International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Integrating MySQL Database Replication In Bahmni– Eager replication

Suhas Mane[1], Pranavi Mohite[2], Renuka Khale[3]

Dombivli East. 421203,India
Sanpada, Maharashtra, India

ABSTRACT :

Master-slave replication is a critical mechanism used in distributed database systems to improve performance, fault tolerance, and data availability.

In this architecture, one server (the master) serves as the primary database node, handling write operations, while one or more other servers (the slaves) replicate data from the master and handle read operations.

This paper explores the design and implementation of a master-slave replication system, focusing on its key components such as data consistency, fault tolerance, scalability, and performance optimization.We have created master-slave replication process for data security purpose.

If master system is crashed then data will get automatically restored in slave system hence chances of data lost are less. In the event of a master server failure, failover mechanisms can promote a slave server to the master role, ensuring high availability of the database system. Master-slave replication finds extensive application in scenarios requiring high availability, scalability, and fault tolerance, such as web applications, e-commerce platforms, and enterprise systems.

Keywords: data replication, bahmni data replication, Eager Replication

Introduction:-

the realm of healthcare information systems, efficient and reliable data management is paramount to ensuring high-quality patient care and operational efficiency. Bahmni, an open-source hospital management system and electronic medical record (EMR) solution, has gained traction for its comprehensive approach to healthcare IT. Integral to the performance and reliability of such systems is the database that underpins it. MySQL, a widely-used relational database management system, offers robust features for data management. However, the challenge of maintaining data consistency and availability across distributed environments remains a critical concern.
Database replication, particularly eager replication, has emerged as a solution to address these challenges by ensuring that changes made to one database are immediately reflected across all replicas. This approach minimizes latency and improves data consistency, which is crucial in healthcare settings where timely and accurate information can significantly impact patient outcomes.
Despite the advantages, integrating MySQL database replication within the Bahmni framework presents several technical and operational challenges. These include configuring and managing replication processes, ensuring minimal downtime, and maintaining performance under heavy load conditions. This research aims to develop and evaluate a robust methodology for integrating MySQL eager replication within Bahmni, leveraging modern containerization technologies to streamline deployment and management.
The primary objective of this study is to design and implement a eager replication system for MySQL within the Bahmni environment. This involves setting up a multi-master replication architecture and evaluating its performance in terms of replication lag, system reliability, and data consistency. By addressing these objectives, the research seeks to contribute to the enhancement of database management practices in healthcare information systems, ultimately leading to improved system reliability and patient care.

Problem Statement

Single Point of Failure: Without replication, the master server becomes a single point of failure. If the master server goes down, the entire database becomes unavailable, leading to service disruptions and potentially significant downtime.

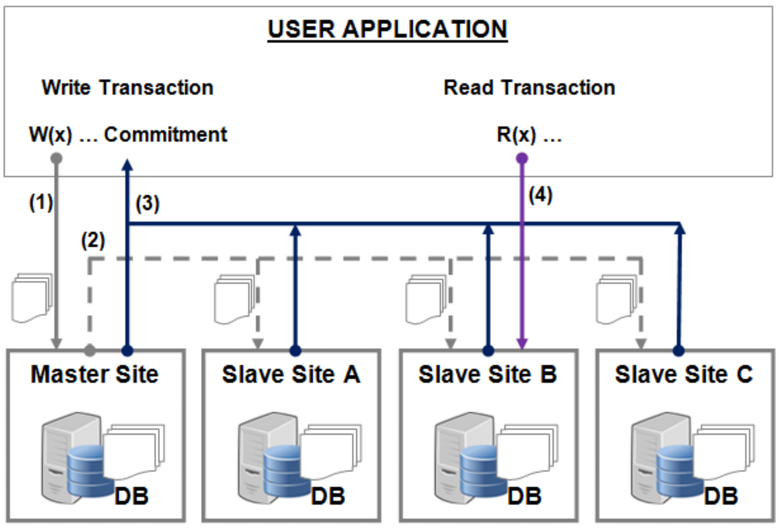
Data Loss: In the absence of replication, there is a higher risk of data loss. If the master server fails and there are no backups or replicated copies of the data, any changes made since the last backup will be lost.

Reduced Performance: With only one server handling both read and write operations, the system's performance may suffer, especially during peak usage periods. Read-heavy workloads can degrade the performance of the master server, leading to slower response times for all clients.

Limited Availability: In the event of maintenance activities or upgrades on the master server, the database may become temporarily unavailable to clients. With replication, clients can be directed to one of the slave servers during such maintenance windows, ensuring continuous availability

Load Balancing: Replication allows for load balancing by distributing read requests among multiple slave servers. Without replication, all read requests must be handled by the master server, potentially leading to performance bottlenecks and uneven resource utilization

1.1. Structure



Proposed Methodology

Primary Server(192.168.1.23) commands :-
Step 1: Ensure that folder `/var/log/mysql/` exists with permissions `mysql:mysql`

```
mkdir -p /var/log/mysql

chown -R mysql:mysql /var/log/mysql
```

Step 2: File '/etc/my.cnf ' should contain this content

```
[mysqld]
bind-address        = 0.0.0.0
server-id           = 1
log_bin             = /var/log/mysql/mysql-bin.log
binlog_do_db        = openmrs
```

Step 3: Restart the `mysqld`

```
sudo service mysqld restart
        OR
sudo service mysql restart
```

Step 4: Create replication user in the DB

```
mysql -h 127.0.0.1 -u root -p

mysql>  CREATE USER 'replica_user'@'192.168.1.47' IDENTIFIED BY
'<replica_user_password>';

mysql> GRANT REPLICATION SLAVE ON *.* TO 'replica_user'@'192.168.1.47';

mysql> FLUSH PRIVILEGES;
```

Step 5: Retrieving Binary Log Coordinates from the Source

```
FLUSH TABLES WITH READ LOCK;

SHOW MASTER STATUS;
```

You will see a table similar to this example in your output:

Output :-

```
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000001 |      899 | openmrs      |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

Step 6: DO NOT CLOSE MYSQL CONSOLE TERMINAL (in order to keep the DB locked during the replication configuration)

Open a new terminal and create a backup of the DB

```
mysqldump -h 127.0.0.1 -u root -p openmrs > openmrs.sql
                              OR
mysqldump -h 127.0.0.1 -u root -p openmrs > /openmrs.sql
```

Step 7: After that you can close previous mysql terminal or run there a command:

```
UNLOCK TABLES;
```

Note:- Copy `openmrs.sql` file from Master to Replica server

Standby Server(196.168.1.47) commands

Step 8: Recover database from the dump

```
mysql -h 127.0.0.1 -u root -p

CREATE DATABASE openmrs;

Exit;

mysql -h 127.0.0.1 -u root -p openmrs < openmrs.sql
```

Step 9: Ensure that folder `/var/log/mysql/` exists with permissions `mysql:mysql`

```
mkdir -p /var/log/mysql

chown -R mysql:mysql /var/log/mysql
```

Step 10: File `/etc/my.cnf` should contain this content

```
[mysqld]
bind-address     = 0.0.0.0
server-id        = 2
log_bin          = /var/log/mysql/mysql-bin.log
binlog_do_db     = openmrs
relay-log        = /var/log/mysql/mysql-relay-bin.log
```

Step 11: Restart the `mysqld`

```
sudo service mysqld restart
             OR
sudo service mysql restart
```

Step 12: Start replication

```
mysql -h 127.0.0.1 -u root -p

CHANGE MASTER TO
MASTER_HOST='192.168.1.23',MASTER_USER='replica_user',MASTER_PASSWORD='<replica_user_password>',MASTER_LOG_F
ILE='mysql-bin.000001',MASTER_LOG_POS=899;
```

Step 13: Start slave using this command

```
START SLAVE;
```

Step 14: Check replication status

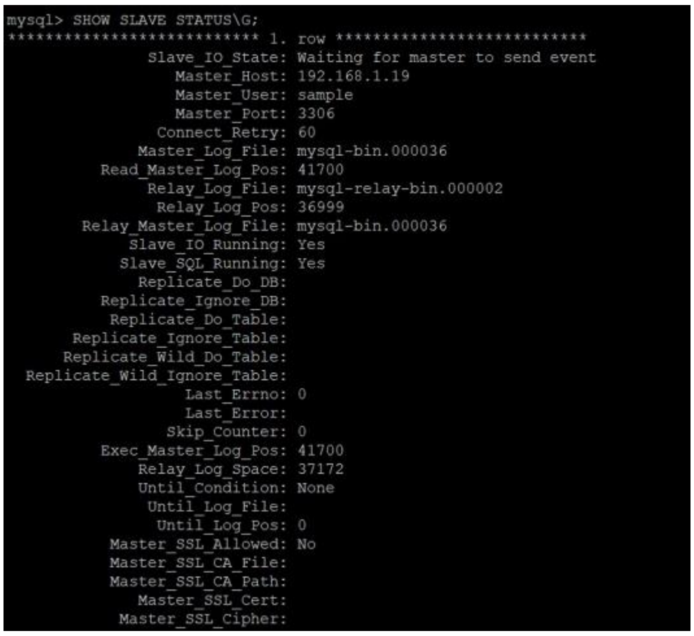
```
SHOW SLAVE STATUS\G;
```

Performance Analysis

Eager replication, also known as single leader replication, can improve system performance by distributing read requests across multiple nodes. This architecture can help with performance issues, such as read-heavy workloads, by:

- **Improving read scaling:**
  Read operations are distributed across slave servers, reducing the load on the master
- **Reducing downtime**
  Slaves act as hot backups, ready to become the new master in case of a primary server failure
- **Improving disaster recovery :-**
  Slaves safeguard data against hardware failures or catastrophic events, ensuring quick recovery

Successful data replication



```
mysql> SHOW SLAVE STATUS\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.19
                  Master_User: sample
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000036
          Read_Master_Log_Pos: 41700
               Relay_Log_File: mysql-relay-bin.000002
                Relay_Log_Pos: 36999
        Relay_Master_Log_File: mysql-bin.000036
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 41700
              Relay_Log_Space: 37172
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
```

Conclusion :

The conclusion of setting up a master-slave replication setup in a database system is a critical milestone in ensuring data reliability, scalability, and fault tolerance. Here are key points included in the conclusion of our setup:

Successful Implementation: There is a successful implementation of the Eager replication setup in bahmni mysql database.

Data Redundancy and Availability: Emphasize the improved data redundancy and availability achieved through replication. With the master-slave setup, data is replicated across multiple servers, ensuring that in the event of a failure, there's a standby server ready to take over without significant downtime.

Fault Tolerance: In case of a master server failure, one of the slave servers can be promoted to a master, ensuring continuous operation with minimal disruption.

Monitoring and Maintenance: It is very important of ongoing monitoring and maintenance of the replication setup. Regular monitoring ensures that replication is functioning as expected, and any issues are addressed promptly to maintain system reliability.

REFERENCES :

1. https://stackoverflow.com/questions/50360376/
2. https://stackoverflow.com/questions/22292089/mysql-replication-slave-error-22?rq=3
3. http://dev.mysql.com/doc/refman/5.5/en/replication-compatibility.html
4. https://talk.openmrs.org/t/db-replication-broken-on-bahmni-active-passive/12973