Deanship of Graduate Studies
Al-Quds University

Performance Comparison for High Availability Solutions of MySQL Database

Alaa Yaser Fayez Mohsen

MSc Thesis

Jerusalem-Palestine

1441-2020

Performance Comparison for High Availability Solutions of MySQL Database

Prepared By:

Alaa Yaser Fayez Mohsen

B.Sc.: Information Technology, Al-Quds University, Palestine

Supervisor: Dr. Nidal Kafri

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Computer Science/ Department of Computer Science & IT/Faculty of Science & Technology/Graduate Studies.

1441-2020

Al-Quds University

Deanship of Graduate Studies

Master of Computer Science

Thesis Approval

Performance Comparison for High Availability Solutions of MySQL Database

Prepared By: Alaa Yaser Fayez Mohsen

Registration No: 21610049

Supervisor: Dr. Nidal Kafri

Master thesis submitted and accepted, Date: 10 June, 2020

The name and signatures of examining committee members are as follows:

1- Head of committee	Dr. Nidal Kafri	Signature: ..N.kafri
2- Internal Examiner	Dr. Rashid Jayousi	Signature: R.Jayousi
3- External Examiner	Dr. Iyad Tumar	Signature: [illegible]

Jerusalem-Palestine

1441-2020

Dedication

I dedicate my thesis to my dearest parents who are always supporting me without stop, and to my precious homeland Palestine.

Alaa Yaser Fayz Mohsen

Declaration

I certify that this thesis submitted for the degree of Master, is the result of my own research, except where otherwise acknowledged, and that this study (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Signed……Ala' Mohsen………

Alaa Mohsen

Date: 10 June, 2020

Acknowledgments

Firstly, I thank God for everything, and I would like to thank my supervisor, Dr. Nidal Kafri and teachers in the program. Without their meticulous effort and support, this thesis would not have been the same.

In addition, I would like to dedicate special and great thanks to my family especially my parents, sisters, grandfather and brothers for their love and encouragement. Also, to my colleagues and friends.

Abstract

MySQL is one of the most common relational database management system that is used around the world and keeping it high available is important for most of its users. This research concentrates on MySQL database high availability solutions, where the database performance is evaluated and compared between two different high availability solutions. Noting that there are multiple choices for high availability solutions of MySQL such as master-slave replication with manual or automatic failover, MySQL NDB cluster, Galera Cluster for MySQL, MySQL with Solaris Cluster, Oracle MySQL Cloud Service. In this research the chosen solutions are InnoDB cluster (MySQL replication) and DRBD (Distributed Replicated Block Device) that depends on replicated disk architecture. InnoDB cluster is provided officially by MySQL. Up to our knowledge, there are no researches found related to performance comparison between these two solutions, and both are considered an effective solution for high availability with automatic failover. Each solution is evaluated using different replication modes of single primary topology. Two modes of DRBD cluster is evaluated, which are Protocol A and Protocol C. Also, two consistency levels are evaluated for InnoDB cluster, which are eventual and before_and_after levels. The results are analyzed and compared, so that would be helpful for decision makers in picking an appropriate high availability database solution for a given application taking into account its cost.

The clusters are configured as recommended (best practice) for best performance and some variables are changed with multiple values to have the best case. For faire comparison purpose a benchmark tool is used as workload testing tool. The clusters are tested using read and write quires. As a result, we found that InnoDB outperforms the performance of DRBD for write tests. It shows higher performance in terms of throughput and total time. On the other hand, for read test it outperforms DRBD when the number of concurrent users is high. This is because InnoDB cluster has two nodes for read queries, and the received requests will be routed on two nodes. But when the number of concurrent users is low the DRBD shows better performance.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

Abbreviation	**Full word**
M.Sc.	Master’s degree
HA	High Availability
CRUD	Create, Read, Update, Delete
DRBD	Distributed Replicated Block Device
OLTP	Online Transaction Processing
VIP	Virtual IP Address
VM	Virtual Machine
MHz	Mega Hertz
NoSQL	Not Only SQL
Mbps	Megabit Per Second
CRM	Cluster Resource Manager
HADBMSs	High Availability Database Management Systems
DBMSs	Database Management Systems
TCP	Transmission Control Protocol
SMEs	Subject-Matter Experts

Chapter 1

Introduction

1.1 Background and Motivation

MySQL database is most popular open source relational database management system that used by many of companies and organizations to organize their data, and for different range of purposes such as data warehousing, logging application, ecommerce purposes etc.,[1]. Also, it is used to store anything from a single record of information to large number of records. High availability is one of the highest priorities for companies/organizations who are caring about their end user satisfaction for the services they provide, where high availability indicates minimal downtime, it is important to ensure data is readily available in situations like server crash, failover, or other data disaster and no organization can afford downtime, even it is merely for a few seconds. Consequently, if their services are not available like an application or site fails to load because of problems with their databases, will result in losing their customers and so decreasing their revenues. Therefore, thinking of high available solution is important for their database, and the database that this research talks about is MySQL DBMS.

Different database high availability solutions may have different effects on database performance. Therefore, the database administrator who is responsible for optimizing the database performance, must decide which appropriate high availability solution for used application. In this case, he must choose a trade-off between (high availability) and the cost [2]. Therefore, this research will be helpful for the decision maker in picking the suitable high availability solution for their requirements, since it compares two different well-known and popular high availability MySQL databases regard database performance (i.e. , how they differ in affecting the database performance), it also clarify the resources needed for each solution which is important for cost calculate, and it discuss the most suitable solution for the business size based on the experiments result, noting these points are important for them when they pick their decision.

There is more than one solution for MySQL database high availability, and organization choose the one that is suitable for its requirements and resources. Furthermore, the DB performance will be taken into consideration to help them in choosing a good and appropriate one according to their requirements (i.e., a feasible solution). In this work a comparison of database performance will be carried out between two databases high availability solutions, which are InnoDB cluster technology and DRBD system (distributed replicated storage system) for Linux platform with MySQL service. They are chosen because both provide an effective high availability with automatic failover (not manual) and can be built over Unix system, also there are no researches did any comparison between them, where this comparison considers throughput and response time in order to maximize customer satisfaction.

Moreover, from my experience as a database administrator in a company, I have been asked about which high availability solution between DRBD and InnoDB cluster has better performance and consider the best for company's requirements, where they often use DRBD solution, but the client asked for an official solution from MySQL. Therefore, to answer this question we felt there is a need for a scientific approach that could help and be needed for other companies and organizations.

Researches about the performance of MySQL high availability solutions are rarely found. Most of previous works talked about database performance evaluations as standalone databases with another, where they focused on doing a comparison between standalone database management systems either relational or non-relational DBMS. One of those researches that is related is carried out by Raju Shrestha [3] where he did a performance comparison between MySQL HA solutions. In [3] they studied and evaluated master-slave and cluster-based high availability database solutions, qualitatively as well as quantitatively. This paper investigated effectiveness of the two major solutions to high availability database: traditional master-slave replication and modern cluster-based techniques (Galera cluster). Authors used SysBench tool to execute tests (read-only and read-write quires) over both solutions, where it is implemented using MariaDB 10.1 with Galera cluster. Results shown that traditional master-slave replication solution performs equal or better in terms of throughput and response time. Despite some performance lag, author summarized that the Galera cluster is an effective solution for applications and services where data consistency and high availability is critical.

Also, in Adfinis SyGroup [4] a comparison for MySQL/MariaDB high availability was done between Galera Cluster vs. DRBD replication, it compares two different High-Availability solutions for MySQL databases, one is a block-device based replication solution, the other extends MariaDB internals to provide synchronous replication. This comparison focused more on network traffic, commit latency, replication, load balancing, failover and resynchronization.

1.2 Problem Statement

Database (MySQL) high availability may affect its' performance. So, achieving extreme high availability, will need extreme cost. A trade-off between availability, performance, cost, locking and complexity can occur comparing to standalone DB , where the data will be replicated on more than one node continuously and this may cause replication latency between nodes which affects badly on DB performance [5]. So, companies/organizations who care about keeping their MySQL database up for long time, will choose one of MySQL high availability solutions, where these solutions could have different effects on performance of database. So, a comparison for DB performance will be carried out for two of these high availability solutions which are, InnoDB cluster and DRBD system with MySQL service.

This research evaluates the performance of the two clusters under different replication types which includes synchronous and asynchronous modes. This will help them to choose the suitable and appropriate solution for their application and satisfies their requirements. It should be noted that the evaluation is done in terms of throughput (i.e., transaction per unit time) and total response time.

Thus, the main question of this work is which of the above mentioned high available MySQL solution (i.e., InnoDB cluster and DRBD) performs better than the other and in which cases?

1.3 Contribution

This research aims to evaluate and compare MySQL database performance for two of high availability solutions for MySQL. These solutions are InnoDB cluster and DRBD system (distributed replicated block device) with MySQL service. The evaluation considers the database transactions throughput and the total/elapsed time needed. Where the testing is carried out with different replications modes and using different read and write queries to evaluate the performance of these clusters/solutions.

The results show that InnoDB cluster outperforms the performance of DRBD for write tests, it shows higher performance in terms of throughput and total time. On the other hand, for read test it outperforms DRBD when the number of concurrent users is high, , but when the number of concurrent users is low the DRBD shows better performance.

The results of this evaluation can have significant help for decision makers in companies or who are caring about having their MySQL database up for long time (high available). In other word, this comparison can help them in choosing a suitable solution that fits their requirements and resources.

Unfortunately, up to our knowledge the absence of recorded data from real-life systems or previous research results on evaluation of these particular solution, makes the validation of our results based on real-life systems difficult. Thus, we faced a limitation problem in validation of the outcomes of our results with either historical real data or other previously obtained results. Therefore, we carried out validation of our results based on subjective approach i.e., subject-matter experts (SMEs) opinions validation approach technique to evaluate the significance differences in performance of the two alternative solutions, and according to the SMEs the results are valid and can be accredited.

1.4 Thesis Organization

This thesis is organized as follows: in next chapter a background introduces definition, notions and theory relevant to this work such as high availability and MySQL replications. Chapter 3 presents literature reviews and background of MySQL performance comparison and evaluation. Chapter 4 introduces the research design and methodology used. In chapter 5 we present the experiments, results, and discussion of the results. Finally, chapter 6 presents the conclusion and future work.

Chapter 2

Background

In this chapter, the main concepts of high availability and replication are discussed in general, then focusing on MySQL database group replication and DRBD (distributed replicated block device).

In computing, high availability refers to systems that are durable and likely to operate continuously without failure for a long time. As databases are increasingly deployed in environments, the need to have them highly available has increased also, where it can be trusted to work properly in cases of hardware or software failures.

2.1 Introduction

The meaning of High Availability varies depending on the requirements of your application and business, where it is measured as a percentage, For instance, 99% availability (two nines) in a period of one year means the system can have up to 3.65 days of down time, 99.99% or (four nines) as is considered excellent uptime , 100% availability indicates that the system is always up and will never come down (zero downtime).

There are basically three component that can help achieving high availability, which are: first is elimination of single point of failure (SPOF), which is causing the system failure if any component of the system failed. This failure can be avoided by adding redundancy in infrastructure and data, which acts as a standby service ready to take over in case the primary failed. Second component is a reliable crossover from a failed to a standby component. The third component is detection of failures, as this will enable the framework to either take corrective actions on the same primary system, or failover the services to a standby system.

Duplication entire MySQL servers could be done by different redundancy options. Shared storage architecture, replicated disk architecture and MySQL replication are some ways that can be followed as redundancy options. Considering that the standby servers have access to the primary server's data. These options are discussed in the following sections and some are used in this research [6].

- **Shared Storage Architecture:**

Is a way to remove some single points of failure. It requires specialist hardware usually with a SAN (storage area networks), where if an active server dies, the standby server can mount the same filesystem, perform any necessary recovery operations and start MySQL on the failed server's files. Considering that, it is Complex to operate (specially for DBAs) [6].

- **Replicated disk architecture:**

Redundancy through disk replication (RAID over ethernet) is another way to keep the data safe in case of failure on a master server. DRBD (Distributed Replicated Block Device) is an example of this option, which is commonly used for MySQL in combination with tools from the Linux-HA. In this research a comparison between this option and another redundancy option (MySQL replication) is carried out regard the database performance.

- **MySQL Replication:**

Redundancy through MySQL replication is at the relational database management system layer, it is a process that enables data from one MySQL database server (the master) to be copied automatically to one or more other nodes of MySQL database servers (the slaves), which has potential for scaling out. Replication works because events written to the binary log are read from the master and then processed on the slave in different format, that is clarified in following sections.

This study presents two different ways of the redundancy options mentioned before. This aims to have a duplicate for MySQL database server for different purposes as automatic failover, scalability, and could be used for analyzing data on the slave in order not to overload the master. These are InnoDB cluster that depends on MySQL group replication technology [7], and a DRBD (distributed replicated block device) with MySQL service, which depends on a replicated device architecture. A comparison between these solutions is carried out based-on the performance of the database, and how the end user will be affected of these clusters regards the response time and throughput.

2.2 MySQL Replication

The MySQL database management system provides mechanism to configure master-slave replication. That allows configuring one or more servers as slaves (replicas) of another server, or even to behave as master for local updates. This mechanism of MySQL replication works in a simple three-part process:

• Firstly, the master records changes/updates on its data from application or client in its binary log (these records are called binary log events).

• Then, the slave nodes copy the master’s binary log events to its relay log.

• Finally, slave replays the events in the relay log, applying the changes to its own data [8].

The events are written to binary logs in different replication format according to the type of events, which is discussed in the following.

2.2.1. Replication Formats:

MySQL uses the primary copy replication method, and supports three kinds of replication, statement-based, row-based formats and can use a mixed format logging [8].

- **Statement-Based Replication**

  In this approach, every SQL statement that could modify the data is logged on the master server [8], where statements are logged to the binary log exactly as they were executed. Then those SQL statements are replayed on the slaves against the same dataset and in the same context. The binary logs do not grow as fast with statement-based replication as they do in row-based replication. It generally requires less data to be transferred between the master and the slave, as well as taking up less space in the update logs. MySQL originally were based on this approach that depends on propagation of SQL statements from master to slave. It has been around from the beginning of MySQL version 3.23 [9].

- **Row Based Replication**

  The master writes events to the binary log that indicate how individual table rows are changed, where every row modification gets logged on the master and then applied on the slave. Instead of replicating the statement that performs the changes, the row-based approach replicates each row being inserted, deleted, or updated separately with values that were used for the operation [1], which is logged to the binary log separately, noting that statements are not logged.

- **Mixed Mode Replication**

  Server uses a combination of statement-based logging and row-based logging, where it can change the binary logging format in real time according to the type of event. Statement-based logging is used by default, but automatically switches to row-based logging in particular case. This is recommended for MySQL version 5.1, to avoid problems for users who upgrade from version 5.0 or earlies because those versions had no row-based replication and users have had to use statement-based replication, where MySQL developers did not want server to make a sudden switch [1].

2.2.2. Replication Protocols:

Classification of replication protocols can be done according to where and when updates and changes happens on master database, can be performed on the replicas database servers [8]. And regarding to when updates can be propagated, different main replication options could be used with MySQL workloads, which are MySQL Asynchronous, Semi-Synchronous Replication and MySQL Group Replication [10].

- ***Asynchronous Replication:***

MySQL replication by default is asynchronous, this type of replication particularly suitable for modern application such as website. It is considered asynchronous because the master does not wait for the slaves to apply the changes. But instead just dispatches each change request to the slaves and assumes they will catch up eventually and replicate all changes [1] as shown in Figure 2.1. With asynchronous replication, if the master crashes, transactions that it has committed might not have been transmitted to any slave, which may have a missing transaction [11].

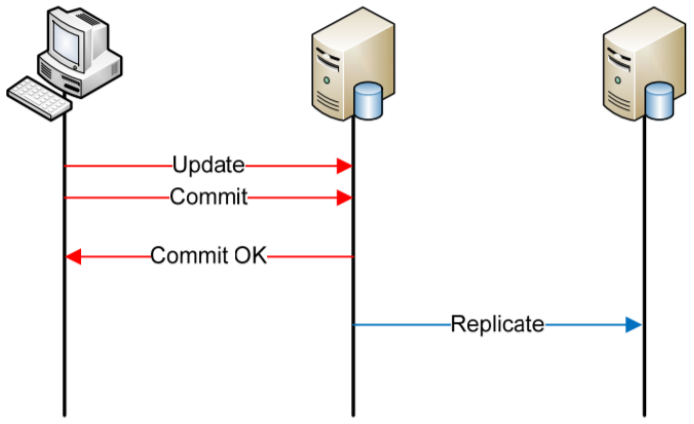

Figure 2.1::Asynchronous replication [11]]

- ***Synchronous Replication:***

In contrast, synchronous replication keeps the master and slaves in sync and does not allow a transaction to be committed on the master unless the slave agrees to commit it as well. The synchronous replication makes the master wait for all the slaves to keep up with the writes [1]. Where a request is sent to all storage nodes being involved in the transaction, and the transaction is not committed till all nodes indicate that they are ready, then it becomes committed and the application/client get informed of the success of the transaction [12] as shown in Figure 2.2.

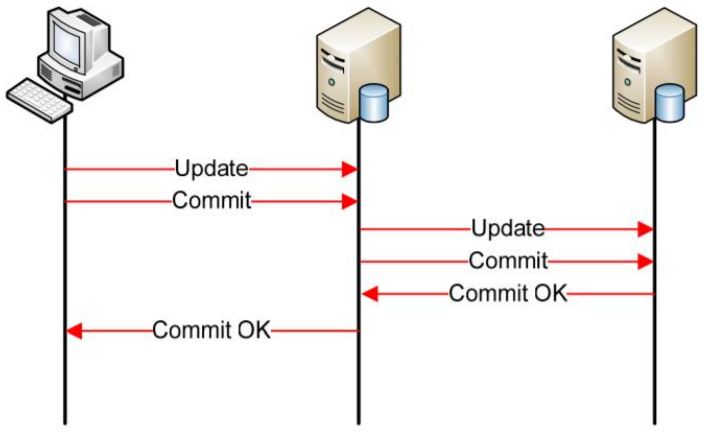

Figure 2.2:Synchronous replication [11]

Asynchronous replication is faster than synchronous for reasons, relates to that it requires extra synchronizations to guarantee consistency, which is usually implemented through protocol called two-phase commit. This guarantee consistency between the master and slaves, which requires extra messages to ping-pong between master and slave [1].

- ***Semi-synchronous Replication:***

Semi-synchronous replication falls between asynchronous and fully synchronous replication. The idea about it is to ensure the changes are written to disk on at least one slave before allowing execution to continue. This action avoids sending a reply to the client until the transaction has been written to the relay log of at least one slave. This means it does not wait for all slaves to acknowledge receipt, it just requires only one receipt [13] as shown in Figure 2.3.

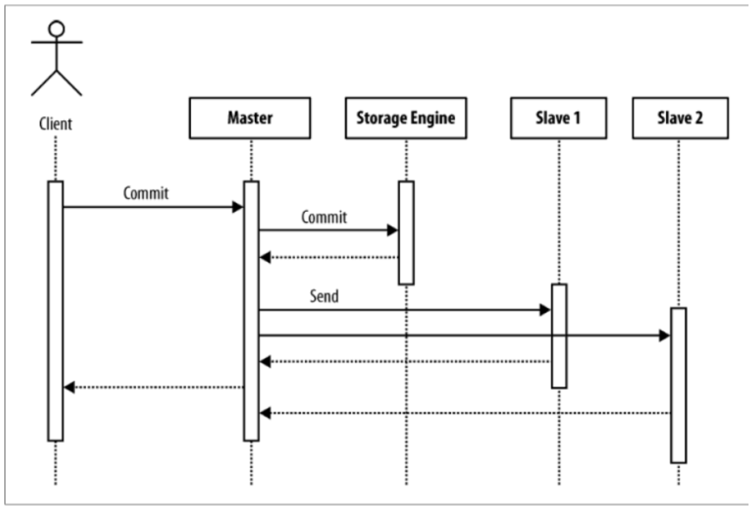

Figure 2.3: Transaction commit with semi synchronous replication [1]

- ***MYSQL Group Replication:***

Database replication is traditionally handled in two ways, either with synchronous (eager) replication or with asynchronous (lazy) replication. But they have problems one can consider such as synchronous replication which is slow and deadlock prone, and asynchronous replication does not enforce consistency between the replicas even though it is efficient. Therefore, to address this problem, replicated databases based on group communication have been proposed for some time [14].

Atomic Broadcast

Techniques based on group communication typically rely on a primitive called total order broadcast or atomic broadcast, it ensures that messages are delivered reliably and in the same order on all replicas [14]. Where a client sends the transaction to the primary and it processes the transaction to all participants using an atomic broadcast, then replicas apply the writes according to delivery order of the atomic broadcast as shown in Figure 2.4. Consequently, conflicts are detected and if a transaction needs to be aborted, it is aborted on all servers [15].

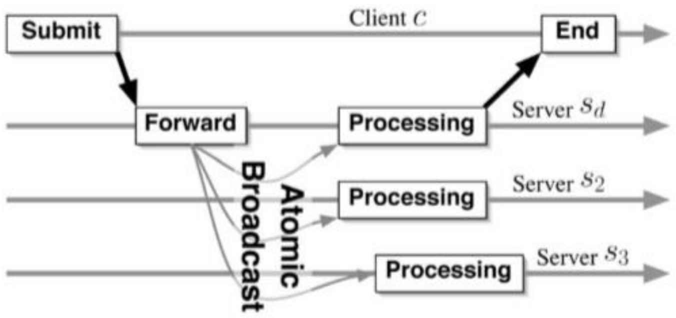

Figure 2.4:Active replication - Atomic broadcast [13]

Group Replication Technique

Group Replication is the latest evolution of MySQL Replication. It is a technique that can be used to implement fault-tolerant systems. Group replication is designed to make data replication more robust and reliable. It consists of set of servers that each have their own entire copy of the data and interact with each other through message passing.

MySQL Group Replication provides distributed state machine replication with strong coordination between servers. In this technique (state machine replication) the whole transaction is put into a message, and the message is broadcast (total order broadcast) to the servers as shown in Figure 2.5. That is provided by the communication layer that apply a set of guarantees such as atomic message and total order message delivery [16].

Group Communication System (GCS) protocols provide a failure detection mechanism, a group membership service, safe and completely ordered message delivery. These key properties ensure that data is consistently replicated across the group of servers. Furthermore, at the very core of this technology lies an implementation of the Paxos algorithm [16], that acts as the group communication engine. Where it is an efficient and highly fault-tolerant algorithm, for reaching consensus in a distributed system [17].

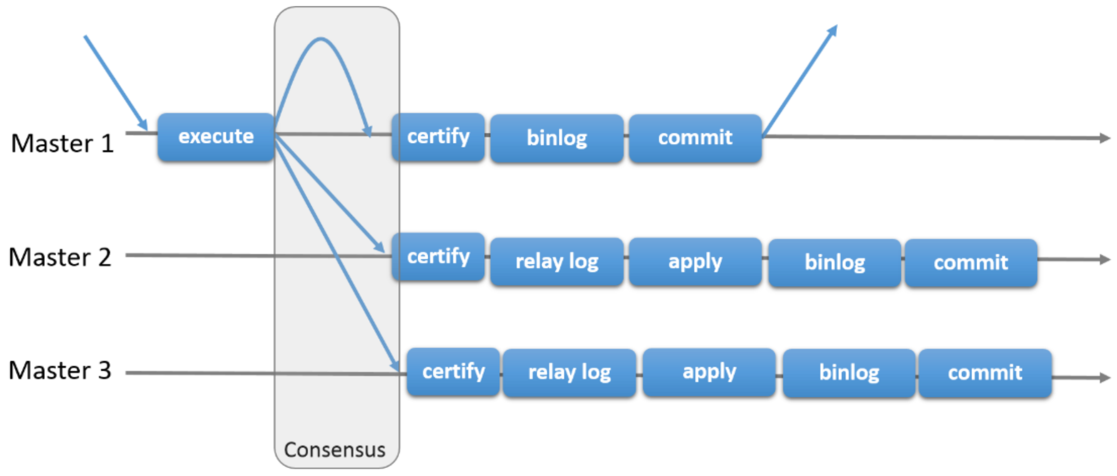

Figure 2.5: MySQL Group Replication Protocol [15]

Performance Tuning:

Group communication is a plugin, and one of the key components of a group replication plugin is a group communication thread (GCT), which runs in loop when the plugin is loaded [18]. This threads' wait value is determined by this variable A group_replication_poll_spin_loops. where this variable represents in InnoDB cluster, the number of times the group communication thread waits for the communication engine mutex. Since group communication thread (GCT), receives messages from the group and from the plugin, handles quorum and failure detection related tasks, sends out some keep alive messages and handles the incoming and outgoing transactions from/to the server/group, and waits for incoming messages in a queue. So, when there are no messages, the GCT waits, and by configuring this wait to be a little longer (doing an active wait) before going to sleep, can prove to be beneficial in some cases [19].

Thus, different values are tried for this variable in the configuration of MySQL cluster in the experiment of this research. That is to check the case that the performance of MySQL can be improved and take it for comparison with the other cluster.

MySQL Group Replication Topologies

It is possible to setup MySQL group replication with different topologies of masters and slaves, which are single primary mode (the simplest topology), and multi primary mode.

- **Single-Primary Mode**

In this topology, the group has a single primary server that is set to read-write mode. Thus, only a single server writes to the group and all other members in the group are set to read-only mode. In this case they don't interact with each other at all, they all connect only to the master. Also, the server which join the group will learn about the primary server and is automatically set to read-only mode. This configuration is useful for a system that has many reads and few writes [8].

- **Multi-Primary Mode**

In multi-primary mode, any member joins the group and compatible with the group members is set to read-write mode, where there are no special roles for any member, even if they are issued concurrently. Consequently, writing simultaneously is possible in this mode [20].

2.3 MySQL InnoDB Cluster

MySQL InnoDB Cluster is a collection of products that work together to provide a complete high availability solution for MySQL. It is composed of three main parts: Group Replication (GR), MySQL Shell and MySQL Router. Three servers are needed at least to configure InnoDB cluster, either in a single primary, or multiple primary mode with InnoDB engine.

InnoDB cluster depends on group replication technology, that is clarified in preceding sections. In this cluster each MySQL server instance runs MySQL group replication that provides the mechanism to replicate data within the cluster with built-in failover [21].

2.3.1. MYSQL Router:

MySQL router is lightweight middleware that is used to route connections between the application and back-end MySQL servers, and handle the failover and load balancing. Where it acts as a proxy to hide the multiple MySQL instances on the network and map the data requests to one of the cluster instances. Thus, it stands between application servers and the GR setup, as shown in Figure 2.6. It makes the cluster transparent for the application, that is why application believes that it is talking to a single MySQL server. But in fact, there is a cluster consists of multiple servers there. Furthermore, MySQL router selects a new MySQL server if there any connection fails between the already connected MySQL server and application. In this case, applications will be designed to retry the connection.

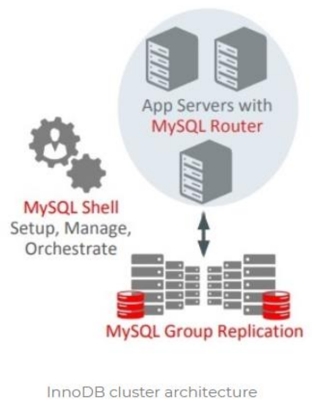

Figure 2.6:InnoDB cluster architecture [22]

2.3.2. MySQL Shell:

MySQL shell is used for configuring InnoDB Cluster. It is an advanced command-line client and code editor for the MySQL Server, which supports development and administration for the MySQL server. Also, it provides the developer and DBA with a single intuitive, flexible, and powerful interface for all MySQL related task. Moreover, it supports different languages such as (JavaScript, python and SQL).

2.3.3. InnoDB Consistency Level:

The configuration of a group's consistency can be guaranteed based on the point at which the transaction is wanted to be synchronized across the group. The points of synchronizing transactions across a group could be at the time of a read operation or at the time of a write operation. The consistency level can have a different impact on read-only (RO) and read-write (RW) transactions processed by the group. Thus, one can determine the consistency level according to his situation with MySQL group replication as following [23]:

- **Eventual Level**: It is the GR default consistency level, where read only and read-write transactions do not wait for preceding transactions to be applied before executing. It could result in outdated values of RO transaction, and RW transactions could result in a rollback when a primary failover happens.
- **Before_On_Primary_Failover**: In this level, the new RO or RW transactions with a newly elected primary that is applying backlog from the old primary are held (not applied), until any backlog has been applied.
- **Before**: A RW transaction waits for all preceding transactions to complete before being applied and A RO transaction waits for all preceding transactions to complete before being executed, that will cause a long wait time.
- **After**: A RW transaction waits until its changes have been applied to all other members. This can be considered as synchronous writes as the return from commit happens only when all members have applied it. This value has no effect on RO transactions, and it has impact on network latency.
- **Before_and_After:** In this level a RW transaction waits for all preceding transactions to complete before being applied and until its changes have been applied on other members. Also, a RO transaction waits for all preceding transactions to complete before execution takes place. That is why this level considered as the highest level that guarantee the transaction consistency.

In this research the experiments are carried out using two of these consistency levels, which guarantee the lowest and the highest transaction consistency (i.e., eventual and before_and_after).

2.4 DRBD System Replicated Disk Architecture

DRBD (device replicated block device) is designed for high availability clusters and software defined storage. Where it is a software-based, shared-nothing, replicated storage solution, and mirroring the content of block devices (hard disks, partitions, logical volumes etc.) between hosts. It is developed by LINBIT, which provides networked RAID 1 functionality for GNU/Linux [24]. Since It implements mirroring across two disks as shown in Figure 2.7. Thus, two copies of information exist, and when a disk fail, the information can still be acquired through the other copy [25]. Each peer of a DRBD resource acts in one of two roles, it may be either secondary or primary node. This is controlled by a cluster manager software called heartbeat, that initiates the failover process in case the primary node (active) leaves the cluster unexpectedly (crashes) [26]. Consequently, all the modifications that happened to data must be initiated on the primary node and reflected on the secondary node. The secondary node can’t be used for neither read nor write access, where it is passive node.

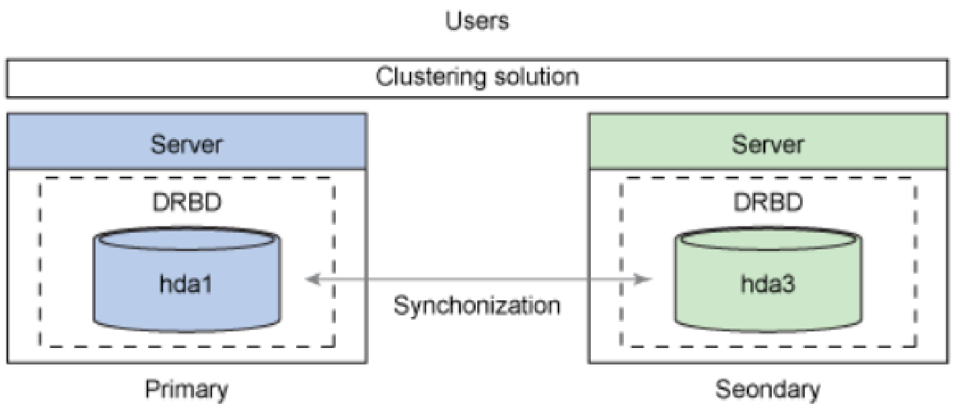

Figure 2.7:DRBD architecture

2.4.1. Replication Mode:

DRBD mirrors the data in real time and transparently. Thus, the applications are not aware of multiple nodes. DRBD replication can be either synchronously or asynchronously replicated modes and there are three degrees of replication synchronicity as the following :

Protocol A

It is asynchronous replication protocol, which means that local write operations are considered achieved on the primary node when local disk write are finished, and the replication packet has been placed in the local TCP send buffer. However, if there is a host forced failover, then some data loss can occur. This setup is more common in replicating stacked resources in a wide area network [24].

Protocol B

This protocol is memory synchronous (semi-synchronous) replication protocol. The local write operations are considered achieved, once it is occurred on primary and replicated data has reached the peer node. In this mode no writes are lost in case of forced fail-over.

Protocol C

Synchronous replication protocol, which is the most commonly used protocol and the default one. Using this mode means that the Local write operations on the primary node are considered completed only after both the local and the remote disk writes have been confirmed, where it guarantees the prevention of any data loss in failover.

2.4.2. DRBD Topologies:

Single-Primary Mode

Primary role in this mode is on one of cluster member, where the application can only write to it, and the secondary node is simply a real time replica of the primary. It is guaranteed that only one cluster node manipulates the data at any moment.

Dual-Primary Mode

In this mode the DRBD resource has primary role on both cluster nodes at the same time, where the application can freely write to both DRBD resources simultaneously, and DRBD kernel driver allows write attempts to happen to DRBD resource on both nodes sharing the resource. By design, a DRBD resource is supposed to have the same contents on both nodes of a cluster. However, it is a bit dangerous, where application can have some form of locking logic. But it is the preferred approach for load-balancing clusters [24].

Optimizing DRBD Performance

There are number of configuration options for tuning the throughput of DRBD. One of the recommendations for tuning DRBD to optimize its performance is tweaking the I/O unplug watermark. The I/O unplug watermark affects how often the I/O subsystem's controller is forced to process pending I/O requests during normal operation. Some storage controllers deliver better performance with small values. However, others perform better when left alone, and others setting as high as max-buffers is advisable. On the other hand, in some cases there is no significant effects of this setting [27]. Where there is no universally recommended setting for this option, since it is hardware dependent. Thus, in this research three values are experimented for this variable. Thus, the best case could be taken for performance evaluation.

2.4.3. Pacemaker / Corosync:

DRBD uses Pacemaker and Corosync tools for communication and managing cluster as illustrated in Figure 2.8. Pacemaker is an open source cluster resource manager (CRM) that performs tasks to control how the cluster behaves as to start, stop, monitor, recover or move around a resource. The resource in high availability configuration can be something as simple as an IP address that floats between cluster nodes or something as complex as a database instance with complex configuration [28].

Corosync is an open source program that provides cluster membership and messaging capabilities referred to as the messaging layer (cluster communication layer), where it serves three primary purposes; it provides reliable message passing between cluster nodes, establishes the cluster membership, and determines quorum. Corosync is the default cluster communications layer in the Linux HA stack [28].

Pacemaker and Corosync also used to manage the failover when a resource becomes unavailable, that is by using a virtual IP method for redirecting clients to the active node as shown in Figure 2.8.

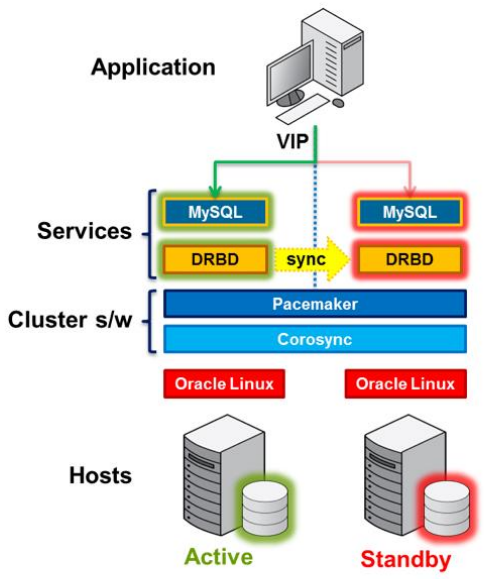

Figure 2.8:DRBD with MySQL Architecture [29]

2.5 SysBench Tool

SysBench is a benchmark suite, open source benchmarking tool, that allows one to quickly get an impression of system performance. It provides benchmarking capabilities for Linux, supports testing CPU, memory, file I/O, mutex performance, and database performance as MySQL benchmarking.

SysBench is used to evaluate MySQL clusters performance in this research. It is simple to use, open source, provide scripts for load testing, and it automatically generates a data into the database.

It includes an Online Transaction Processing (OLTP) test profile, where it is a true database-backed benchmark that conducts transactional queries to an instance of MySQL in a CentOS environment. A LUA [30] scripts can be used to execute benchmarks by this tool, where those scripts, handle input from command line parameters, that define the modes of benchmark that is supposed to use which are: prepare, run and cleanup. The prepare command should be executed before run command, that is to generate a data into database by defining the number of tables and number of rows for each one. Moreover, these scripts define how the benchmark will be executed [31].

Chapter 3

Literature Review and Related Work

While using the database high availability HA, the performance also should be taken into consideration. High availability database may affect its’ performance. There is a trade-off between performance, cost, locking and complexity comparing to standalone DB [1]. Companies and organizations who care about keeping their MySQL database up for long time, will choose one of MySQL high availability solutions, recall that these solutions have different effects on performance of the database.

While there has been much research on evaluating the performance of database as standalone server, and there are researches for comparison between standalone servers for relational and non-relational database management systems, few researchers have taken clusters or high availability solutions into consideration. So, a comparison for DB performance is carried out in this research for two of high availability solutions, which are InnoDB cluster and DRBD system with MySQL service. This comparison aims to evaluate the performance of two clusters under different replication types, taking into account the throughput and response time, which are the most common metrics used in previous works.

3.1 Performance Evaluation Techniques

Paul, Subharthi [32] in his survey tried to emphasize the importance of database systems in enterprise setups and looked at the methods and metrics that are used to evaluate the performance of database systems. In [32] he discussed some of the analytical modeling methods for evaluating systems that are applicable for database systems, which are: queuing models, cost models, simulation modeling and benchmarking (which is used in this research). Where benchmarking method is considered the best, when multiple database systems need to be evaluated against each other. But it suffers from the inherent setback, that it assumes all systems to be fully installed and operational. It relies on the effectiveness of the synthetic workloads as real workloads are non-repeatable and hence not good for effective benchmarking.

Also, he classified the Database Performance Evaluation Techniques for specialized Databases as Web-Database Systems, which serves the back-ends of Web-Servers, Real-time Database Systems, Enterprise Data Mining System. These DBs are huge database often called (data mining systems) that stores historical and redundant data, and Object-Oriented Systems performance Evaluation. Moreover, they listed different benchmarks that could be used for evaluating different performance aspects of such systems [32].

3.2 MySQL High Availability Evaluation

A performance comparison between MySQL high availability solutions had been made by Raju Shrestha [3], where he studied and evaluated master-slave and cluster-based high availability database solutions, qualitatively as well as quantitatively. He investigated effectiveness of the two major solutions to high availability database which are: traditional master-slave replication, and modern cluster-based techniques (Galera cluster). Author used SysBench tool to do the experiment of executing tests (read-only and read-write quires) over both solutions, where it is implemented using MariaDB 10.1 with Galera cluster. Results show that traditional master-slave replication solution performs equal or better in terms of throughput and response time. This is because of simpler setup and better performance. However, cluster-based solution is superior when it comes to high availability, data consistency and scalability as it offers instantaneous failover, no data inconsistency and loss of data. Also, at the same time providing both read and write scalability. Therefore, despite some performance lag, author summarized that the Galera cluster is an effective solution for applications and services where data consistency and high availability is critical.

The ways of fault tolerant of MySQL database have been talked about by Ari J. Flinkman [33], he looked at ways to build a fault-tolerant MySQL installation in his research, by creating an active/passive setup using either MySQL's standard replication, shared storage or DRBD, he also talked about MySQL Cluster and NDB Storage Engine, (where this research focused on InnoDB storage engine). This can be used to create a setup with multiple active server instances with redundant, distributed storage for data. He summarized that the shared storage is always problematic when used with databases which are not designed for it, and the replication with DRBD might be good low-cost alternative to shared storage but it is limited to Linux. He considered the MySQL cluster is definitely the best bet in such quest, and it might prove influential for future designs of HADBMSs if it lives to deliver all the promises made.

Bart Oles [5] looked at the two main high availability solutions for MySQL and MariaDB, and how they can each be affected by latency issues. He clarified the effects of master/slave replication for MySQL high availability and multi-master replication. While the term latency refers to several kinds of delays incurred in the processing of data, but he talked about it as a definition of how long it takes for a piece of information to move from stage to another. Where master slave replication for MySQL comes with multiple configuration options to optimize replication process. He talked about essential replication related parameters, which are: parallel apply, logical clock algorithm, compression, selective master-slave replication and replication mode. Moreover, author discussed the multi master replication (MariaDB) and the points that cause common latency issues. These points are the slowest node in the cluster, horizontal scaling, write operations, geolocated clusters, high ping and transaction size. Where some of them are considered of the experiment for this research, for tuning the database to give better performance.

In [4] (Adfinis SyGroup website) a comparison for MySQL/MariaDB high availability is carried out between Galera cluster vs. DRBD replication. Where they compared two different high availability solutions for MySQL databases, one is a block-device based replication solution and the other extends MariaDB internals to provide synchronous replication. This comparison focused on some points such as network traffic, commit latency and replication.

DRBD supports synchronize and asynchronies modes. However, the galera cluster can only be used synchronously. Moreover, load balancing in DRBD is typically used in an Active/Passive setup, in contrast, Galera cluster is a pure Multi-Master solution. Authors also explained about the failover. In DRBD environment, if the active node goes down, the Cluster stack (typically

Pacemaker with Heartbeat or Corosync) has to detect the problem and switch the services over to another node. While in Galera Cluster, when a single node goes down, the remaining nodes in the cluster continue working without interruption and the client currently connected to the failing node would retry the connection via a load balancer without notice any interruption [4].

Pukdesree et al. [34] had published their research with name of "Performance Evaluation of Distributed Database on PC Cluster Computers", which aims to evaluate the distributed database approach, that can improve the performance of database system. They used an open source DBMSs and evaluated the distributed database system using SysBench benchmark tool. Where the test was executed using two types of operations, which are read/write and read only scripts. It was in term of the number of processed requests in specific time period. Authors configured the cluster using MySQL Cluster 7.0. On the other hand, some researches articles claimed that the version of MySQL Cluster 7.0 has greatly higher performance than previous versions, while they used Red Hat Enterprise Linux 5 operation system. Moreover, the test was over different number of storage nodes to check the performance when it increased i.e., its scalability. They summarized that the number of succeeded transactions per second (throughput) improved significantly, when number of data storage increased, which depends on the results they got. However, their evaluation was limited by the maximum number of data storage nodes to eight data storage machines.

3.3 Standalone Database Comparison

Other researches introduce different type of database management systems and compare the database performance between them as standalone database. Where Shivani [35] evaluated the performance of different NoSQL databases, where NoSQL refers to non-relational database management systems. These NoSQL databases are (MongoDB, Couchbase, Cassandra, HBase) under various parameters like creation, insertion, update, and throughput. YCSB benchmark (Yahoo Cloud Serving Benchmark) has been used to drive performance tests, which provides data generator and a set of workloads that are defined as a set of CRUD operations. Also, the article analyzed the result of running the CRUD operation over each database instance and compared the time needed for insertion, update, creation operations and the throughput for each one. It concluded that MongoDB performs better than Cassandra for insert operations for various sizes of data sets, and Couchbase and MongoDB perform better than Cassandra regard the throughput [35].

Some researches tried to compare between relational and non-relational database system to prove which outperform other as Lokesh Kumar [36], who attempts to use NoSQL database to replace the relational database, where he focused on one of NoSQL database (MongoDB) and made a comparative study with MySQL. Also, a method is suggested to integrate with different two technologies of these two types of databases by adding a middleware (Metadata) between application layer and database layer. Where the comparison was based on terms/concepts and behalf the quires and based on Query Execution speed/performance (for basic and complex queries) between mongo database and MySQL. Consequently, the result showed that MongoDB spends less time than MySQL, and it concluded that NoSQL(MongoDB) is better to be used instead of MySQL because of two factors which are, ease of use and timing performance [36].

Response time, throughput, latency, **c**reation/Insertion time and delay are the most used metrics in previous work for researches related to database performance evaluation. It should be noted that most of these researches do comparison between two or more DBMS (relational or non-relational database) for standalone database.

Chapter 4

Research Approach and Methodology

This chapter describes the experiment design used in this research to achieve the goal of performance evaluation, and the comparison between the two high availability solutions for MySQL. It describes the methodology and stages that are followed to answer the question of this research, and how the experiments are implemented.

4.1 Methodology

Recall that the aim of this research is to evaluate the database performance of two high availability solutions of MySQL and to make a quantitative comparison between them. The experiment design is determined after studying the factors that could affects the performance for each cluster, which aims to compare the best case for InnoDB cluster and DRBD in two different replication modes. The experiments are carried out to choose the best configuration, that gives best performance of InnoDB cluster and the best one for DRBD. Then to compare between these best cases as clarified in the experimental design section.

The comparisons occurred between best cases to give more accurate results where changes of some variables value can affect the performance, it could make it better or worse. Therefore, in this research we try to change all the variables that could affects database performance to recommended values. The purpose of these trials is to achieve fairness in performance comparisons between the best case of each solution.

After studying and determining the experiment design for this research, we followed the following steps:

- Preparing the servers needed for implementing the experiments, taking into consideration the experiment design, where InnoDB cluster needs at least three servers to be configured and DRBD needs two servers. These servers have the same specifications (Unix servers) for both clusters.
- Prepare SysBench tool, which is the benchmarking tool that used in this research to evaluate the performance of MySQL database. It automatically generates the data into database and allow the use of transactional queries for testing. It is installed on a server on the same virtual machine of the database servers used for high availably. This is to mitigate the network latency, that could occur between the SysBench (which is considered as the client) and the database server.
- Installation and configuration of first cluster InnoDB cluster, where it is installed on three Unix servers as database servers, and one server for MySQL router which is installed on separate one on the same VM. Where it is configured on bases of what is recommended to have the best performance from MySQL [37].
- MySQL database is tuned also to have better performance by modifying the database configuration file.
- SysBench tool is used to test the database performance for two types of tests (write only and read only test). The experiment is repeated 30 times (replicas) for each case. Which is the reasonable number of replicas by which we obtained means of results with relatively low variance.
- Throughput and total time are saved in excel files from SysBench results for each run and the average of the 30 times is calculated for each case, where the experiment is repeated for two consistency levels of InnoDB cluster.
- The results are analysed and best case for InnoDB cluster is determined, for two of consistency levels.
- Installation of DRBD cluster (second cluster) on two Unix servers, where one node is active and the other is passive. It is also configured with settings recommended by LINBIT to have best configuration for performance and throughput [24].

- Same database settings are added into MySQL configuration file in both clusters.
- The same generated database is taken from first cluster as backup and restored to the DRBD cluster to have the same data on both clusters. SysBench tool is utilized by applying the same way and the same commands that are used for InnoDB cluster for read and write tests.
- Experiment is repeated 30 times too,for each of two replication modes of DRBD cluster, and the average of the result is taken for each case.
- Results are analysed for DRBD, and the best case is determined for two of replication modes.
- The best cases of the two replication modes of InnoDB cluster and DRBD are compared, and results are analysed.

4.2 Experiment Design

The experiment design laying out a detailed experimental plan for doing this quantitative experiment. It aims to describe and explain the variation of information under conditions that are hypothesized to reflect the variation. Next section shows the details of the plan for this research.

Service and Topology: This experiment aims to evaluate the performance of MySQL database service with high availability (HA), where high availability aims to keep the database up for longer time and do automatic failover. InnoDB cluster (MySQL Group Replication) and DRBD (Replicated Disk Architecture) are the high availability solutions that are chosen to be evaluated. This evaluation achieved by analysing the results of experiments and comparison between them. Where the used topology is Single-Primary Mode for each cluster, recall this topology was discussed in chapter 2.

Replication Mode: The replication mode used in this experiment for DRBD cluster is protocol A (asynchronous replication) and protocol C (synchronous replication). In contrast the used consistency levels for InnoDB cluster are the eventual level (the default) and before_and_after level, which represent simplest and most complex level, respectively, in InnoDB. These configurations are discussed in section 2.3.3.

Dependent Variables: Represent the output or outcome of the experiment, which are the metrics used to evaluate the performance of MySQL database of the two clusters. They are throughput (number of transactions per second) and response time (the total time needed for the CRUD transactions to be completed).

Independent Variables: The independent variables are controlled inputs. It should be noted that the variation in the value of the dependent variable (i.e., output) is due to the different inputs. In this experiment, independent variables refer to the factors that could be manipulated to take the best performance for each MySQL cluster. Where there are multiple factors that affect the performance for each cluster including network latency, hardware specifications, and the configurations of each cluster itself. For this experiment, the goal is to compare between two MySQL clusters. Thus, the used hardware specifications and network bandwidth between servers are identical. Where they are installed on same virtual machine to mitigate the network latency. The variables that could be changed for each cluster configuration is taken into consideration separately to have the best performance for each one. Then do comparison between the best cases for each cluster. Some variables value depends on the hardware specifications other variables don't have optimal values. Therefore, multiple input values are tried for each cluster and the best one is taken. For InnoDB cluster a variable group_replication_poll_spin_loops value is tried with different three values through the configuration (see section 2.2.2). Also, for DRBD the value of variable is called I/O unplug watermark is also tried with three different values in the DRBD configuration file.

4.2.1. Data and Workload Characterization:

The data used is auto generated from workload testing tool (SysBench), that auto generates the data into database based on the command line option you specified. In this experiment the following command is used to generate the data, using 'prepare' command.

```
SysBench  --MySQL-host=database-IP --MySQL-port=port# --db-driver=MySQL  --MySQL-
user=db_user --MySQL-password=*** --MySQL-db=test cluster  --tables=5 --table_size=2000000  --
percentile=99 --rand-type=uniform test_scripts.lua prepare
```

Where the “MySQL-db” is the created database into which the generated data to be saved, “tables” determine the number of tables for the specified database, “table_size” specifies the number of rows for each table, “rand-type” is the random numbers distribution.

In this experiment rand-type specified with uniform value which will equally stress the dataset and will have more chances to read/write all over the place, and the percentile option allows to specify a percentile rank of query execution times to count which is specified to be 99 % . Note that, the data generated contains (integer and character) data types and with specific template (groups) generated from SysBench as shown in Appendix B. Furthermore, the size of the generated database is around 2 Gigabyte as a result of five tables and 2000000 rows for each table.

Workload: SysBench provides OLTP workloads to be executed, where the read only and write only workloads are executed on the two clusters. Hence, read-only script consists of different types of select queries and write-only script has a mix of delete, insert and update quires.

SysBench works with three commands (prepare, run and cleanup), with multiple options. The following are the steps that followed in this experiment with SysBench:

- Preparing the database by generating the data using prepare command (SysBench), which generate the data and has option to determine the number of tables and rows for each table.
- After the data is prepared (loaded), write and read workloads are executed using run command in SysBench, which executes transactional queries on the database node as shown in the following command used for testing (read_only.lua script):

```
 SysBench  --MySQL-host=DB-server-IP  --MySQL-port=port# --db-driver=MySQL  --
MySQL-user=db_user --MySQL-password=*** --MySQL-db=test_cluster --
table_size=2000000  --tables=5  --percentile=99 --rand-type=uniform  --events=1000 -
-threads=1  /usr/share/SysBench/oltp_read_only.lua run
```

  Where “threads” option in the command determines the number of concurrent users in the experiment, and it is tried with these (1, 4, 8, 16 and 32) concurrent users, “events” option determines the maximum number of transactions to be executed through time period (default is used 10 seconds), and the database server IP is the router IP in InnoDB cluster and the virtual IP (VIP) in DRBD system.

- Each test is repeated 30 times (using for loop) for each thread number (number of concurrent users) to have more accurate results and saved in excel sheets, then the average of results is calculated, where the concurrent users are changed to check how the cluster performs with increasing number of concurrent users.

4.3 Experiment Environment

This section discusses the hardware and software platforms used and how they are used in this experiment.

4.3.1. Physical Platforms:

The specification of the VM (VMware virtual machine) used in this experiment for both clusters InnoDB and DRBD are the same for each server:

CPU: 2 processor cores, 2593.906 MHz, Intel.

Memory: 8 Gigabyte (GB).

Hard disk: 50 Gigabyte.

Network Speed: 10000 Mbps

Attached Disk: 20 Gigabyte, it is attached just for DRBD cluster for two nodes as it is replicated hard disk architecture.

InnoDB cluster: Installed on three VMs used for database servers and one server (VM) for MySQL router. The client/ application will connect to the router IP and it will route the received requests to the read or read/write node of the cluster based on the used TCP port number. The topology used is single primary topology, Figure 4.1 illustrates this architecture.

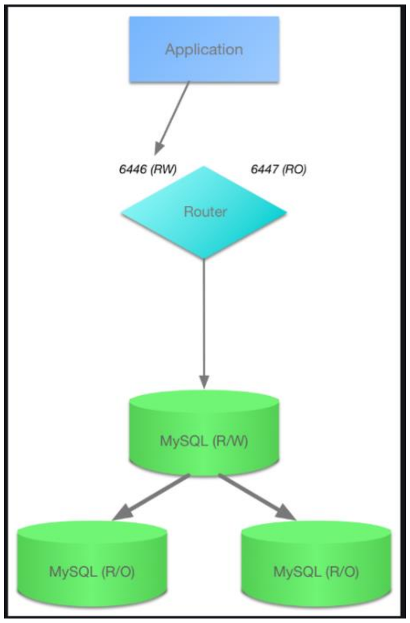

Figure 4.1:InnoDB cluster architecture of single primary topology [38]

DRBD: Installed on two servers (VMs) with identical hardware specifications and with single primary topology. The disk device is attached to the two nodes with 20 GB, as illustrated in Figure 4.2.

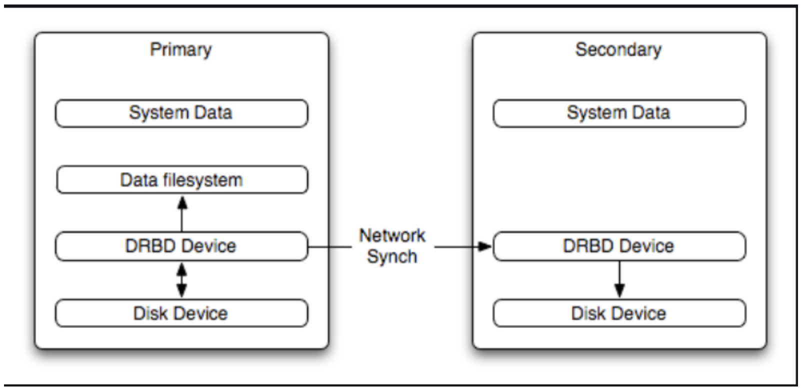

Figure 4.2:DRBD architecture of single primary topology [39]

4.3.2. Software Platforms:

The operating system used in this experiment is Linux on all servers. The DRBD doesn't configured on other OS and no firewall enabled on any server, where the OS specification for all servers as following:

Operating System: CentOS Linux 7

Kernel: Linux 3.10.0-957.21.2.el7.x86_64

InnoDB Cluster: Three types of software are needed to be installed for this cluster. Thus, the InnoDB cluster could be configured as MySQL database server, MySQL shell and MySQL router and all are of the same version:
MySQL Server, MySQL shell and MySQL Router version is 8.0.18.

DRBD Cluster: More than one software needed to be configured for DRBD cluster with MySQL service and doing automatic failover. Following is the version for each software:
DRBD version: 8.4.11-1
MySQL version: 8.0.18
Pacemaker version: 1.1.20-5
Corosync version: 2.4.3-6

SysBench: Installed on separate server but on same VMware with same specification, that is to mitigate the network latency issue between SysBench (which acts as client) and the database server, where the used SysBench version is 1.0.18-1.

4.3.3. Experiment Settings:
This section discusses the settings used for each cluster, how it is implemented, and the used commands.

- ***InnoDB Cluster:***

MySQL server and MySQL shell software are installed on three nodes (VMs), which are the database servers, and MySQL router installed on different node. To achieve good performance, the following two variables are changed in MySQL configuration file (my.cnf), as recommended from MySQL. The most commonly followed practice is to set this value InnoDB_buffer_pool_size at 70% – 80% of the system RAM and the InnoDB transaction logs should be approximately 50-100% of the size of the InnoDB buffer pool.

```
InnoDB_buffer_pool_size = 6G

InnoDB_log_file_size = 3G
```

After installing MySQL server on three nodes, MySQL shell is installed and used to configure the cluster. The next command is executed on three nodes using MySQL shell to prepare the configuration:

```
dba.configureLocalInstance('user@server-name1:3306');
```

After preparing the three nodes, where each node has different ID in my.cnf file, the cluster is initialized on one node and the rest of nodes are added by executing following commands using MySQL shell:

```
var cluster = dba.createCluster('cluster-name');
cluster.addInstance('user@server-name2:3306');
cluster.addInstance('user@server-name3:3306');
```

Also, the command used to check the status of cluster:

```
cluster.status()
```

After ensuring that the status of cluster is successfully configured, MySQL router installed on different server, and configured to start managing it, where the read/write node (primary node) is the DB server that is used in next command for this configuration.

```
MySQLrouter --bootstrap user:password@server-name:3306 --user=MySQLrouter --force
```

The result of this command is that, router will start routes the transactional requests into the cluster on two TCP ports, one for the read only node which is 6447, and the other for read/write node which is 6446.

To take the best performance case of MySQL cluster that could be have in this experiment, this variable group_replication_poll_spin_loops, which is discussed in chapter 2, is changed with three different values and the best case is taken in the comparison. These values are 0, 1000 and 1500 that are tested using SQL language. For each value, the experiment is repeated 30 times (i.e., 30 replicas/trials) and the average is taken. Since smaller number of used replicas generates high variance between the result values. But with 30 replicas the results are closer based on the calculated standard deviation and the coefficient of variation as shown in Appendix A. Also, the results fall in the calculated confidence interval.
Regard changing the consistency level of InnoDB cluster, the following are utilized:

```
set global group_replication_consistency='before_and_after';

set global group_replication_consistency='eventual';
```

After the settings of InnoDB cluster and MySQL router are configured successfully, the SysBench tool is used to generate the database using prepare command, and the generated database size is around 2 Gigabytes. Then the quires (write and read quires) are executed over the cluster, and the results are saved to excel sheets.

- ***DRBD (Distributed Replicated Block Device):***

DRBD software is installed on the two nodes, and a file is created for the DRBD configuration under this path /etc/drbd.d/ for both nodes, where the attached disk on both nodes are prepared to be used for DRBD configuration.
The configuration file for DRBD is changed with the recommended settings, where this variable (unplug-watermark) has no recommended value as discussed in chapter 2. Therefore, it is experimented with three values, which are 16, 32 and 64 to find out the best performance case. Other variables are assigned values as recommended to have best performance and throughput [27]. The following are the assigned values:

```
max-buffers 8000;
max-epoch-size 8000;
```

```
sndbuf-size 0;
no-disk-barrier;
no-disk-flushes;
al-extents 3389;
```

The utilized command to initialize DRBD and to create the metadata for the DRBD resource on both nodes is:

```
drbdadm create-md resource_name
```

DRBD replication after that should synced successfully, where the `drbdadm status` command used to ensure that DRBD replication is works correctly and the disks are up to date (consistent) on both nodes.

Then MySQL database server is installed on both nodes, where the MySQL configuration file for DRBD has the same settings that is used in InnoDB cluster. Then the attached disk drive is mounted on the MySQL data directory to replicate the MySQL data directory on both nodes using this command: `mount /dev/drbd0 /data` , where /data is the MySQL data directory that is determined in MySQL configuration file (my.cnf) as `datadir=/data/MySQL`.

After that Pacemaker and Corosync software are installed on both nodes and configured with three resources which are file system which is the DRBD disk, virtual IP and MySQL service.

`pcs status` command provided from Pacemaker/Corosync used to check the status of these three resources i.e., check the status of the cluster and the active node.

After the configuration works successfully for DRBD cluster, the database is cloned from the first cluster (InnoDB) and restored into the DRBD primary node to have the same database, then SysBench tool is used for the test using read and write scripts.

Chapter 5

Experiment Results, Discussion and Analysis

This chapter presents the results of the experiments in details, discussion and the analysis of these results. It should be noted that the experiments followed the methodology mentioned in chapter 4 to achieve the goal of this research. Recall that the goal is to compare the performance between two clusters. Two high availability solutions are implemented and tested with two modes and experimented with multiple values for a variable that affects the performance to get the best one, then the best cases for each high availability solution are compared as clarified in the following sections.

5.1 InnoDB Cluster

The experiment is executed for two consistency levels. These levels are eventual and before_and_after i.e., the simplest and most complex consistency levels of InnoDB cluster as discussed in chapter 2. Three different values for variable group_replication_poll_spin_loops are experimented. The value of this variable may affect the performance of throughput as discussed before. Thus, the best performance could be taken for the comparative analysis. Read and write test repeated 30 times (i.e., number of replicas or trials) for each of these input values to obtain results with reasonable low variance. It should be noted that the variance of the results was relatively high when the number of replicas (trials) were less than 30. Where the calculated coefficient of variation has low values (i.e., result is more precise) when tried with 30 times, which is calculated based on the standard deviation of the results, as shown in Appendix A. Also, with 95% confidence, the means of the results are within the 95% confidence interval.

The throughput and total time represent the average of the obtained results of all trials for each case i.e., summation of 30 values divided by 30. Furthermore, each experiment is repeated for different number of concurrent users (threads) to evaluate the behaviour and performance of the high availability solutions with increasing number of concurrent users.

5.1.1. Eventual Consistency Level:

The read and write tests are executed over the cluster for this type of consistency level (eventual), which is the default for InnoDB cluster. The averages are calculated for the 30 times of experiments to have more accurate results as shown in Appendix A.

Write tests are executed over this level with different values of this variable group_replication_poll_spin_loops, so we can have the best case. Table 5.1 shows and clarifies the throughput (transactions per second) of the experiment by changing the number of concurrent users for write test.

Table 5.1: SysBench results of throughput for write test of eventual InnoDB consistency level

	Number of threads				
Values of group_replication_poll_spin_loops	**1**	**4**	**8**	**16**	**32**
0	80.94567	218.8977	352.4203	537.8833	817.5393
1000	46.874	123.2027	240.4377	367.0267	790.0937
1500	78.08233	175.0093	283.2623	383.793	873.155

As shown in Figure 5.1, the throughput has higher values (best) when group_replication_poll_spin_loops variable value equal to 0 in this experiment for write test but has no significant difference when the concurrent users equal to 32.

In other words, the best case in this experiment environment, is when number of waiting times equal to zero for the group communication thread to wait for the communication engine mutex.

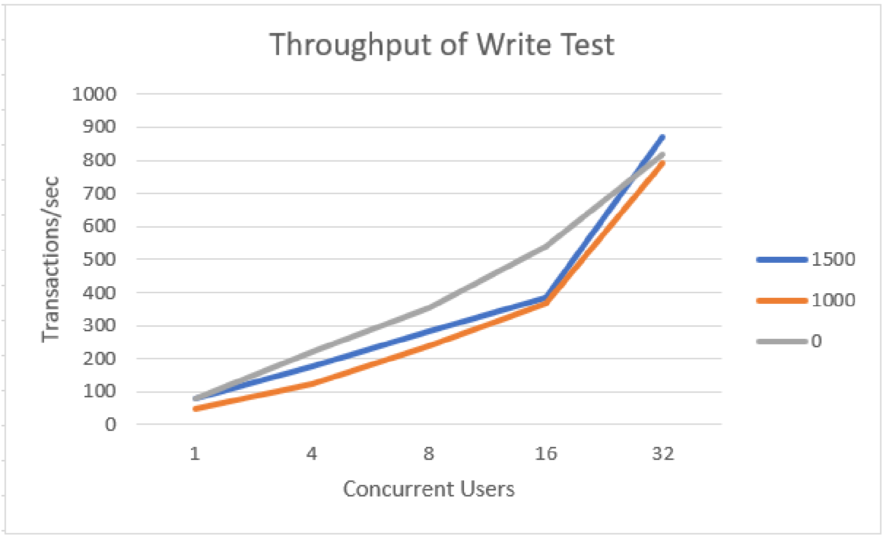

Figure 5.1:Throughput for write test for eventual InnoDB consistency level

Moreover, read tests are executed over this level with same environment of write tests and with different values of this variable group_replication_poll_spin_loops to have the best case. Table 5.2 presents the throughput (transactions per second) of the experiment when changing the number of concurrent users for read test.

Table 5.2: SysBench results of throughput for read test for eventual InnoDB consistency level

	Number of threads				
Value of group_replication_poll_spin_loops	**1**	**4**	**8**	**16**	**32**
0	72.73133	270.1647	499.506	912.4237	1387.351
1000	74.15733	276.737	531.776	849.8547	1271.261
1500	68.25567	257.227	470.1163	847.091	1290.459

It is noted that results of throughput have no significant difference when changing the value of this variable group_replication_poll_spin_loops, where results are very close to each other as shown in Figure 5.2. So, changing this variable has no significant effects to throughput when read queries are executed.

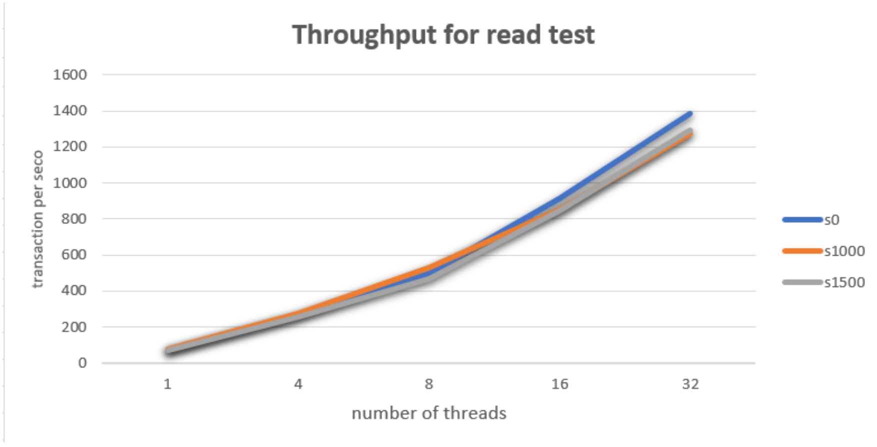

Figure 5.2: Throughput for read test for InnoDB eventual consistency level

5.1.2. Before_and_After Consistency Level:

This experiment has been done for the second chosen consistency level of InnoDB cluster which is before_and_after level. Similar to the previous experiments, this experiment was repeated 30 times for read and write tests. As indicated in Table 5.3, the averages of throughput for write test when the value of this variable group_replication_poll_spin_loops is tested and changed using three values, and when number of concurrent users are changed to record the behavior of the performance.

Table 5.3:SysBench result of throughput for write test of before_and_after InnoDB consistency level

	number of threads				
Value of group_replication_poll_spin_loops	**1**	**4**	**8**	**16**	**32**
0	45.695	82.78967	91.96033	78.78433	80.06967
1000	60.25433	101.368	109.229	91.65967	85.59733
1500	55.76067	83.677	74.25567	62.4	62.332

It is noted that, throughput is increasing till 8 concurrent users (threads), then it has light decrease (lower values), when concurrent users are getting higher than 8 users. This decrease in throughput may occur because the waiting time needed for the cluster to commit the transaction is higher. Since the higher number of concurrent users, the higher the number of transactions. Consequently, the waiting time for the transaction commitment is longer for this level of consistency (strict consistency). Since it waits all transactions to be applied i.e., waits the acknowledgments from all online members [40]. Also, the cause of this decrease can be referred to large number of threads management overheads. As shown in Figure 5.3, throughput has best values (higher), when the value of group_replication_poll_spin_loops is equal to 1000.

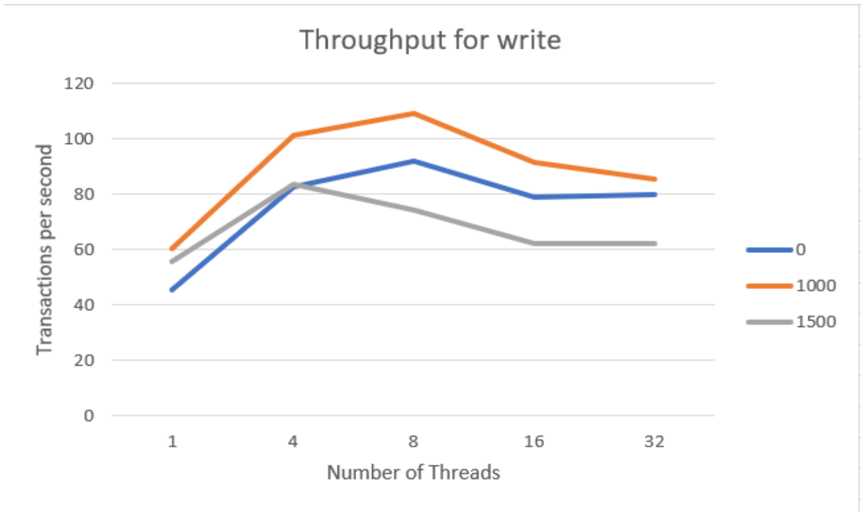

Figure 5.3:Throughput for write test for InnoDB before_and_after consistency level

For read test, the results are so close to each other, which is the same as what obtained from eventual level, where changing this variable has no effects on the read test (select queries) on both consistency levels of InnoDB cluster. As shown in Table 5.4 and Figure 5.4 that illustrate the results.

Table 5.4: SysBench result of throughput for read test of before_and_after InnoDB consistency level

	number of threads				
Value of Group_replication_poll_spin_loops	**1**	**4**	**8**	**16**	**32**
0	73.21133	270.5327	484.1593	838.714	1314.211
1000	72.238	262.1983	453.389	814.3987	1150.635
1500	72.099	259.4527	485.5373	799.074	1275.382

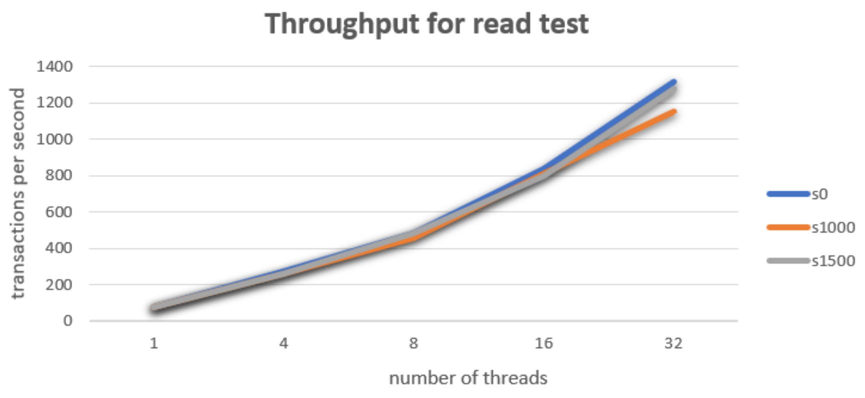

Figure 5.4: Throughput of read test for before_and_after InnoDB consistency level

The throughput for both consistency levels of InnoDB cluster has approximately same values for read test. But for write test the eventual consistency level has higher throughput. As shown in Figure 5.5, where the read operations don’t change the data of MySQL database, but the write operations can do insert, update and deletion for data, which commits the transaction. Since the database should be consistent on all cluster nodes (strict consistent), these modifications and changes should be replicated on all cluster nodes and make it permanent before commitment of the request. Thus, the overheads variation in assuring the different consistency levels of the database makes the throughput different for both consistency levels of InnoDB cluster for both tests.

Since the before_and_after consistency level do wait for all preceding transactions to be applied i.e. waits the acknowledgments from all online members (strict consistency), and this wait cause this low throughput. But the transactions in eventual consistency level do not wait for preceding transactions to be applied before executing.

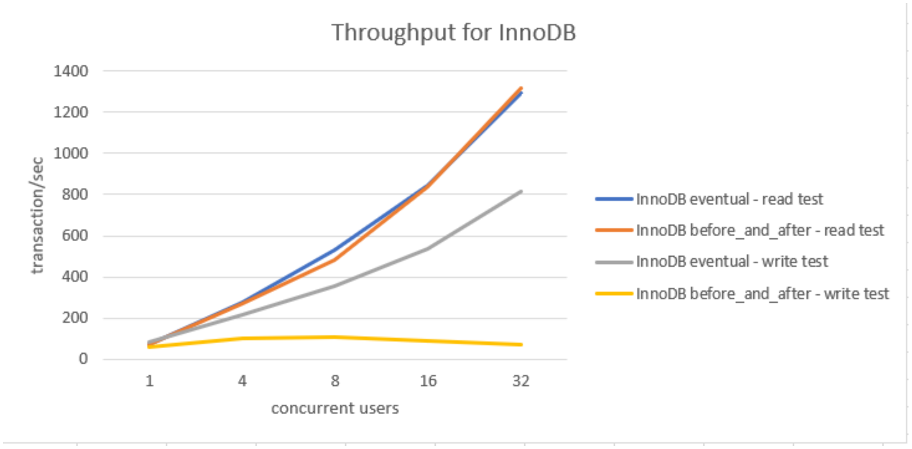

Figure 5.5 : Throughput of InnoDB cluster for read and write tests for both levels

5.2 DRBD System

The test experiment for this cluster (DRBD) is executed in the same way as in InnoDB cluster. So, the comparison could be done between them fairly. Where the same commands of SysBench tool are used for read and write tests and are executed for two types of replication modes i.e., protocol A and protocol C, which are (asynchronous and synchronous replication modes).

The value of this variable I/O unplug watermark is changed in DRBD configuration file with three possible values to take the best case for comparison. Where it can affect the performance of this cluster as discussed. Moreover, read and write tests were repeated 30 times for each of these values to take more accurate and closed results based on the calculated coefficient of variation that has low values as shown in Appendix A. The average of throughput and total time were calculated for performance evaluation. Also, each test is experimented with different number of concurrent users to check how this high availability solution behaves when concurrent users are increasing. Following sections illustrate these behaviours.

5.2.1. Protocol A:

This experiment is executed for protocol A mode (Asynchronies replication) of DRBD system for read and write tests. It is experimented with three different values of this variable (I/O unplug watermark) that may influence the performance of the cluster. So, the best case can be taken for the comparison with the other cluster. Table 5.5 shows the throughput (transactions per second) of the experiment when changing the number of concurrent users for write test.

Table 5.5: SysBench results of throughput for write test of DRBD-Protocol A

	number of threads				
Value of I/O unplug watermark	**1**	**4**	**8**	**16**	**32**
16	41.967	91.09167	161.1593	286.249	506.5947
32	27.75867	78.58433	138.8707	390.686	683.1583
64	39.77367	105.294	180.4633	237.6617	540.912

The results show no significant difference between the three cases when concurrent users are low, but when they are increasing, the performance of cluster is better when the value of this variable (i.e., I/O unplug watermark) equals to 32, as shown in Figure 5.6.

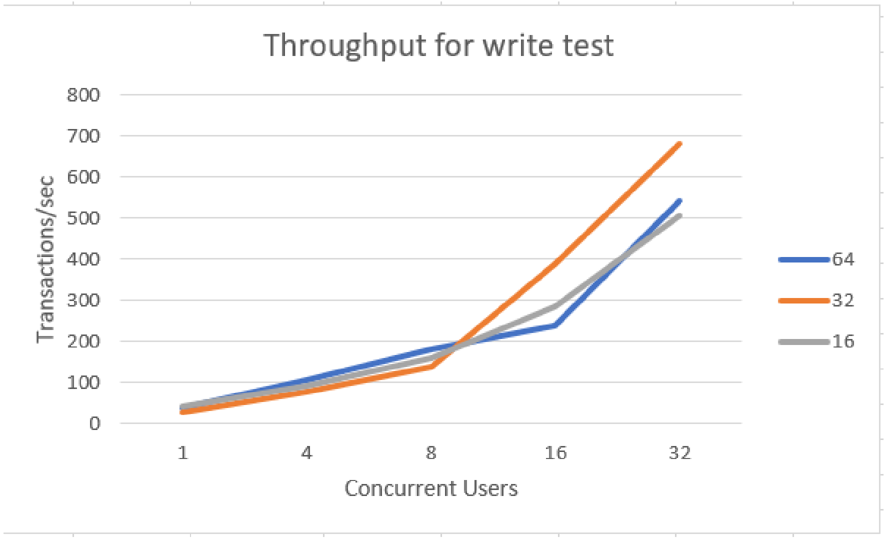

Figure 5.6:Throughput of write test for DRBD-Protocol A

5.2.2. Protocol C:

This experiment of DRBD system is executed for the second replication mode, which is protocol C (synchronize replication). The tests are executed in the same way of previous experiments. Where write tests are executed over this protocol with different values of this variable (I/O unplug watermark), so we can have the best case. Table 5.6 shows that the throughput (transactions per second) of this experiment when changing the number of that variable and changing the number of concurrent users for write test.

Table 5.6:SysBench results of throughput for write test of DRBD-Protocol C

	number of threads				
I/O unplug watermark	**1**	**4**	**8**	**16**	**32**
16	43.69	129.8963	224.103	365.5103	597.931
32	45.02467	85.70067	174.449	455.317	678.2077
64	30.17567	106.8957	226.333	332.576	553.201

The throughput gets higher when value of I/O unplug watermark equal to 32 for high number of concurrent users comparing to others. In contrast there is not big difference between results when changing this variable value, when concurrent users are low as shown in Figure 5.7, which is the same result as we obtained in protocol A.

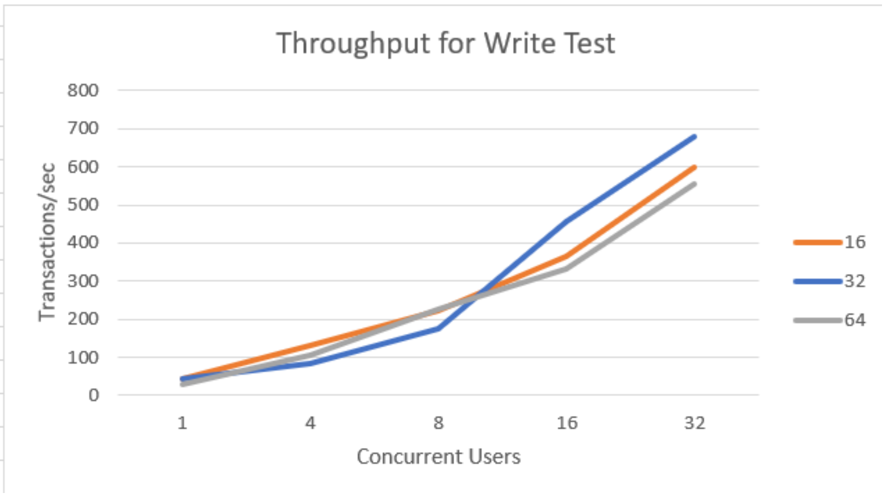

Figure 5.7:Throughput for write test for DRBD-Protocol C

When the read tests are executed over the DRBD cluster for both protocols, the results have no significant difference when the value of I/O unplug watermark changes too.

Where read tests are executed over both protocols with different values of this variable (I/O unplug watermark). As indicated in Table 5.7 that clarifies the throughput (transactions per second) for protocol C when changing the value of this variable and changing the number of concurrent users for read tests.

Table 5.7: SysBench results of throughput for read test of DRBD-Protocol C

	number of threads				
I/O unplug watermark	**1**	**4**	**8**	**16**	**32**
16	142.1603	489.4743	803.2713	1152.219	1243.397
32	144.301	491.8507	811.3977	1178.851	1197.87
64	145.271	510.7057	793.5857	1149.818	1255.616

As depicted in Figure 5.8, throughput for read test almost have similar results for all three cases, which indicates that this variable has no effects on performance when the transactions read the data from database.

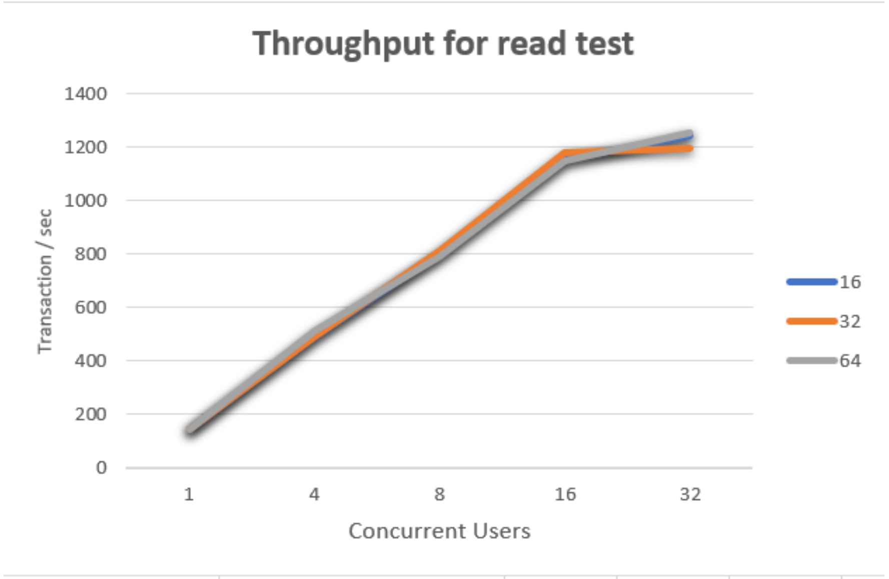

Figure 5.8 :Throughput of read test for DRBD-Protocol C

Changing the replication mode doesn't have effects on the throughput for DRBD system, where the choice of replication protocol influences two factors of deployment: protection and latency. Throughput, by contrast, is highly independent of the selected replication protocol [24], as shown in Figure 5.9 in this research for write and read tests. Which illustrates the best cases of the two replication protocols for read and write tests that has approximately nearly/closed values for DRBD cluster.

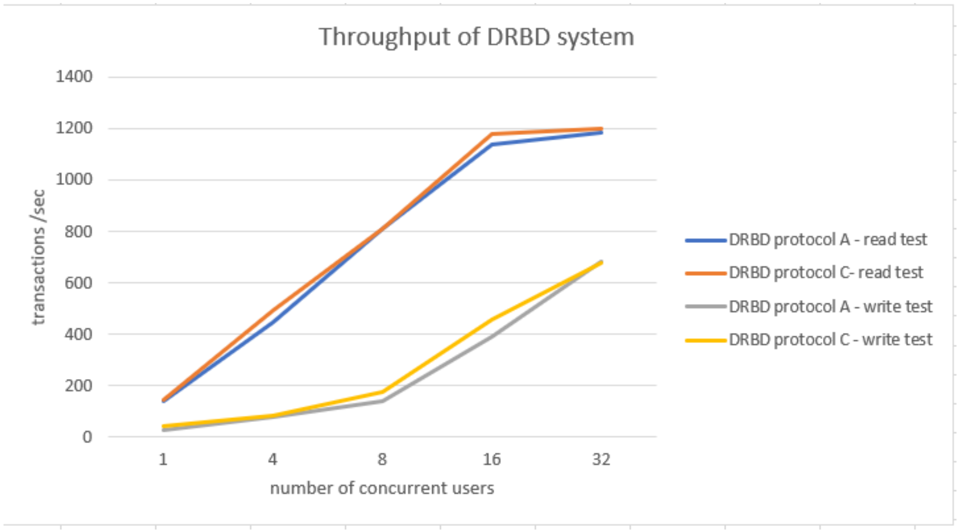

Figure 5.9 : SysBench result of throughput for best cases of write/read tests of two modes of DRBD cluster

5.3 Comparison Between Two Clusters

The best case that shows best performance for the two clusters is taken for write and read tests for each replication mode. Then these results are compared based on their throughputs and the total time taken for write and read operations i.e. the duration from start to finish, which is clarified in following sections.

5.3.1. Throughput:

InnoDB cluster of eventual consistency level has the best performance for write test with all used number of concurrent users comparing to all cluster's cases used in this research as indicated in Table 5.8. This table shows throughput i.e. transactions per second, for write test of the used clusters with all four cases (InnoDB eventual , InnoDB before_and_after, DRBD Protocol A, DRBD Protocol C) with different number of concurrent users.

Table 5.8: SysBench results of throughput for write test for all clusters with all types

	number of threads				
Cluster Type	**1**	**4**	**8**	**16**	**32**
DRBD protocol A	27.75867	78.58433	138.8707	390.686	683.1583
DRBD protocol C	45.02467	85.70067	174.449	455.317	678.2077
InnoDB eventual	80.94567	218.8977	352.4203	537.8833	817.5393
InnoDB before_and_after	60.25433	101.368	109.229	91.65967	72.59733

However, the InnoDB of before_and_after consistency level has the worst case comparing to other cluster's types which is caused by the strict consistency overheads of this level.

According to DRBD system with MySQL service, the two protocols (Protocol A and Protocol C) have close throughput results. But both have throughput results less than InnoDB eventual level as illustrates in Figure 5.10 .

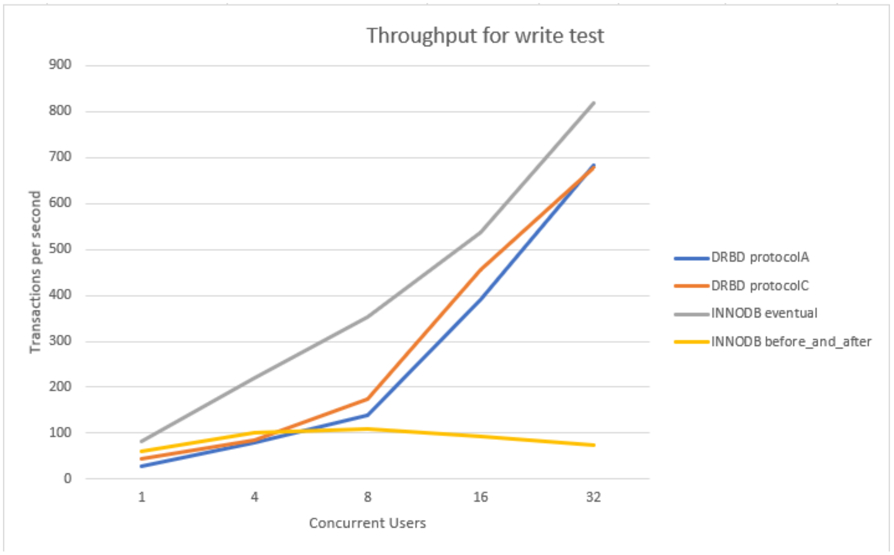

Figure 5.10 : Throughput of write test for two clusters with all modes

In contrast the DRBD system of both protocols performs better (i.e., has better throughput) for read test when number of concurrent users is low comparing to others as indicated in Table 5.9. This table shows the obtained throughput i.e. transactions per second, for read test for used clusters with all four cases with different number of concurrent users.

Table 5.9: SysBench results of throughput for read test for all clusters with all types

	Number of Threads					
Cluster Type	**1**	**4**	**8**	**16**	**32**	**64**
DRBD protocol A	138.728	446.506	812.6197	1138.211	1181.895	1200.025
DRBD protocol C	144.301	491.8507	811.3977	1178.851	1197.87	1188.484
InnoDB eventual	74.15733	276.737	531.776	847.091	1290.459	1503.022
InnoDB before_and_after	73.21133	270.5327	484.1593	838.714	1314.211	1434.031

But when the number of concurrent users start increasing, throughput for InnoDB cluster with both levels (eventual and before_and_after) is getting higher than DRBD for read test, and DRBD of both protocols (Protocol A and Protocol C) become stable as shown in Figure 5.11. This is because InnoDB cluster has two nodes for read queries, and the received requests will be routed to both nodes. Consequently, this will balance the loads on two servers, which makes the throughput for InnoDB cluster better than DRBD cluster in this case.

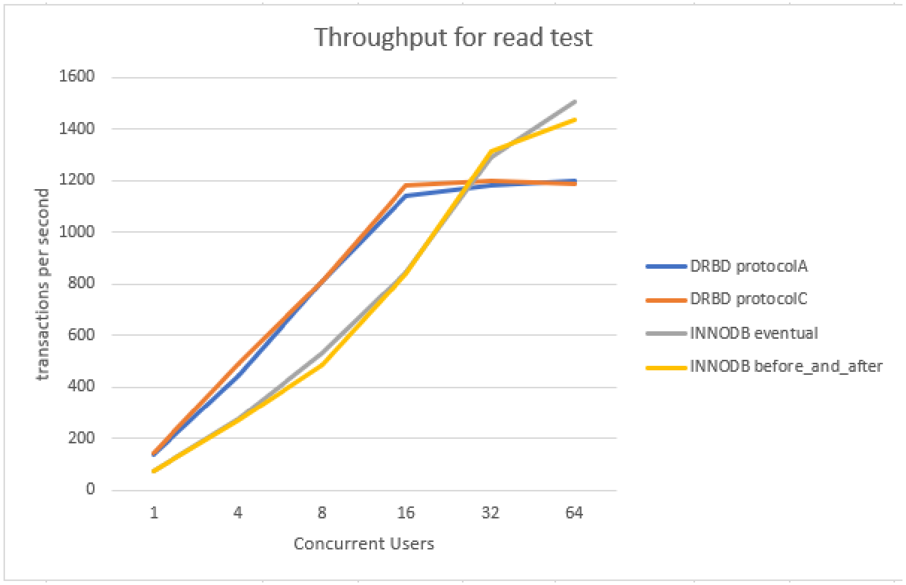

Figure 5.11:Throughput of read test for two clusters with all modes

5.3.2. Total Time:

Total time represents the response time i.e. the time needed for the CRUD transactions to be completed. Table 5.10 presents the total time for write test of the used clusters with all four cases (InnoDB eventual, InnoDB before_and_after, DRBD Protocol A, DRBD Protocol C) with different number of concurrent users. The InnoDB cluster of eventual consistency level outperforms the others regarding the performance for write test, where the total time needed for its requests is the least. Note that the lowest the total time needed the better the performance. As a result, clients will be more satisfied when their request takes less time to be completed.

Table 5.10: SysBench results of total time for write test for all clusters with all types

	Number of Threads				
Cluster Type	**1**	**4**	**8**	**16**	**32**
DRBD protocol A	10.03234	9.919637	7.382933	2.971787	1.676903
DRBD protocol C	10.03842	9.260047	6.165483	2.496223	1.663133
InnoDB eventual	9.951893	4.88357	3.121573	2.239877	1.236633
InnoDB before_and_after	10.02834	9.354733	8.66595	9.71932	10.47004

By contrast, as shown in Figure 5.12 the InnoDB of before_and_after consistency level shows the worst case amongst the other configurations, where requests need more time to be completed. Also, the time is getting higher when number of concurrent users are increasing. Moreover, the two protocols of DRBD system has approximately same values of total time and both perform better than InnoDB before_and_after consistency level.

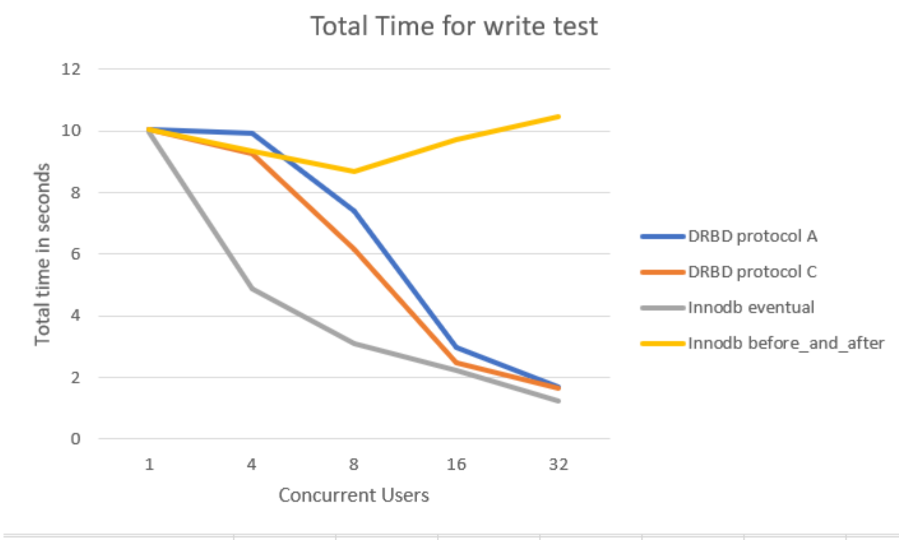

Figure 5.12: Total time for write test for two cluster with all modes

Table 5.11 presents the obtained total time for read test of the used clusters with all four cases (InnoDB eventual, InnoDB before_and_after, DRBD Protocol A, DRBD Protocol C) with different number of concurrent users.

Table 5.11::SysBench results of total time for read test for all clusters with all types

	Number of Threads					
Cluster Type	**1**	**4**	**8**	**16**	**32**	**64**
DRBD protocol A	7.22165	2.246007	1.232273	0.883133	0.845817	0.832103
DRBD protocol C	6.934123	2.033067	1.23244	0.847787	0.833847	0.840883
InnoDB eventual	10.01867	3.705107	2.002983	1.093983	0.71941	0.667993
InnoDB before_and_after	10.00832	3.69567	2.06427	1.19112	0.762643	0.728457

Both protocols of DRBD system are better (i.e., need less time) than InnoDB both levels when concurrent users are low. But when number of concurrent users are increasing the total time is getting closer to each other for all cases and the time is getting lower as shown in Figure 5.13.

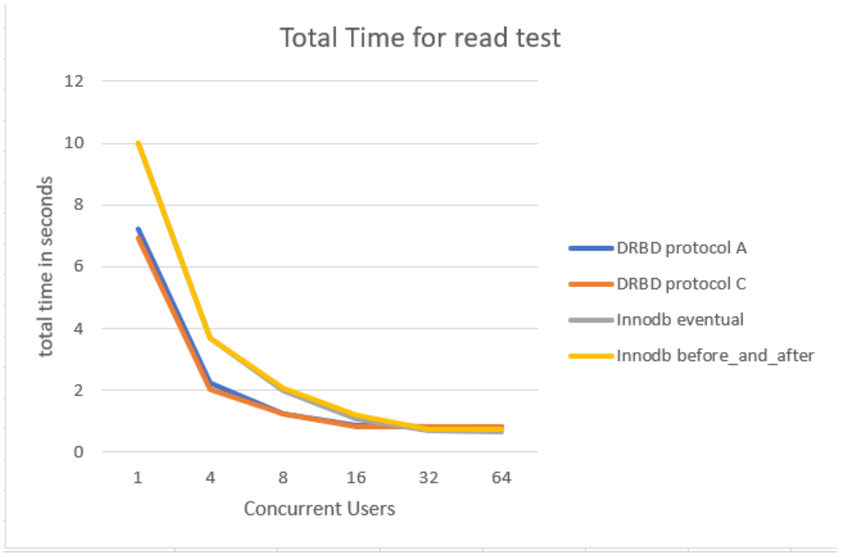

Figure 5.13 : Total time for two clusters for read test of all types

5.3.3. Cost:

Both clusters need different number of servers to be configured, where InnoDB cluster needs at least three servers to be configured, but DRBD needs at least two servers to be configured. It should be noted that both of clusters are scalable i.e. can accept more servers to be added to the cluster if needed which raises the cost. This variation of requirements can affect the decision makers of companies and organizations. So, when they decide to choose one of both those clusters, they will consider the overheads of the configuration in addition to the performance of the selected MySQL database solution.

5.4 Limitations

We have faced some difficulties in finding either real recorded historical data from real-life systems data or previous researches about performance evaluation of InnoDB cluster and DRBD performance with MySQL service. Up to our knowledge, there is neither recoded historical data for real-life system that uses InnoDB cluster or DRBD solutions nor previous research that evaluates MySQL performance of these solutions. Most of previous work focused on evaluating MySQL performance for standalone servers not as cluster. Consequently, the choice of the validation approaches and methods of the obtained results are affected and limited to statistical approach to validate the output data and subjective approach for the models [42].

Moreover, the results are limited for the used server's hardware specification, and for the characteristics used in the experiments like the data size and type, where may these characteristics affect the result of performance evaluation. Noting that all the used servers were on same VMware to mitigate the network latency issue.

5.5 Verification and Validation of the experimental Model

Verification of the Simulation Tool:

Since we have used a high-level simulation programming software/tool (i.e., SysBench tool) for carrying out the experiments with recommended parameters, we do not need to verify the correctness of our simulation model. Thus, we concentrate on parameters settings. Therefore, we do not need to verify the correctness of the utilize too [41]. Also, validation of the input data distribution, the dataset, and the transaction for the model were generated by the utilized software, which is designed for such experiments. Moreover, the number of replicas (i.e., the experiments and trials) should be sufficient in order to achieve a reasonable low variance in the means of the results. Therefore, the outputs were tested by incrementing the number of the trials until the variances of the results were relatively small (see Appendix A), which is achieved by replicating the experiments 30 times.

Validation of the Output Data:

Regarding the validity of the output data (results) and the sufficiency of replication trials (i.e., 30 replicas) are tested with 95% confidence (see Appendix A). The results in tables of Appendix A show the means, standard deviations, coefficient of Variance C.O.V and confidence intervals CI for the obtained results. It is clear that the C.O.V is relatively low, and all of the means are within their confidence intervals. These statistical measures indicate that the number of replications (i.e., 30 times of trials) is sufficient and the results (i.e., the means) are valid [42]. For more details on the utilized statistical measures see Appendix A.

Recall that absence of recorded data from a real-life system or results from related researches, limits the choices of the validation methods. Consequently, it is hard to accomplish statistical approach for validation of the results based on a comparison with existing data, since it will be time consuming and the time for this research is limited for recording new data. Therefore, in addition to the validity testing of the output results using statistical approach, we used subjective approach i.e., subject-matter experts (SMEs) opinions validation approach and sensitivity analysis for the validity of the output data using comparison-based approach [42]. Furthermore, in the sensitivity analysis we changed the affecting factors in the simulation and tested the resulting performance (i.e., throughput and response time). Thus, from the statistical measures and according to the SMEs the results are valid and can be accredited [42].

5.6 Summary of the results

The results are summarized and concluded as in Table 5.12, which will be helpful for decision makers to choose between these two solutions in terms of performance. Where the table shows the best cases obtained from the results. The outcome is concluded as following:

Table 5.12: Summary of the results as the best high availability solution in terms of performance (low <25 , high > 25)

Concurrency	**Transactions type**	
	Write	**Read**
Low	InnoDB cluster	DRBD
High	InnoDB cluster	InnoDB cluster

Where concurrency means the number of concurrent users that need to connect to MySQL database simultaneously. It should be noted that, the low value indicates that the number of users is from 1 to 25 user in these experiments. However, the high value represents a number more than 25 concurrent users. Also, transaction type refers to the type of query that the end user needs to use, where it can be read or write queries. Since some business needs to read from database more than to write, and others need the vice versa i.e., need to modify and save the data more than retrieving it.

Also, Table 5.12 shows that the InnoDB cluster with eventual consistency level with write queries performs better than before_and_after. But read queries column refers to both InnoDB consistency levels. However, DRBD system has no difference between its synchronization types, both have approximately the same values for read and write tests.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The goal of this research is to compare the database performance of two high availability MySQL solutions. In this work the two MySQL high availability solutions are compared and evaluated the performance of database, and how the end user will be satisfied when using MySQL clusters in terms of throughput and response time. There are many high availability solutions for MySQL. Two of them are chosen in this research, which are InnoDB cluster and DRBD. Where they are configured based on what is recommended with best practice to have best performance. They are tested for two of replication modes using benchmark tool (SysBench). Thereafter, the results are analyzed and compared, so the decision maker who is concerns of having a high available database can choose the cluster that is suites his resources and needs.

The results show that for write test, InnoDB eventual consistency level outperforms the performance of DRBD. However, InnoDB before_and_after consistency level shows worst performance compared to other cases. Moreover, DRBD both modes have approximately same throughput values, and its performance getting better when concurrent users increasing.

For read test, when the concurrent users are low, DRBD shows better performance than InnoDB in term of throughput and total time. But when number of concurrent users starts increasing, throughput becomes stable in DRBD and InnoDB cluster outperforms it. This is because it has two nodes for read queries. For read test both replication modes of InnoDB cluster have approximately same throughput, and DRBD both modes also have nearly same throughput.

Choosing a suitable high availability between these two solutions depends on different factors i.e., resources and business size. The resources that could be available, where InnoDB needs at least three servers to be configured, but DRBD can be configured by two servers. Moreover, the size of business that is related to the database, which may limit the number of concurrent users i.e. the load. Recall that the performance of database has been affected when number of concurrent users is increased. Where DRBD has just one active node and the another is passive. While, InnoDB has three active nodes, two for read only queries and one for read/write queries.

Recall that the absence of either recoded historical data of real-life systems that use InnoDB and DRBD solutions or previous research that evaluate such systems, validation of the obtained results is limited in selecting the validation approach. Therefore, we use subject-matter experts (SMEs) approach.

6.2 Future work

This work can be considered as first step towards the study of performance for possible MySQL high availability database solutions. Future work could be to extend this with an extensive study of other MySQL high availability solutions, evaluate the database performance for it with multi primary topology with more testing and failover scenarios.

References

[1] Bell C, Kindabl M and Thalmann L, MySQL High Availability: Tools for Building Robust Data Centers, 1st ed. Sebastopol, CA : O'Reilly, 2010, pp. 598. Accessed on: Jun 19,2020. [online]. Available: http://docs.linuxtone.org/ebooks/MySQL/MySQL_High_Availability_Tools_for_Building_Robust_Data.pdf

[2] Książek K, "Designing HA for MySQL", Jul 29, 2015. [slides]. Available: https://www.slideshare.net/Severalnines/dba-series-ha, Accessed on: Jun 19,2020.

[3] Shrestha R, "High Availability and Performance of Database in the Cloud - Traditional Master-slave Replication versus Modern Cluster-based Solutions," 7th International Conference on Cloud Computing and Services Science,Porto,Portugal,2017,pp 413-420.

[4] *MySQL/MariaDB HA: Galera Cluster vs. DRBD replication*, Adfinis, Aug 20, 2016. Accessed on: Jun 19, 2020. [online]. Available: https://adfinis.com/en/blog/mysql-mariadb-ha-galera-cluster-vs-drbd-replication

[5] Oles B, *Understanding the Effects of High Latency in High Availability MySQL and MariaDB Solutions*. severalnines, Mar 26, 2019. Accessed on: Jun 19, 2020. [online]. Available: https://severalnines.com/database-blog/understanding-effects-high-latency-high-availability-mysql-and-mariadb-solutions

[6] Schwartz B, Zaitsev P, Tkachenko V, Zawodny J, Lentz A and Balling D, High performance MySQL: optimization, backups, and replication, and more, 2nd ed. Sebastopol, CA : O'Reilly, 2008, pp. 684. Accessed on: Jun 19,2020.[online]. Available: https://www.it.iitb.ac.in/frg/wiki/images/b/b0/OReilly.High.Performance.MySQL.Optimization.Backups.Replication.And.More.2nd.Edition.Jun.2008.ISBN.0596101716.pdf

[7] *Group Replication*, MySQL, Jun 12, 2020. Accessed on: Jun 19, 2020. [online]. Available: https://docs.oracle.com/cd/E17952_01/mysql-5.7-en/group-replication-summary.html

[8] Araújo, M. G. D, "Database replication in large scale systems," M.S. thesis, college of Eng., do Minho Unv., Jun.2011. Accessed on: Jun. 19,2020. [Online]. Available: https://repositorium.sdum.uminho.pt/bitstream/1822/28946/1/eeum_di_dissertacao_pg13367.pdf

[9] *Replication Formats* ,MySQL, Jun. 18,2020. Accessed on: Jun. 19,2020. [Online].Available:https://dev.mysql.com/doc/refman/8.0/en/replication-formats.html

[10] Vanoverbeke D, *Different Types of MySQL Replication Solutions*, PERCONA, Feb. 7,2017. Accessed on: Jun. 19,2020. [Online]. Available: https://www.percona.com/blog/2017/02/07/overview-of-different-MySQL-replication-solutions/

[11] Mauchle F, "Database Replication with MySQL and PostgreSQL," college of Softw. and Syst., Applied Sciences Rapperswil Univ., Switzerland, 2008. Accessed on: Jun. 19,2020. [Online]. Available: https://wiki.hsr.ch/Datenbanken/files/Mauchle_Replication_MySQL_Postgres_Paper.pdf

[12] *MySQL Cluster Architecture Overview*, MySQL, Apr 2004.Accessed on: Jun. 19,2020. [Online]. Available: https://confluence.oceanobservatories.org/download/attachments/16418744/MySQL-cluster-technical-whitepaper.pdf

[13] *Semisynchronous Replication*, MySQL, Jun. 18,2020. Accessed on: Jun. 19,2020. [Online]. Available: https://dev.MySQL.com/doc/refman/5.5/en/replication-semisync.html

[14] Wiesmann M and Schiper A, "Comparison of database replication techniques based on total order broadcast," TKDE, vol.17,no.4,pp.551 - 566,Apr.2005. Accessed on : Jun. 19,2020. [Online]. Available:https://ieeexplore.ieee.org/document/1401893/authors#authors

[15] Malioutina E, "Replication Technology and Failover Solution Development for the MySQL Open Source Database Management System," M.S. thesis, college of comput. Sci.,Stockhom Univ,Sweden,2008. Accessed on: Jun. 19,2020. [Online]. Available : http://kiosk.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2008/rapporter08/malioutina_elena_08081.pdf

[16] *Group Replication Background*, MySQL, Jun. 19,2020. Accessed on: Jun. 20,2020. [Online]. Available: https://dev.MySQL.com/doc/refman/8.0/en/group-replication-background.html

[17] De Prisco, R., Lampson, B., & Lynch, N. Revisiting the Paxos algorithm, In International Workshop on Distributed Algorithms ,Springer, Berlin, Heidelberg,2005, pp 111-125.

[18] Vnier E,Shah B and Malepati T, Advanced MySQL 8: Discover the full potential of MySQL and ensure high performance of your database,1st ed. Birmingham, UK: Packt Publishing, 2019, pp. 286. Accessed on: Jun. 20,2020. [Online]. Available: https://searchworks.stanford.edu/view/13241768

[19] *Fine Tuning the Group Communication Thread,* MySQL, Jun. 19,2020. Accessed on: Jun. 20,2020.[Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/group-replication-fine-tuning-the-group-communication-thread.html

[20] *Multi-Primary Mode*, MySQL, Jun. 19,2020. Accessed on: Jun. 20,2020. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/group-replication-multi-primary-mode.html

[21] *Introducing InnoDB Cluster*, MySQL, Jun. 18,2020. Accessed on: Jun. 19,2020. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/mysql-innodb-cluster-introduction.html

[22] *MySQL InnoDB Cluster – A complete High Availability solution for MySQL*, dinfratechsource, Nov. 10,2018. Accessed on: Jun. 20,2020. [Online]. Available: https://dinfratechsource.com/2018/11/10/MySQL-InnoDB-cluster-a-complete-high-availability-solution-for-MySQL/

[23] *Configuring Transaction Consistency Guarantees*, MySQL, Jun. 18,2020. Accessed on: Jun. 19,2020. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/group-replication-configuring-consistency-guarantees.html

[24] *The DRBD9 User's Guide,* LINBIT. Accessed on: Jun. 20,2020. [Online]. Available: https://www.linbit.com/drbd-user-guide/drbd-guide-9_0-en/#about

[25] Jones M, *High availability with the Distributed Replicated Block Device*, IBM Developer, Aug. 4,2010. Accessed on: Jun. 19,2020. [Online]. Available: https://developer.ibm.com/tutorials/l-drbd/

[26] Philipp R and Ellenberg L,"Replicated Storage with Shared Disk Semantics," In Proceedings of the 12th International Linux System Technology Conference.,Germany,2005,pp 111-119.

[27] *Optimizing DRBD performance*, LINBIT. Accessed on: Jun. 20,2020. [Online]. Available: https://www.linbit.com/drbd-user-guide/drbd-guide-9_0-en/#p-performance

[28] Haas F, "Ahead of the Pack: The Pacemaker High-Availability Stack," LINUX JOURNAL, Jun. 18,2012. Accessed on: Jun. 20,2020. [Online]. Available: https://www.linuxjournal.com/content/ahead-pack-pacemaker-high-availability-stack#:~:text=And%20to%20ensure%20maximum%20service,communications%2C%20resource%20management%20and%20applications.

[29] Mike Z, *MySQL now provides support for DRBD*, MySQL High Availability, Sep 29,2012. Accessed on: Jun. 20,2020. [Online]. Available: https://mysqlhighavailability.com/MySQL-now-provides-support-for-drbd/

[30] Lua.(2019). Accessed on: Jun. 20,2020. [Online]. Available: https://www.lua.org/start.html

[31] Ksiazek K, *How to Benchmark Performance of MySQL & MariaDB Using SysBench*, severalnines, Jun. 12,2018. Accessed on: Jun. 20,2020. [Online]. Available:

https://severalnines.com/database-blog/how-benchmark-performance-mysql-mariadb-using-sysbench
[32] Paul S, "Database systems performance evaluation techniques," St. Louis, MI, USA, 2008. Accessed on: Jun. 20,2020. [Online]. Available: https://www.cs.wustl.edu/~jain/cse567-08/ftp/db.pdf
[33] Flinkman A J, “High-availability mechanisms in MySQL, Helsinki Univ,” 2007. Accessed on: Jun. 20,2020.[Online]. Available : https://pdfs.semanticscholar.org/035a/f7a72d61de6021940fc5e971a59cc26826a4.pdf?_ga=2.47921255.1337970251.1592644642-99908994.1592644642
[34] Pukdesree S, Lacharoj V and Sirisang P, "Performance Evaluation of Distributed Database on PC Cluster Computers," WSEAS transactions on computers,vol. 10,no. 1,pp. 21-30 , Jan 2011. Accessed on: Jun. 20,2020. [Online]. Available: https://www.researchgate.net/publication/228749856_Performance_evaluation_of_distributed_database_on_PC_cluster_computers
[35] Shivani, "An Empirical Study on Performance Evaluation of NoSQL Databases," International Journal of Electronics Engineering, vol. 10, no. 1, pp. 235-244, Jan. 2018.
[36] Kumar L, Rajawat S and Joshi K, "Comparative analysis of nosql (mongodb) with MySQL database,". International Journal of Modern Trends in Engineering and Research, vol. 2, no. 5, pp. 120-127, May. 7,2015. Accessed on: Jun. 20,2020. [Online]. Available: https://www.semanticscholar.org/paper/Comparative-analysis-of-NoSQL-(MongoDB)-with-MySQL-Kumar-Rajawat/b84d9920c7a4fe3ae7f625293166183153044618
[37] *Group Replication Performance*, MySQL, Jun. 19,2020. Accessed on: Jun. 20,2020. [Online]. Available:https://dev.mysql.com/doc/refman/8.0/en/group-replication-performance.html
[38] Kojima A, *MySQL InnoDB Cluster – A Hands on Tutorial*, MySQL Server Blog, Dec. 12,2016. Accessed on: Jun. 20,2020. [Online]. Available: https://mysqlserverteam.com/MySQL-InnoDB-cluster-a-hands-on-tutorial/
[39] *High Availability and Scalability*, MySQL, Accessed on: Jun. 20,2020. [Online]. Available: http://www.mysql.ru/docs/mysql-man-5.0-en/ha-overview.html
[40] Gryp K,Araújo M, "MySQL InnoDB Cluster-Tutorial", Sep ,2019. [slides]. Available: https://static.rainfocus.com/oracle/oow19/sess/1552388032514001eAhU/PF/TUT2070%20-%20MySQL%20InnoDB%20Cluster%3A%20High-Availability%20Tutorial_1568750112091001LM5l.pdf
[41] Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds, Verification and validation of simulation models, Proceedings of the 2011 Winter Simulation Conference, 2011.
[42] Averill M. Law, Simulation Modeling and Analysis, 2nd ed. Tucson, Arizona, USA, 2013, pp. 800.Accessed on: Jul 14,2020. [online]. Available: www.averill-law.com

Appendices

Appendix A

This appendix introduces some basic statistical analysis needed for evaluation the validity of the output results using central tendency measurement. It presents indices of dispersion measures (i.e., variance, standard deviation, coefficient of variance C.O.V, and confidence interval). Also, it shows the details of the obtained results of our experiments [42]. It presents the means, standard deviation, coefficient of variance and the confidence intervals of the means of throughputs of the experiment cases through Tables A.1- A.8.

Variability can be specified using one of the following indices dispersion measures such as Range (minimum and maximum of the data set), variance or standard deviation.

Range:

Range can be easily calculated by tracking minimum and maximum values,

Range= maximum – minimum. In many cases it is not useful. Since minimum often can be zero, and maximum might be far from typical value (outlier). With more samples, max may continue to rise, and min may continue to decrease → no "stable" point. Range is useful if system performance is limited.

Variance and Standard Deviation:

If there is a sample of n observations $\{x_1, x_2, \ldots, x_n\}$, the sample variance is calculated as follows:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \text{, } where \ \ \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Where s^2 is called sample variance and its square root s is called standard deviation.

Notice that in computing the variance, the sum of squares $\sum_{i=1}^{n}(x_i - \bar{x})$ is divided by (n-1) and not n, because there are n-1 out of n differences are independent. If there is a (n-1) differences, the nth difference can be calculated, since the sum of all n differences must be zero (Section 12.8 [42]). The number of independent terms is the "degrees of freedom" (df). The main disadvantage of variance is that it is expressed in square of the units of the observations. Therefore, it is better to use standard deviation.

Standard deviation:

Standard deviation SD and mean have the same units. Below are two examples with high and low variabilities:

Example **a)**

Mean = 2 seconds, SD = 2 seconds; high variability.

Example **b)**

Mean = 2 seconds, SD = 0.2 seconds; Low variability.

Coefficient of Variation (C.O.V):

C.O.V is another widely used measurement. It represents the ration of standard deviation to the mean (i.e. C.O.V= SD/mean) . This measurement can be considered as a better measurement than standard deviation, because it ignores the units of measurement in variability consideration.

Thus, a C.O.V of 5 is large, while a C.O.V. of 0.2 (20%) is small variation (regardless of the units).

It should be noted that the C.O.V in (Example a) is 1 and in (Example b) is 0.1.

Mean absolute Deviation:

Mean absolute deviation is calculated without multiplication or square root:

$$mean\ absolute\ deviation = \frac{1}{n}\sum_{i=1}^{n}|X_i - \bar{X}|$$

Confidence Interval (CI) for the Mean:

Recall that each sample mean is an estimate of the population mean. If we have *k* samples, we have *k* different estimates [42]. The goal is to obtain a single estimate. Actually, it is not possible to obtain an accurate estimate of the population mean from any finite number of finite sample size. However, we can obtain probabilistic bounds. Consequently, we can get two bounds, for example c1 and c2, such that there is a high probability (1-±) or (1-α), the mean is in the interval (c1, c2). The confidence interval is calculated as follows:

(mean-$z_{1-\alpha/2}$s/sqrt(*n*), mean+$z_{1-\alpha/2}$s/sqrt(*n*)), Where

(c1, c2) - is the confidence interval,

±, α - is the significance level (e.g., 0.1, 0.05, 0.01…)

100(1-α) - confidence level (e.g., 90%, 95%, 99%…),

(1-α) - confidence coefficient, and s is the SD of the sample.

$Z_{1-\alpha/2}$ - is the (1-α / 2)- quantile of the unit normal variate, which can be obtained from the table! (see Table A2 in [42]), in our case of 95% confidence and degree of freedom df=29, this value is 2.042. Thus, in our case the confidence interval (c1, c2) is:

(mean-2.042(s/sqrt(*n*)), mean+2.042(s/sqrt(*30*))) .

Thus, we can see in the tables below (Table A.1-Table A.8) that the obtained means of all throughputs fall within their confidence intervals.

Table A.1: coefficient of variation for results of InnoDB eventual level throughput of 30 trials of experiments for write test

	Number of threads				
InnoDB Eventual when value of group_replication_poll_spin_loops =0	**1**	**4**	**8**	**16**	**32**
Standard Deviation	17.11898718	54.26450632	98.76165	199.2143	305.5664
Mean	80.94566667	218.8976667	352.4203	537.8833	817.5393
Coefficient of Variation	0.211487383	0.247898971	0.280238	0.370367	0.373764
Confidence Interval (c1,c2)	74.56,87.32	198.66,239.12	315.6,389.24	463.61,612.15	703.61,931.45

Table A.2:coefficient of variation for results of InnoDB before_and_after level throughput of 30 trials of experiments for write test

	Number of threads				
InnoDB before_and_after when value of group_replication_poll_spin_loops =1000	**1**	**4**	**8**	**16**	**32**
Standard Deviation	12.26278578	17.35815349	40.21242	26.49273	18.1441
Mean	60.25433333	101.368	109.229	91.65967	72.59733
Coefficient of Variation	0.203517077	0.171238986	0.368148	0.289034	0.249928
Confidence Interval (c1,c2)	55.68,64.82	94.89,107.83	94.23,124.22	81.78,101.53	65.83,79.36

Table A.3: coefficient of variation for results of DRBD Protocol A throughput of 30 trials of experiments for write test

	Number of threads				
DRBD Protocol A when value of I/O unplug watermark =32	**1**	**4**	**8**	**16**	**32**
Standard Deviation	8.279121	20.09607	29.41284	130.2008	213.8107
Mean	27.75867	78.58433	138.8707	390.686	683.1583
Coefficient of Variation	0.298254	0.255726	0.2118	0.333262	0.312974
Confidence Interval (c1,c2)	24.67,30.84	71.09,86.07	127.9,149.83	342.14,439.22	603.44,762.87

Table A. 4: coefficient of variation for results of DRBD Protocol C throughput of 30 trials of experiments for write test

	Number of threads				
DRBD Protocol C when value of I/O unplug watermark =32	**1**	**4**	**8**	**16**	**32**
Standard Deviation	17.9989	44.68397	55.4905	127.2264	232.8679
Mean	45.02467	85.70067	174.449	455.317	678.2077
Coefficient of Variation	0.399756	0.521396	0.31809	0.279424	0.343358
Confidence Interval (c1,c2)	38.31,51.73	69.04,102.35	153.76,195.13	407.88,502.74	591.39,765.02

Table A.5: coefficient of variation for results of InnoDB eventual level throughput of 30 trials of experiments for read test

	Number of threads					
InnoDB Eventual when value of group_replication_poll_spin_loops =0	**1**	**4**	**8**	**16**	**32**	**64**
Standard Deviation	2.18747	6.977974	17.13637	65.42918	63.61652	122.1812
Mean	74.1573	276.737	531.776	847.091	1290.459	1503.022
Coefficient of Variation	0.02949	0.025215	0.032225	0.07724	0.049298	0.08129
Confidence Interval (c1,c2)	73.34,74.97	274.13,279.33	525.38,538.16	822.69,871.48	1266.74,1314.17	1457.47,1548.57

Table A.6: coefficient of variation for results of InnoDB before_and_after level throughput of 30 trials of experiments for read test

	Number of threads					
InnoDB before_and_after when value of group_replication_poll_spin_loops =1000	**1**	**4**	**8**	**16**	**32**	**64**
Standard Deviation	2.744366	7.598095	16.95814	32.91803	82.40658	261.1866
Mean	73.21133	270.5327	484.1593	838.714	1314.211	1434.031
Coefficient of Variation	0.037486	0.028086	0.035026	0.039248	0.062704	0.182135
Confidence Interval (c1,c2)	72.18,74.23	267.7,273.36	477.83,490.48	826.44,850.98	1283.48,1344.93	1336.6,1531.4

Table A.7: coefficient of variation for results of DRBD Protocol A throughput of 30 trials of experiments for read test

	Number of threads					
DRBD Protocol A when value of I/O unplug watermark =32	**1**	**4**	**8**	**16**	**32**	**64**
Standard Deviation	6.5164	28.43437	43.43331	93.34602	50.32001	33.64047
Mean	138.728	446.506	812.6197	1138.211	1181.895	1200.025
Coefficient of Variation	0.046972	0.063682	0.053449	0.082011	0.042576	0.028033
Confidence Interval (c1,c2)	136.29,141.15	435.9,457.1	796.42,828.81	1103.41,1173.01	1163.13,1200.65	11787.48,1212.56

Table A.8: coefficient of variation for results of DRBD Protocol C throughput of 30 trials of experiments for read test

	Number of threads					
DRBD Protocol C when value of I/O unplug watermark =32	**1**	**4**	**8**	**16**	**32**	**64**
Standard Deviation	4.374087	15.00438	30.53194	47.52411	39.71019	54.42307
Mean	144.301	491.8507	811.3977	1178.851	1197.87	1188.484
Coefficient of Variation	0.030312	0.030506	0.037629	0.040314	0.033151	0.045792
Confidence Interval (c1,c2)	142.67,145.93	486.25,497.44	800.01,822.78	1161.13,1196.56	1183.06,1212.67	1168.19,1208.77

Appendix B

This appendix presents the created data by the SysBench simulation tool for the experiments.

Table B.1: sample of the data that is generated into database by SysBench tool and used in the experiment

id	k	C	pad
1	1002923	68487932199-96439406143-93774651418-41631865787-96406072701-20604855487-25459966574-28203206787-41238978918-19503783441	22195207048-70116052123-74140395089-76317954521-98694025897
2	1004960	13241531885-45658403807-79170748828-69419634012-13605813761-77983377181-01582588137-21344716829-87370944992-02457486289	28733802923-10548894641-11867531929-71265603657-36546888392
3	1028745	71749523403-03621984679-84246721148-54647104962-38952275016-06464896823-14571026186-24257421698-84141554112-49791540502	41234744688-51641643591-52697224766-15020694408-68856618037
4	1005291	54133149494-75722987476-23015721680-47254589498-40242947469-55055884969-23675271222-20181439230-74473404563-55407972672	88488171626-98596569412-94026374972-58040528656-38000028170
5	59	21497257486-14006031362-20954507792-52915987979-03910680647-21006633133-10715894604-38455907716-44361535976-97953247731	94919115396-98824649950-78033843611-50697887870-70826638081
6	1004112	27259561572-06414927280-46715228218-84893654561-76731023738-20557067759-19583561006-08735004037-78310463408-71967134144	66200784124-83118543608-45044785936-98738951684-25087136427
7	1731816	26016962040-84401935623-82022405434-08698958003-65003728728-28968010755-58339263572-01789602703-39811082909-08525005410	02358180554-15815773302-92697093801-06820739293-88955012668
8	1000507	97747537709-99230977413-59153069647-62890934635-11729491470-07415848518-61416594520-16559727341-46432156293-62484046618	94633519802-41908941667-78242579588-16891567524-49216405523
9	912351	52787004896-03713561258-89030827639-79870405712-99021527444-79036649924-19443487452-29368183864-55806406503-87227017685	82343182218-75122207739-98082475690-33929140169-00985969597
10	569367	00415922346-91627720052-70099716576-52830063896-15696986113-66985501675-77110646865-39221301044-85038367777-40632164178	07449874956-85062155369-35881996476-14158883642-64897422580
11	1009644	68305043604-07392484646-78480928447-88155597080-08908465928-35357008626-44894171482-13904841657-13998032237-49278517178	82525607519-86035420736-51783285478-30216080554-19134890096
12	995201	84225420767-95119807827-48689909948-04145663437-29723649568-88238910120-61256632514-12324871715-71270848294-09484980067	91218229232-45669484074-60159597493-60558506146-76239412517
13	1538098	23701604887-44671845013-88591550229-83592334526-03371749712-02384181617-26831671666-20188224074-64547017922-97641736034	18878447083-51518715478-08505273803-98186184968-06782243308

14	1000477	64842668315-60792497498-62750863276-45625090087-75713057101-27160306431-79536585757-87296141069-47125388407-35079320070	55933840247-47337803868-79571705078-03223267064-04605777986
15	1216352	95419146678-14093409261-09962384278-71846018805-74601632203-97534503889-75491510988-93003967293-09007318291-32570920479	58506892021-02842031308-87774929720-59501190013-74028991815
16	1864373	32094662126-89225014700-79152064371-08455689349-03137928650-55175661781-49326570184-45630878875-69816385564-33358002874	73609511824-17903462079-43512858897-43546610069-94699539415
17	1006810	71448009014-71834791236-61164972585-70749298280-80542194040-57578052588-95202958716-10334483721-64225565322-50266191520	82792905088-72706133952-00547821102-64831494800-59296447588
18	1492030	78838609893-46348600857-07072734072-15051355461-52622106468-04383929752-42738380886-21129359785-07287880963-16697504004	81103306942-71781756861-75275133854-25619966886-10225409478
19	1121987	90051425691-25789925225-11346061836-86027481503-56888390198-29184614639-20597516557-59209249918-30962581385-32551385958	56649223465-05395201594-30598644923-67595771828-36110187986
20	1859585	30110320760-92038482878-35745432461-23903545970-76501097089-02250160299-17775142558-19081672308-36543406505-99353980746	11308973902-51522513977-11061491231-08659548052-02924801260
21	107524	16690530377-63987168491-72172776123-80117937302-92509900367-95921774112-79338680468-42947724750-62405208456-01987401862	65136224409-33706000949-96439751543-67924295910-44782822297
22	1001744	15263455457-58075632796-44895195635-64246175113-09306502497-70016121495-35716128361-81345533735-43361537551-40713590514	80385226158-87059079968-82865735167-90330450895-98846983203
23	1882671	43578579884-68500126356-36545380402-53520257942-40842833170-36650534101-80609975436-04837531511-84168834240-66986216142	93655585782-21092397652-11370478330-93836554012-92574026450
24	1086311	23092203816-47751292915-52445457243-37532319024-52754430958-94884358644-92217537524-69630011136-58075625595-12276707316	16668590421-97960154678-96679208706-82117037773-28162486749
25	1411922	78839613741-27483355158-07765701483-18206947477-18511609137-54680149758-12877124569-57138303285-69932369353-21819235763	49300657906-25122359700-61884477637-37175101640-91187374434
26	998809	94028665321-27368401356-46526587557-73692072145-37888180908-21695018941-29833499555-88732094221-81150570616-33523419975	11589584481-27280373851-09621824390-62557697530-87005061141
27	1003004	76133995643-03630441214-63521419347-94780563180-43900509498-15133366912-64213550032-42596168518-60920051261-45367112919	64436625952-42090500434-15425150061-32331186139-83808254918
28	1006442	57776552922-11880051588-58075076667-23997402256-69520667378-97791137572-99703195908-45604024869-11666111212-97020413349	12237021043-92068457966-22695573331-53702842209-86646342777
29	998950	81169731792-73974368494-07505456106-79874259318-27707978883-25191598376-01617415993-11613717249-19708218490-51428364177	28886928848-52163576179-56338762801-66820300137-36760460345

مقارنة أداء حلول ذات التوافر العالي لقاعدة البيانات (MySQL)

إعداد: الاء ياسرفايز محسن

إشراف: د. نضال الكفري

ملخص:

الكثير من الشركات والمؤسسات على مستوى العالم اليوم تهدف الى تنظيم وتخزين بياناتهم لتسهيل الحصول عليها وتسهيل ادارتها ,وذلك من خلال توفير نظام ادارة لقواعد البيانات, اي مجموعة برامج حاسوبية تتحكم في ذلك. دون شك هذه المؤسسات تهدف الى الحصول على اداء عالٍ لهذا النظام لزيادة عدد المستخدمين وبالتالي زيادة الارباح , في هذا البحث تم اجراء مقارنة لاداء قاعدة البيانات MySQL بعد ان تم اعدادها على نوعين من حلول التوافر العالية High Availability.

تم عمل مقارنة لاداء قاعدة البيانات MySQL بعد اعدادها على نوعين من حلول High Availability التي ستساعد اصحاب القرار في الشركات والمؤسسات في اختيار الحل الانسب الذي يتوافق مع احتياجاتهم, علماً أن حلول التواجد العالي لقاعدة البيانات Database High Availability عبارة عن حل تكنولوجي يمكن المستخدم من الوصول الى قاعدة البيانات لأطول فترة ممكنة دون الشعور بمشكلة في قاعدة البيانات في حالة وجود فشل ما كاعطال الشبكة وغيرها من المشاكل مع ضمان عدم حصول اي فقدان لهذه البيانات ,وذلك من خلال تواجد نسخة متزامنة لقاعدة البيانات هذه على اكثر من خادم بأكثر من طريقة.

تمت هذا المقارنة على خوادم بنفس المواصفات Hardware Specification, على أساس الإنتاجية; اي المعدل الأقصى من العمليات التي يمكن تنفيذها خلال وحدة زمن معينة, وزمن الاستجابة; اي الوقت الذي تحتاجه العملية حتى يتم تنفيذها.

هذان الحلان هما نظام MySQL InnoDB ونظام DRBD ,علما أن كل منهما يعتمد على تكنولوجيا مختلفة وبمواصفات مغايرة. حيث ان نظام InnoDB يعتمد على تكنولوجيا MySQL Group Replication, ونظام ال DRBD يعتمد على نظام Replicated Disk Architecture , تم اختيارهما لعدم وجود ابحاث سابقة تتحدث عن المقارنة بينهما بخصوص اداء قاعدة البيانات MySQL,كما ان كلاهما يقدمان خدمة automatic failover اي انه في حالة وجود فشل او مشكلة باحدى الخوادم سيتم تحويل التطبيق (المستخدم) بشكل اوتماتيكي للخادم الفعال, وايضا يعتبران فعالين في تقديم توافرية عالية للمستخدم, لذا مقارنة اداء قواعد البيانات بينهما سيكون مفيد لاصحاب القرار حتى يتسنى لهم اتخاذ القرار الانسب والاكثر دقة لاحتياجاتهم ومتطلباتهم.

يوجد لكل حل من هاذين الحلين أكثر من طريقة لتزامن البيانات ,اي تزامن بيانات الخادم الرئيسي مع الخوادم الاخرى (الثانوية) , تم اختيار نوعين من هذه الطرق لكل حل وتقييم اداء قاعدة البيانات بالنسبة لهما. أيضاً تم اجراء التجربة لكل نظام بهيكلية single primary topology , التي تعني أن يتواجد جهاز خادم رئيسي واحد, الذي يُسمح بالكتابة عليه وباقي الخوادم يتم تزامن البيانات عليها مع الرئيسي دون السماح بالكتابة عليها.

تم إعداد هذه الأنظمة بناءً على ما هو موصى به من الاعدادات اللازمة للحصول على أفضل أداء لقاعدة البيانات, حيث تم إجراء نفس الإختبارات (اختبارات القراءة والكتابة) على نفس البيانات مع تغيير عدد المستخدمين المتزامنين لكلا النظامين باستخدام أداة SysBench التي تعتمد على معايير معينة, وتقدم قيم الانتاجية وزمن الإستجابة كنتيجة لهذه الاختبارات.

أظهرت النتيجة أن نظام InnoDB قد تفوق على الآخر في اختبارات الكتابة بجميع حالات اختلاف عدد المستخدمين المتزامنين, كما أنها تفوقت أيضا باختبار القراءة على نظام DRBD عندما كان عدد المستخدمين المتزامنين مرتفع, ولكن عندما كان عددهم قليل نسبيا, نظام DRBD أظهر أداءً أفضل بهذا الاختبار(القراءة) .