for
```

2) LBI aggregation algorithm

```
Procedure KT_node_report_LBI(KT_node X)

1: if (X is a KT leaf node) then
2:      <Lx, Cx, Lx,min> ← receive <Li, Ci, Li,min> from X.host
3: else
4:      Receive <Li, Ci, Li,min>s from k children /* i=1,…,k*/
5:      Lx ← ∑_{i=1}^{k} Li
6:      Cx ← ∑_{i=1}^{k} Ci
7:      Lx, min ← the smallest Li,min
8: end if
9: if (X is not a KT root node) then
10:     Report <Lx, Cx, Lx,min> to X.parent /* report to the parent node*/
11: end if
```

3) Virtual server assignment algorithm

```
Procedure KT_node_VSA(KT_node X)

1: VSA.pool ← φ
2: if (X is a KT leaf node) then
3:      VSA.pool ← VSA information from X.host /* receive the VSA information from its hosting virtual server */
4: else
5:      VSA.pool ← VSA information from k children
6: end if
7: if (VSA.pool.size ≥ pairing_threshold || X is a KT root node) then
8:      KT_node_rendezvous_point(X, VSA.pool) /* X serves as a rendezvous point */
9: else
10:     Report VSA.pool to X.parent /* propagate the VSA information to its parent */
11: end if


Procedure KT_node_rendezvous_point(KT_node X,

VSA information pool)
1: light_list ← remove all <ΔLj = Tj – Lj, ip_addr(j)> s from pool /* light_list maintains the VSA information of light nodes */
2: heavy_list ← remove all <li,r, vi,r, ip_addr(i)> s from pool /* heavy_list maintains the VSA information of heavy nodes */
3: pool ← φ
4: while (heavy_list ≠ φ) do
5:      Remove the most loaded virtual server vi,r from heavy_list, and assign it to a DHT node j in light_list such that ΔLj is minimized and subject to the condition that ΔLj ≥ Li,r
6: if (vi,r can be assigned) then
7:      Remove <ΔLj, ip_addr(j)> from light_list
8: if (ΔLj – Li,r ≥ Lmin) then
9:      Insert <ΔLj – Li,r, ip_addr(j) > into light_list
10: end if
11:     Send the assigned information <vi,r, ip_addr(i), ip_addr(j)> to DHT nodes I and j /*prepare for virtual server transferring */
12: else
13:     pool ← pool U {<Li,r, vi,r, ip_addr(i)>}
14: end if
15: end while
16: pool ← pool U light_list
17: if (pool.size > 0 && X is not a KT root node) then
18:     Report pool to X.parent /* report un-assigned VSA information to its parent */
19: end if
```

IV. PROPOSED SYSTEM

As we seen above the traditional data storage methods require high cost of hardware and data management. The cloud storage techniques in the cloud systems range from small sized files to large-scale commercial application. A resource-integrated heterogeneous system achieves the best storage, along with the optimal performance of the load balancing, and it also reduce the risk of losing information due to failure of storage device. The content of the cloud may be duplicated due to redundant data present on the cloud storage. This data is uploaded by different users at different time. To handle the load caused by redundant data, this paper presents cloud management architecture, named as Index Name Server (INS), this is a de-duplication with access point selection optimization techniques which improves the performance of the cloud system.

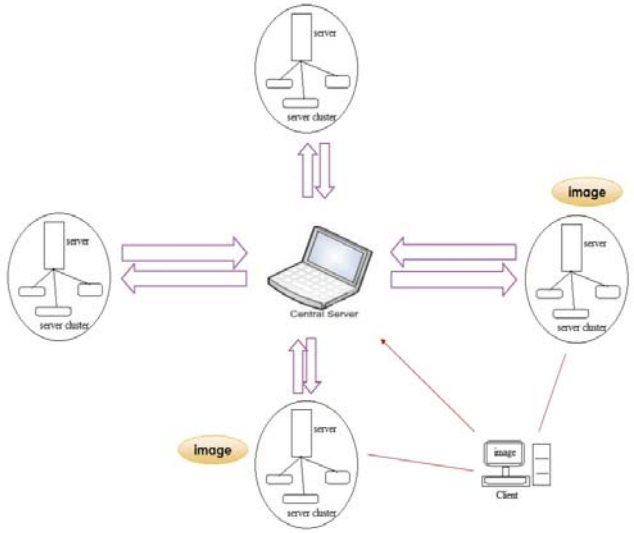

Fig 5: System architecture.

Our system consists of following methodologies.

1) Cloud balancing by primary server

When a client requests for multimedia upload and download, then the request of client is being handle by central server. This is also known as a primary server. This primary data then divide the data into various categories depending upon the request of the client. This categorisation is done by the cluster head which is present in present inside the cluster server. The cluster server is the server which handles the all task of clustering inside the cloud. The clustering is done on the basis of user requested data.

2) Proximity Weight calculation module

Main task of the central primary server is to maintain the load along the cluster server. The capacity of the cluster server is maintained and not allows to be exceeded beyond the memory limit of that server. After that consistency of data provided by various servers is checked by central server. This consistent data is provided to the client by limiting client request load. For this it calculate the network proximity values. After that we measure the latency of each server in specific landmark. Then by computing the landmark order we calculate the server utilization ratio. This ratio minimises the weight calculation and the link assessment.

3*) Load balancing*

The client request is h handle by specific cluster head. Then we consider weighted bipartite graph of client requested cluster head. After that we remove links which do not provide consistent data. Then we again go for the calculation of the network proximity value. Finally we compute latency order to balance the load.

V. CONCLUSIONS

Here we proposed a cloud management mechanism which objective of improving the accuracy in backup selection, like considering habits, data rate and formats of user. This includes statistic parameters and those may be classified based on file formats; or ignoring peak hours. We can collect the information for more efficient cloud load computing to propose various parameters for various service types. In this way, the load of the data cloud can be reduced more accurately and the performance of heavily loaded cloud storage system can be greatly improved by using this system.

REFERENCES

[1] Wenzheng Li, Hongyan Shi, "Dynamic Load Balancing Algorithm Based on FCFS"_ 2009 Fourth International Conference on Innovative Computing_ Information and Control (ICICIC), 2009, pp. 1528-1531.

[2] Sun Nian, Liang Guangmin, "Dynamic Load Balancing Algorithm for MPI Parallel Computing"_ 2009 International Conference on New Trends in Information and Service Science (NISS '09), 2009, pp. 95-99.

[3] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network"_ 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2010), vol. 1_2010_pp. 108-113.

[4] Ananth Rao_Karthik Lakshminarayanan_Sonesh Surana_Richard Karp _Ion Stoica_"Load Balancing in Structured P2P Systems"_ Lecture Notes in Computer Science_vol. 2735_2003, pp. 68-79_ DOI: 10.1007/978-3-540-45172-3_6.

[5] Y. Zhu, Y. Hu, "Efficient_Proximity-Aware Load Balancing for DHTBased P2P Systems"_IEEE Transactions on Parallel and Distributed Systems, vol. 16_Issue 4_ 2005_pp. 349-361.

[6] Jiexi Zha, Junping Wang, Renmin Han, Maoqiang Song, "Research on load balance of Service Capability Interaction Management"_2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT 2010)_2010_pp. 212-217.

[7] Ruixia Tong, Xiongfeng Zhu, "A Load Balancing Strategy Based on the Combination of static and Dynamic"_2010 2nd International Workshop on Database Technology and Applications (DBTA)_2010_pp. 1-4.

[8] Lin Xia, Han-Cong Duan, Xu Zhou, Zhifeng Zhao, Xiao-Wen Nie, "Heterogeneity and Load Balance in Structured P2P Syste" 2010 International Conference on Communications, Circuits and Systems (ICCCAS), 2010, pp. 245-248.

[9] Yonghui Zhang, Chunhong Zhang, Yang Ji, Wei Mi, "A Novel Load Balancing Scheme for DHT-BASED Server Farm"_ 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), 2010, pp. 980-984.

[10] Lu Gao, Min Peng, "Optimal Super peer Selection Based on Load Balance for P2P File-sharing System" _ 2009 International Joint Conference on Artificial Intelligence (JCAI '09), 2009_pp. 92-95.

[11] Isaac Keslassy, Cheng-shang Chang, Nick Mckeown, Duan-Shin Lee, "Optimal Load-Balancing"_Proceedings of IEEE Infocom_2005