Efficient Horizontal Scaling of Databases using Data Sharding Technique

Ragul R, Arokia Paul Rajan R

Abstract*: *As the new era of computing technology begins with the start of the 21st century, the data produced by each person are increasing enormously day-by-day. Companies started to track one's personal details from applications like media sharing, social networks, eCommerce, etc. since they have started to generate a lot of data traffic. The Database, where these heavy loads of generated data have been stored, had to scale out to handle the network traffic. Cloud computing arrival carves a solution for this trafficking problem by making data sharding a suitable option for scaling the data. This paper discusses about the importance of database sharding and the distribution of the database over the different server and also how sharding is helpful to scale the incoming amount of enormous data to control the data trafficking issues in the cloud database. Also compares the difference in the performance of database with and without sharding techniques using SQL server instance.

Keywords*: *Distributed Database, Scalability, Database sharding, Horizontal sharding.

I. INTRODUCTION

The beginning of the 21st century made a way for technology improvising where people started to use online applications almost for everything. Web application requirements have been increased dramatically in scaling, Web 2.0 technologies made an interactivity attraction for the users. The new version and features in web application tools help the scripting programmer to improve the webpage more attractive which made the people use online applications more. Due to this, the data incoming is growing huge in number even behind our imagination. The data produced in social networks, eCommerce and media sharing are some of the best examples for increasing data production because of the huge number of users [1]. Concurrently, many companies developed (Service-oriented architecture) SOA-based applications making easier the integration between different systems and increasing the reuse of functionalities through services.

Scaling a Database system can be done in two ways, Scale up and Scale out. Scale up (Vertical Scaling) is enlarging the available resources to achieve the desired state of performance [2]. It is done by adding additional resources to the existing system. It is expensive since it requires disk storage, bigger machine, memory, and processors. So scaling up a database server is always not possible. The other way is Scale out (Horizontal Scaling) associates with different distributed architecture [3][4]. It distributes the larger database into several servers to handle and maintain the data and retrieve it back when needed.

Revised Manuscript Received on March 30, 2020.
* Correspondence Author

Ragul R, MSc Student, Department of Computer Science, CHRIST (Deemed to be University), Bangalore, India. Email: ragul.r@cs.christuniversity.in

Dr Arokia Paul Rajan *, Associate Professor, Department of Computer Science, CHRIST (Deemed to be University), Bangalore, India. Email: arokia.rajan@christuniversity.in

A. Sharding

The method of splitting the single logical dataset and storing it in multiple servers (shown in figure 1) or databases is known as Sharding. Let's assume a situation, an application has been created and becomes more popular suddenly without exception. So, the incoming data and traffic will start growing and the database will start getting overloaded day by day. So to handle the overwhelming of unexpected data, Sharding might play a vital role by scaling the dataset with different types [5].

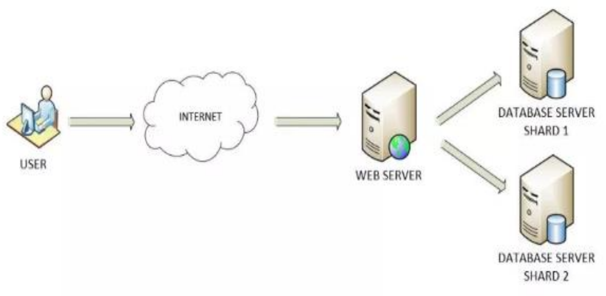


Fig. 1. A Simple sharding scheme

This paper discusses the Database Sharding techniques like vertical, horizontal, hash pattern, etc., and the aim is to achieve scale in the cloud database using any of this technique and to extemporize the situation where the unexpected overload of incoming data to be scaled efficiently using data sharding. Data sharding has various types to scale the database and to increase the performance. The different types are briefly explained below:

1.1.1. Horizontal Sharding

Sharding is almost most of the time referred to the horizontal way. It is a data architecture pattern related to horizontal partitioning, which is a method of splitting the one big table into different smaller tables. This way is also called partitioning. Here each partition will have columns and schema, but the rows will be with different datasets. So the tables which hold data will have different datasets that are independent and unique from other partition tables.

1.1.2. Hash Sharding

Hash or Key Sharding uses the hash index (value) which is the resultant generated when a value from each entity passed to a hashFunction. The resultant hash index decided which shard (database server) should be used to store the specific data. For example, corresponding to the figure below, assume 4 database servers and each server has an application id which will be incremented every time by 1 when a new application gets registered[5][6]. We can simply do modulo operation the application id by 4 since the number of servers is 4 and the resultant will decide the server for the application to be stored.

The drawback of this schema is that the dataset will not be evenly distributed. Again, assume the same student databases. What if more number of students studied in 2017 than 2007? Obviously, that will be the case. There are an increasing number of students each year. So here, then each shard will be unevenly distributed where some of the shards will be experiencing much heavier loads and some will experience fewer loads which results in system bottleneck.

1.1.3. Vertical Sharding

Vertical partitioning is one way of sharding technique which is a key idea in the utilization of various databases for different tasks. For different business purpose, an organization's database can consist of multiple data tables which were before sharding. But once the sharding was performed the tables will be arranged with their specific business and clustered to different databases, to balance the workload among the organization's various databases.

The drawback of this vertical sharding schema is that the application layer might need to combine back the dataset from the scaled shards to solve any queries depend on the system which increases the complexity of operation and development of the system [5][7]. And also if the system is updated with new features, there will be a need of providing new shards for the features to maintain the scalability.

II. RELATED WORKS

This research reviewed the literature extensively and the findings are summarized as follows:

[8] presents an idea about the Distributed Database problems and approaches. The ultimate goal of the paper is to find the issues in the Distributed Database and suitable way to solve it. Distributed Database is one modern way to store databases on various servers by splitting it with constraints. With Distributed Database, users can also create and store the dataset random in a network where the database is located. Because of this the problems raise in the database which includes data allocation, deadlock, replication. This research work also discusses about the various issues in the distributed database such as transaction management, distributed database design problem, concurrency control, etc. The following approach will help to reduce the Distributed Database issues. They are (i) Top-Down and (ii) Bottom-Up approach. Top-down will only be used when the database at the beginning implements the Distributed Database. Bottom-up will be used to add a new database to the existing one which is implemented with the Distributed Database. So, the paper discusses also about the various issues while implementing the Distributed Database into a server and different approaches to solve the problems.

[9] discussed how to implement the Distributed Database using Sharding technique. The authors briefly explained about the sharding technique, fault tolerance and also the scalability of an enormous load of databases in the cloud. Sharding (Horizontal partitioning) is the way to split the data among the data nodes, which is placed in the server. The use MySQL Cluster is to demonstrate the Sharded database prototypes with various data node clusters deployed using datasets. Horizontal partitioning is the way of splitting two rows into two different tables with a single logical unit. Storage area network (SAN) is a single shared where the partitioned data will be still considered in one place. The data will be scalable and always available in the cloud server even will any disaster occur. This architecture of the database which is sharded exhibits agility, scalability, and availability will be very helpful to manage big data on the cloud.

[10] focussed on the hash partitioning method of sharding. Since the start of the 21st century, the amount of data from social networks, media sharding, etc., has created a huge traffic on the network server. The database manages this server had to scale the data for efficiency. This issue had been handled well with the emergence of Cloud technology. Cloud computing provides not only the scalability but also security to the data which is also an affordable solution. The aim of this paper to provide the best sharding practice to handle the overwhelming data where the author thought Database Scalability Pattern where the data cluster use hash technique of sharding for load balancing in the server. This hash type of sharding technique formalizes the database scalability pattern to improve the data cluster's performance with assigning a server for the particular dataset. However, it is not the complete solution in all the cases because it does not solve all the database scalability problems. The authors continue their work with a database horizontal scalability pattern with the goal to solve the problems.

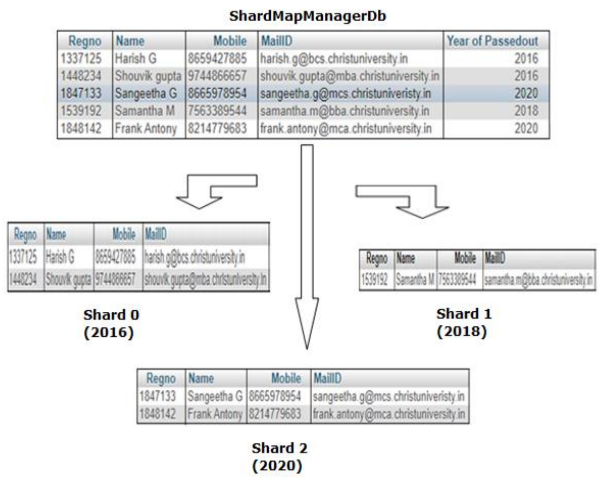
[11] worked on document-oriented open-source database MongoDB to use the sharding technique for scaling. MongoDB uses both vertical and horizontal technique for scaling where horizontal sharding use auto-sharding method to divide and distribute the data to different server where the databases administer applies the key to divide the dataset. The key here is to provide the right key for the performance and capability of the database. The selection in different shard key will help the MongoDB admin to understand about the shards and choose the correct key for the particular system. The work discusses about the range-based key to understand the performance of MongoDB after sharding key applied. The performance of databases depends on how widen the shard cluster is where the range of key is vital on some shards.

[12] deals with the security level of sharding in NoSQL databases. The drawback of sharding is that data is being distributed over the unsecured networks to different servers. This paper proposes comprising security techniques to divide and distribute the data in NoSQL sharded databases.

The analysis of this paper will help various organizations provide a reliable and trustworthy database with security. The problems in the existing system are identified and the databases were analyzed with assessment criteria. This helps for the security of the sharded database along with the analysis of the databases which helps the NoSQL customers in improvising the security of the sharded databases. It is also clear that there is no single complete solution for the security of the shards, but the organization still needs to consider each level of security for the data cluster which plans to implement the sharding technique in the database server.

III. PROPOSED METHODOLOGY

Horizontal way of sharding can also be referred to Range-Based sharding which splits the data in using the value of ranges that are derived from each entity. For example, let's assume a student database of the entire college, now we need to send an invite to each alumnus. This will be so easy if the database of the student has been partitioned by each year as a row because the other information like regno, name, mobile, email, etc. will be in columns. So, as each year of students can be stored in each shard as shown in figure 2.


ShardMapManagerDb

Regno	Name	Mobile	MailID	Year of Passedout
1337125	Harish G	8659427885	harish.g@bcs.christuniversity.in	2016
1448234	Shouvik gupta	9744866657	shouvik.gupta@mba.christuniversity.in	2016
1847133	Sangeetha G	8665978954	sangeetha.g@mcs.christuniveristy.in	2020
1539192	Samantha M	7563389544	samantha.m@bba.christuniversity.in	2018
1848142	Frank Antony	8214779683	frank.antony@mca.christuniversity.in	2020

Regno	Name	Mobile	MailID
1337125	Harish G	8659427885	harish.g@bcs.christuniversity.in
1448234	Shouvik gupta	9744866657	shouvik.gupta@mba.christuniversity.in

Shard 0
(2016)

Regno	Name	Mobile	MailID
1539192	Samantha M	7563389544	samantha.m@bba.christuniversity.in

Shard 1
(2018)

Regno	Name	Mobile	MailID
1847133	Sangeetha G	8665978954	sangeetha.g@mcs.christuniveristy.in
1848142	Frank Antony	8214779683	frank.antony@mca.christuniversity.in

Shard 2
(2020)

Fig. 2. Horizontal or Range based sharding

It also has its own benefits like it is the simplest sharding schema available. Also, the schema of the original database will be similar to each shard. In major scenarios, the application layer will be much simpler, combining the dataset from the scaled shard is no longer needed to solve any queries. It also works well for relative non-static data [13][14].

IV. EXPERIMENTAL SET UP & RESULTS DISCUSSION

Here, to demonstrate the simple shard technique an application has been created and the key capability of the database was also explored. Use cases for multi-shard querying, data-dependent routing, and shard map management are being focused. The client libraries are available for Java.

A. Configuration of Maven and SQL Server Instance

First Maven, SQL Server Instance and JDK have to be configured. Java program has been written for the configuration, to create shard, multi shard query, SQL database, etc. Build the JAR files and start the experiment. Student details were used in the program as a dataset.

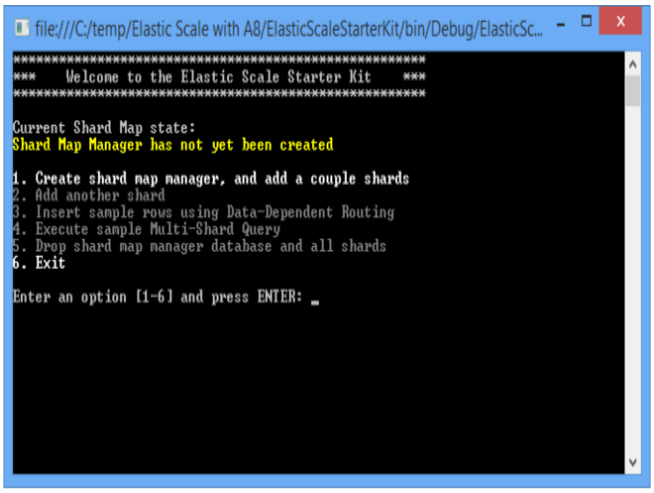
Maven has to be installed using the below command and to build the application in the ./sample directory,

```
mvn install
```

Execute the following command in command prompts to start the experiment:

```
mvn -q exec:java
"-Dexec.mainClass=com.microsoft.azure.elasticdb.samples.
elasticscalestarterkit.Program"
```

Once the server name, username and password in the file resources.properties sync with the SQL Server Instance, the output screen is presented in figure 3.

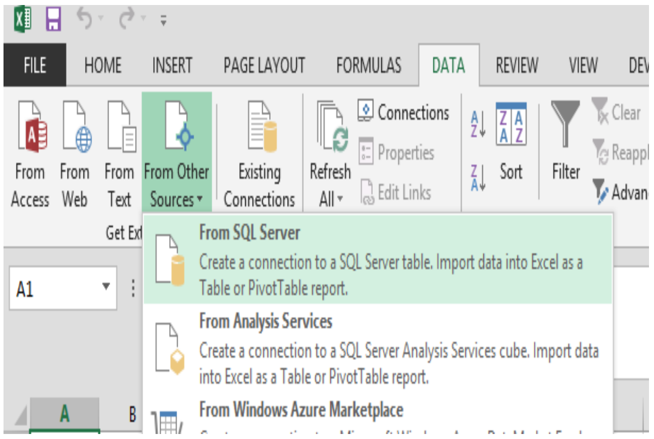

file:///C:/temp/Elastic Scale with A8/ElasticScaleStarterKit/bin/Debug/ElasticSc...

```
*************************************************************
***    Welcome to the Elastic Scale Starter Kit    ***
*************************************************************

Current Shard Map state:
Shard Map Manager has not yet been created

1. Create shard map manager, and add a couple shards
2. Add another shard
3. Insert sample rows using Data-Dependent Routing
4. Execute sample Multi-Shard Query
5. Drop shard map manager database and all shards
6. Exit

Enter an option [1-6] and press ENTER: _
```

Fig. 3. Application after the successful build

B. Creating the Shard using Microsoft Azure

In figure 4, Datasets are imported using the database query into the SQL Server Instance (i.e. import using the option "Select how you want to view this data in your workbook")



From SQL Server
Create a connection to a SQL Server table. Import data into Excel as a Table or PivotTable report.

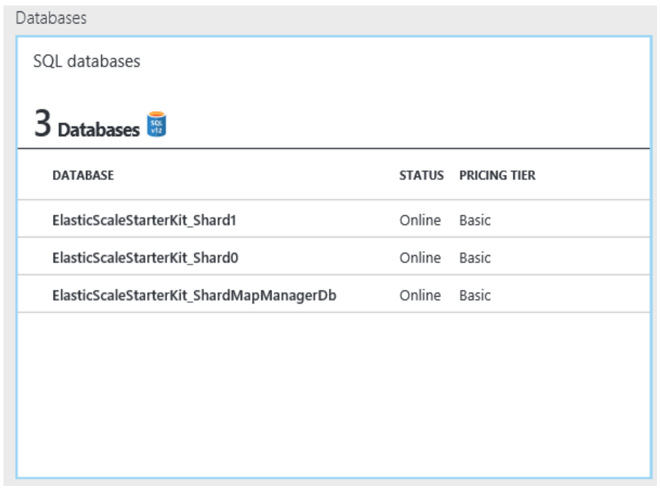
From Analysis Services
Create a connection to a SQL Server Analysis Services cube. Import data into Excel as a Table or PivotTable report.

From Windows Azure Marketplace

Fig. 4. Importing Datasets

Before creating the shard the imported datasets have to be saved in a database in Microsoft Azure. So, create an Azure SQL Database with the same server name used in the configuration. Then create the database objects. To create the object Open SQL Server Management Studio and connect the database using the SQL commands.

Run the application build in the above step. In command prompt, enter 1 to create the shard in the Azure [15]. This will create shards shown in figure 5.



Databases

SQL databases

3 Databases

DATABASE	STATUS	PRICING TIER
ElasticScaleStarterKit_Shard1	Online	Basic
ElasticScaleStarterKit_Shard0	Online	Basic
ElasticScaleStarterKit_ShardMapManagerDb	Online	Basic

Fig. 3. Creating the Shards using the same application

In this work, a database has been created for storing the students' detail. We used Christ University students' database for the process of sharding. Based on a few constraints like year wise students' details, sharding was done. After sharding, various queries related to the students were posted and the performance of the system has been recorded. It was observed that the response time of the system was minimal when the database sharded. The experiments were repeated many times and the performance was measured by the parameter response time. The results from the experiments were recorded and figure 6 represents the performance of the system with and without sharding.

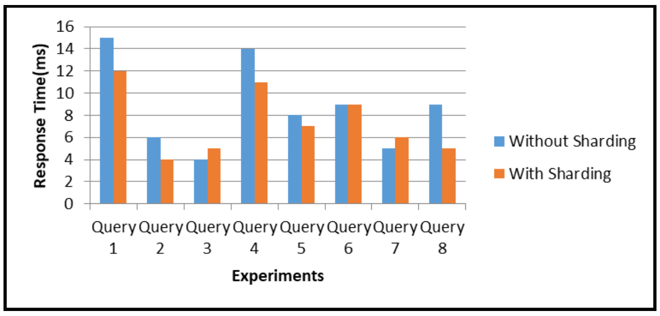


Fig. 6. Response time respect to Experiments conducted

V. LIMITATIONS

The different ways of sharding which have been seen above might make to think that data sharding is like an easy way to scale out the data on a server. But even the database sharding has its own limitations.

• Database Joins are not feasible and become more expensive in some cases.

• Database Referential Integrity is compromised by Database Sharding.

• Database Schema changes become extremely expensive.

Each time the database has been sharded, the set of various constraints is set on the operations that need to be performed on the server (database) [5]. Once, after sharding, the same table will not allow the multiple rows in it to run on the same server.

VI. CONCLUSION

This paper presented the detailed working and the importance of database sharding along with various types in it. Each sharding technique that is horizontal, hash and vertical has its own demerits and merits. So understanding the working principle of sharding is the key to control the performance and scalability of a huge amount of increasing data in the database server. The basic concept behind the database sharding is much simpler and not tedious to understand which will help to implement and scale the server efficiently. The experiments show a significance difference in the performance of systems with and without the sharding techniques.

REFERENCES

1. Principles of Sharding for Relational Databases, https://dzone.com/articles/principles-of-sharding-for-relational-databases.
2. Shadi Aljawarneh, (2015), "Advanced Research on Cloud Computing Design and Applications", 1st IGI Publishing Hershey.
3. Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, John Wilkes, (2011). "CloudScale: Elastic Resource Scaling for Multi-Tenant Cloud Systems". SOCC '11 Proceedings of the 2nd ACM Symposium on Cloud Computing Article No. 5.
4. Niloofar Khanghahi, Reza Ravanmehr, (2013). "Cloud Computing Performance Evaluation: Issues and Challenges", International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.3, No.5.
5. System Design Interview Concepts – Database Sharding, https://www.acodersjourney.com/database-sharding/
6. Róbert Lovas, Enikő Nagy, József Kovács, (2018)."Cloud agnostic Big Data platform focusing on scalability and cost-efficiency", Advances in Engineering Software, Vol.125, pp.167-177.
7. Neelu Sinha, Laila Khreisat, (2014). "Cloud Computing Security, Data, And Performance Issues", Wireless and Optical Communication Conference (WOCC), https://doi.org/10.1109/WOCC.2014.6839924.
8. D.S.Hiremath, Dr.S.B.Kishor, "Distributed Database Problem areas and Approaches", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727 PP 15-18.
9. Sikha Bagui et al, (2015). "Database Sharding: To Provide Fault Tolerance and Scalability of Big Data on the Cloud", International Journal of Cloud Applications and Computing, 5(2), 36-52.
10. Caio H. Costa, Joao Vianney B. M. Filh et al, (2015). "Sharding by Hash Partitioning A database scalability pattern to achieve evenly sharded database clusters", International Conference on Enterprise Information Systems.
11. Pakorn Kookarinrat, Yaowadee Temtanapat, (2015). "Analysis of range-based key properties for sharded cluster of MongoDB", 2nd International Conference on Information Science and Security (ICISS).
12. Rahat Masood, Awais Shibli, (2014). "Security of sharded NoSQL databases: A comparative analysis", Conference on Information Assurance and Cyber Security (CIACS).
13. Ab Rashid Dar, Dr.D.Ravindran, (2017). "Survey On Scalability In Cloud Environment", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 5, Issue 7.
14. Javier Espadas, Arturo Molina, (2013). "A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures", Future Generation Computer Systems 29, 273–286.
15. Get started with Elastic Database Tools, https://docs.microsoft.com/bs-cyrl-ba/azure/sql-database/sql-database-elastic-scale-get-started