IJDCST @October Issue- V-1, I-6, SW-41
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Load re Balancing in cloud using Signature Load Manage Management

Shaik Shakeela[1], P.Radha Krishna[2]

[1]Student, Nova College of Engineering and Technology for Women, Ibrahimpatnam, Krishna Dist, Andhra Pradesh, India

[2] Associate Professor,HOD CSE, Nova College of Engineering and Technology for Women, Ibrahimpatnam, Krishna Dist, Andhra Pradesh, India

Abstract: Due to the large infrastructure capacity of cloud providers, there is a common myth that an application can scale up unlimitedly and automatically when application demand increases. The common provisioning with fixed capacity would either result in unsatisfied customers. We are going to investigate to build a large scale web server farm in the cloud. Load balancing is the wide area that achieved by instructing users to access a particular cloud based on their location. To distribute the load across the server with equivalent client requests. Due to these assurances, In this paper we introduce Signature load management algorithm for providing load in between clients across the server. Resource monitoring is the main aspect in our proposed work because based on resources present in both hardware and software we are arranging the client operations on cloud. And also we maintain equivalent load considerations based on quality of services provided by the cloud provider.

Index Terms: IaaS, PaaS, SaaS, Cloud Computing, Load balancing, Quality-of-Service, Resource monitoring

I. INTRODUCTION

Cloud computing gets its name as a metaphor for the internet. Formally the internet can be represented in network diagram as a cloud. The cloud Icon represents "all that other Stuff" that makes networks work.

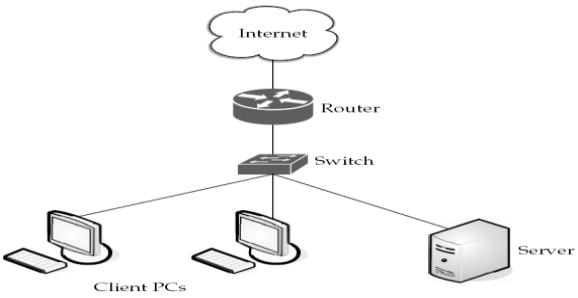

Figure 1: A cloud is used network diagrams to depict the internet.

Cloud computing promises to cut operational and capital costs and, more importantly, let IT departments focus on strategic projects instead of keeping the data center running. In commercial cloud computing applications they are providing practical unlimited infrastructure capacity on client's request. For example Amazon EC2 web services in real time oriented applications of cloud services. Amazon EC2 provides two types of services according to the client request. They are On-demand and free web services, Amazon EC2 disabled many networking layer features, such as ARP, promiscuous mode, IP spoofing, and IP multicast. Cloud architecture shows

scaling applications on cloud is more difficult due to the following several aspects.

a) First, in enterprises, application owners can choose an optimal infrastructure for their applications amongst various options from various vendors. In comparison, a cloud infrastructure is owned and maintained by the cloud providers. Because of their commodity business model, they only offer a limited set of infrastructure components.

b) Second, again due to its commodity business model, a cloud typically only provides commodity Virtual Machines (VM). The computation power and the network bandwidth is typically less than high-end servers.

c) Third, unlike in an enterprise, application owners have little or no control of the underlying cloud infrastructure. For example, for security reasons, Amazon EC2 disabled many networking layer features, such as ARP, promiscuous mode, IP spoofing, and IP multicast. Application owners have no ability to change these infrastructure features.

d) Last, commodity machines are likely to fail more frequently. Any architecture design based on cloud must handle machine failures quickly, ideally in a few milli-seconds or faster, in order not to frequently disrupt service.

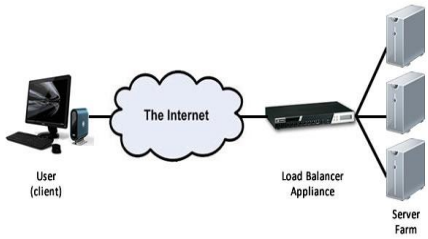

Figure 2: Load balancing in cloud using proxy servers.

Consider the above results present in cloud balancer, we are introducing Signature Load management algorithm for increasing resource service in cloud. To provide more efficient result analysis with load balancing.

II. BACK GROUND WORK

Traditional techniques for designing a scalable web server farm would not work in a cloud environment; we need to devise new techniques which leverage scalable cloud components while getting around their limitations. The load balancer assumes the ip address of the web applications all communications are hits the load balancer. The load balancer is connected to one or more identical web servers in backend. Depending on the session present in the web server they are allocating the in between cloud clients.

IJDCST @October Issue- V-1, I-6, SW-41
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

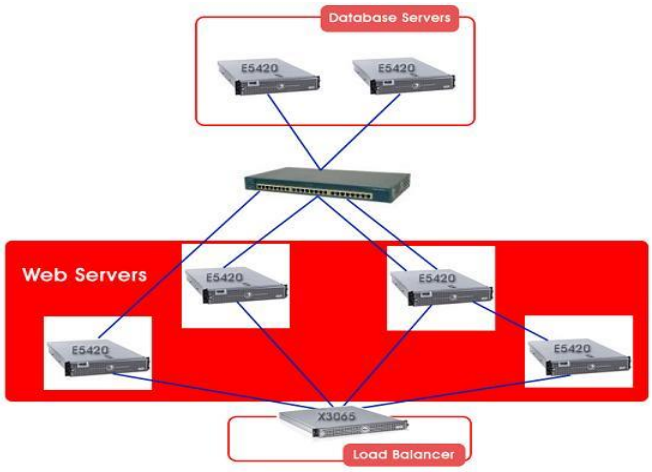

Figure 3: Hardware load balancer using clusters.

A hardware-based load balancer uses application specific hardware-based components, thus it is typically expensive. Because of cloud's commodity business model, a hardware-based load balancer is rarely offered by cloud providers as a service. Instead, one has to use software- based load balancer running on a generic server.

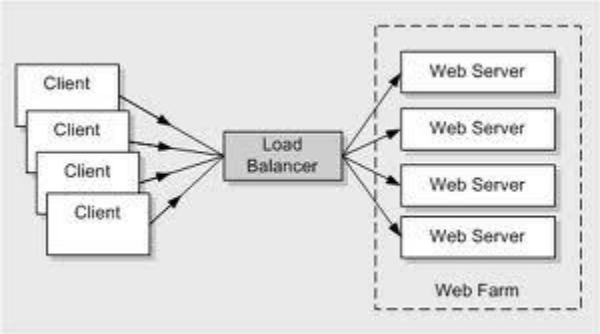

Figure 4: Software load balancer using clusters.

A software-based load balancer is not a scalable solution, though. The scalability is usually limited by the CPU and network bandwidth capacity of the generic server that the load balancer runs on and a generic server's capacity are much smaller than that of a hardware-based load balancer. We use back-end web servers for dynamic content processing, and use client-side load balancing technique to distribute the traffic across the back-end web servers.

III. PROPOSED APPROACH

Here in this load balancing task we will set a threshold value for each resource. And we run a thread to monitor the resource load. Once this load crosses its threshold value then we first gather all information about process task and then shift that task to another node's resource without disturbing running task. The Algorithm we are going to use is Signature Driven Load management for Cloud Computing Infrastructure.

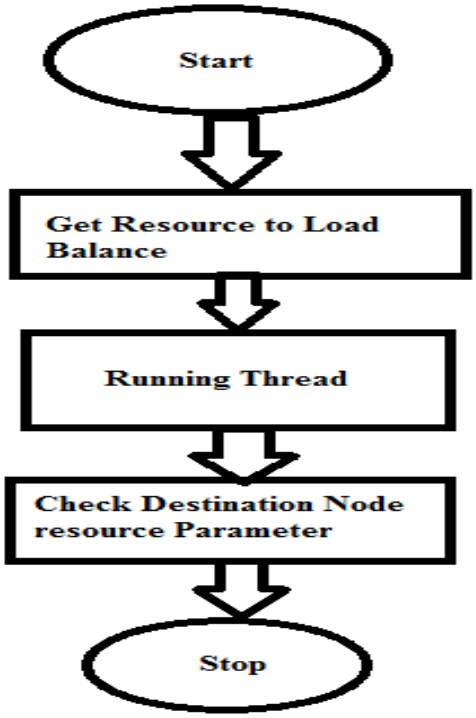

Figure 5: Signature Driven Load Management.

Threads which are ready to be submitted are checked by or load balancing algorithm along with that it also verifies the threshold value of the node as well as threshold value of the upcoming load if it satisfied the request will be forwarded to server.

IV. PERFORMANCE ANALYSIS

SigLM provides dynamic runtime load management for executing long-running data-intensive computing jobs in cloud systems. To achieve runtime load management, each cloud node needs to periodically update its multi-attribute resource signatures. SigLM performs dynamic matching between currently running tasks and existing cloud nodes based on the maintained load and resource signatures. For each newly arrived task, the system first instantiates the task on some lightly loaded node to collect the task's load

Signature.

Input:
$V = \{v_1, ..., v_n\}$: nodes in the cloud system
t_i: a task that needs to be placed in the cloud system
W: signature sliding window
f_i: pre-filtering qualifying function for resource type $r_i \in R$
DHT : P2P signature lookup system
UpdateResourceSignature(V , |W|, DHT)
1. **for** every signature window |W| **do**
2. **for** each node v_i in V **do**
3. **for** each resource attribute r_k in R **do**
4. Construct the resource signature Sig_k
5. Construct the index MBRs for Sig_k
6. Insert the MBRs into R-trees
7. Push Sig_k and its index into DHT

Algorithm 1: Update resource Signature.

MatchTaskSignature(t_i, V , DHT)
1. Construct the MBRs for the load signature Sig_L of t
2. Send load matching request to DHT nodes
3. **for** each DHT node **do**
4. **for** each cloud node resource signature Sig_R **do**
4. flag = TRUE;
5. **for** each resource type Sig_n **do**
6. flag = flag ∧ $f_i(Sig_i, Sig_n)$ /* MBR matching*/
7. if (flag == TRUE)
8. insert the cloud node signature into a DTW list
9. return the DTW list to the initiating node for t_i
4. merge DTW lists received from all DHT nodes
7. **for** every node resource in the DTW List **do**
8. Invoke DTW algorithm to get a matching score

9. Sort the DTW List based on

10. **for** every node v_j in the sorted DTW list **do**
11. Invoke admission control func. between t_i and v_j
12. **if** admission control func. returns **TRUE**
13. Allocate t_i to v_j
14. Break

Algorithm 2: Match task signature for comparing each client in cloud.

We have implemented the SigLM system and conducted both extensive trace-driven experiments and prototype evaluation. To perform extensive controlled experiments, we perform trace-driven experiments where SigLM node software is fully implemented but only task load and node resources are emulated. We have collected real application workload and node resources on the Planet Lab to drive the trace driven experiments. Specifically, we collect a set of resource metrics (e.g., CPU, memory) to indicate the available resources on different Planet Lab nodes. We also collect application load metrics (e.g., CPU load, memory consumption) of those applications running on the Planet Lab nodes. In trace-driven experiments, we set the node resources and task load requirements based on the collected traces. We also conducted prototype evaluation of the SigLM on the Planet Lab by running computation-

intensive applications on top of SigLM.

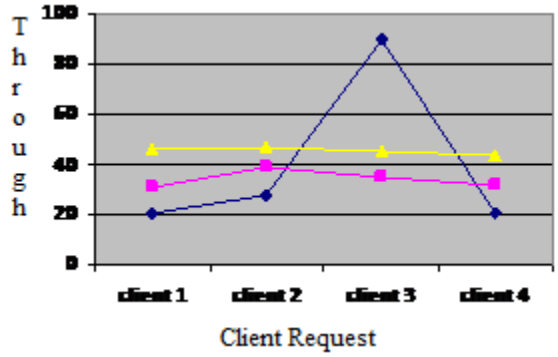

Figure 6: Load balancing analysis for each client.

As shown in above figure efficient client load balancing across the server.

V. CONCLUSION

A cloud is an attractive infrastructure solution for web applications since it enables web applications to dynamically adjust its infrastructure capacity on demand. we have presented SigLM, a new signature driven load management system for large-scale cloud computing infrastructures. Different from traditional coarse grained approaches, SigLM can capture detailed patterns of dynamic system resources and application workloads using fine-grained, dynamically updated, time series signatures. To the best of our knowledge, SigLM makes the first attempt to apply time series analysis techniques to achieve more efficient load management in large-scale distributed infrastructures. As a further improvement of load balancing in cloud in different commercial techniques with both hardware and software.

VI. REFERENCES

1) Sewook Wee, Huan Liu," **Client-side Load Balancer using Cloud**", SAC'10 March 22-26, 2010, Sierre, Switzerland. Copyright 2010 ACM 978-1-60558-638-0/10/03 ...$10.00.

2) Miss.Rudra Koteswaramma**,"** Client-Side Load Balancing and Resource Monitoring in Cloud**",** (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 6, November- December 2012, pp.167-171.

3) Zhenhuan Gong, Prakash Ramaswamy, Xiaohui Gu," SigLM: Signature-Driven Load Management for Cloud Computing Infrastructures", *ACMTransactions on Internet Technology*,8(2):113–124, May 2008.

4) Amazon Web Services. Amazon Web Services (AWS).
http://aws.amazon.com.

5) V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. IEEE Internet Computing, 3(3):28{39, 1999.

6) L. Cherkasova. FLEX: Load Balancing and Management Strategy for Scalable Web Hosting Service. IEEE Symposium on Computers and Communications, 0:8, 2000.