International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 03, Issue – 03, March 2018, PP – 54-60

An Innovative Approach for Managing Ever-growing Voluminous data of eGovernance Applications using NewSQL - Sharding

Lavanya S,
Computer Science and Engineering, Panimalar Engineering College/ Anna University, India)

Priyanka V,
Computer Science and Engineering, Panimalar Engineering College/ Anna University, India)

Preethi T,
Computer Science and Engineering, Panimalar Engineering College/ Anna University, India)

Mrs. Kavitha Subramani,
Associate Professor,
Computer Science and Engineering, Panimalar Engineering College/ Anna University, India)

Abstract: Now a days, the web applications especially e-governance applications are becoming slower because of the heavy load at the backend especially due to the data uploads and downloads, and not processing it efficiently. Thus in this research, we introduce a new database architecture for efficiently managing ever-growing voluminous data and thereby improving the speed of the web applications' response.
Keywords: NewSQL, Sharding, Aging, Scalability, Consistency, Partition tolerance

I. INTRODUCTION

Relational Database Management Systems (RDBMS) are de facto for e-governance for more than 3 decades. RDBMS which ensures consistency is restricted to a Single Box Server by nature and cannot be horizontally scaled. For some e-governance applications, 5-10 years data has to be retained for a meaningful decision making and so the old data cannot be aged and has to be kept in a single server. Government wants its services to reach citizens at their door step. So, many applications are adding every day. On the other side, NoSQL technology provides availability and partitioning but sacrifices consistency unlike RDBMS. This technology is used mostly for social media applications like Facebook, Twitter etc. Many API's evolved in the Open Source Domain to incrementally push the RDBMS to NoSQL Cluster but lack of consistency of data, hardware , software requirements impeded its adoption. NewSQL is a class of modern relational database management systems that seek to provide the same scalable performance of NoSQL systems for online transaction processing (OLTP) read-write workloads while still maintaining the ACID guarantees of a traditional database system. It's alternative to RDBMS for processing large datasets collected via e-governance applications. Most of the NewSQL products like Google Spanner, VoltDB, NuoDB, Clusterix DB are commercial whereas e-governance is a free service to the society. Citizens' data cannot be locked by vendor, besides the cost has to be kept nominal. Hence, Open Source is the best option. Cockroach DB is the best NewSQL product available as on date. Cockroach DB is almost an extension to PostgreSQL, the most widely used RDBMS. SQL, Drivers etc which are used in PostgreSQL can be used as it is. Here, migrating the data is easier since it is similar to PostgreSQL. The ultimate aim of this project is to impart scalability and consistency in rapidly growing and no-aging e-governance data through NewSQL-CockroachDB.

II. OBSERVATIONS

NIC is technically managing many websites for State and Central Govt. Government want to serve citizens at their doorsteps or from Community Service Centres(CSC). Most come applications require RDBMS as their default strategy. Some applications like Police etc aging data is impossible.

III. EXISTING SYSTEM

As the data multiplies, partitioning becomes unavoidable but RDBMS does not allow us to partition the data and provide consistency at the same time. Horizontal scaling is not possible. It is bound to a Single Box by nature. RDBMS jobs are CPU centric. It is not possible to increase CPU count every year. Since Sockets and CPU cores are limited. As the data grow the performance will degrade.

1. NoSQL

NoSQL(Not Only SQL) technology is based on the Availability and Partition without Consistency. The technology is used mostly for Social Media Applications like Facebook, Twitter etc. Many API's evolved in the Open Source Domain incrementally push the RDBMS to NoSQL Cluster. But lack of Consistency of data, Hardware, Software requirements impeded its adoption. Even with effort, the real time Online Transaction Processing (OLTP) feature was unavailable. Except for Social Media Applications, Time Series Data Application NoSQL is nowhere near to replace the RDBMS.

2. RDBMS

Advances in the complexity of information cause another drawback to relational databases. Relational databases are made for organizing data by common characteristics. Complex images, numbers, designs and multimedia products defy easy categorization leading the way for a new type of database called object-relational database management systems. These systems are designed to handle the more complex applications and have the ability to be scalable. Some relational databases have limits on field lengths. When you design the database, you have to specify the amount of data you can fit into a field. Some names or search queries are shorter than the actual, and this can lead to data loss. Complex relational database systems can lead to these databases becoming "islands of information" where the information cannot be shared easily from one large system to another. Often, with big firms or institutions, you find relational databases grew in separate divisions differently. For example, maybe the hospital billing department used one database while the hospital personnel department used a different database. Getting those databases to "talk" to each other can be a large, and expensive, undertaking, yet in a complex hospital system, all the databases need to be involved for good patient and employee care.

IV. PROPOSED SYSTEM

NewSQL is a class of modern relational database management systems that seek to provide the same scalable performance of NoSQL systems for online transaction processing (OLTP) read-write workloads while still maintaining the ACID guarantees of a traditional database system. It is also Horizontally Scalable. It is deployed on a Cloud environment. Alternative to RDBMS for processing large datasets which are collected through e-governance applications. Here, primary Key for each table is a must.

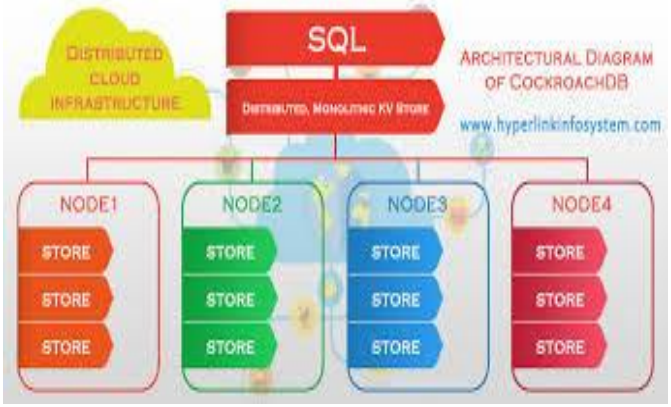
1. CockroachDB

Cockroach DB is almost an extension to PostgreSQL, the most widely used RDBMS. SQL, Drivers etc which are used in PostgreSQL can be used as it is. Migrating the data is easier since it is similar to PostgreSQL. Aim of the Project is to impart scalability and consistency in e-governance projects through newSQL-CockroachDB. This is the only Open Source equivalent to our current RDBMS products like PostgreSQL.

2. Why CockroachDB?

NewSQL most of the products are Commercial/in-memory. Google Spanner, VoltDB, NuoDB, Clusterix DB are commercial products. e-governance is a free service to the society. Citizens' data cannot be locked by Vendors. Besides the cost has to be kept nominal. Hence, Open Source is the best option. Cockroach DB is the best community software available in NewSQL as on date.

V. ARCHITECTURE OF COCKROACH DB



VI. CAP THEOREM

The CAP theorem, also named Brewer's theorem after computer scientist Eric Brewer, states that it is impossible for a distributed computer system to simultaneously provide more than two out of three of the following guarantees: Consistency (Consistency in database systems refers to the requirement that any given database transaction must change affected data only in allowed ways. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.), Availability (availability is the condition wherein a given resource can be accessed by its consumers at any given tim), Partition tolerance i.e. any Database be it RDBMS or NoSQL-Store has at-least two features out of three which is mentioned above. But in our proposed system, we have overcame this drawback by having Hybrid Database Architecture (Any RDBMS, for instance postgreSQL for storing Transaction based information and the metadata and Apache Cassandra NoSQL datastore for storing attachments) we can attain all the three guarantees namely, Consistency, Availability, Partition tolerance.

VII. MODULES

1. Configuration

Configure the New-SQL database and CockroachDB database (Back-end) To configure the new-SQL datastore to manage file attachments, first the Cockroach Labs should be installed in a server node and while installing, the number of replicating nodes with its IP address is configured. It may even be altered after the installation process also. On each server, add a non-root user with sudo privileges.

Make sure that TCP traffic is allowed on the following ports. If you have set up UFW as your firewall you will need to allow these two ports on each server:

- 26257 for inter-node and application communication: sudo ufw allow 26257/tcp
- 8080 for the Admin UI: sudo ufw allow 8080/tcp

For configuring CockroachDB database for storing the metadata and Non-Transaction Based data(in order to maintain concistency), the CockroachDB executable file is downloaded from online and installed in the main server; but all the eGovernance applications are using PostgreSQL as the backend. So, for maintaining Backward compatabilty, we are not changing any table structure that is already present in the existing system.

2. Setting up a cluster

A cluster can be set up using two methods as follow From Binary and using Docker. Here, we use binary.

STEPS:

Step 1. Start the first node

```
cockroach start
--insecure \
--host=localhost
CockroachDB node starting at 2018-03-12 15:10:52.34274101 +0000 UTC
build: CCL v1.1.6 @ 2018/03/12 14:48:26 (go1.8.3) admin: http://localhost:8080
sql: postgresql://root@localhost:26257?sslmode = disable
```


logs: cockroach-data/logs
store[0]: path=cockroach-data status: initialized new cluster
clusterID:{dab8130a-d20b-4753-85ba-4d8956a294c}
nodeID: 1
This command starts a node in insecure mode, accepting most cockroach start defaults.
• The --insecure flag makes communication unencrypted.
• Since this is a purely local cluster, --host=localhost tells the node to listens only on localhost , with default ports used for internal and client traffic (26257) and for HTTP requests from the Admin UI (8080).
• Node data is stored in the cockroach-data directory. • The standard output gives you helpful details such as the CockroachDB version, the URL for the admin UI, and the SQL URL for clients.

Step 2. Add nodes to the cluster
At this point, your cluster is live and operational. With just one node, you can already connect a SQL client and start building out your database. In real deployments, however, you'll always want 3 or more nodes to take advantage of CockroachDB's automatic replication, rebalancing, and fault tolerance capabilities. This step helps you simulate a real deployment locally.

In a new terminal, add the second node:

```
cockroach start \
 --insecure \
 --store=node2 \
--host=localhost \
--port=26258 \
 --http-port=8081 \
--join=localhost:26257
```

In a new terminal, add the third node:

```
cockroach start \
 --insecure \
 --store=node3 \
 --host=localhost \
 --port=26259 \
 --http-port=8082 \
 --join=localhost:26257
```

The main difference in these commands is that you use the --join flag to connect the new nodes to the cluster, specifying the address and port of the first node, in this case localhost:26257 . Since you're running all nodes on the same machine, you also set the --store, --port, and --httpport flags to locations and ports not used by other nodes, but in a real deployment, with each node on a different machine, the defaults would suffice.

Step 3. Test the cluster
Now that you've scaled to 3 nodes, you can use any node as a SQL gateway to the cluster. To demonstrate this, open a new terminal and connect the built-in SQL client to node 1:
Note: The SQL client is built into the cockroach binary, so nothing extra is needed.

```
cockroach sql --insecure
Run some basic CockroachDB SQL statements: CREATE DATABASE bank;
CREATE TABLE bank.accounts (id INT PRIMARY KEY, balance DECIMAL);
INSERT INTO bank.accounts VALUES (1, 1000.50);
SELECT * FROM bank.accounts;
+----+---------+
| id | balance |
+----+---------+
```



```
| 1 | 1000.5 |
+----+---------+
(1 row)
```

Exit the SQL shell on node 1:
\q

Then connect the SQL shell to node 2, this time specifying the node's non-default port:
cockroach sql --insecure --port=26258
Note: In a real deployment, all nodes would likely use the default port 26257 , and so you wouldn't need to set the --port flag.

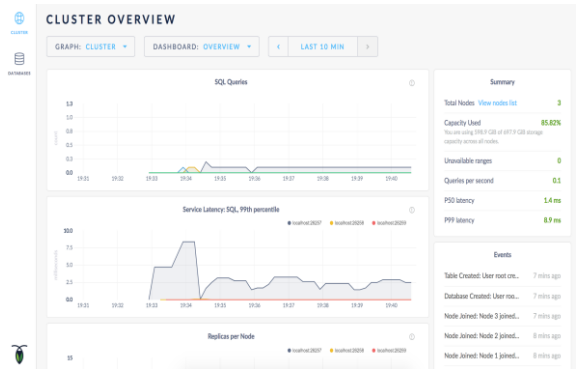
Now run the same SELECT query:
SELECT * FROM bank.accounts;

```
+----+---------+
| id | balance |
+----+---------+
| 1 | 1000.5 |
+----+---------+
(1 row)
```

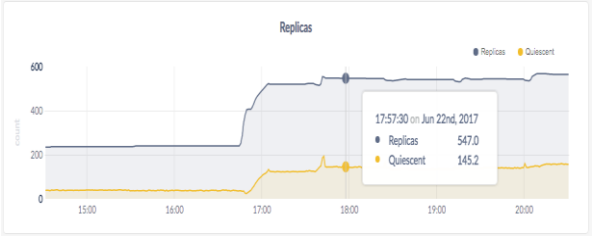
As you can see, node 1 and node 2 behaved identically as SQL gateways.

Exit the SQL shell on node 2:
\q

Step 4. Monitor the cluster
To access the Admin UI for your cluster, point a browser to http://localhost:8080 , or to the address in the admin field in the standard output of any node on startup:



As mentioned earlier, CockroachDB automatically replicates your data behind-the-scenes. To verify that data written in the previous step was replicated successfully, scroll down to the Replicas per Node graph and hover over the line:



Step 5. Stop the cluster

Once you're done with your test cluster, switch to the terminal running the first node and press CTRL + C to stop the node. At this point, with 2 nodes still online, the cluster remains operational because a majority of replicas are available. To verify that the cluster has tolerated this "failure", connect the built-in SQL shell to nodes 2 or 3. You can do this in the same terminal or in a new terminal.

```
cockroach sql --insecure --port=26258
SELECT * FROM bank.accounts;
+----+---------+
| id | balance |
+----+---------+
| 1 | 1000.5 |
+----+---------+
(1 row)
```

Exit the SQL shell:

```
\q
```

Now stop nodes 2 and 3 by switching to their terminals and pressing CTRL+C

Step 6. Restart the cluster

If you decide to use the cluster for further testing, you'll need to restart at least 2 of your 3 nodes from the directories containing the nodes' data stores. Restart the first node from the parent directory of cockroach-data/ :

```
cockroach start \
--insecure \
--host=localhost
```

In a new terminal, restart the second node from the parent directory of node2/ :

```
cockroach start \
 --insecure \
--store=node2 \
--host=localhost \
--port=26258 \
--http-port=8081 \
--join=localhost:26257
```

In a new terminal, restart the third node from the parent directory of node3/ :

```
cockroach start \
--insecure \
--store=node3 \
--host=localhost \
--port=26259 \
--http-port=8082 \
--join=localhost:26257
```

3. Front-end Web-Application Development

The front is a simple webpage that contains some text, date and other datatype fields along with byte a fields that should be input by the citizens when they are to upload some information, and there is an option of show details that when clicked by the citizen, must produce the respective details along with some byte a type fields as requested. So, both for input and showing output, the webpage should be designed. This webpage is

designed using the Web Application development languages like HTML5, CSS3, JSP. AJAX is also used for adding better user interface experience.

VIII. BENEFITS

Fast uploading and retrieval of voluminous data(images, text, documents, etc). Enable the Web Applications to support Consistency, Availability and partition tolerance simultaneously. Flexible schema per table is possible. Low latency and high performance can be achieved through this method. High scalability is achieved. As everything is OpenSource, it is Cost Effective

IX. CONCLUSION

Thus, the proposed new Database Architecture based on NewSQL- CockroachDB will be used to manage the ever-growing voluminous data through Web Applications. Response for the Users' Queries through Web Applications will be faster at the same time ensuring Consistency, Availability and Partition tolerance. So, when the Web Applications especially the e-Governance application will be made to response faster even with huge number of users requesting for http sessions especially during the announcement of some news or results or issues that is expected by huge number of citizens at the same time, where everyone will be trying to access the same site at a same point of time.

Naturally, every solution has its pros and cons. Due to performance being the top priority, NewSQL databases tend to have more security gaps than traditional SQL databases. These issues need to be researched in-depth to overcome the situation.

References

[1] Ismail Hababeh, Issa Khalil, and Addallah Khreishah, "Designing High Performance Web-Base Computing Services to Promote Telemedicine Database Management system," IEEE Transactions on services computing, Vol.8, No.1, January-February 2015.

[2] Zhou Wei, Guillaume Pierre, and Chi-Hung Chi, "CloudTPS: Scalable Transactions for Web Applications in the Cloud," IEEE Transactions on services computing, Vol.5, No.4, October-December 2012.

[3] Mainak Ghosh, Wenting Wang, Gopalakrishna Holla, and Indranil Gupta, "Morphus: Supporting Online Reconfigurations in Shared NoSQL Systems",IEEE TRANSACTIONS ON EMERGING Topics IN COMPUTING, 6 December 2017.

[4] Besma Khalfi, Cyril De Runz," A New Methodology for Storing Consistent Fuzzy Geospatial Data in Big Data Environment ", JOURNAL OF TRANSACTIONS IEEE COMPUTER SOCIETY, VOL. 3, NO. 4, DECEMBER 2017.

[5] Jun Gao, Jiashuai Zhou, Jeffrey Xu Yu, and Tengjiao Wang," Shortest Path Computing in Relational DBMSs," IEEE transactions on knowledge and data engineering, April,2014

[6] A B M Moniruzzaman," NewSQL: Towards Next-Generation Scalable RDBMS for Online Transaction Processing (OLTP) for Big Data Management, International journal of computer application, Vol No. 7, Jan.,2014 engineering, April.

[7] Carlos Ordonez," Integrating K-Means Clustering with a Relational DBMS Using SQL," IEEE transactions on knowledge and data engineering vol. 18, no. 2,February. 2006

[8] Mahmoudreza tahmassebpour, "A New Method for Time-Series Big Data Effective Storage,"IEEE access June 27. 2017

[9] Michael Joseph Mior, Ashraf Aboulnaga, "NoSE: Schema Design for NoSQL Applications" , IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 29, NO. 10, OCTOBER. 2017

[10] Dixin Tang, Hao Jiang, Aaron J. Elmore, "Adaptive Concurrency Control: Despite the Looking Glass, One Concurrency Control Does Not fit All" , 8th Biennial Conference on Innovative Data Systems Research (CIDR '17) January 8-11.2017

[11] Olga Kurasova, Virginijus Marcinkevicius, Viktor Medvedev, Aurimas Rapecka and Pavel Stefanovic, Strategies for Big Data Clustering, Institute of Electrical and Electronic Engineer, 15 December 2014.

[12] Saurabh Arora, Inderveer Chana, A survey of clustering techniques for big data analysis, Institute of Electrical and Electronic Engineer, 15 December 2014.

[13] Ajay Jangra, Vandhana Bhatia, Upsana Lakhinaza, Niharika Singh, An efficient storage framework design for cloud computing: Deploying compression on de-duplicated No-SQL DB using HDFS.

