Query routing at infrastructure level as load balancer at application and database server

Sourabh Joshi[#1], Nitin Patil[*2]

[#]Department of Computer Science, Savitribai Phule Pune University, Pune

[1]joshiss@hotmail.co.uk

[2]nitindocs@gmail.com

Abstract:

With growing innovation of technology and computing we also need to focus on optimization and performance of our computing resources, Consider working of application server, web server and DB server which are almost used in every Application/project that we design and develop, amongst these DB server are most important aspect. In Database server architecture we always use master and slave db, we always think and used slave Db for replication and as life saver if master db went down, here we need to think on the other side and start using slave db to serve the all read the request which comes from the client and segregate read and write queries at infrastructure level not from the application/code level changes as writing code every time is very complex and time consuming as well. We have a approaches to load balance the queries but that model only works for the multi master where only single slave is used between multiple writers. In this project we have used the multiple slave but not the multiple writer as frequency for read queries are more rather then write queries for most of the application today. Here we have tried to route the queries between the master and slave as a load balancer between application server and Database server, By which we can improve the performance of master Db and overall application performance at infrastructure level. We have use AWS service RDS Aurora Mysql which provide better performance than simple Mysql. In implemented system we have calculated the performance with respect to queries which are executed on writer only then calculated the performance of when the queries are routed properly using the round robin algorithm and based on the cluster endpoints provided for read cluster and write cluster.

Keywords - AWS Cloud, RDS, Master-slave Replication, Database, Load balancer

I. Introduction

In today's day and age, when any person hits the address bar of the of his/her web browser and type a URL, that each hit owes a big credit to Sir Mr Tim Berners-Lee, a British Scientist who gave a human community a master-key to the galaxy of the Information from the CERN Institute , Geneva in 1989. The three technologies designed and defined by Berners-Lee i.e. The URL, The HTML and The HTTP has played a significant role in creation of the Dot – Com – Bubble during 2001 which forced a generation of a decade to witness the IT revolution to design the bright future. When the critical analysis of WWW or "The Web" needs to be taken into account, the Web Servers, application servers and the database servers have become the real script writers of this whole story. This three tier architecture has become a foundation and performing a critical role to satisfy the client requests by through the HTTP. However, the more layers like load balancers have been added which have underlined and magnified the integration of this three tier architecture with new age technologies like cloud computing.

As the first Tier, the Web Servers defines themselves as the server software or the hardware dedicated to the running software which performs the processing of the incoming network requests by using the HTTP (Hypertext Transfer Protocol). The web servers act as a storage points for one or more websites aka web pages which not only include the text but also the images, videos and many more. However the web servers don't have any backend logic and can give the HTML as an output to all the requests received through the web browsers. However, this also underlines the limitations of the web servers i.e. processing of the high end web applications. The NGINX, GWS and Netware can be named as popular web servers which are preferable for computer geeks.

While defining the drawbacks of the web servers, the need of more enhanced, complex and powerfully integrated server automatically comes into play. The application servers has successfully won the battle over the web servers by showing their capabilities of processing and storing the high end applications, the addition of the backend logic and ability to work in sync with the Database servers. The layers like Load Balancers, failover and clustering also make them efficient in all the way and increase their reliability by handling the services as well as requests which a web server can't handle. The application servers like Apache, Glass Fish have marked their presence in this three tier Client/Server architecture. Working in sync with the application servers, the Database Servers are defined as the storage points or the repositories of the required files, queries and database services which satisfy the client's queries through either front –end or back-end. The Database Management Systems or DBMS like Mysql become the power house for the Database Servers which provides the database services functionality to the users. Database servers process the request by understanding them in the form of the Query Language of their own and convert each submitted query to server-readable form and execute it to retrieve results.

Adding to the database servers, the Master and the Slave architecture in MySQL makes them efficient and capable to serve the client's queries. Here the concept of MySQL replication is referred as the process that allows to maintain multiple copies of My SQL data that will get copied automatically from Master to multiple slave databases. In the one way replication process from Master to Slave, the Master database is used as Writer while the read operation can be spread across multiple Slave databases which provides the significant improvisation in scalability and can also be used to rectify the issues like the failover.

A. Problem Statement

In current system/application architecture of any project Database is core and crucial part, here in this , I would like to address the problem related to routing query or we can say query splitting. Which means write request will be routed to writer i.e. our master and read request will be routed to slave db which is always sitting idea in most cases. To take the full utilization of our master db as well as slave Database can we have the approach to manage them at infrastructure level just like ALB (application load balancer), we can also say it as query load balancer. By which we are not only improving the performance of our master but also the total turnaround time required to serve one request come to application.

B. Goals

- Design and implement a query routing to load balance queries on Database server so that load can be spit for reader and writer respectively, designing different architecture that will be used to improve the threshold and cpu utilization of database with multiple salve and multiple master.
- Utilizations of existing cloud resources and RDS aurora services with cluster endpoints which provide high fault tolerance.
- Compare the existing web server, application and Database and also various architecture by integrating with aurora mysql to split the queries in the right direction.

C. Motivation

In current use case we are using the Database server for every application also for every project it may be related to small scale project to a large extent like use of Artificial intelligence system or for data science project, we might come across a situation that database server stops responding to queries after a certain amount of load on database server. There are numerous of factors affecting it, some of them are like

1. Queries are not optimize,
2. Heavy traffic unable to bear the load.
3. Large amount of data is there and it takes time to fetch and scan each row in a table.

Some time a situation came that all these things come at the same time on database server, what we would do to improvise this is the question. It may be due to any of the thing from the above one can cause a memory or cpu utilization of the database server to go at peak, and a deadlock situation may occur and leads to scaling of server or either improve the queries which are causing the problem. There are some other techniques example use of indexing, but still we doing all the above workarounds will not help the database server to have the lesser load until the master-master or multi master database concept came across the database and load balance queries on two server or more then two master servers. With this we also develop the technique for slave database in case the master fail and become unresponsive it is used as a recovery server for master database. But it also has some of its advantages and some limitations. Unlike the master-master server serving the request to user or application, slave does not do anything for application it just seats ideal and used as a backup mechanism. In many applications to segregate load for master

and slave many code developers write a code module to send some functionality of application to send queries on reader and writer but its a complex and takes time to write and find a workable code that do not affect other functionality of application.

In this project we would be focusing on dividing the work between the slave and master not at code level but at infrastructure level, to achieve this we study the working of master and slave to figure out which type of work slave can perform. We check the performance of master and slave when the query wise routing is used, where master serves only write request and slave serves read request. Depending on that we have created different architectural approach to improve the performance of database server. Also checked tried to learn about some load balancing algorithm to be used while multiple slaves are in place for work.

II. Literature survey

A. Introduction

Today's era every application system design will be distributed and series of network connected with each other in structure, which includes many servers, storage node and network devices communicate in network. each node is formed using a series of resources such as CPU, Memory, network bandwidth, etc. Each resource has its own corresponding properties. It may include the Web server, Application servers, Subnets, Load Balancers, database servers, memcache or redis servers. They have different functionality at various levels. Many algorithms have been made improve the efficiency of communication and to increase the response time. Among them is load balancer which is used for distribution of load on the servers if one server is heavily loaded or it depends on the algorithm how it distribute traffic. But it is been used at web server and application server load to handle not a database level. until the cloud computing takes the boom in market, and use of technology was limited to some organisation and some particular number of clients only earlier the data set did not face any issues of data management as it was small and manageable in single bare metal server without the use of load balancing as workload was also less on that particular server. To make effective for the modern day business technology SQL organizations update and move there resources to costly, complex and bigger ones which can possibly scale as per the need for bare metal server but cloud provides as a reliability of cheaper rate and same configuration with built in advance features. Here in this project we tried to go in a systematic way, studying the current need of project with respect to application server database server and their communication, then we went to study various databases available on cloud and various services used in cloud as we need to match and cope up with upcoming and blooming technologies. We went on studying the AWS RDS service and its cluster mode. As it was useful for us in setting up master slave replication is an easy way and had its own benefits like we can have some ten replica server running for a single cluster, which help as to split the queries at various slaves and one master instance(server). Then we went to read some paper related with load balancing done at web server when the first client request come the study related to algorithm are well explained in later chapters.

B. Related study

Mayur M Patil [1] they proposed and describe choosing an index of the MongoDB, competently as a shared key for further horizontal scaling of the database. Even in there study it tried to study load balancer. more focused is provided on need for NoSQL databases in the current situation and elaborated advancement of document-oriented database - especially MongoDB by explaining a quantitative example that SQL databases are prone to deteri-
oration when data is overloaded as MongoDB comes with inbuilt load balancer features which makes it an advance solution in applications with high data load and high query performance. It describes the technology of sharding with auto load balancing functionality of MongoDB. What basically a sharding do is that the work load is distributed amongst the different shards which ultimately reduces the load on one server.

Jin-Ha Kim, Gyu Sang Choi, Member, and Chita R. Das [3], in their research use a cluster concept to handle the load using back-end forwarding scheme, called ssl-with-bf, that employs a low-overhead user-level communication mechanism like Virtual Interface Architecture (VIA) to achieve a good load balance among server nodes. They have well explain about the concepts of application server, database server and web server how they communicate with each other. They even tried to explain about cryptography algorithms for SSL. which help us to understand the detailed working of particular request. Simranjit Kaur and Tejinder Sharma. well explain about the cloud computing and some application of load balancing done in cloud, they also provide some algorithm used for load balancing. And concluded with central load balancing algorithm selection of better machine would be possible with no resource wastage.

Jian-Bo Chen, Tsang-Long Pao, and Kun-Dah Lee [4], In this experiment, they have checked the performance of load balancing architectures with a centralized database arrangement and a decentralized database arrangement. With a series of experiments, we can find the most appropriate arrangement for the database server. The experimental results show that when the number of client requests is small, using the

decentralized arrangement results in a lower average response time because no network communications are needed. But when the number of client requests is large, using the centralized database architecture can achieve higher performance because the database server can share the load of web Servers.

We try to study the working of aurora cluster with mysql and postgres, where in AWS documentation it explains about the Amazon Aurora DB cluster, which consists of one or more database instances, where instance are called as virtual server, and a cluster volume that manages the data for those DB instances. An Aurora cluster volume is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the DB cluster data. Two types of DB instances make up an Aurora DB cluster that is master and replica server.

III. Load balancing and Mysql Replication

A. Load balancing

Load balancing is used to facilitate networks and resources by providing a maximum throughput with minimum response time. Dividing the traffic between servers, data can be sent and received without major delay. Working of load balancing is an easy way is utilization of all the available server not letting them seat ideal. There are various kinds of algo available that will distribute traffic load between available servers. Its a basic need for every cloud project in today's era. Load balancing solutions usually apply redundant servers which help a better distribution of the communication traffic so that the website availability is conclusively settled [5]. There are various types of load balancing algorithms available, which can be fall mainly under two groups. In this section we will discuss these
two main categories of load balancing algorithms.

B. Algorithms

1) Static algorithms:

It divides the traffic equally between two or more than two servers. Using this method traffic on the servers will be maintained easily and consequently, it make the server to work in more efficient way and perfectly. The divides traffic equally, is known as round robin algorithm. but still there were lots of problems appeared in this algorithm and many reference materials can be found for improving this algorithm, one of them is weighted round robin was defined to improve the challenges present with round robin. In this algorithm each servers have been assigned a weight and according to the highest weight they received more connections. In the situation where all the weights are equal, servers will receive balanced traffic.

2) Dynamic Algorithms:

Dynamic algorithms, on the other hand, distribute tasks using the current activity and state of information of the server. If the workload of a any of the server becomes heavy then the request will be transferred to another server which is not loaded so heavily. but, selecting a proper server needs real time communication with the networks, which will lead to extra traffic added on system. In comparison between these two algorithms, although round robin algorithms based on simple rules, more loads conceived on servers and thus imbalanced traffic. Moreover, a different and detailed approach for load balancing is discussed in upcoming chapters, by using round robin algorithm. The main concept here is not to dig out the algorithm of load balancing but to use this load balancing algorithm in middle communication of application server and Database server. Currently, these algorithms are only being used in communication between DNS name resolver to web server and Application server. we would be using load balancing algorithms for distributing query request to master and slave databases.

C. Mysql Replication

Mysql Replication is a process where data from one MySQL database is copied automatically based on the configuration to one or more MySQL database servers. Copy is created and named as Master or also called as a Slave. Working of master and slave is completely different then each other which is explain in short.

1) Master and Slave:

Master can be referred as leader for all the slave databases, where Master always has all the read and write privileges for himself, it means on master we can perform read as well as write operation both. Where in case on slave we cannot perform the operation of write, it can only be in sync with master with read queries executed on it. Here is the diagram which illustrates the working of Master and slave replication.

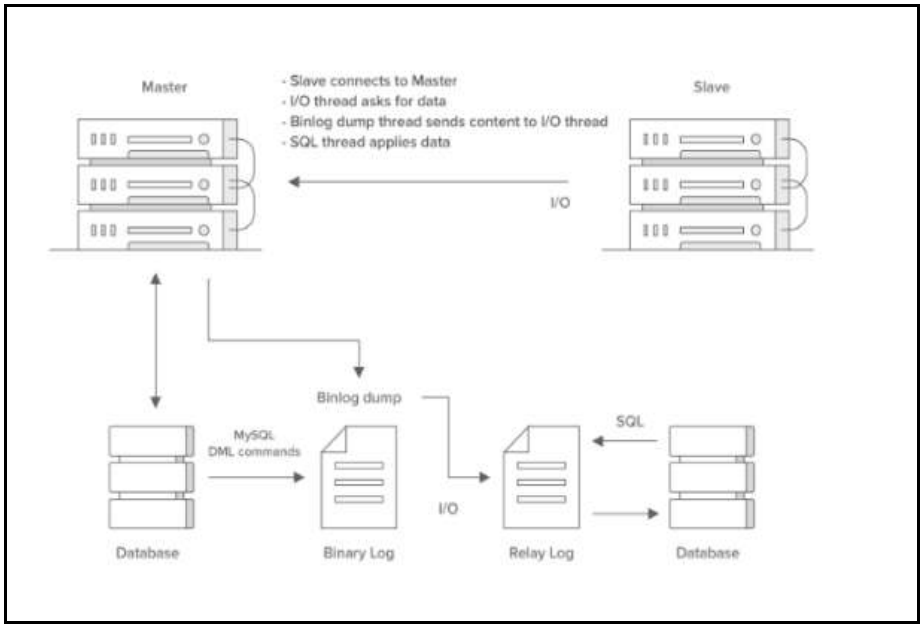

Figure. 1 Master Slave synchronization diagram

Based on these Master and slave concept different architecture are design and system are

2) Multi Master and Multi Reader Architecture:

Multi-master replication is also called as master master replication, It is a process in database replication in which data to be stored by a group of computers in a synchronous Way. In which all database group members are responsible to respond to client queries. The multi-master replication system have the responsibility to propagate the data modified by each writer or master member to the rest of the group, and resolve any conflicts that might arise between concurrency changes done by any other member.

Multi-master replication is opposite with master-slave replication, In which a single member of the group is designated as the ”master” for a given piece of data and is the only node allowed to modify that data item. Other members wishing to modify the data item must first contact the master node. Allowing only a single master makes it easier to achieve consistency among the members of the group, but is less flexible than multi-master replication. it distribute a write and read load on multiple master computers.

Where as the exact opposite behavior is shown for the multi reader architecture, where a single master is used and multiple read replica computers are in sync with the data. Where in this case only one master is present, while one or more other servers acts as slave. Master server writes updates in for every query in binary log files, and maintains an index of the files also keeps track of log rotation using not a single file will be of a huge size. Then this logs serve as a copy of updates to be sent to all slave servers were all slave updates its database to be in sync. Whenever any slave server form group of multiple slaves try to connect to the master server, it tells the master about its last position of logs since the last successfully update was done on its database. We can also check this in a Fig 1. where it gives you some detail insights, The slave check and takes up all the updates that have occurred since then last update was done.

IV. Query routing and load balancer at application and database level

A. Introduction

In this section, we will discuss system architecture of the proposed model using proxysql method and its working and how we can use it with various algorithms previously used in literature. In the preceding chapters, I have discussed on some terminology of Load balancing with respect to keeping two master and a single slave. As we already know, a lot of work has been done in the literature on the imputation of data but those work has a lot of limitations to route the queries to reader at infrastructure level. Our System architecture consists of some well known attribute which has different application in our paper and those are well explained using the diagrammatic way in Fig 2.

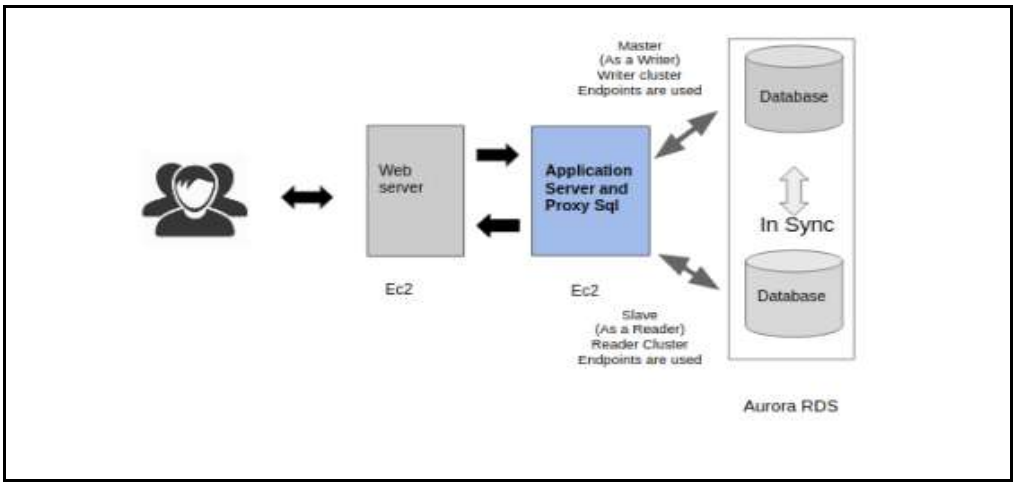

Fig. 2 Implemented Load balancing basic Architecture

B. System Architecture

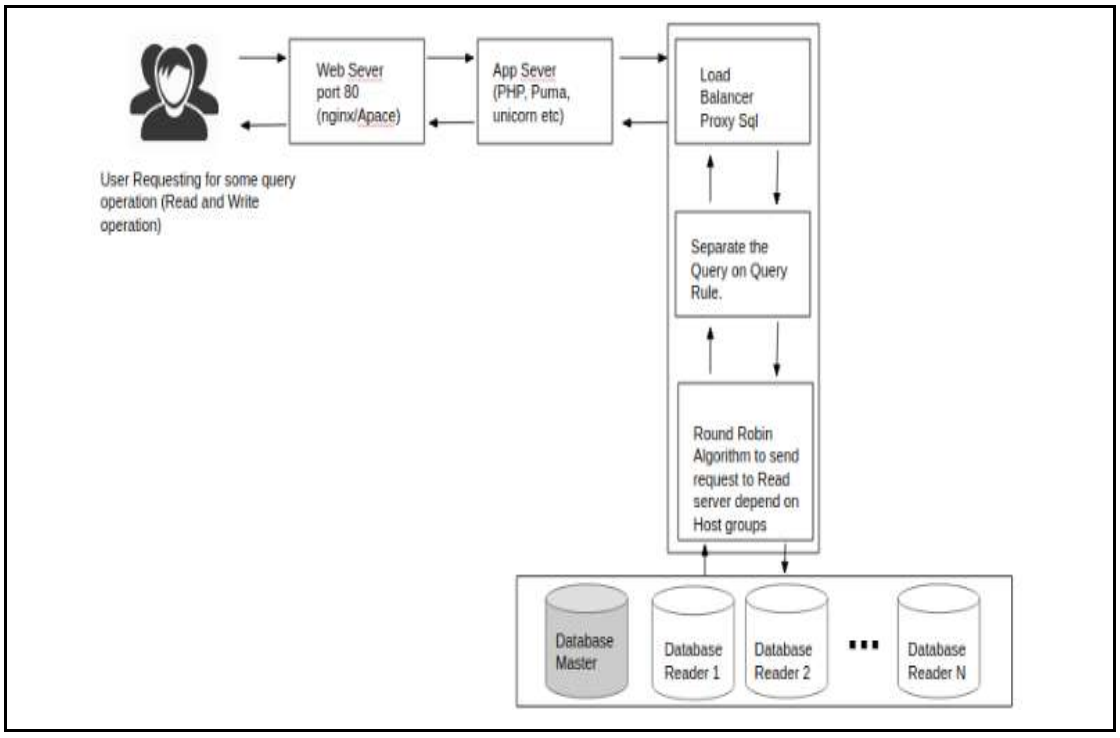

Fig. 3 Proposed System Architecture

C. Formulation approach

We begin by presenting the basic concepts and notations used in cloud computing then some advantages of AWS Aurora RDS and Proxysql structures in detail.

1) EC2: EC2 is used as storage, processing, and Web services of AWS for customer as a virtual server. EC2 is a virtual computing environment, that can be launched with a variety of operating systems and instance types and configure them with your custom application.

2) RDS: Amazon Relational Database Service is a web service from amazon that it makes easy and ready made set up, operate, and scale a relational database in the cloud with a minimum efforts.

3) Endpoints: It is a remote server IP address that is communicated back and forth within a network to which it is connected.

4) Writer Endpoints: writer endpoints is the master server IP address mapped with certain naming conventions, if use the dig +short ¡Endpoint¿ it will provide as mapped IP address. Similar goes with the reader which is acts a slave Database.

Writer endpoint example : test-db-mysql57.cjgxubxnop988ge.ap-south-1.rds.amazonaws.com

Reader Endpoint example: test-db-mysql57-ap-south-1b.cjgxxnosp98ge.ap-South-1.rds.amazonaws.com

5) Cluster: I had also taken advantage of clustering used in mysql, as cluster is a technology providing shared-nothing clustering and auto-sharding for the MySQL database management system. It is designed to provide high availability and high throughput with low latency, while allowing for near linear scalability.

Example of cluster endpoints :
Writer Cluster: test-db-mysql57.cluster-cjgxxsunsop98ge.ap-south-1.rds.amazonaws.com
Reader Cluster: ech-db-mysql57.cluster-ro-cjgop98ge.ap-south-1.rds.amazonaws.com

The implemented model explains Aurora ProxySQL configuration that can be used to test query routing and load balancing basic setup would be needed is explained below.

Aurora RDS with a Cluster (one writer and one reader) defined as the ’mtech-db- mysql57’ cluster. Each of the Aurora instances are added to separate host groups using the instance endpoints (not the reader / writer endpoints).

EC2 server for proxysql where mysql query rule and hostgroup is added. And Algorithm is changed to divert the queries to proper slave server.

In addition to this we can also setup a web and application server for purposed model which is explained in detail in system architecture. Once our cloud server are up and running we need to configure them to communicate with each other for that we would need endpoints and some configuration at proxysql level. Which is well explained in proxysql configuration part.

D. Proxysql Configuration and Aurora RDS Integration

Proxysql Includes following table to configure, these are the table which are used for making connection with RDS mysql.
1) myql_server
2) mysql_user
3) Mysql_query_rules

Configuration for RDS endpoints are provided using the host groups in mysql_server table, concept of hostgroup is a group of host with logical functionalities.

For example, you can have the production master in hostgroup123, all the production slaves in hostgroup321 and DB slaves in hostgroup432 if required.

A very basic query split can be done by sending to slaves all SELECTs without FOR UPDATE, and to send every other query to master, these rules are specified in the mysql_query_rules, depending on user requirement we can update the rules my looking into the query digestion logs Mysql_user table is used for configuration of connecting
user to RDS database, which has all the privileges for accessing DB.

Command to test the connection to test read queries are routed to proper

***reader only**: mysql -u utest -ptest -h 127.0.0.1 -P6033 -e "SELECT @@hostname"*

Command to test the connection of all other queries are redirected to proper hostgroups :

mysql -u <username> -p<password> -h 127.0.0.1 -P6033 -e "SELECT @@hostname FOR UPDATE

E. Algorithms

In this section we will explain to you how the various load balancing algorithms can be used with proxysql to load balance the multiple reader and multiple writers as well.

1) Round Robin Algorithm:

Similar to our old round robin algorithms, It selects the server sequentially in the list. Once it reaches the end of the server, the algorithm forwards the new request to the first server in the list. types in round robin algorithm explained below.

• Weighted round robin: uses the weight allocation for the server to forward the request.
• Dynamic Round Robin Algorithm: uses real time updated weight list of the server to forward the request.

2) Pseudo Code for Task Scheduling Process Based on RR:

1. The scheduler maintains a queue of ready Processes and a list of blocked and swapped out processes.

2. The PCB of newly created process is added to the end of the ready queue. The PCB of terminating process is removed from the scheduling data structures.

3. The scheduler always selects the PCB at the head of the ready queue.

4. When a running process finishes its slice, it is moved to the end of the ready queue.

5. The event handler perform the following action,
(a) When a process makes an input -output request or swapped out, its PCB is removed from the ready queue to blocked/swapped out list.
(b) When input-output operation awaited by a process finishes or process is swapped in its process control block is removed from blocked/swapped list to end of the ready Queue.

3) Least connection Algorithm:

This algorithm selects the server with few active trans-action and then forwards the query request to available database depending on the read or write query.

V. Implementation and Results

We will use the available dataset for measuring the performance of aurora and proxysql with respect to the algorithms used in with integration of proxysql. We provide comparisons with multiple read write queries executed on mysql Aurora RDS. We will also check the outcome when the load balancer is not used and what is the outcome when we redirect the queries on reader and load on master is also checked, Finally the experimental outcome will be provided and practical implications of the implemented system.

To measure the performance of the database we are also using the sysbench tool where multiple queries can be fired in a single shot, as it

provides the builtin script to check the performance but we modify the script to check the performance based on standard dataset.

A. Dataset Description

The dataset used for our implemented model is taken from the mysql site, The Employees sample database was developed by Patrick Crews and Giuseppe Maxia and provides a combination of a large base of data (approximately 160MB) spread over six separate tables and consisting of 4 million records in total.[3].

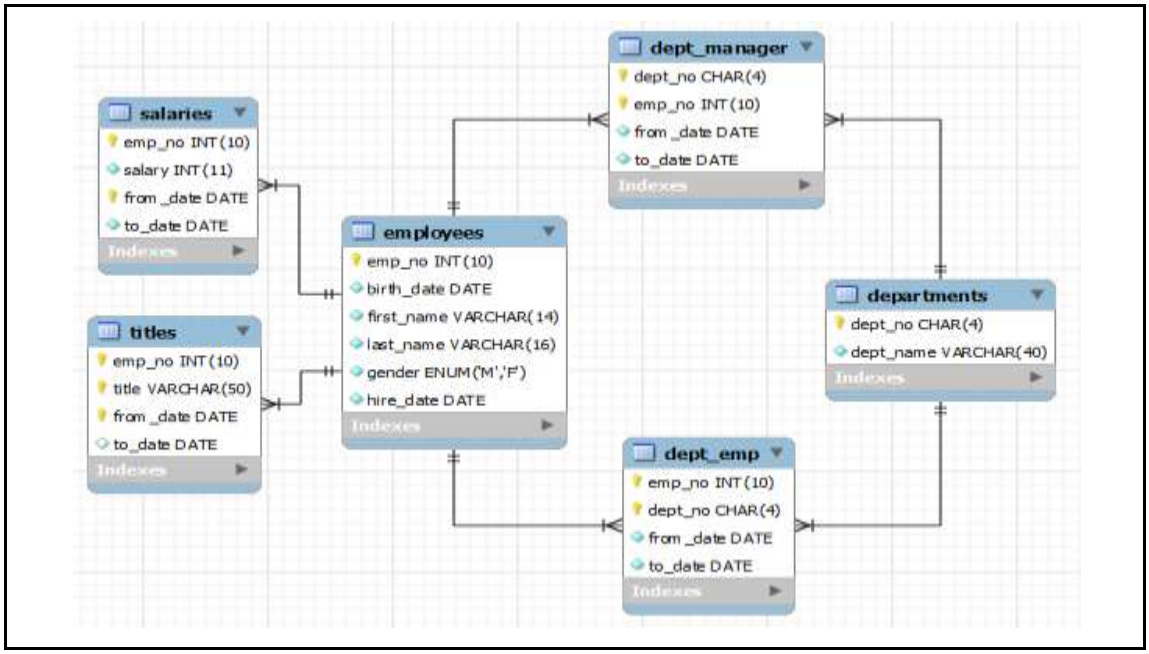

Fig. 4 dataset schema architecture

B. Performance Evaluation

For performance evaluation we have used the sysbench tool to fire up some multiple queries on a database server which include the script for testing performance of database. SysBench is a modular, cross-platform and multi-threaded benchmark tool for evaluating OS parameters that are important for a system running a database under intensive load. The idea of this benchmark suite is to quickly get an impression about system performance without setting up complex database benchmarks or even without installing a database at all.

There are some default oltp benchmarking script present in sysbench, which include by default database which create some default table with multiple rows in it. oltp This test mode was written to benchmark a real database performance. In our paper we made the changes in sysbench script to work with provided employee data set. Which is explained above.

Some examples of how to test the performance, here are the two commands

sysbench --test=<scriptname> --mysql-table-engine=myisam --oltp-table-size=1000000 --mysql-socket=/tmp/mysql.sock prepare

sysbench --test=<scriptname> --mysql-table-engine=myisam --oltp-table-size=1000000 --mysql-socket=/tmp/mysql.sock run

1) Response Time: Response time is the time it takes to start responding, not the time it takes to output the response. Large response time is a drawback in round robin architecture as it leads to degradation of system performance.

2) CPU Utilization: the amount of cpu used when the complete operation performed on server.

3) Turnaround Time: Turnaround time is sum of periods spent waiting to get into memory, Waiting in ready queue, executing on CPU and doing input output. It should be less.

4) Throughput: Throughput is called as a number of processes completed per unit time. Throughput will be slow in round robin scheduling implementation. Context switch and throughput are proportional to each other.

5) Waiting Time: Waiting time is the amount of time a process has been waiting in the ready queue. The CPU scheduling algorithm does not affect the amount of time during which a process executes or does input-output it affects only the amount of time that a process spends waiting in ready queue.

C. Results and Discussion

Latency and cpu measure sysbench Read and Write script executed on Aurora db cluster directly.

1) Sysbench command executed:

sysbench /usr/share/sysbench/employeedb_read_write.lua --mysql-db=legend --mysql-user=flash -mysql- host=mtech-db-mysql57.cluster-cjgxxnop98ge.ap-south-1.rds.amazonaws.com --threads=16 --db- driver=mysql --mysql-password=<dbpassword> run

2) Output :

```
Running the test with following options:
Number of threads: 16
Initializing random number generator from current time

Initializing worker threads...

Threads started!

SQL statistics:
    queries performed:
        read:                            28546
        write:                           8154
        other:                           4077
        total:                           40777
    transactions:                        2038   (202.50 per sec.)
    queries:                             40777  (4051.60 per sec.)
    ignored errors:                      1      (0.10 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          10.0624s
    total number of events:              2038

Latency (ms):
         min:                                   33.71
         avg:                                   78.81
         max:                                  567.46
         95th percentile:                      161.51
         sum:                                160622.86

Threads fairness:
    events (avg/stddev):           127.3750/4.27
    execution time (avg/stddev):   10.0389/0.02
```

Fig. 5 Snapshot for Script executed on Aurora write cluster directly

3) Cpu Utilisation graph

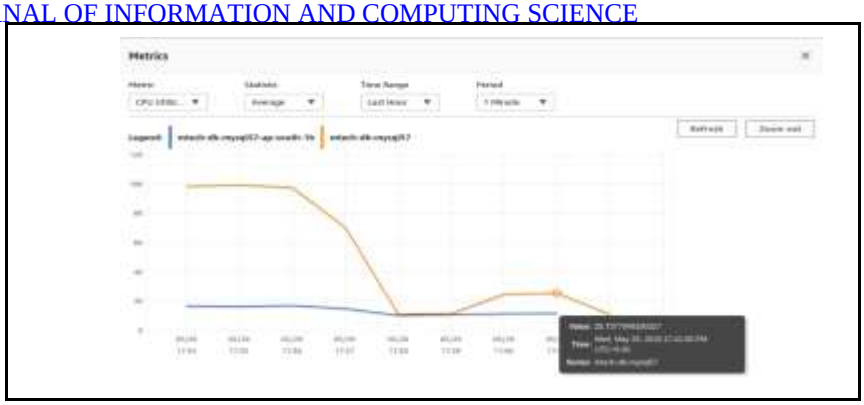

Fig. 6 cpu utilisation graph

The first graph Fig. 6 shows as the named "legend" master and slave cpu utilization. Yellow line is for master database and blue line is used for slave database. Its show when the queries executed on db server cpu went to 25.73 percent in case of master database, but for slave it remain ideal. second graph shows as the legend master and slave cpu utilization. Yellow line is for master database and blue line is used for slave database. Here in this execution we have executed query on both the server read queries are executed on slave database that is blue line and write queries are executed on master. Its show cpu utilization of master is improved than the previous case and slave db cpu utilization went to 22 percent.

Latency and CPU measure sysbench script Read and Write script executed on Proxy sql and Aurora Mysql RDS cluster.

1) Sysbench command executed :

sysbench /usr/share/sysbench/employeedb_read_write.lua --mysql-db=legend --mysql-user=flash --mysql-host=127.0.0.1 --mysql-port=6033 --mysql-password=<dbpasowrd> --threads=16 --db-driver=mysql run

2) Output:

```
Running the test with following options:
Number of threads: 16
Initializing random number generator from current time


Initializing worker threads...

Threads started!

SQL statistics:
    queries performed:
        read:                            29302
        write:                           8372
        other:                           4186
        total:                           41860
    transactions:                        2093   (208.23 per sec.)
    queries:                             41860  (4164.50 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
    total time:                          10.0497s
    total number of events:              2093

Latency (ms):
         min:                                   28.69
         avg:                                   76.63
         max:                                  424.97
         95th percentile:                      158.63
         sum:                                160390.90

Threads fairness:
    events (avg/stddev):           130.8125/9.81
    execution time (avg/stddev):   10.0244/0.01
```

Fig. 7 Snapshot for Script executed on Proxy server and Aurora cluster

3) Cpu Utilisation graph

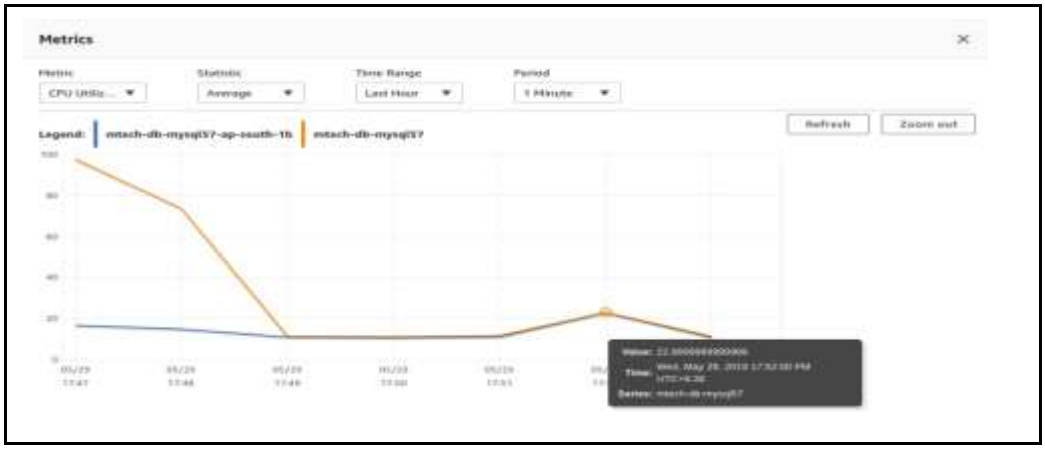

Fig. 8 cpu utilisation graph when query is routed.

VI. Conclusion

As the technology is moving in a faster way, cloud technology is taking a rapid growth in the IT Industry, all the real world and IT infrastructure is begin move to Cloud, where more resources with less price is available, but the basic concepts are still being used with different naming conventions so the working remains the same. Load balancing and Database with application server for read replica is not begin used in most case, we have seen in different article and research work related to load balancing is related with Multi Master database only, but the case with slave remain untouched and as it still consumes the memory and CPU utilization which is only used in case of disaster recovery, here in this work we tried to balance the load between the master and slave by splitting the queries, by separating the all the select queries to route to reader only and rest of the queries on writer so as to minimise the load which was coming on every master in case of multi master architecture is used. We have got an improved calculation based on our implemented system where we used dataset of employee where each table include some millions of row and by using sysbench tool we have evaluated the result which are in positive side. We have created only a single read replica and tested the bulk of queries on single master and single reader only, it can be possible to create multiple reader and use the same script to load balance among them it can be useful in the application where the database size is very huge and user request count is very high this scenario can have the scope to extend this work. We can also think about implementing this solution on some other databases like postgres.

References

[1] Mayur M Patil, Akkamahadevi Hanni, CH Tejeshwar, Priyadarshini Patil., Sharding in MongoDB and its advantages, International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), IEEE, 2017.

[2] Saranya, D., and L. Sankara Maheswari, Load balancing algorithms in cloud computing: a review, International Journal of Advanced Research in Computer Science and Software Engineering Research Paper, 2015.

[3] Jin-Ha Kim, Gyu Sang Choi, Chita R. Das, An SSL Back-End Forwarding Scheme inCluster-Based Web Servers, IEEE Transaction on Parallel and Distributed system, 2007, vol 18, no. 7.

[4] Jian-Bo Chen, Tsang-Long Pao , Kun-Dah Lee, Effect of Database Server Arrangement to the Performance of Load Balancing Systems, Springer, 2009

[5] Employee dataset, https://dev.mysql.com/doc/employee/en/employees- introduction.html.

[6] Cloud computing bible BY BARRIE SOSINSKY, 2011.

[7] Choi, E. Lim, Y.Min, Performance Comparison of Various Web Cluster Architectures, LNCS, Springer, 2005, vol. 3398.

[8] Harvinder singh and Rakesh Chandra Gangwar, Comparative Study of Load Balancing Algorithms in Cloud Environment, International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169, Volume: 2 Issue: 10.

[9] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Approach, Tata McGraw-HILL Edition, 2010.

[10] Hellerstein, J.M., Stonebraker, M., Hamilton, Architecture of a Database System.Foundations and Trends , October 2007.

[11] Multi master replication, https://en.wikipedia.org/wiki/Multi-master-replication

[12] Haney, D, Madsen, K.S Load-balancing for MySQL Kobenhavns Universitet, 2003

[13] Sysbench manual, 28 http://imysql.com/wp-content/uploads/2014/10/sysbenchManual.pdf

[14] AWS service RDS Aurora Mysql, https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide

[15] Simranjit Kaur, Tejinder Sharma, Efficient Load Balancing using Improved Central Load Balancing Technique, IEEE, 2018.