Cloud Database Services improve Using load balancing Technique

[1]Md. Asad, [2] Dr.Madhurendra Kumar, [3]Dr. Arbind Kumar Sinha, [4] Dr. Prakash Kumar
[1]Research Scholars, MagadhUniversity, Gaya 2 PhD Co-Guide, Project Manager, National Informatics Center (NIC), Delhi
[3] PhD Guide, Associate Professor, MagadhUniversity,Gaya (Bihar) ,[4] Assistant Professor ,JRSU(Ranchi)

Abstract:

Cloud database load blanching services is an innovative application. It is a column-oriented and relational database. It also has many options to filter or report the needed data. It has many features like flexibility, scalability, etc. It is also very user-friendly. The problem with this is that it has a slow loading time for huge data. I will improve cloud database services data availability using Round Robin algorithm.

Keyword: Round Robin, Load balancer, working process proposed work, Security and privacy.

Introduction:

Today there is an enormous amount of data all around the world, making it kind of difficult for organizations to manage and store the data. Cloud computing has made the process a bit simple with a proper cloud database management system. Just like Relational Database Management System, there needs to be proper planning and system to store chunks of essential data, and manipulate and access the data from all across the world whenever it is required.

The three schema architecture looks at three layers necessary for separating the data level from the physical level. It normally goes with the storage of the data, forming the base or the data layer. The data gets stored at a server with the application part forming the second layer. The final layer is the physical layer which can be accessed and used by the common people. There can be modifications in the architecture such as the 5-tier architecture or the 7 level architecture but the 3 layer architecture forms the base with necessary changes for the cloud services.

A more convenient model is "database as a service" or DBaaS. With this approach, the provider shoulders far more of the burdens, typically including provisioning, backup, scaling, monitoring, and other management services. Users don't have to fiddle with infrastructure concerns and can simply connect and use the systems they need, again saving time, money, and frustration.

It is important to mention, however, that DBaaS offerings still typically involve some administrative burdens. Consumers must often choose between tiers of performance or storage, tweak or tune scalability parameters as needed.

Moving to the cloud can often require a change of model and/or technology, which can also require redesigning or tweaking data schemas for flexibility, performance, etc. Transitioning to modern application architectures such as cloud-native development with micro services, using server less functions such as AWS Lambda or Cloud flare Workers, or building JavaScript heavy applications via Jam stack provides an opportunity to reassess and restructure to maximize the benefits of cloud databases.

As such, a multi-step approach is advised, working from least to most important. Begin with your least mission-critical applications (or services) and rebuild those database schemas, moving those to the cloud. As the difficulty of moving data and operations increases, weigh the costs and benefits, engaging only with vendors who meet your data-security standards along the way. You might not be able to move everything to the cloud, or at least not today, but good

planning will help you maximize your benefits while minimizing your risks.

A bit of Customization: Cloud data is used by a variety of users with different requirements. Therefore, cloud database services for data management must be a bit customized catering to the needs of all the users.

Modeling concept: Since there will be different types of users, each user is going to have his or her view with regards to his or her requirements.

Hiding the necessary information: This a crucial step when it comes to the safety of the process. Unauthorized people must not have any access to the information stored on a cloud-based database.

1. Round Robin – The most basic load distribution technique, and considered rather primitive by network administrators.

In a round robin scenario the load balancer simply runs down the list of servers, sending one connection to each in turn, and starting at the top of the list when it reaches the end.

The same principle as Round Robin, but the number of connections that each machine receives over time is proportionate to a ratio weight predefined for each machine.
For example, the administrator can define that Server X can handle twice the traffic of Servers Y and Z, and thus the load balancer should send two requests to Server X for each one request sent to Servers Y and Z. However, given that most enterprises use servers that are uniform in their processing power, Weighted Round Robin essentially attempts to address a nonexistent problem

2. Load balancer:

Cloud load balancing is the process of distributing workloads and computing resources in a cloud computing environment. Load balancing allows enterprises to manage application or workload demands by allocating resources among multiple computers, networks or servers

Cloud load balancer is a virtual, scalable, high availability component that improves user response time and system stability by distributing workloads across two or more working servers and resources to maximize performance and avoid overload.

After verifying server availability using HTTP, TCP or UDP checks, the Cloud Load Balancer routes incoming network traffic to the servers that are available for response. Your performance will therefore remain stable even at times of peak usage, or in the unlikely event that one of your servers fails.

Clouds Load Balancer can handle almost any number of requests and is not restricted by the number of servers you have, among its features are:

1. Load balancing algorithm based on Round Robin

2. Handling of SSL decryption (sometimes referred to as SSL offloading termination)

3. Least connections and weight

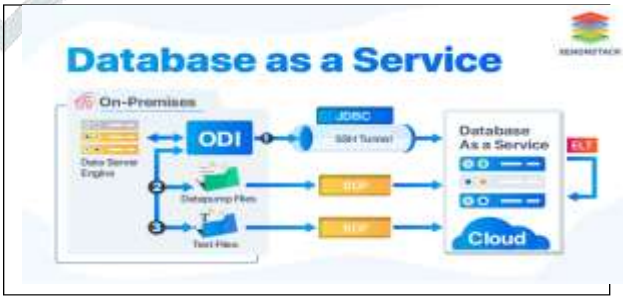
4. Session persistence across HTTP/S protocols and more

While maximum performance will be achieved by using Cloud Servers, this is not mandatory; you can use cloud load balancing for 3rd party servers located outside

3. Working Process:

The load balancer uses a predetermined pattern, known as a load balancing algorithm or method. This ensures no one server has to handle more traffic than it can process. Different algorithms manage the process using different techniques. You, therefore, have multiple options to choose from when making a decision on what type of load balancer to use.

Cloud database services flow



3.1 How a load balancer works:

1. A client, such as an application or browser, receives a request and tries to connect with a server.
2. A load balancer receives the request, and, based on the preset patterns of the algorithm, it routes the request to one of the servers in a server group (or farm).
3. The server receives the connection request and responds to the client via the load balancer.
4. The load balancer receives the response and matches the IP of the client with that of the selected server. It then forwards the packet with the response.
5. Where applicable, the load balancer handles SSL offload, which is the process of decrypting data using the Security Socket Layer encryption protocol, so that servers don't have to do it.
6. The process repeats until the session is over.

4. My proposed work:

With help from java programming I am implanting reduce the load balance time of cloud database services.

I will show a classic interaction with IP request and response. A request creates messages and places them into a queue, while a client reads them out and displays them. To be realistic, I will give the queue a maximum request and response in an ideal way.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Set
public class Client {
public static void main(String[] args) {
int NUM_OF_REQUESTS = 15;
        Client client = new Client();
        ArrayList <String> ipPool = new ArrayList <>();
        ipPool.add("192.168.0.1");
        ipPool.add("192.168.0.2");
        ipPool.add("192.168.0.3");
        ipPool.add("192.168.0.4");
        ipPool.add("192.168.0.5");
        Map <String, Integer> ipPoolWeighted = new HashMap<>();
        ipPoolWeighted.put("192.168.0.1", 6);
        ipPoolWeighted.put("192.168.0.2", 6);
        ipPoolWeighted.put("192.168.0.3", 3);
        client.printNextTurn("Random");
        LoadBalancer random = new RandomLoadBalancer(ipPool);
        client.simulateConcurrentClientRequest(random,
NUM_OF_REQUESTS);
        client.printNextTurn("Round-Robin");
        LoadBalancer roundRobbin = new
RoundRobinLoadBalancer(ipPool);
        client.simulateConcurrentClientRequest(roundRobbin,
NUM_OF_REQUESTS);

        client.printNextTurn("Weighted-Round-Robin");
        LoadBalancer weightedRoundRobin = new
WeightedRoundRobinLoadBalancer(ipPoolWeighted);
        client.simulateConcurrentClientRequest(weightedRoundRobin,
NUM_OF_REQUESTS);

        System.out.println("Main exits");

    }
    private void simulateConcurrentClientRequest(LoadBalancer
loadBalancer, int numOfCalls) {

        IntStream
                .range(0, numOfCalls)
                .parallel()
                .forEach(i ->
                        System.out.println(
                                "IP: " + loadBalancer.getIp()
                                + " --- Request from Client: " + i
                                + " --- [Thread: " +
Thread.currentThread().getName() + "]")
                );
    }

    private void printNextTurn(String name) {
        System.out.println("---");
        System.out.println("Clients starts to send requests to " + name
+ " Load Balancer");
        System.out.println("---");
    }

}
```

Clients starts to send requests to Random Load Balancer

```
IP: 192.168.0.3 --- Request from Client: 9 --- [Thread: main]
IP: 192.168.0.2 --- Request from Client: 10 --- [Thread: main]
IP: 192.168.0.5 --- Request from Client: 8 --- [Thread: main]
IP: 192.168.0.1 --- Request from Client: 14 --- [Thread:
ForkJoinPool.commonPool-worker-9]
IP: 192.168.0.2 --- Request from Client: 13 --- [Thread:
ForkJoinPool.commonPool-worker-5]
IP: 192.168.0.1 --- Request from Client: 11 --- [Thread:
ForkJoinPool.commonPool-worker-9]
IP: 192.168.0.4 --- Request from Client: 7 --- [Thread: main]
IP: 192.168.0.2 --- Request from Client: 0 --- [Thread:
ForkJoinPool.commonPool-worker-5]
IP: 192.168.0.3 --- Request from Client: 12 --- [Thread:
ForkJoinPool.commonPool-worker-23]
IP: 192.168.0.1 --- Request from Client: 5 --- [Thread:
ForkJoinPool.commonPool-worker-13]
IP: 192.168.0.5 --- Request from Client: 3 --- [Thread:
ForkJoinPool.commonPool-worker-5]
IP: 192.168.0.3 --- Request from Client: 4 --- [Thread:
ForkJoinPool.commonPool-worker-19]
IP: 192.168.0.2 --- Request from Client: 6 --- [Thread:
ForkJoinPool.commonPool-worker-9]
```

```
IP: 192.168.0.1 --- Request from Client: 2 --- [Thread:
ForkJoinPool.commonPool-worker-5]
IP: 192.168.0.4 --- Request from Client: 1 --- [Thread:
ForkJoinPool.commonPool-worker-27]
```

Round Robin: We can observe that each thread gets ordered IPs

Clients starts to send requests to Round-Robin Load Balancer

```
IP: 192.168.0.1 --- Request from Client: 9 --- [Thread: main]
IP: 192.168.0.2 --- Request from Client: 3 --- [Thread:
ForkJoinPool.commonPool-worker-21]
IP: 192.168.0.3 --- Request from Client: 5 --- [Thread:
ForkJoinPool.commonPool-worker-21]
IP: 192.168.0.4 --- Request from Client: 7 --- [Thread:
ForkJoinPool.commonPool-worker-21]
IP: 192.168.0.5 --- Request from Client: 10 --- [Thread:
ForkJoinPool.commonPool-worker-3]
IP: 192.168.0.1 --- Request from Client: 6 --- [Thread:
ForkJoinPool.commonPool-worker-7]
IP: 192.168.0.3 --- Request from Client: 2 --- [Thread:
ForkJoinPool.commonPool-worker-21]
IP: 192.168.0.2 --- Request from Client: 13 --- [Thread:
ForkJoinPool.commonPool-worker-5]
IP: 192.168.0.3 --- Request from Client: 1 --- [Thread:
ForkJoinPool.commonPool-worker-19]
IP: 192.168.0.5 --- Request from Client: 4 --- [Thread:
ForkJoinPool.commonPool-worker-27]
IP: 192.168.0.4 --- Request from Client: 14 --- [Thread:
ForkJoinPool.commonPool-worker-23]
IP: 192.168.0.2 --- Request from Client: 8 --- [Thread:
ForkJoinPool.commonPool-worker-9]
IP: 192.168.0.1 --- Request from Client: 12 --- [Thread:
ForkJoinPool.commonPool-worker-13]
IP: 192.168.0.5 --- Request from Client: 11 --- [Thread:
ForkJoinPool.commonPool-worker-17]
IP: 192.168.0.4 --- Request from Client: 0 --- [Thread:
ForkJoinPool.commonPool-worker-31]
```

Clients starts to send requests to Weighted-Round-Robin Load Balancer

```
IP: 192.168.0.2 --- Request from Client: 9 --- [Thread: main]
IP: 192.168.0.2 --- Request from Client: 10 --- [Thread:
ForkJoinPool.commonPool-worker-27]
IP: 192.168.0.1 --- Request from Client: 3 --- [Thread:
ForkJoinPool.commonPool-worker-7]
IP: 192.168.0.1 --- Request from Client: 6 --- [Thread:
ForkJoinPool.commonPool-worker-3]
IP: 192.168.0.2 --- Request from Client: 13 --- [Thread:
ForkJoinPool.commonPool-worker-17]
IP: 192.168.0.1 --- Request from Client: 14 --- [Thread:
ForkJoinPool.commonPool-worker-5]
IP: 192.168.0.2 --- Request from Client: 4 --- [Thread:
ForkJoinPool.commonPool-worker-31]
IP: 192.168.0.2 --- Request from Client: 11 --- [Thread:
ForkJoinPool.commonPool-worker-19]
IP: 192.168.0.2 --- Request from Client: 7 --- [Thread:
ForkJoinPool.commonPool-worker-23]
IP: 192.168.0.3 --- Request from Client: 2 --- [Thread:
ForkJoinPool.commonPool-worker-3]
IP: 192.168.0.3 --- Request from Client: 5 --- [Thread:
ForkJoinPool.commonPool-worker-7]
IP: 192.168.0.3 --- Request from Client: 1 --- [Thread:
ForkJoinPool.commonPool-worker-21]
IP: 192.168.0.1 --- Request from Client: 0 --- [Thread:
ForkJoinPool.commonPool-worker-27]
IP: 192.168.0.1 --- Request from Client: 8 --- [Thread:
ForkJoinPool.commonPool-worker-13]
IP: 192.168.0.1 --- Request from Client: 12 --- [Thread:
ForkJoinPool.commonPool-worker-9]
```

5. Security and privacy: The cloud administrators must be sure of granting access to the right people. Also, orchestration may be required at each step of the process.

Know about the characteristics to understand more about the working and the structure of the cloud database services' platforms.

The majority of the services use web consoles by which the end-user can configure several database instances.

Like all kinds of database services, the database manager controls different underlying data modifications and storage and instances with the help of API. This service API is provided to the user for scaling operations on the database.

The scalability feature also differs among the different existing types, with some offering APIs and others going for auto-scaling options.

The software stack which is used for the entire system includes the database system, operating system(OS), and third-party software. Generally, the service providers are responsible for installing and updating the system for its smooth functioning.

Cloud services must be available and must ensure to protect the data without losing any part of it and without any kind of unwanted changes.

Conclusion:

Cloud database services load balancing delivers multiple benefits by optimizing resource use, data delivery, and response time. In high-traffic environments, load balancing is what makes user requests go smoothly and accurately. They spare users

the frustration of wrangling with unresponsive applications and resources.

Load balancing also plays a key role in preventing downtime and simplifying security, reducing the likelihood of lost productivity and lost profits for your organization. Other benefits of load balancing include the following:

Besides directing traffic to maximize efficiency, load balancing delivers the flexibility to add and remove servers as demand dictates. It also makes it possible to perform server maintenance without causing disruption for users since traffic gets rerouted to other servers during maintenance.

As the use of an application or website increases, the boost in traffic can hinder its performance if not managed properly. With load balancing, you gain the ability to add a physical or virtual server to accommodate demand without causing a service disruption. As new servers come online, the load balancer recognizes them and seamlessly includes them in the process. This approach is preferable to moving a website from an overloaded server to a new one, which often requires some amount of downtime..

7. References

[1]. Antony Rowstron and Peter Druschel, "Pastry:

Scalable, decentralized object location and routing for largescale peer-to-peer systems", IFIP/ACM International Conference on Distributed SystemsPlatforms.

[2]. Buyya R., Ranjan R., and Calheiros N., "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Cloudsim," in Proceedings of the Conference on Interactionand Confidence Building Measures in Asia,Lecture Notes in Computer Science, Istanbul,Turkey, pp. 189-196, 2010.

[3]. Hsu C. and Chen T., "Adaptive Scheduling Based on Quality of Service in HeterogeneousEnvironments," in Proceedings of the IEEEInternational Conference on Multimedia and Ubiquitous Engineering, California, USA, pp. 1-6, 2010.

[4]. Ijaz S., Munir E., Anwar W., and Nasir W.,"Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment," theInternational Arab Journal of Information Technology, vol 10, no. 5, pp. 486-492, 2013.

[5].Kessaci Y., Melab N., and Talbi E., "A Pareto- Based GA for Scheduling HPC Applications on Distributed Cloud Infrastructures," in Proceedings of the IEEE International Conference on High Performance Computing andSimulation, Istanbul, Turkey, pp. 456-462, 2011.

[6] M. Velicanu. Baze de date prinexemple, Ed. ASE, Bucharest, 2007.

[7] M. Velicanu. Dicţionarexplicativ al sistemelor de baze de date, Ed. Economică, Bucharest, 2005.

[8] M. D. Solomon, "Ensuring a Successful Data Warehouse Initiative ," Information Systems Management; 2005, pp. 26,.

[9] J. M. Hickand and J. L. Hainaut Database application evolution: A transformational approach, 2005

[10]. Hsu C. and Chen T., "Adaptive Scheduling Based on Quality of Service in HeterogeneousEnvironments," in Proceedings of the IEEEInternational Conference on Multimedia and Ubiquitous Engineering, California, USA, pp. 1-6, 2010.

[11]. Ijaz S., Munir E., Anwar W., and Nasir W.,"Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment," theInternational Arab Journal of Information Technology, vol 10, no. 5, pp. 486-492, 2013.

[12].Dr.Madhurendra Kumar,"Cloud computing Network Problem and Storage solutions Using Ant colony optimization"

International Journal of Scientific & Engineering Research Volume 7, Issue 12, December-2016, ISSN 2229-5518

[13] Shelly Rohilla, Pradeep Kumar Mittal, Database Security: Threats and Challenges, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013.

[14] Deepika, Nitasha Soni, Database Security: Threats and Security Techniques, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 5, May2015.

[15] Debasish Das, Utpal Sharma & D.K. Bhattacharyya, An Approach to Detection of SQL Injection Attack Based on Dynamic Query Matching, International Journal of Computer Applications, Volume 1, 2010.

[16] Shivnandan Singh, Rakesh Kumar Rai, A Review Report on Security Threats on Database, International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014.

[17] Debasish Das, Utpal Sharma, D.K. Bhattacharyya, An Approach to Detection of SQL Injection Attack Based on Dynamic Query Matching, International Journal of Computer Applications, Volume No.1– 25,2010.

[18]. Lorpunmanee S., Sap M., Abdul A., andChompoo C., "An Ant Colony Optimization forDynamic Job Scheduling in Grid Environment,"in Proceedings of World Academy of Science, English and Technology, 2007.