International Journal of Scientific Research & Engineering Trends
Volume 8, Issue 6, Nov-Dec-2022, ISSN (Online): 2395-566X

Load Balancing in Cloud Computing Through Multiple Gateways

Research Scholar Rani Danavath, Asst. Prof. Dr. V. B. Narsimha
Dept. of Computer Science
Osmania University
Hyderabad, India

Abstract-Cloud computing is a structured model that defines computing services, in which data as well as resources are retrieved from cloud service provider via internet through some well formed web-based tool and application. As the numbers of users are increasing on the cloud, the load balancing has become the challenge for the cloud provider. As most of the traffic is oriented towards the Internet and may not be distributed evenly among different IGWs, some IGWs may suffer from bottleneck problem. To solve the IGW bottleneck problem, we propose an efficient scheme to balance the load among different IGWs within a WMN Our proposed load-balancing scheme consists of two parts: a traffic load calculation module and a traffic load migration algorithm. The IGW can judge whether the congestion has occurred or will occur by using a linear smoothing forecasting method. When the IGW detects that the congestion has occurred or will occur, it will firstly select another available IGW that has the lightest traffic load as the secondary IGW and then inform some mesh routers (MPs) which have been selected by using the Knapsack Algorithm to change to the secondary IGW. The MPs can return to their primary IGW by using a regression algorithm.

Keywords-Cloud Computing, IGw's, Bottleneck problem, Wireless Mesh Network, Load Balancing.

I. INTRODUCTION

In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses in the form of web services—now commonly known as cloud computing. One of the key benefits of cloud computing is the opportunity to replace up-front capital infrastructure expenses with low variable costs that scale with your business. With the cloud, businesses no longer need to plan for and procure servers and other IT infrastructure weeks or months in advance. Instead, they can instantly spin up hundreds or thousands of servers in minutes and deliver results faster.

Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world.From an Amazon web services Elastic Load Balancing (AWS ELB) some key issuesare taken such as limited capacity and end-to- end fairness problem pose serious challenges to the wide spread addition to the aforementioned challenges, load deployment of this technology. In balancing of the traffic at the gateway nodes is also another important challenge to be addressed.

Envisioned to serve a large community of user so, the average volume of traffic is significantly higher than in a cloud environment. We are using. Multiple Internet Gate Ways (IGWs) to prevent the problems. Here we can use load balancing protocol to reroute flows from one internet gate way to another internet gateway. And we are using Prediction Packet Loss Rate algorithm for reducing packet loss rate and time.

1. Amazon web services:
Amazon's load balancer supported both Web Sockets over SSL and the ability to have 256-bit SSL encryption. It still leaves a few things to be desired (like being able to have a "backup" server which web traffic is routed to only in the case of a primary server failure).Amazon uses Elastic Load balancing Algorithm for load balancing.

2. Elastic Load Balancing Products:
Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones.

Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.

3. Key Benefits AWS ELB:
- Distribution of requests to Amazon EC2 instances (servers) in multiple Availability Zones
- Continuous monitoring of the health of Amazon EC2 instances registered with the load balancer so that requests are sent only to the healthy instances.
- Support for end-to-end traffic encryption on those networks that use secure (HTTPS/SSL) connections.

- The ability to take over the encryption and decryption work from the Amazon EC2 instances, and manage it centrally on the load balancer.
- Support for the sticky session feature, which is the ability to "stick" user sessions to specific Amazon EC2 instances.

4. Key Limitations of AWS ELB:

- ELB Doesn't Work for Non-AWS Servers
- Taking Hosts In and Out of Service
- Special DNS Requirements for DNS Apex

II. OBJECTIVE

Amazon Web Service ELB (Elastic Load Balancer) uses Traditional topology and Round Robin scheduling algorithm for load balancing.

So I want to use Mesh topology that will give you the better performance i.e.High band width and broadband services and I implemented MGMR load balancing algorithm to give better results

III. MOTIVATION

There are two main categories of reasons for the uneven load distribution across Clouds.

- The First reason is the uneven client distribution. Generally, users are unevenly distributed throughout the cloud, and furthermore, the distribution also changes with time.
- The second reason is the uneven user demands. The user demands may vary widely within a given network domain.

The traffic consists of several short flows and long flows. Thus, the demands of users greatly vary as well. These two factors cause a strong difference in the aggregated load imposed on each IGW and thus some IGWs may suffer congestion problems, whereas some IGWs may be severely underutilized.

The congestion may have an effect on the network performance, such as aggregated network throughput, packet delivery ratio, and end-to-end delay. In order to improve the network performance, we propose an efficient method to balance the traffic load among different IGWs.

1. Gateway Load Balancing Scheme:

Even though the MGMR-HWMP described in Section 3 is configured with multiple IGWs, it does not enable a directly and linear increase of the network performance. To achieve better WMN performance, I propose an efficient gateway load-balancing scheme based on the MGMR-HWMP, which is called Multi-Gateway Multi-Root Hybrid Wireless Mesh Protocol with Load Balancing (MGMR-LB). The proposed gateway load-balancing scheme consists of two parts—a traffic load calculation mode and a traffic load migration algorithm. The basic procedures are shown as follows:

- When the RANN interval is coming, each gateway should calculate the traffic load of the current RANN interval and predict the traffic load of the next interval by using Brown's Linear Smoothing method.
- The gateway should judge whether it has suffered or will suffer congestion problem. ① If the gateway has suffered or will suffer congestion problems, it should use the load Balancing scheme.
- The gateway should calculate the traffic load it intends to transfer.
- After calculating the transferred traffic load, the gateway should use the Knapsack Algorithm to find the best set of MAPs (mesh access points) that need to change their primary gateway to the secondary gateway to relieve the traffic load of the primary gateway. In the Knapsack Algorithm, we consider the traffic load of the MAP, the transferred traffic load, and the number of MAPs.
- After getting the best MAP set, the gateway sends notification messages to those MAPs.
- When the MAP receives the notification message, it should change its primary-Gateway to the selected secondary gateway.
- After using the load balancing scheme, if the congested gateway detects that it is in the stable state for several RANN intervals, it should use the mesh router regression algorithm to inform the MAP return to the primary gateway. We propose the mesh router regression algorithm, as we want to reduce the potential congestion problem of the secondary gateway.

2. The Traffic Load Calculation Mode

The exponential smoothing assigns exponentially decreasing weights over time. In our case, the time series data is the average traffic load of the IGW. There are many methods for double exponential smoothing, but in my paper i use Brown's Linear Smoothing.The raw traffic load data sequence is often represented by $\{TLi\}$. We use $\{Si\}$ to present the smoothed value for interval *i*, and $\{bi\}$ is our best estimate of the trend at interval *i*.

The output of the algorithm is now written as $TLi+m$, which is an estimate of the value of the traffic load in the $(i+m)$ th interval, where $m>0$ is based on the raw traffic load information up tointerval *i*. In our case, we are just trying to p predict the traffic load of next interval, which means*m* equals to 1.

The common calculation is:
si, = ă x TLi + (1- ă) x si-1,
si" = ă x si, + (1- ă) x si-1"
ai= 2 x (si, - si")
bi = ă1-ă x (si, - si")

We set the initial value of S0' and S0" to TL1 which is the traffic load of the first calculation interval. Then we can predict the traffic load of the next interval as:

TLi= ai+bi

Since we have obtained the traffic load information of both the current and next interval, we can judge the state of the IGW. There are three states of the traffic load: Low State, the Medium State, and the High State.

1. Low state: TLi<TLlower_threshold
2. Medium state: TLlower_threshold ≤TLi＜TLupper_threshold
3. High State: TLi ≥ TLupper_threshold

Table 1. Internet Gateway state judgment.

Predicted Current	Low state	Medium state	Higher state
Low state	Under utilized	Stable	Stable
Medium state	Under utilized	Stable	Congestion
Higher state	Stable	Stable	Congestion

IGWs. For IGWs in underutilized state, they can work in a stable state and can be considered as the best choice to be the secondary IGW for the congested IGW. For IGWs in the stable state, they can work in a stable state and can be considered as the second choice to be the secondary IGW for the congested IGW. The traffic load information and Internet gateway state information will be packaged in the RANN message.

3. The Mesh Router Migration Algorithm:-

When the IGW is in the congestion state, it should consider whether it needs to use the load-balancing scheme. To achieve this target, the congested IGW should calculate the average traffic load of all the available IGWs recorded in its Internet Gateway Information Table (IGIT) and itself. It can be calculated as:

TLIGW = 1NumfoIGW x i=1numofIGW TLIGW_i

Where *NumOfIGW* is the total number of IGWs including itself and *TL* IGW i. In addition, we present the traffic load of the IGW itself as: TL_{IGW_self}.Now we can judge whether this congested IGW needs to use the load-balancing scheme. If

$TL_{IGW\_self} \geq \beta \times TL_{IGW}$($\beta$ is the experienced parameter), this congested IGW should use the load-balancing scheme. Fig. 2 describes the load migration algorithm. If not, there is no need to use it. When the congested IGW decided to use the load-balancing scheme, it firstly selects an IGW with the lightest traffic load among all the available none congested IGWs by using the SearchAvailableIGWwithLightestTL function. If there is an available IGW, the IGW should calculate the traffic load it intends to transfer $TL_{transfer}$. The congested IGW should also get the traffic load information TL_{IGW_Light} of the IGW_{Light} by using the GetTrafficLoadInfo function. If TL_{IGW_Light} is smaller than TL_{IGW}, the congested IGW continues to calculate the traffic load the IGW_{Light} can provide $TL_{Provide}$. If $TL_{Provide}$ is larger than $TL_{transfer}$, then we can use the Knapsack Algorithm to find the best MAP set to change the primary IGW to the IGW_{Light}.

The exact Kanpsack Algorithm in the mesh router regression algorithm is the 0-1 knapsack algorithm. In the 0-1 knapsack algorithm, there are n items (xn). Each item has value (vn) and weight (wn). The problem is that if the weight (W) of packets is limited, then we have to figure out the best set of these items, as shown in Equation (8).

Max i=1nvi xi Subject to i=1nwixi ≤ W, xi €{0,1};

In our 0-1 knapsack algorithm, the item is the MAP. The value and the weight of the item I the metric of the MAP (MP_metric) and the transferred traffic load of MAP (TLMPi). The totalvalue W is TLtransfer. We can show the algorithm as shown in Equation (9):

Min i=1nMPmetric i xi Subject to i=1nTL mpi xi ≤ TLtransfer,xi €{0,1};

As we know the decision of the 0-1 Knapsack Algorithm is a NP-complete, the complexity of running a traditional Knapsack Algorithm is O(nW). The complexity of our Knapsack Algorithm is O(Settransfer(MAP)* TLtransfer).

Load Migration Algorithm

```
01  IGW_Light=SearchAvailableIGWwithLightestTL(IGIT);
02  If(exist IGW_Light){
03      TL_transfer= TL_IGW self- TL_IGW;
04      TL_IGW Light=GetTrafficLoadInfo(IGW_Light);
05      If(TL_IGW Light< TL_IGW){
06          TL_Provide= TL_IGW- TL_IGW Light;
07          If(TL_Provide > TL_transfer){
08              Set_transfer(MAP)=
09   KnapsackAlgorithm(IGIT(AvailableMPInfo));
10                  If(Set_transfer(MAP)= = NULL){
11  Set_transfer(MAP)= SearchAvailableMPwithHighestTL
12  (IGIT(AvailableMPInfo));}
13  SendNotificationMsg(Set_transfer(MAP));
14  }
15      }
16  }
```

Fig 2. Procedure for the load migration algorithm.

The complexity is based on the number of the MAPs set and the total traffic load that we want to transfer. We want to select the MAPs that maintain a better routing metric to the selected secondary IGW. In Equation (9), we want to get the minimum value of the total routing metric of the selected MAP set .We also constrain the traffic load of the MAPs as we want to get the best set of MAPs to change the gateway, and then the congested gateway can be relieved better.

After selecting the MAPs, the congested IGW should send notifications to the selected MAP set to inform them to change from the primary gateway to the secondary gateway by using the SendNotificationMsg function.

IV. SIMULATION RESULTS

1. Response Time:
In this section, proposed method represents mathematical model behalf of response time in equation. Here, Response time is calculated based every user base maximum and minimum response by proposed approach. Response time contains following factorTFinish, which time is taken data center to receive the request from user base. TArrival is recorded by datacenter, which request came. TDelay Is dealing out of the request to reach in datacenter.

TResponse =TFinish- TArrival+ TDelay

Table 2. Response time comparision

Load balancing Algorithms	VM's	Overall Response ime		
		Average (ms)	Min	Max
Round Robin	25	74.97	36.38	65.46
Round Robin	50	50	36.26	65.76
Round Robin	75	50.26	36.38	65.63
Round Robin	100	50.56	36.51	66.12
Equally Spread Current Execution	25	74.54	36.2	65.10
Equally Spread Current Execution	50	51.2	36.4	65.92
Equally Spread Current Execution	75	50.3	36.38	65.63
Equally Spread Current Execution	100	50.23	36.35	66.0
Throttled	25	74.2	36.02	64.99
Throttled	50	51.1	36.2	65.76
Throttled	75	50.4	36.41	65.80
Throttled	100	50.5	36.7	66.20
MGMR	25	74.2	35.2	62.56
MGMR	50	51.1	35.5	64.4
MGMR	75	50.4	35.6	65.0
MGMR	100	50.5	35..6	65.3

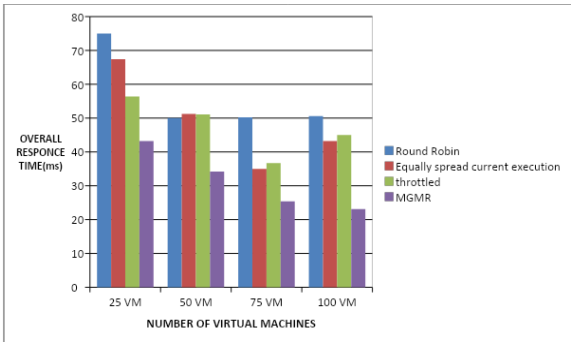

Fig 2.Graph for response time

2. Data Center Processing Time:
Here, datacenter processing take care the entire request which are given user in various regions. Hence, it processes for retrieve the requested query from database. Here, is calculated based user request per hour with respect of bandwidth allocation (1000 mbps).

TDatacenter Processing time = RBWUserbase BW – Band width, R--- User Request

Table 3. Datacenter processing comparison.

VM's	Datacenter Processing time											
	Round Robin			Equally Spread Current Execution			Throttled			MGMR		
	AVG	MIN	MAX	AVG	MIN	MAX	AVG	MIN	MAX	AVG	MIN	MAX
5	0.57	0.00	8.47	0.58	0.00	8.5	0.57	0.00	8.3	0.55	0.00	7.8
10	0.56	0.00	7.46	0.57	0.00	7.57	0.56	0.00	7.90	0.54	0.00	6.2
15	0.51	0.00	5.61	0.57	0.00	5.9	0.52	0.00	5.7	0.52	0.00	5.10
20	0.51	0.00	5.09	0.55	0.00	5.96	0.52	0.00	5.8	0.51	0.00	4.70
25	0.51	0.00	5.83	0.56	0.00	5.88	0.51	0.00	5.7	0.48	0.00	4.42
30	0.51	0.00	5.61	0.52	0.00	5.70	0.51	0.00	5.69	0.45	0.00	4.0
35	0.51	0.00	5.75	0.52	0.00	5.9	0.51	0.00	5.74	0.45	0.00	4.68
40	0.51	0.00	5.11	0.51	0.00	5.22	0.51	0.00	5.34	0.44	0.00	3.9
45	0.51	0.00	5.43	0.51	0.00	5.3	0.51	0.00	5.45	0.44	0.00	3.60
50	0.51	0.00	5.61	0.51	0.00	5.45	0.51	0.00	5.56	0.44	0.00	3.56

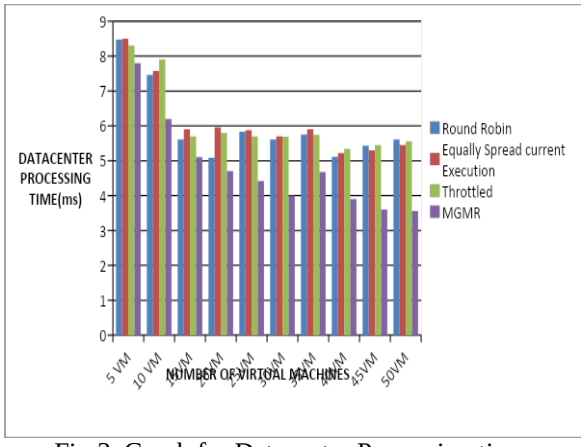

Fig 3. Graph for Datacenter Processing time

3. Delay Time:

Delay time calculates the difference between request arrival time and response time i.e. when user receive the response from datacenter. It also contains delay time between two regions when user is available in different location and datacenter in different locations. Here TFinish represents total time taken to complete the task and TEstimate represent assumption time which are assigned by analyzer before start the task. TArrived Indicate the time when request reach to data center and TDatacenter Processing indicate processing time.

TDelay= TFinished-TEstimated-TArrival-TDatacenter Processing

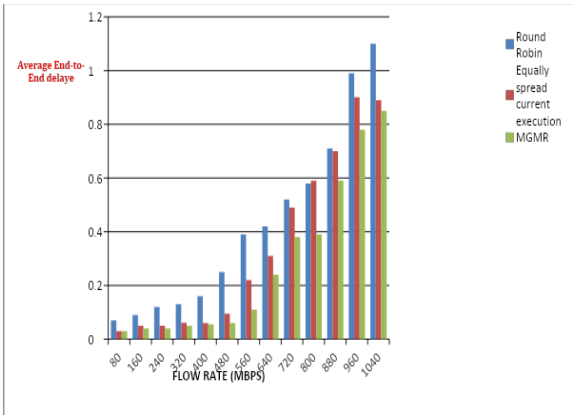

Fig 4.Graph for Datacenter Processing time

V. CONCLUSION

The main purpose of load balancing is to satisfy the customer requirement by distributing load dynamically among the nodes. And need proper load balancer for congestion problem and my work will avoid Limited capacity, End to End fairness, Delay of Taking Hosts In and Out of Service: ELB supports server health checks and will automatically remove servers if they becomeUnhealthy. Unfortunately, the configuration options around the health checks are somewhat limited and occasionally frustrating. It's impossible to immediately force a server into production. Instead, a minimum number of health checks must succeed.Although this is a reasonable to control after a server fails, it also creates a tricky situation when trying to add new servers to the load balancing pool, because they are not immediately available to serve traffic.And my research will improve Throughput, associated Overhead, resource utilization and reduces the Migration time.Minimizes data Processing time and overall response time

REFERENCES

[1] Deepti Nandiraju, Lakshmi Santhanam, Nagesh Nandiraju, and Dharma P. Agrawal " Achieving Load Balancing in Wireless Mesh Networks Through Multiple Gateways"

[2] Cai Ling, Jinkuan Wang, Cuirong Wang, Xu Peng"Load Balancing Algorithm Based on Time Series Prediction of Packet Loss Rate"

[3] Juan J. Gálvez,Pedro M. Ruiz,Antonio F. G. Skarmeta" A Distributed Algorithm for Gateway Load-Balancing in Wireless Mesh Networks"

[4] Klaithem al nuaimi, anader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms".

[5] L. Roychoudhuri and E. Al-Shaer, "Real-time packet loss predction based on end-to-end delay variation, "IEEE Trans. Network Service Manager.

[6] Y. Bejerano S.J. Han, and A. Kumar, "Efficient load-balancing routing for wireless mesh networks", Comput.Network.

[7] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks," IEEE Transactions on Information Theory, pp. 388–404, 2000.

[8] L. Ma and M. K. Denko, "A Routing Metric for Load-Balancing in Wireless Mesh Networks," in AINAW '07, Advanced Information Networking and Applications Workshops, 2007, pp. 409–414.

[9] Anjaly Sara Panicker, Seetha S, Sharmila J "Survey on various Load Balancing Techniques in Wireless Mesh Networks".