Why load balancing and auto-scaling ?

Yi-Pei, Tu

1. Introduction

Nowadays, the cloud-computing service is used by many modern websites tremendously. Most of the cloud-computing services advertise that the service provides the mechanisms such as load balancing and auto-scaling to give better performance and high-scalability to reduce the traffic load and cost for deploying and hosting the websites on the cloud. As a newbie in DevOps, acquiring knowledge about both of them could understand why modern websites rely on them. Moreover, the issues while using practically should be addressed as well. In the essay, the author will introduce the concept, the approaches and the issues in each section. In the discussion section, the author will address the relation between them, the risk of using it alone, and issues for reality.

2. Load balancing

"In computing, load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives.[1]" is the definition of load balancing from Wikipedia. The concept of load balancing originates from the 1990s. During that time, the internet has expanded all over the world [2]. The internet service provider (ISP) offers billions of data by a single upstream provider, or multihoming to prevent redundancy and reduce workload. Load balancing was applied for hardware such as multilayer switch and software such as the Domain Name System service. Due to the tremendous amount of users on the internet, load balancing is applied almost everywhere to reduce the workload for each service such as internet-based services, telecommunications, and data center.

In DevOps, the most common application of load balancing is to provide a single service to distribute the traffic to multiple servers. The mechanism of server-side load balancing is a service that is listening on a specific port to wait for upcoming requests from external. When the clients send the request to the load balancing service, the load balancing service will forward the request to one of the backend servers which is available to handle the request as in Figure 1. The advantage of using load balancer is to prevent users from sending the requests to backend servers which make sure that the users won't know the internal functionalities. Moreover, it may hide the internet strucutre and protect the core stack from outside. Usually, the load balancers are pairs to prevent a single point of failure. As long as one load balancer is shut down, the

other one could handle it.

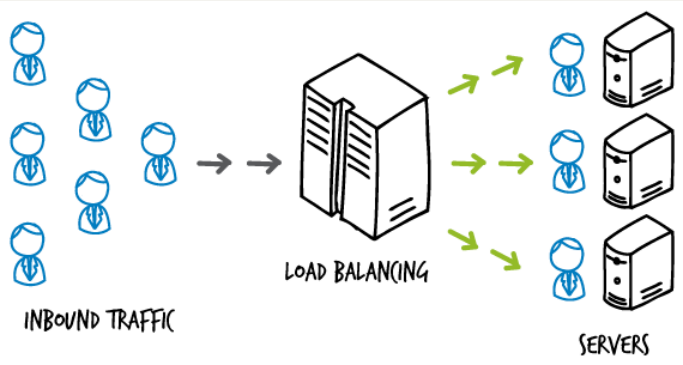

Figure 1: Load Balancing| Image Retrieved From CenturyLink

2.1. Load balancing approcahes

The number of load balancing methods such as random choice and least connections could take into account which depends on the developers' need. For example, the status report for each backend server could be one of the main factors for the load balancer to decide how to dispatch the request. The status report could contain the server load, the server response time and the number of activating requests.

2.2. Issues

The most crucial issue should be addressed while using load balancer is data persistent. The problem is how to make sure the user session data are consistent on all backend servers. If one of the backend servers stored the user data locally and the load balancer forwards the user request to another backend server, the other might have trouble with handling the request due to data inconsistent. The possible solution is to store the user data in a shared database or memory cache. However, storing user session data in a shared database would increase the workload for the database which may have a bad performance. Another common solution is to store user data in client-side with cookies. URL rewriting could be one of the solutions as well. However, users could modify the URL easily which is a security issue.

	OSI Layer	TCP/IP	Datagrams are called
Software	**Layer 7** Application	HTTP, SMTP, IMAP, SNMP, POP3, FTP	Upper Layer Data
	Layer 6 Presentation	ASCII Characters, MPEG, SSL, TSL, Compression (Encryption & Decryption)	
	Layer 5 Session	NetBIOS, SAP, Handshaking connection	
	Layer 4 Transport	TCP, UDP	**Segment**
	Layer 3 Network	IPv4, IPv6, ICMP, IPSec, MPLS, ARP	**Packet**
Hardware	**Layer 2** Data Link	Ethernet, 802.1x, PPP, ATM, Fiber Channel, MPLS, FDDI, MAC Addresses	**Frame**
	Layer 1 Physical	Cables, Connectors, Hubs (DLS, RS232, 10BaseT, 100BaseTX, ISDN, T1)	**Bits**

Figure 2: OSI Network Model| Image Retrieved From Abdulaziz Hameda

Before the cloud-computing became mature, the common application, layer 7 in Figure 2, load balancers are Nginx and Apache HTTP Server. Nowadays, developers prefer to deploy the service on the cloud such as Amazon Web Service [3], Azure [4], and Google Cloud [5]. All of them support not only application load balancing but also Network Load Balancer, layer 4 in Figure 2.

3. Auto-scaling

Auto-scaling is a method to control the amount of activating servers on the cloud [6]. The primary concept of auto-scaling is intuitive which shows in Figure 3. The amount of activating servers is usually in the desired capacity for regular usage. If the traffic is lower than normal usage, the number of activating servers will reduce to the minimum size. When the work loading is increasing over the average usage, the backup server(s) will activate as needed till reaching the maximum size.

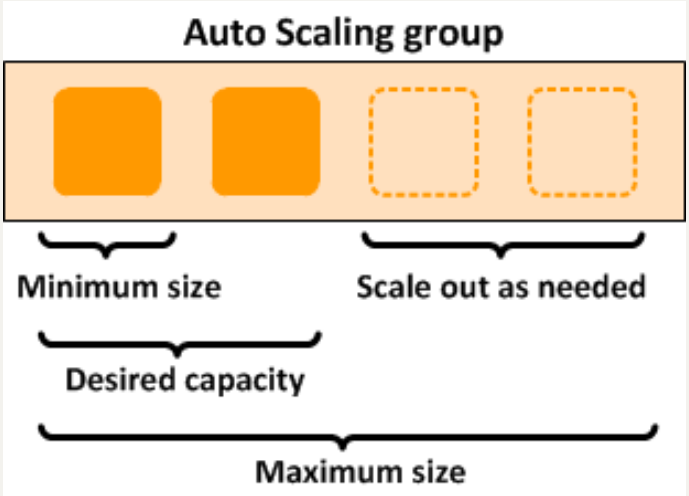

Figure 3: Auto-Scaling| Image Retrieved From AWS-Amazon EC2 Auto-Scaling

Auto-scaling could save the cost and optimize the usage. Before auto-scaling, the developers usually deploy the number of physical servers based on the prediction of traffic in Figure 4. However, the forecast is not 100% accurate in reality. If the unexpected traffic spike happens, the coming requests cannot be processed which might cost a lot by losing customers in Figure 5. To prevent the risk, the developers might run the maximum number of servers which also cost a lot for maintaining all servers. The companies had to decide cost-saving for everyday usage or traffic-handling for the highest peak.

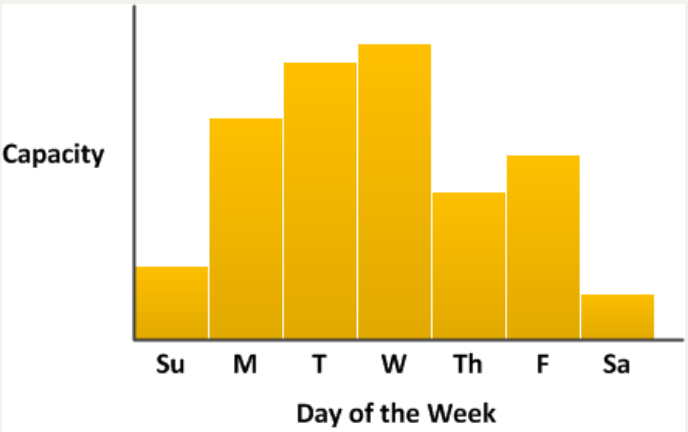

Figure 4: Usage in a week | Image Retrieved From AWS-Benefits of Auto-Scaling

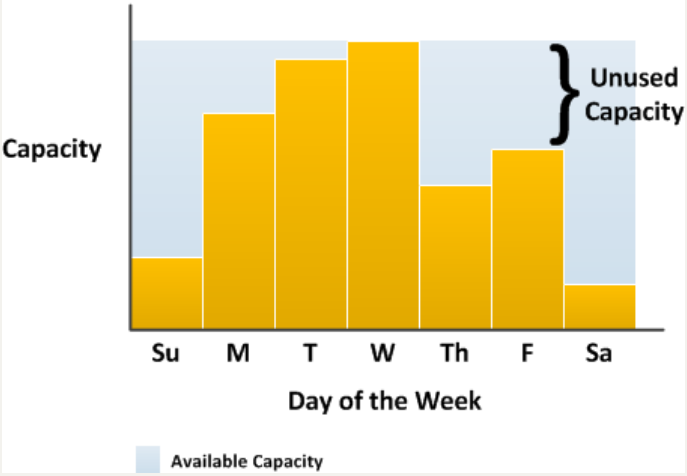

Figure 5: Unused Capacity | Image Retrieved From AWS-Benefits of Auto-Scaling

After auto-scaling launched in Figure 6, the companies could save the cost on servers and handle the traffic spikes with auto-scaling.

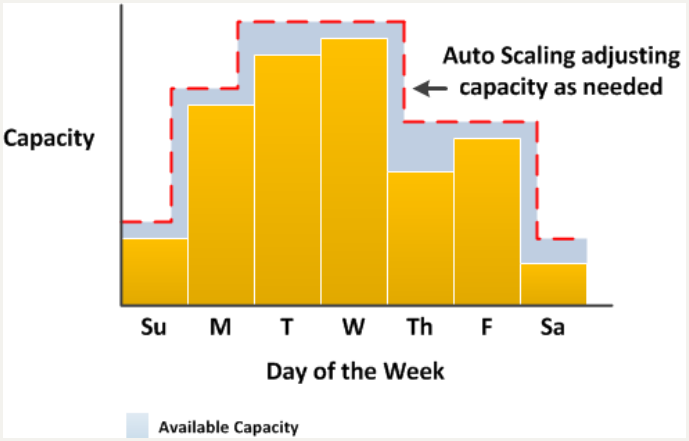

Figure 6: Auto-Scaling| Image Retrieved From AWS-Benefits of Auto Scaling

3.1. Auto-scaling approaches

There are three types of auto-scaling approaches. The **reactive approach** is to scale the group based on real-time traffic. The drawback is that the response may not handle sufficient if the changes are rapid. The **scheduled approach** is that the load changing time has known at a specific time or date. This approach could support a reactive approach to handle the known rapidly changes. The **predictive approach** is to predict the changes by learning from the past with recent usages. The advantages of this approach are to scale the group in a bit advance based on prediction. The types and policies of auto-scaling are general but slightly differs from products such as Amazon Web Service Auto-Scaling [7] and Google Cloud Platform Auto-Scaling [8].

3.2. AWS auto-scaling

AWS supports reactive, scheduled, and predictive approach for auto-scaling. Developers could scale the size at any time with **manual scaling**. Another one is that the developers have known the load increasing time or date, the developer could **schedule scaling**. Besides these two, AWS also provides **dynamic scaling** with three policy types which could apply in combination.

- Target tracking scaling: adjust the size based on a specific metric such as CPU usage.
- Step scaling: adjust the size based on a set of scaling.
- Simple scaling: adjust the size based on single scaling.

3.3. Google cloud platform auto-scaling

Google Cloud Platform offers reactive approach for auto-scaling. Developers should use at least one policy and could apply multiple policies as well. The most basic one is **CPU utilization** that the auto-scaler will adjust the group size based on the CPU usage. **Stackdriver Monitoring Metrics** provides

developers to scale by metrics not only per instance but also per group. Developers could decide the metrics and the auto-scaler will follow the metrics to adjust the group size. The combination of the load balancer and auto-scaler, **Load balancing serving capacity**, is one of the policy for scaling. The auto-scaler modifies the group size according to the HTTP load serving capacity.

4. Disscusion

Auto-scaling relies on load balancing [6] especially for cloud-computing. Figure 7 shows the realtion between load balancer and auto-scaler. Each auto-scaling group contains at least one instance or one back-end server to handle the request. The load balancer will forward the request to the available instance based on the metrics. If the auto scaling group detects the CPU usage reaches the critieria or the scheleded approach applied, the auto-scaler will create new instance to handle the traffic. Thus, the load balancer could dispatch the new request to the new instance. As long as the traffic becomes lower, the auto-scaling group could stop the available instance.

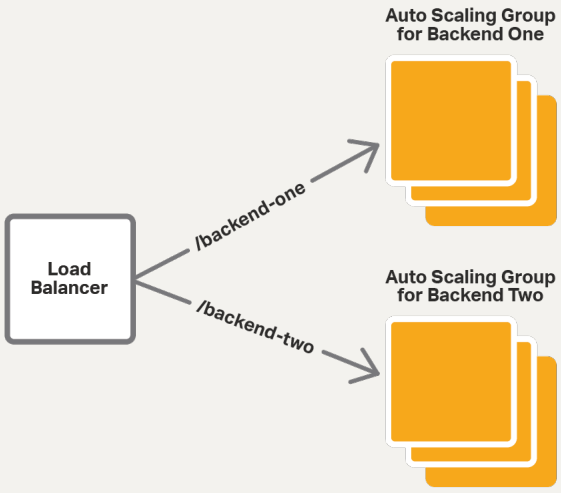

Figure 7: Load Balancer and Auto-Scaling| Image Retrieved From NGINX Plus

4.1. Risks

The risk of **load balancer alone** is that developers have to measure the capacity of coming requests to decide the number of the instance as back-end servers in advance to deal with all the requests. When the traffic spikes happen which means all the servers might be 100% usage, the load balancer cannot dispatch more requests to the available server. The waiting requests are all queuing on the load balancer which will also increase the workload of the load balancer. Figure 8 shows how the request is dispatched to the instance.

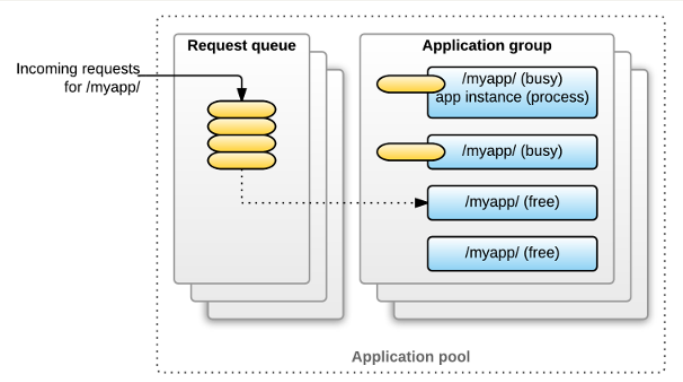

Figure 8: Request queue| Image Retrieved From Request load balancing

The risk of **auto-scaling alone** is that one of instance might be very busy and the rest of instances are free to handle the request which is waste of money as well. Also, it's complicated to maintain the IP entries and update the IP modifcation to the Domain Name System by real-time which is to handle the coming request to map the right IP with domain name. Thus, the auto-scaling should always company with load balancing. Otherwise, the devleopers should deal with the connection between Domain Name Server and the back-end servers.

4.2. Issues in practical

From the book Load Balancing in the Cloud by Derek DeJonghe [9], it says that *"as capacity is increased or reduced, however, the nodes being added or removed must be registered or deregistered with a load balancing solution."* Thus, the approaches of **adding new nodes** and **deregistering nodes** should consider comprehensively with all aspects.

If the auto-scaler added new nodes but not inform the load balancer, the new nodes are idle which means a waste of money. Another problem might be the auto-scaling detects the workload is still uneven, so it scales up again if the metrics are the average of CPU usage. However, the phenomenon is named **rubber banding** which means *"it takes action to serve demand but does not actually provide the intended effect. [9]"*

If the auto-scaler removes the nodes but not inform the load balancer, the client will receive a timeout response if the deleted nodes were handling his requests. The worse case is session loss which might affect the website if it relies on session persistence. The right way to removing the nodes is to drain connections, persistent sessions, deregister the nodes, and inform the load balancer.

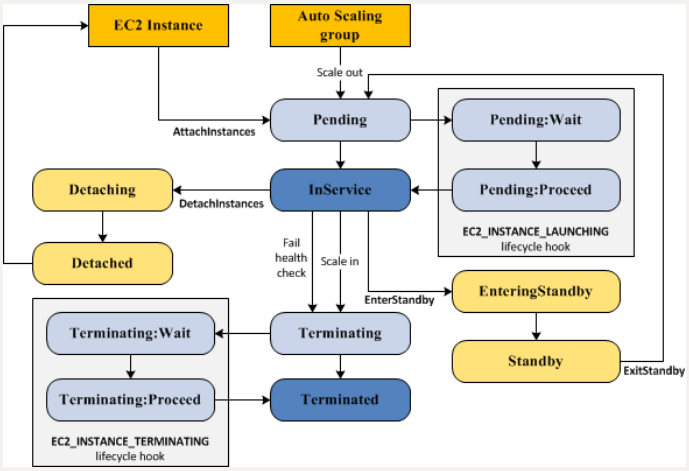

Figure 9: Add/Remove Nodes| Image Retrieved From Auto Scaling Lifecycle

After adding or deregistering the nodes, the workload should redistribute under the promise of session persistence. There are two approaches, **proactive** and **reactive**, to achieve it. Figure 9 shows a provocative way with a lifecycle hook to prevent unpredictable problems. The reactive approach is the developers fetch the number of nodes in the auto-scaling groups by API such as AWS API and update the changes to the load balancer which is asynchronous updating after the node is alive or already gone.

5. Conclusion

Load balancing is a service to distribute the workload, and auto-scaling is to increase or decrease the number of servers to handle heavy traffic based on the specific metrics such as CPU usage or the amount of activating requests. Auto-scaling usually relies on load balancing. If the load balancing doesn't exist as a bridge between the internet and the auto-scaling group, it's complicated to scale down or scale up the services due to IP updating issues. The collaboration of load balancing and auto-scaling is to distribute the traffic and make the back-end servers more flexible to handle the tons of requests. The problems of adding or removing the nodes in the group are controlled by the cloud-computing service automatically. However, it's better to understand how to keep data persistence and the auto-scaling lifecycle to prevent crucial problems after launching.

6. References

[1] *Load balancing (computing)*. (2010). *En.wikipedia.org*. Retrieved 26 April 2019, from https://en.wikipedia.org/wiki/Load_balancing_(computing)

[2] *Internet*. (2019). *En.wikipedia.org*. Retrieved 26 April 2019, from https://en.wikipedia.org/wiki/Internet#History

[3] *Elastic Load Balancing - Amazon Web Services*. (2019). *Amazon Web Services, Inc.*. Retrieved 27 April 2019, from https://aws.amazon.com/elasticloadbalancing/

[4] *Load Balancer*. (2019). *Docs.microsoft.com*. Retrieved 27 April 2019, from https://docs.microsoft.com/en-us/azure/load-balancer/

[5] *Google Cloud Load Balancing - High Performance, Global & Scalable | Load Balancing | Google Cloud*. (2019). *Google Cloud*. Retrieved 28 April 2019, from https://cloud.google.com/load-balancing/

[6] *Autoscaling*. (2019). *En.wikipedia.org*. Retrieved 29 April 2019, from https://en.wikipedia.org/wiki/Autoscaling

[7] *What Is Amazon EC2 Auto Scaling? - Amazon EC2 Auto Scaling*. (2019). *Docs.aws.amazon.com*. Retrieved 29 April 2019, from https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

[8] *Load Balancing and Scaling | Compute Engine Documentation | Google Cloud*. (2019). *Google Cloud*. Retrieved 30 April 2019, from https://cloud.google.com/compute/docs/load-balancing-and-autoscaling#autoscaling

[9] *Load Balancing in the Cloud*. (2019). *O'Reilly | Safari*. Retrieved 30 April 2019, from https://www.oreilly.com/library/view/load-balancing-in/9781492038009/ch04.html